PROTOTIPO DE HERRAMIENTA PARA LA MEJORA EN LOS PROCESOS DE DESIGNACIÓN DE PQRSD DE LAALCALDIA DE BUCARAMANGA

WILFREDO ARIEL GÓMEZ BUENO
EDSON ANDRÉS GÓMEZ CÁRDENAS

Nota de Aceptación

Certificamos que el presente Trabajo de Grado Satisface, en alcances y calidad, todos los requisitos que demanda un Trabajo de Grado de Maestría.

Diego Luis Linares
Director del Trabajo de Grado

Gloria Inés Álvarez
Co Directora del Trabajo de Grado

Maria Constanza Pabón
Jurado

Juan Carlos Martínez Arias
Jurado

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ciencia de Datos.

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Cali, 30 de agosto de 2023

Acta de Correcciones al Documento de Trabajo de Grado

Santiago de Cali, 30 de agosto de 2023

Autor: Wilfredo Ariel Gómez Bueno y Edson Andrés Gómez Cárdenas

Título del Trabajo de Grado: "PROTOTIPO DE HERRAMIENTA PARA LA MEJORA EN LOS PROCESOS DE DESIGNACIÓN DE PQRSD DE LA ALCALDIA DE BUCARAMANGA"

Director:

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.

Diego Luis Linares
Director del Trabajo de Grado

Gloria Inés Álvarez
Co Directora del Trabajo de Grado

Santiago de Cali, 14 de julio de 2023

Ingeniero:
Juan Carlos Martínez Arias
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ciencia de Datos, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado *"Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga*", el cual será realizado por el (la) estudiante Wilfredo Ariel Gómez Bueno y Edson Andrés Gómez Cárdenas con código 8972719 y 8972701 respectivamente perteneciente al énfasis en Sistemas y Computación, bajo la dirección del profesor Diego Luis Linares Ospina y la profesora Gloria Inés Álvarez.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

EDSON ANDRÉS GÓMEZ CÁRDENAS
ESTUDIANTE

WILFREDO ARIEL GÓMEZ BUENO
ESTUDIANTE

DIEGO LUIS LINARES
DIRECTOR

GLORIA INES ÁLVARES
CO DIRECTORA

Bucaramanga, 6 de noviembre de 2022

Señores
PONTIFICIA UNIVERSIDAD JAVERIANA
SEDE CALI
E. S. D.

Asunto: Autorización de uso de base de datos.

Respetados señores,

En mi calidad de Asesor Despacho Alcalde Grado 26 a cargo del proceso de Gestión de las TIC del Municipio de Bucaramanga, identificado con C.C. 91535007 de Bucaramanga me permito autorizar para realizar uso de la base de datos del Sistemas de Información y/o software denominado **GESTIÓN DE SOLICITUDES DEL CIUDADANO -GSC.** Lo anterior, para uso exclusivo en el desarrollo de actividades académicas asociadas a la tesis de grado adelantada dentro del programa de posgrado **MAESTRÍA DE CIENCIA DE DATOS, relacionada al *Desarrollo de un prototipo de herramienta utilizando técnicas y modelos de analítica de datos, para el soporte de los procesos de designación de PQRSD al interior de las dependencias de la alcaldía de Bucaramanga.***

Ahora bien, teniendo en cuenta que dicha información solicitada se encuentra alojada en la infraestructura tecnológica del municipio de Bucaramanga y que la misma contiene datos de índole privado y/o sensible conforme a la normatividad vigente, se recomienda cumplir las formalidades, condiciones, requisitos, reserva, manejo, uso salvaguarda y disposición final de la información de acuerdo con la Ley y sus normar reglamentarias. tener presente, las siguientes consideraciones para el manejo de la información:

1. Tratamiento de la Información:
   Se deberá llevar a cabo el Tratamiento de los Datos Personales de los Titulares que se suministrarán con las bases de datos requeridas, en cumplimiento de lo siguiente:

- Política de Seguridad de la Información de la Alcaldía de Bucaramanga.
- La Política de Tratamiento de Información Personal de la Alcaldía, publicada en la página web www.bucaramanga.gov.co.
- Los principios establecidos en la Ley 1581 de 2012.
- Las normas contenidas en la Ley 1581 de 2012 y demás normas que la reglamenten o la modifiquen y que hagan parte integrante del Régimen General de Protección de Datos Personales en Colombia.

2. Propiedad de la Información:
   La Pontifica Universidad Javeriana y/o el tercero que legalmente tenga acceso a la información entregada, deben tener presente que, las bases de datos que contienen datos personales a las que tiene acceso, son de exclusiva propiedad del Municipio de Bucaramanga.

3. Finalidad:

El Municipio de Bucaramanga sólo autoriza a la Pontifica Universidad Javeriana y/o el tercero que legalmente tenga acceso a la información entregada, a realizar el Tratamiento de esta conforme a las finalidades establecidas en su recolección.

4. Confidencialidad de la Información:
Se deberá mantener bajo reserva, absoluta confidencialidad y sin revelación a terceras personas no autorizadas, la Información y Datos Personales que sean entregados por el Municipio de Bucaramanga y en consecuencia, garantizar la confidencialidad, integridad y disponibilidad de dicha información.

La información y los Datos Personales entregados por el Municipio tienen el carácter de confidencial, cualquiera que sea el medio bajo el cual hayan sido facilitados, comprendiendo la información suministrada en software o en medios de almacenamiento electrónico.

5. Custodia de la información:
Se recomienda que, de suministrar la información a un tercero legalmente autorizado, al momento de la terminación del vínculo legal o contractual por cualquier causa, proceda con la devolución de las Bases de Datos entregadas por parte del Municipio de Bucaramanga, sin derecho a conservar ni original ni copia alguna.

6. Manejo de la información:
Se recomienda al Pontifica Universidad Javeriana y/o el tercero legalmente autorizado poner en conocimiento de sus contratistas, empleados y demás personal, la finalidad del tratamiento de la información y será responsable de que estos cumplan con el correcto manejo de la información entregada. En igual sentido deberá proceder el o el tercero que legalmente autorizado que acceda a la información entregada.

Cordialmente,

WILFREDO ARIEL GOMEZ BUENO
Asesor Despacho Alcalde
Oficina Asesora TIC

Maestría en Ciencia de Datos
Facultad de Ingeniería y Ciencias

FICHA RESUMEN
PROTOTIPO DE HERRAMIENTA PARA LA MEJORA EN LOS PROCESOS DE DESIGNACION DE PQRSD DE LA ALCALDÍA DE BUCARAMANGA

1. **ÁREA DE TRABAJO:** *PROCESAMIENTO DE TEXTO Y PROCESAMIENTO DE LENGUAJE NATURAL*
2. **TIPO DE PROYECTO:** *PROYECTO APLICADO*
3. **ESTUDIANTE(S):** *WILFREDO ARIEL GÓMEZ BUENO, EDSON ANDRÉS GÓMEZ CÁRDENAS*
4. **CORREO ELECTRÓNICO:** *wiargobu@javerianacali.edu.co, edsongomezc@javerianacali.edu.co*
5. **DIRECCIÓN Y TELEFONO:** *Calle 55 #23-19 Bucaramanga*
6. **DIRECTOR:** Diego Luis Linares
7. **VINCULACIÓN DEL DIRECTOR:** *Docente Asociado*
8. **CORREO ELECTRÓNICO DEL DIRECTOR:** *dlinares@javerianacali.edu.co*
9. **CODIRECTOR:** Gloria Inés Álvarez Vargas
10. **GRUPO O EMPRESA QUE LO AVALA (Si aplica):** *ALCALDIA DE BUCARAMANGA*
11. **OTROS GRUPOS O EMPRESAS:** NA
12. **PALABRAS CLAVE:** pqrsd, npl, procesamiento textos, solicitudes, soporte decisiones.
13. **ODS QUE APLICA EL PROYECTO (Agenda 2030): 11.** Ciudades y comunidades sostenibles
14. **FECHA DE INICIO:** *04 de junio de 2022*
15. **DURACIÓN ESTIMADA (En meses):** 12 Meses
16. **RESUMEN:**

Este proyecto de grado surgió de la identificación de un problema clave en la gestión de las Peticiones, Quejas, Reclamos, Sugerencias y Denuncias (PQRSD) en la Alcaldía de Bucaramanga: cuando una PQRSD se asigna incorrectamente, se producen reprocesos que disminuyen el tiempo para su respuesta y generan incumplimientos. En el peor de los casos, una PQRSD puede incluso perderse administrativamente debido a la rotación del personal.

Frente a este escenario, el objetivo principal de nuestro proyecto fue desarrollar una solución que pudiera impactar positivamente la asignación de las PQRSD en la Alcaldía. Para ello, utilizamos diversas técnicas de clasificación para analizar una base de datos de PQRSD compuesta por más de 590.592 registros con baja calidad de datos. El proyecto incluyó varias etapas, desde la extracción, exploración y limpieza de datos, hasta la creación y prueba de una arquitectura de clasificación mixta o híbrida basado en técnicas tradicionales.

El modelo propuesto considera el desequilibrio presente en los registros entre las distintas dependencias de la Alcaldía y utiliza herramientas como GridSearchCV para seleccionar y optimizar los hiperparámetros de los clasificadores utilizados como Regresión Logística y ComplementNB en los cuatro (4) componentes de la arquitectura. Los resultados obtenidos,

brindar un resultado de 65.1%, lo que sugiere que este enfoque puede ser eficaz para mejorar la gestión de las PQRSD en la Alcaldía de Bucaramanga, minimizando reprocesos, evitando incumplimientos y asegurando una gestión eficaz de las mismas, incluso en el contexto de la rotación del personal.

PROTOTIPO DE HERRAMIENTA PARA LA MEJORA EN LOS PROCESOS DE DESIGNACIÓN DE PQRSD DE LA ALCALDIA DE BUCARAMANGA

WILFREDO ARIEL GÓMEZ BUENO
EDSON ANDRÉS GÓMEZ CÁRDENAS

Informe Final del Proyecto Aplicado para optar al título de Magister en Ciencia de Datos

Director
Diego Luis Linares
Codirectora
Gloria Inés Álvarez

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI
2023

FICHA RESUMEN

INFORME FINAL DE TRABAJO DE GRADO

PROTOTIPO DE HERRAMIENTA PARA LA MEJORA EN LOS PROCESOS DE DESIGNACION DE PQRSD DE LA ALCALDÍA DE BUCARAMANGA

1. **ÁREA DE TRABAJO:** *PROCESAMIENTO DE TEXTO Y PROCESAMIENTO DE LENGUAJE NATURAL*
2. **TIPO DE PROYECTO:** *PROYECTO APLICADO*
3. **ESTUDIANTE(S):** *WILFREDO ARIEL GÓMEZ BUENO, EDSON ANDRÉS GÓMEZ CÁRDENAS*
4. **CORREO ELECTRÓNICO:** *wiargobu@javerianacali.edu.co, edsongomezc@javerianacali.edu.co*
5. **DIRECCIÓN Y TELEFONO:** *Calle 55 #23-19 Bucaramanga*
6. **DIRECTOR:** Diego Luis Linares
7. **VINCULACIÓN DEL DIRECTOR:** *Docente Asociado*
8. **CORREO ELECTRÓNICO DEL DIRECTOR:** *dlinares@javerianacali.edu.co*
9. **CODIRECTOR:** Gloria Inés Álvarez Vargas
10. **GRUPO O EMPRESA QUE LO AVALA (Si aplica):** *ALCALDIA DE BUCARAMANGA*
11. **OTROS GRUPOS O EMPRESAS:** NA
12. **PALABRAS CLAVE:** pqrsd, npl, procesamiento textos, solicitudes, soporte decisiones.
13. **FECHA DE INICIO:** *04 de junio de 2022*
14. **DURACIÓN ESTIMADA (En meses):** 12 Meses
15. **RESUMEN:**

Este proyecto de grado surgió de la identificación de un problema clave en la gestión de las Peticiones, Quejas, Reclamos, Sugerencias y Denuncias (PQRSD) en la Alcaldía de Bucaramanga: cuando una PQRSD se asigna incorrectamente, se producen

reprocesos que disminuyen el tiempo para su respuesta y generan incumplimientos. En el peor de los casos, una PQRSD puede incluso perderse administrativamente debido a la rotación del personal.

Frente a este escenario, el objetivo principal de nuestro proyecto fue desarrollar una solución que pudiera impactar positivamente la asignación de las PQRSD en la Alcaldía. Para ello, utilizamos diversas técnicas de clasificación para analizar una base de datos de PQRSD compuesta por más de 590.592 registros con baja calidad de datos. El proyecto incluyó varias etapas, desde la extracción, exploración y limpieza de datos, hasta la creación y prueba de una arquitectura de clasificación mixta o híbrida basado en técnicas tradicionales.

El modelo propuesto considera el desequilibrio presente en los registros entre las distintas dependencias de la Alcaldía y utiliza herramientas como GridSearchCV para seleccionar y optimizar los hiperparámetros de los clasificadores utilizados como Regresión Logística y ComplementNB en los cuatro (4) componentes de la arquitectura. Los resultados obtenidos, brinda un resultado de 65.1%, lo que sugiere que este enfoque puede ser eficaz para mejorar la gestión de las PQRSD en la Alcaldía de Bucaramanga, minimizando reprocesos, evitando incumplimientos y asegurando una gestión eficaz de las mismas, incluso en el contexto de la rotación del personal.

TABLA DE CONTENIDO

TABLA DE TABLAS

TABLA DE ILUSTRACIONES

INTRODUCCIÓN

La Constitución Política Colombiana, en sus artículos 23 y 74, establece que todas las personas tienen derecho a presentar peticiones respetuosas a las autoridades por motivos de interés general o particular, y a acceder a documentos públicos, excepto en los casos que la ley establezca como reservados. La función administrativa está al servicio de los intereses generales de la comunidad y se rige por los principios de debido proceso, igualdad, imparcialidad, buena fe, moralidad, participación, responsabilidad, transparencia, publicidad, coordinación, eficacia, economía y celeridad (Artículos 209 C.N, artículo 3° ley 1437 de 2011 Código de Procedimiento Administrativo y de lo Contencioso Administrativo).

En este contexto, el presente proyecto desarrolló un prototipo de herramienta que utiliza técnicas de análisis de datos para apoyar los procesos de asignación de PQRSD dentro de las dependencias de la Alcaldía de Bucaramanga. Esta herramienta permite, a través del seguimiento y las funcionalidades de alertas tempranas, reducir significativamente la cantidad de PQRSD que no se responden en los tiempos legales y aquellas que se responden de manera extemporánea. Además, permite la medición de indicadores de gestión de políticas y planes a nivel municipal y nacional en el sistema de gestión y desempeño MIPG.

1. DEFINICIÓN DEL PROBLEMA

El costo reputacional, el desgaste administrativo por responder a instancias judiciales, las sanciones por incumplimiento en los tiempos y procesos de investigación son las consecuencias de contar con un proceso de gestión de PQRSD deficiente. Así mismo, tener un sistema PQRSD requiere personal constantemente capacitado que, en el sector público se ve afectado por escenarios comunes como la alta rotación de personal que evidencian frecuentes asignaciones equivocas y errores humanos en el uso de los sistemas de PQRSD. Estos errores finalmente generan reprocesos y favorecen posibles incumplimientos en las respuestas a los ciudadanos. Muchos de los sistemas actuales tienen procesos de generación de reportes muy pasivos, donde solo se obtienen estadísticas básicas que rara vez pueden retroalimentar el sistema y permitir la toma de decisiones o acciones frente a posibles patrones que puedan ser obtenidos del procesamiento de grandes volúmenes de datos que producen estos sistemas.

1.1 PLANTEAMIENTO DEL PROBLEMA

La Alcaldía de Bucaramanga, a través de su Administración Municipal, implementó un Software de Gestión de Servicio a la Ciudadanía (GSC), que permite la unificación de las solicitudes que ingresan a la Alcaldía por los diferentes canales de atención. Sin embargo, se evidencian procesos de generación de reportes muy pasivos, con datos muy básicos, que dificultan la retroalimentación del sistema y la generación de soporte o información importante para la toma de decisiones o acciones frente a posibles comportamientos que puedan ser obtenidos del procesamiento de grandes volúmenes de datos que producen estos sistemas.

Durante el primer trimestre de 2022, la Administración Municipal recibió 73.254 solicitudes por los diferentes canales de atención, de las cuales el 91% fueron radicadas a través del módulo PQRSD. Comparado con el total de solicitudes del trimestre

inmediatamente anterior, que fue de 40683, hubo un aumento del 80,06%. Asimismo, se detectó una variación en los radicados efectuados a través del módulo PQRSD [1].

Al radicar la solicitud, los usuarios realizan consultas que, debido a los temas requeridos, deben ser asignadas a diferentes dependencias. Esto genera que el número de asignaciones aumente respecto al número de radicados, dado que se generan asignaciones múltiples con el mismo número de radicado, lo que evidencia el tratamiento dado a cada una de las PQRSD por dependencias.

En la siguiente tabla se puede evidenciar que, del cuarto trimestre del año 2021 al primer trimestre de 2022, las solicitudes asignadas en el módulo PQRSD se incrementaron en un 86,50% [1].

Tabla 1. ***Solicitudes asignadas en el módulo PQRSD***

NOVEDADES POR TRIMESTRE	**CANT. AÑO 2019**	**CANT. AÑO 2020**	**CANT. AÑO 2021**	**CANT. AÑO 2022**
Primer Semestre	2.676	3.774	59.605	**67.372**
Segundo Semestre	2.194	22.588	43.846	**N/A**
Tercer Semestre	1.966	28.210	33.967	**N/A**
Cuarto Semestre	1.623	29.600	36.123	**N/A**
TOTAL	**8.459**	**84.172**	**173.541**	**67.372**

Fuente: Informes Comportamentales vigencias 2019-2020-2021-2022

Se puede evidenciar el incremento del 80% en las asignaciones para el primer trimestre del 2022 respecto al trimestre anterior, esto debido al aumento de solicitudes de impuestos (predial e industria y comercio) en la secretaría de Hacienda.

Análisis de trámites de acuerdo con los tiempos de vencimiento legales estipulado. La tabla que se presenta a continuación representa el acumulado de los casos contestados, según los términos establecidos en la normatividad leal vigente [1]:

Tabla 2. ***Análisis de trámites en el marco legal***

DEPENDENCIAS	CONTESTADAS DENTRO DEL TRÁMITE DE LEY	VENCIDAS SIN RESPUESTA A LA FECHA DE CORTE DEL INFORME	CONTESTADAS DESPUÉS DE LA FECHA DE VENCIMIENTO	SIN RESPUESTA A LA FECHA DE CORTE DEL INFORME (Dentro de términos para otorgar respuesta)	TOTAL
Secretaría Administrativa	2.308	75	2.270	2.656	**7.309**
Secretaría de Desarrollo Social	290	183	126	307	**906**
Secretaría de Planeación	832	39	60	474	**1.405**
Secretaría del Interior	1.384	162	280	428	**2.254**
Secretaría de Salud y ambiente	1.424	27	138	420	**2.009**
DADEP	182	9	14	49	**254**
Secretaría de Infraestructura	544	46	41	119	**750**
Secretaría de Educación	417	3	73	78	**571**
OFAI	3	0	0	1	**4**
Secretaría Jurídica	346	11	45	40	**442**
Oficina de Valorización	146	0	0	22	**168**
Sisbén	171	1	4	14	**190**
Secretaría de Hacienda	46.807	1.098	509	2.458	**50.872**
Despacho Alcalde	63	0	1	4	**68**
Oficina Asesora TIC	43	0	0	0	**43**
Oficina Control Interno Disciplinario	32	0	0	0	**32**
Oficina de Control Interno de Gestión	3	0	0	0	**3**
Oficina de Prensa y comunicaciones	82	0	0	0	**82**
Unidad Técnica de Servicios Públicos	10	0	0	0	**10**
TOTAL	**55.087**	**1.654**	**3.561**	**7.070**	**67.372**

Fuente: Informes Comportamentales vigencias 2019-2020-2021-2022

Al revisar durante todo el funcionamiento del aplicativo de PQRSD, desde el año 2016 hasta el mes de junio de 2022, se observa que el acumulado de respuestas extemporáneas sin respuesta o con respuesta superan las 148 mil y que se encuentran 7.312 sin respuesta, como se observa en la tabla a continuación:

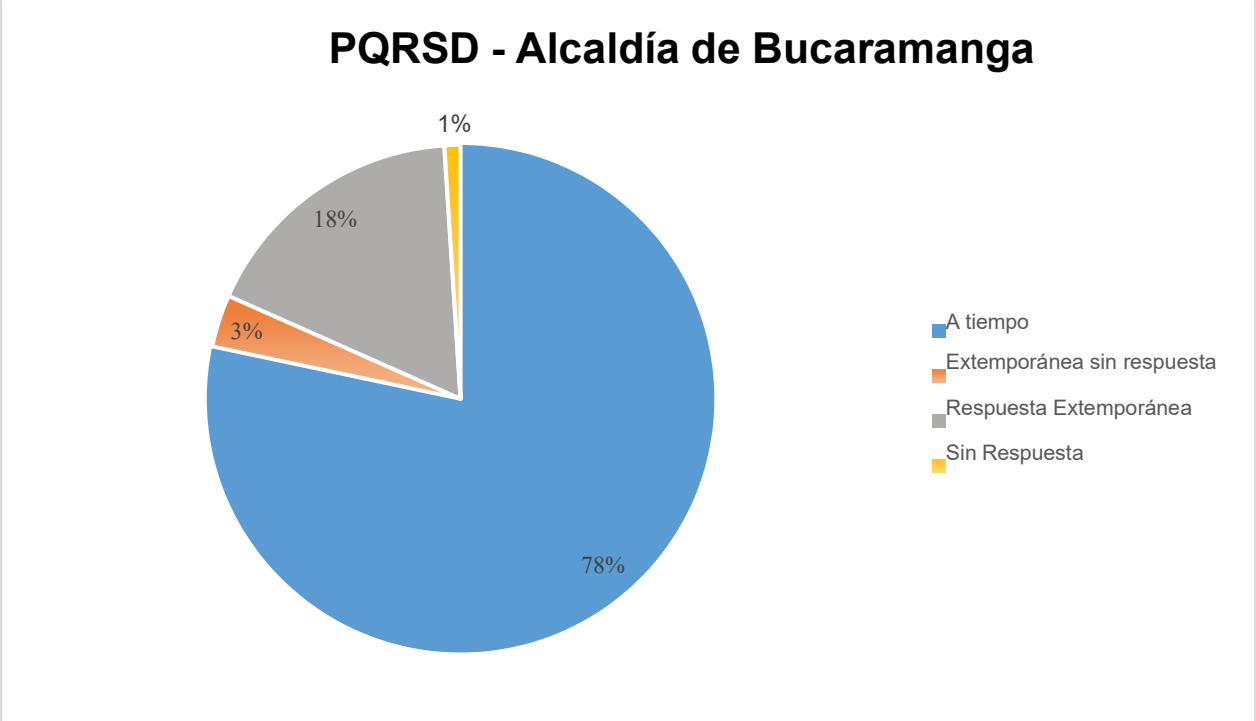
Tabla 3. ***Conclusión de los datos***

PQRSD	**Total**
A tiempo	561.569
Extemporánea sin respuesta	23.894
Respuesta Extemporánea	124.354
Sin Respuesta	**7.312**

Fuente: Elaboración Propia

De acuerdo con la información presentada en la tabla anterior, se establece que el 78.3% de las PQRSD, son contestadas a tiempo, el 10% se encuentran sin respuesta, con respuesta extemporánea el 17.3% y PQRSD extemporánea sin respuesta el 3.3%, como se presenta a continuación.

Ilustración 1. ***Diagramación de los datos.***



Fuente: Elaboración Propia

Con respecto a la anterior información, se puede concluir que cuando una PQRSD se designa mal, se generan reprocesos, porque esta se devuelve y debe ser nuevamente reasignada, haciendo que se disminuyan los tiempos para su contestación, generando incumplimientos, y en el peor de los casos, a quese pierda administrativamente, por no contar con el personal responsable de la misma por temas de rotación del talento humano.

1.2 FORMULACIÓN DEL PROBLEMA

Para el presente proyecto se planteó la siguiente formulación:

¿Cómo las técnicas y modelos de analítica de datos implementadas en una herramienta prototipo puedentener un impacto efectivo en la mejora de los procesos de designación de PQRSD para la Alcaldía de Bucaramanga?

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Desarrollar un prototipo de herramienta utilizando técnicas y modelos de analítica de datos, para el soporte de los procesos de designación de PQRSD al interior de las dependencias de la alcaldía de Bucaramanga.

2.2 OBJETIVOS ESPECÍFICOS

2.2.1 Realizar un reporte de análisis exploratorio de la Base de Datos del Sistema PQRSD de la alcaldía de Bucaramanga, para la comprensión preliminar de la estructura, calidad y la lógica de negocio de la situación problema

2.2.2 Definir técnicas y modelos para el análisis de los datos, que permitan dar respuesta a las necesidades del proyecto con un buen desempeño para los procesos de designación de las PQRSD de la Alcaldía de Bucaramanga.

2.2.3 Desarrollar las visualizaciones e interfaces de la herramienta que apoyen la designación de las PQRSD a través de métricas e indicadores relativos a la dependencia probable que los modelos y técnicas de analítica sugieren a partir del entrenamiento de los datos.

2.2.4 Evaluar el desempeño de la herramienta prototipo en entornos controlados de pruebas y preproducción con usuarios internos validando el impacto en el proceso de designación de PQRSD para la Alcaldía de Bucaramanga.

3. MARCO TEÓRICO Y ANTECEDENTES

3.1 MARCO TEÓRICO

La Constitución Política de Colombia, establece como fines esenciales del Estado, el de servir a la comunidad, promover la prosperidad, garantizar la efectividad de los principios, derechos y deberes consagrados en la carta magna, además de las necesidades demandadas por las diferentes ciudadanías. En este sentido es un deber y prioridad para las instituciones públicas organizarse para dar cumplimiento a los derechos fundamentales y velar por entregar servicios de calidad a la ciudadanía, haciéndose necesaria la búsqueda constante de estrategias, alternativas y métodos, q permitan el mejoramiento en la calidad de atención y satisfacción de los usuarios de una forma digna, efectiva y moderna, en torno a los trámites y servicios que las mismas ofrecen, por cuanto los ciudadanos se convierten en la razón de ser de la Administración Pública. La alcaldía de Bucaramanga, a través de su página https://pqr.bucaramanga.gov.co/, permite a los usuarios y ciudadanos a presentar sus peticiones, quejas y reclamos, como se enuncia en la figura:

Ilustración 2. ***Interfaz de la página web de la alcaldía de Bucaramanga***

Fuente: Interfaz https://pqr.bucaramanga.gov.co/

Teniendo en cuenta que la profundización planteada se basa en los principios de calidad, es importante realizar una reseña de las filosofías y planteamientos de los

precursores de la calidad. A partir de los planteamientos y aportes de los autores que se mencionan a continuación, surge lo que se llamó el movimiento moderno de la calidad, que involucra necesariamente los conceptos de productividad y competitividad [2].

En este contexto, la calidad no puede ser un concepto propio de las organizaciones privadas, en consecuencia, la Administración Pública debe garantizar óptimos niveles de calidad frente a los trámites y servicios que ofrece a la ciudadanía, razón por la cual el Gobierno Nacional a través de la Ley 872 de 2003 creó el Sistema de Gestión de la Calidad en la Rama Ejecutiva del Poder Público y en otras entidades prestadoras de servicios. Así mismo, la Administración municipal adopta la Norma.

De la misma manera, la ley 1712 de 2014 determina que el acceso a la información sólo podrá ser restringido excepcionalmente. Las excepciones serán limitadas y proporcionales, deberán estar contempladas en la ley o en la Constitución y ser acordes con los principios de una sociedad democrática.

Algunas de estas excepciones son:

a. Información pública clasificada cuyo acceso pudiere causar daño al derecho a la intimidad, a la vida, la salud o la seguridad, o los secretos comerciales, industriales y profesionales.
b. Información pública reservada relacionada con la defensa y la seguridad nacional, la seguridad pública, las relaciones internacionales, los derechos de la infancia y la adolescencia, la estabilidad macroeconómica y financiera del país, la salud pública.

Con el fin de regular el derecho de petición en el país, fue expedida la ley 1755 de 2015, la cual dispone la determinación del objeto de la petición y los plazos para su respuesta, la presentación y contenido de las peticiones, la información reservada, el trámite del derecho de petición ante entidades privadas, la atención prioritaria a peticiones de reconocimiento de un derecho fundamental. Así mismo, en su artículo 22

establece: “Las autoridades reglamentarán la tramitación interna de las peticiones que le corresponda resolver, y la manera de atender las quejas para garantizar el buen funcionamiento de los servicios a su cargo”.

3.2 ANTECEDENTES

En el documento de informe de práctica: Diseño e implementación de mejora en la gestión de peticiones, quejas, reclamos, sugerencias y denuncias, en la Dirección de Bienes y Seguros de la Gobernación de Antioquia, por Gómez. Claudia, de la Universidad de Antioquia en el año 2021 [3], se plantea que, el Gobierno colombiano ha establecido lineamientos precisos en cuanto a la garantía del derecho de petición, como un derecho fundamental que afianza la participación ciudadana y constituye un importante elemento de transparencia en la actividad de las entidades públicas. Por su parte, la Gobernación de Antioquia, adoptó la Política Pública de Atención a la Ciudadanía, con el propósito de contribuir al fortalecimiento de las relaciones de confianza entre la ciudadanía y las entidades de la administración departamental. Dentro de dicha política se contempla la gestión del servicio al ciudadano, por medio de la atención de trámites y peticiones, quejas, reclamos, sugerencias y denuncias (PQRSD), como una actividad integral y de carácter obligatorio.

En cumplimiento de la normatividad vigente relacionada con la atención del derecho de petición, la Dirección de Bienes y Seguros, dependencia adscrita a la Secretaría de Suministros y Servicios de la Gobernación de Antioquia, debe dar respuesta a las PQRSD interpuestas por la ciudadanía y que sean de su competencia, las cuales se reciben a través del sistema de gestión documental de la entidad, denominado Mercurio. El trámite de respuesta debe realizarse dentro del término legal establecido, brindando al peticionario información clara, completa, oportuna y congruente con lo solicitado. El proceso para este trámite al interior de la Dirección, comprende varias etapas:1) Ingreso de la PQRSD radicada, 2) Disposición del documento para conocimiento del Director, 3) Instrucción de respuesta o delegación del documento para ser gestionado por un funcionario competente, 4) Consecución de la información

requerida y proyección de oficio de respuesta, 5) Envío de oficio proyectado para revisión del Director, 6) Radicación y firma de oficio de respuesta, 7) Envío de respuesta al peticionario.

A través del estudio se pretendía mejorar la gestión de PQRSD, controlando las situaciones que se viene presentando, mediante un sistema de medición que permita realizar un monitoreo permanente del proceso, de manera que se pueda observar su comportamiento, identificar oportunamente retrasos o irregularidades y aplicar los correctivos que sean necesarios.

Este trabajo aporta perspectivas y procesos relevantes para el proyecto actual, como, por ejemplo, su gestión de la información a través de un proceso bien demarcado que consta de varias etapas mediante las cuales se extraen los datos más relevantes del PQRSD, para posteriormente enviar sus propuestas al director para que estas sean aprobadas, de igual manera un aspecto importante que puede ser tomado en cuenta para el proyecto es el proceso de seguimiento que se plantea en el trabajo, ya que, este tiene como fin mantener un monitoreo permanente, identificando retrasos o irregularidades para que estos puedan ser solucionados.

En el documento de Tesis: Clasificación de la Prioridad de Atención a Reclamos Presentados por Clientes Utilizando Machine Learning", por Aliaga. Jimena y Chahuara Lesly, de la Universidad Peruana Unión, en diciembre 2019 [4], se menciona como el Machine Learning, en la última década ha marcado un rápido y significativo crecimiento del mercado global de automatización de almacenes de datos. El mayor desafío radica en la identificación y manejo de las variables. Se requiere un conjunto de datos grande para reducir el sobreajuste y aumentar la precisión de la validación. Cuando se alcanzan suficientes precisiones de clasificación, se pueden implementar estrategias de selección inteligente para manejar la data de manera eficiente. ¿Por qué agregar aprendizaje automático a la mezcla? Con los modelos de aprendizaje automático apropiados, las organizaciones tienen la capacidad de predecir continuamente cambios en el negocio para que son más capaces de predecir qué sigue. Como los datos son constantemente

agregado, los modelos de aprendizaje automático aseguran que la solución sea constantemente actualizada.

Un aporte relevante de este proyecto es principalmente su enfoque en la automatización de datos, aspecto importante en el proyecto actual, siendo así que sus aportes respecto al “Machine Learning” son bastante relevantes, al igual que su búsqueda por solucionar algunos fallos de la misma, como lo es el manejo de variables, que en este caso es apoyado por la selección inteligente, basada principalmente en el aprendizaje automático, con el que se busca mejorar la eficiencia del procesamiento de datos, debido a que estos serán organizados en base a aquellos que son más frecuentes.

En el documento tesis de grado Diseño de una metodología analítica para el tratamiento de quejas y reclamos para el sector hogares y movilidad ETB, por Jeimmy Espine. Jeimmy y Amórtegui. Rodrigo, de la Universidad de los Andes en el 2019 [5], se plantea una metodología basada en analítica de datos, que le permita a la empresa procesar de una manera más eficiente los reclamos de los clientes del segmento Hogares y Personas, así mismo realizar la clasificación de los reclamos de manera estructurada y apoyada en herramientas de analítica de datos, con el fin de identificar las causas de estas para que sean enrutadas a las áreas que deben dar solución.

A pesar de la principal diferencia entre los proyectos que radica en los fines de estos, se pueden tomar de apoyo de la clasificación de los reclamos de manera estructurada que es mencionada en el documento, de igual manera el apoyo en herramientas de analítica de datos, con el fin de que las quejas sean enrutadas a las áreas que deben dar solución, así liberando la carga de trabajo de una clasificación manual.

En la tesis, Diseño del módulo de PQRS para la recepción de peticiones, quejas, reclamos y sugerencias del portal empleo de la Universidad Católica de Colombia implementando Business Intelligence para optimizar acciones correctivas, por Mora. Jonatan, de la Universidad Católica de Colombia en 2019 [6], se referencia a los sistemas

PQRS, como una de las formas en que una institución puede recibir información sobre las actividades, productos o servicios y con ello medir el nivel de satisfacción que tienen sus clientes. Sirve como herramienta gerencial para tener control y mejoramiento continuo de los productos y/o servicios ofrecidos, permite obtener información de lo que sucede, quejas, sugerencias y felicitaciones que tienen los usuarios relacionados con el cumplimiento y objetivos de la institución. En la actualidad los sistemas que utilizan Business Intelligence son los más usados ya que debido a la capacidad que tienen para proporcionar acceso a los datos y en tiempo real, se convierte en un gran aliado para la gerencia ya que pueden llevar a cabo análisis de datos actuales, estados y rendimientos para que sea más fácil tomar alguna decisión. Es por esta razón que se utilizaran herramientas de análisis propias del BI. El implementar un módulo para la gestión de solicitudes que utilice Business Intelligence demostrara cuales pueden ser las razones del poco uso del portal de empleo de la Universidad.

La implementación de un modelo automatizado de PQRS, busca que la Universidad Católica de Colombia desarrolle e implemente este módulo dentro de su portal web de empleo para el año 2022 mejorando el nivel de atención de los usuarios y encaminando al usuario a obtener respuestas eficaces y poder hacer seguimiento de cada solicitud. Con el desarrollo de este módulo se busca identificar inconvenientes que tengan los estudiantes con el portal.

En este caso es necesario el tener en cuenta la posibilidad de la implementación de visualizaciones con enfoque de Business Intelligence en el proyecto actual, debido al aporte en el proceso de toma de decisiones, y como esto es de utilidad para cumplir los objetivos del proyecto para la designación de PQRSD en la administración municipal.

En la tesis, Transformación digital para el mejoramiento de los servicios al ciudadano beneficiarios de la formalización de tierras en Colombia por Rodríguez Néider de la Universidad Libre en 2018 [7], se realizó una investigación con el objetivo de generar una plataforma de transformación digital que permitiera realizar un análisis de datos de los sentimientos de los ciudadanos, proporcionando distintas percepciones, al

igual que información respecto a requerimientos y necesidades del grupo de interés con el fin de poder generar mejoras oportunas en la oferta al ciudadano. Transformando los territorios a la legalidad y productividad y dotando de integralidad la política de tierras sostenible.

Esta investigación fue impulsada por las diversas dificultades a las que se ha enfrentado el proceso de formalización de tierras en los últimos años, debido principalmente a que es un trámite realizado por medio de la Agencia Nacional de Tierras, la cual según cifras recibe alrededor de 25.000 consultas con el fin de resolver conflictos relacionados con la tierra lo cual equivale a un promedio de 66 diarias, lo cual combinado a las largas series de tramites que deben ser realizados gracias a una gran cantidad de entes como lo son: el Ministerio de Justicia y Derecho, la Superintendencia de Notariado y Registro, el Departamento Nacional de Planeación y el Instituto Geográfico Agustín Codazzi, entre otros. Dan lugar a la necesidad de racionalizar tanto la información general de las consultas como los trámites, para erradicar aquellas falencias generadas por la saturación de datos.

La revisión de la anterior investigación permite el entendimiento de los insumos necesarios que puedan requerirse para análisis de este tipo dentro del proceso de gestión de PQRSD, con el fin de aportar a futuras implementaciones que puedan abordar este tipo de alcance que pueda ser importante para la identificación de problemáticas al interior de la información descrita en las solicitudes peticiones reclamos y sugerencias que llegan de los ciudadanos bumangueses.

El documento, Análisis etiquetado de textos para predicción de la polaridad, enfóquese mi supervisado y etiquetado automático, por María Luque y Luis Cortes, de la Universidad Pontificia Javeriana 2020[8], se enfoca en agregar valor a las soluciones ofrecidas a empresas colombianas a problemas de analítica relacionados con el etiquetado de la data para procesamiento de textos.

Este análisis se basa en la estrategia y definición de la Alianza CAOBA, como el centro de excelencia y apropiación que apoya el uso de las tecnologías de Big Data y Data Analytics, a través de diferentes frentes que incluyen la formación del talento humano, la investigación aplicada y el desarrollo de productos a la medida. Este centro está constituido por las empresas Grupo Bancolombia, Grupo Nutresa, IBM de Colombia, SAS Institute Colombia, DELL, Cluster CREATIC y las Universidades ICESI, EAFIT, los Andes y la Pontificia Universidad Javeriana, que actúa como ejecutor del proyecto. La potencia de esta alianza le permite interactuar en diferentes campos de análisis, tales como la investigación aplicada, la formación, el apoyo al emprendimiento y la consultoría.

En este documento, se presenta como estrategia, la solución Data Label Server que ofrecería CAOBA, cuyo enfoque principal es hacer más eficientes los procesos relacionados con el procesamiento de data, incluido el etiquetado de bases de datos. Para CAOBA, este servicio podría presentarse dentro del flujo actual para modelado de NLP, junto con un módulo de Modelación y otro de Visualización. El módulo de Data Label Server permitiría cobrar por el etiquetado manual de una porción o de la totalidad de la base de datos que es entregada por el cliente. La decisión del cliente y de CAOBA de cuanto es necesario etiquetar, no solo radicará en su costo, sino también apoyada en un desempeño estimado del modelo con ciertos niveles de etiquetado.

El conjunto de datos proporcionado por CAOBA pertenece al Servicio de Atención al Cliente en una entidad financiera. La información corresponde a las conversaciones entre los clientes y el personal de la entidad encargado de atender las solicitudes e inquietudes de clientes y usuarios. Por medio de este canal los clientes se ponen en contacto con el área de servicio para que les resuelvan algún problema específico, y las múltiples solicitudes y requerimientos como proporcionar información de productos y asesorías, sobre los diferentes servicios y canales de atención que la entidad tiene disponible. Por medio de la innovación en las técnicas de procesamiento del lenguaje natural se tiene acceso a una mejor comprensión de este tipo de datos junto con los modelos de conversación para el estudio de las prácticas modernas de atención al cliente y análisis de polaridad.

4. METODOLOGÍA

El desarrollo del presente proyecto, se basó en la metodología Team Data Science Process (TDSP) de Microsoft. Este enfoque, ágil e iterativo, se orienta hacia la entrega eficiente de soluciones de análisis predictivo y aplicaciones inteligentes. Si bien es adaptable a procesos de análisis exploratorio o improvisados, la TDSP comparte similitudes con la metodología CRISP-DM de la Unión Europea, e incorpora elementos de metodologías ágiles como Scrum para facilitar el desarrollo iterativo e incremental de soluciones.

En el proyecto, se incorporaron prácticas recurrentes y habilitantes en el contexto de la metodología y de la realidad de la información como, por ejemplo, las limitantes por la calidad de la información, el tiempo y restricciones de recursos para el desarrollo del proyecto. Sin embargo, los resultados y actividades realizadas se describen a continuación:

a. **Business Understanding:** Esta fase implica identificar el problema y los objetivos del proyecto. En el proyecto, esta fase se centró en identificar el problema de la designación incorrecta de las PQRSD y establecer el objetivo de desarrollar un modelo que mejore este proceso en la Alcaldía de Bucaramanga.
b. **Data Ingest & Understanding:** Durante esta etapa, los datos se recolectan, exploran y limpian para su posterior análisis. En el desarrollo del proyecto, se realizó la extracción de los datos de la base de PQRSD de la Alcaldía, su limpieza y preprocesamiento, y el análisis exploratorio de los datos para entender sus características y calidad.
c. **Modeling:** Esta fase consiste en el desarrollo de modelos utilizando técnicas de ciencia de datos. En nuestro caso, se aplicaron algoritmos, la ingeniería de características, la validación de modelos y la optimización de hiperparámetros a

través del uso de GridSearchCV, la implementación de una arquitectura de clasificación mixta entre los diferentes componentes y sus características.

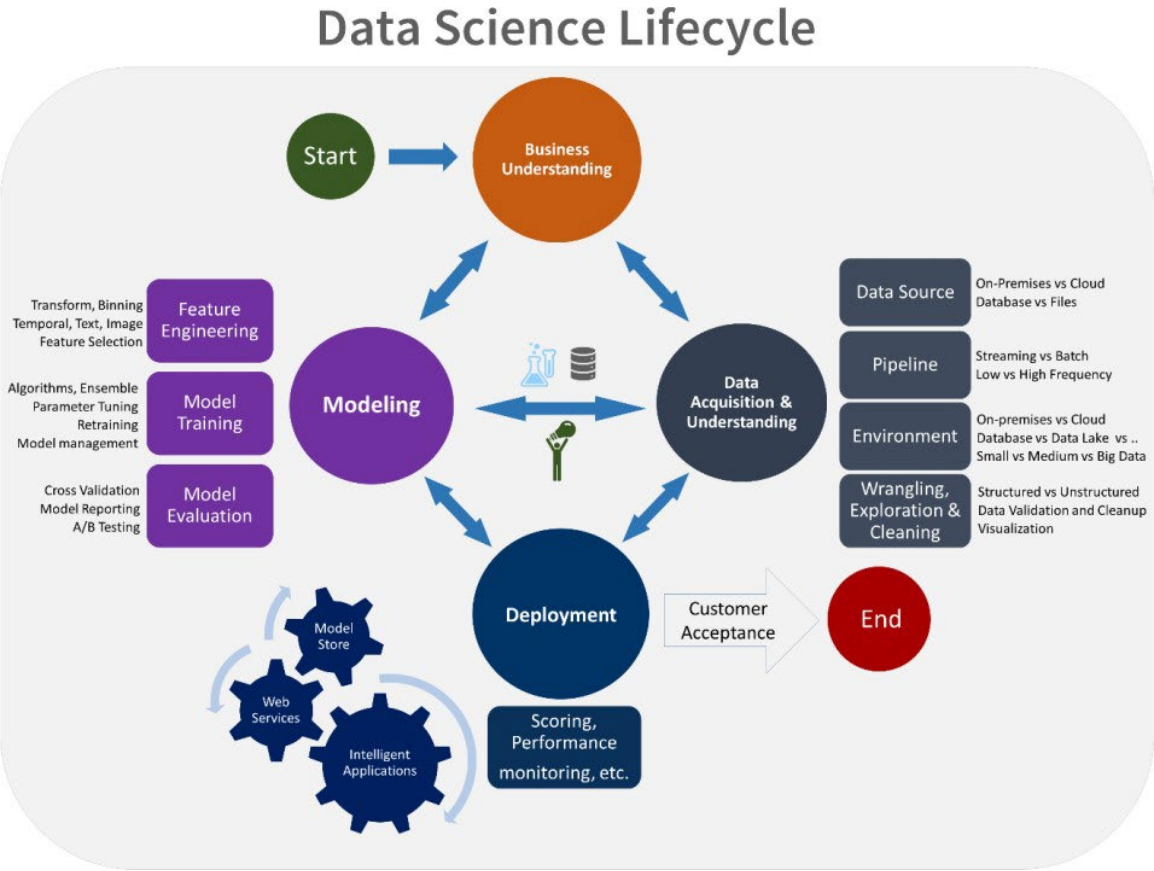
d. **Deployment:** Una vez construido el modelo, se despliega para que pueda ser utilizado. Aquí, se llevó a cabo una prueba con 1.000 nuevos registros para medir los resultados obtenidos y confirmar la viabilidad de la implementación del modelo en la designación de PQRSD.

e. **Customer Acceptance:** Esta etapa se trata de obtener retroalimentación de los usuarios finales e iterar sobre el modelo según sea necesario. En el desarrollo del ejercicio con usuarios de la Alcaldía de Bucaramanga, se determinaron aspectos clave de relación en los procesos de asignación equiparables con los definidos por nuestro modelo y ejercicios de asignación manual, que es como actualmente se realizan.

Estas fases forman parte del ciclo iterativo de la TDSP, que puede repetirse a medida que se refinan los modelos y se adapta la solución para satisfacer mejor las necesidades del problema.

Estos componentes se adaptaron y utilizaron respectivamente, como referencia principal para el desarrollo de nuestro proyecto, como se verá en las etapas subsiguientes. A continuación, se presenta una figura que muestra la adaptación del Ciclo de vida del dato correspondiente a nuestro proyecto.

Ilustración 3. ***Ciclo de vida de ciencia de datos, Microsoft 2020***



Fuente: *Ciclo de vida de ciencia de datos, Microsoft 2020*

4.1 ENTENDIMIENTO DEL NEGOCIO

Durante esta etapa, se revisó documentación asociada a los informes de PQRSD que realiza la secretaria Administrativa reportando el estado de la información en periodos trimestrales, donde se observaban las dinámicas de asignación y estadísticos del proceso de gestión de estas solicitudes.

Se examinó el sistema de Información para la gestión de PQRSD y se realizaron inspecciones visuales a su Base de Datos para determinar la estructuración de la información y la calidad de la misma, se solicitó un proceso de entendimiento al equipo

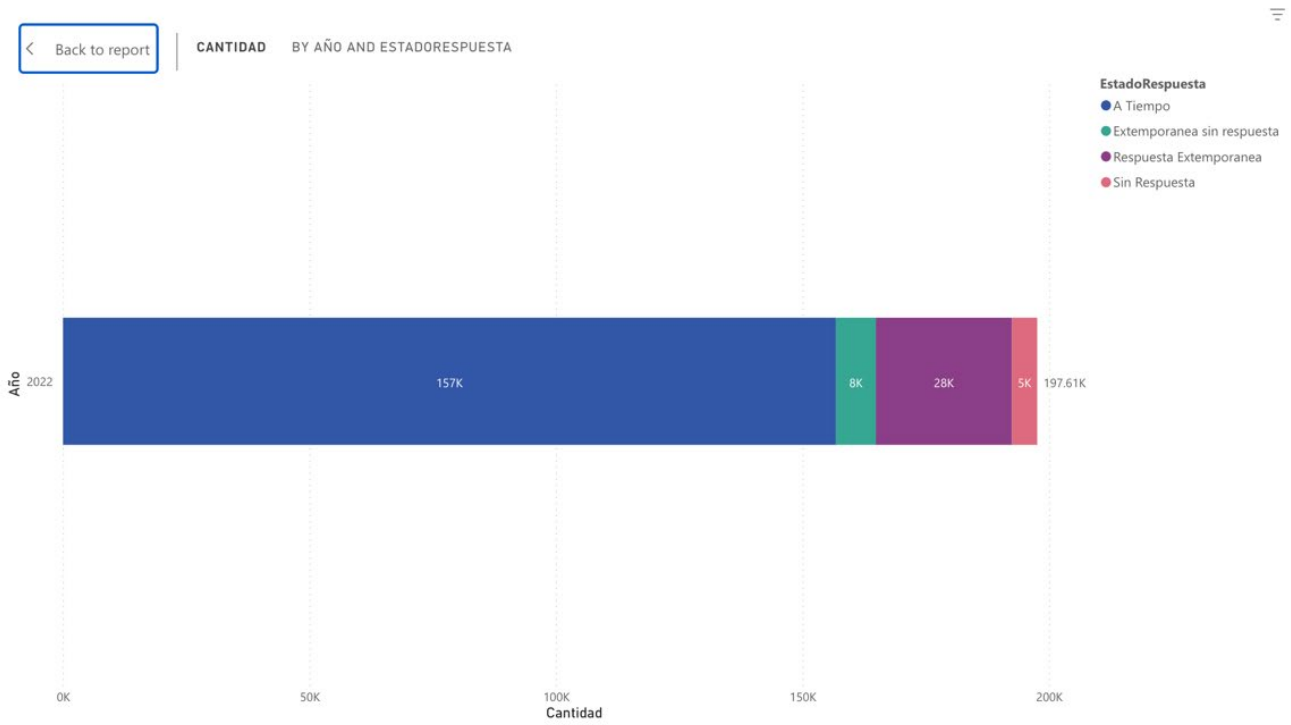
de desarrollo responsable de la administración y soporte del mismo, con el fin de detallar la lógica de negocio que tiene este proceso en la entidad

De igual manera, para el entendimiento fue suministrada una herramienta de Inteligencia de Negocio que tiene la entidad desarrollada desde la Oficina Asesora TIC, encargada de la operatividad de la plataforma GSC para llevar control y seguimiento a los procesos de asignación y respuesta.

Reporte Estado, Tipo y Dependencia – Alcaldía de Bucaramanga

Este ejercicio permitió observar de manera clara los comportamientos que la asignación de PQRSD debe resolver, como, por ejemplo, las respuestas extemporáneas y las solicitudes sin respuesta con volúmenes superiores a las treinta mil solicitudes, tal como se observa en la siguiente captura de pantalla del tablero de control.[11]

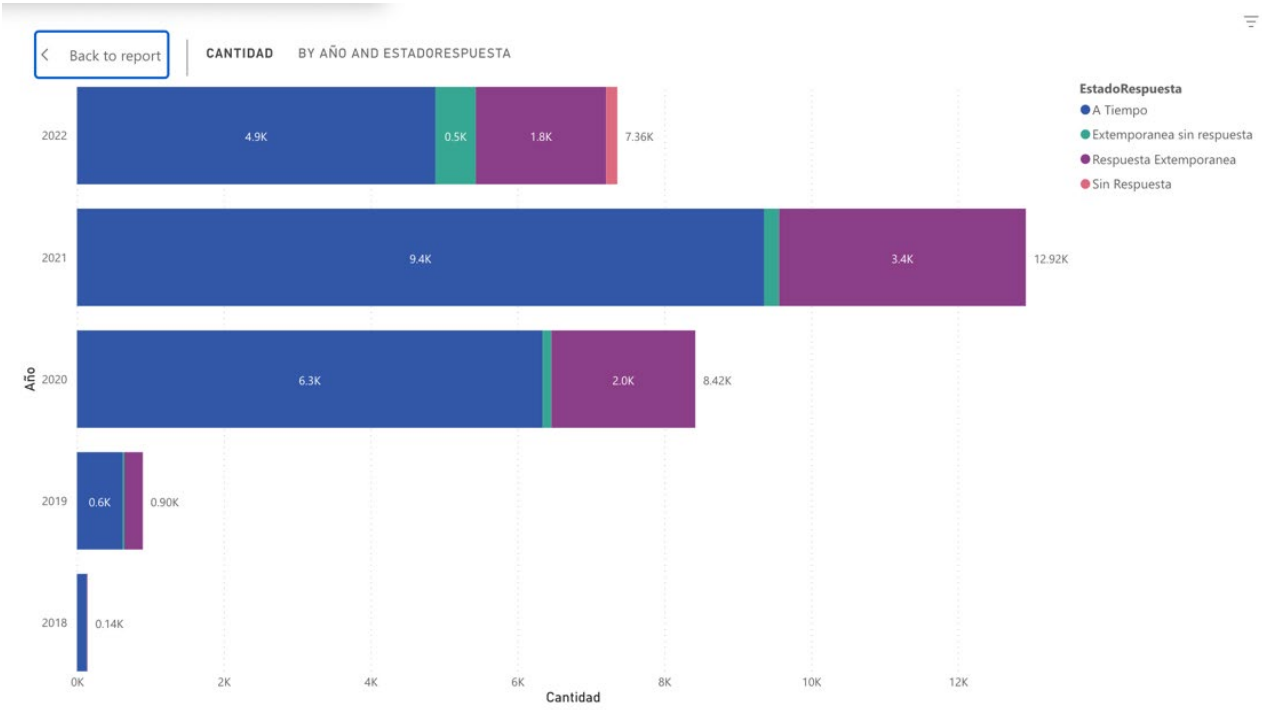
Ilustración 4. ***Tablero de Solicitudes Asignadas***



Fuente: Alcaldía de Bucaramanga

De igual manera, se identificó una gran cantidad de respuestas que fueron prorrogadas, teniendo en consideración que las condiciones para que se pueda realizar dicha acción están sometidas a condiciones muy específicas que deben tener una justificación detallada para asegurar el cumplimiento normativo. Lo anterior igual evidencia un comportamiento a considerar pues el volumen de respuestas prorrogadas corresponde a cerca de 34 mil solicitudes como se observa en la gráfica a continuación.

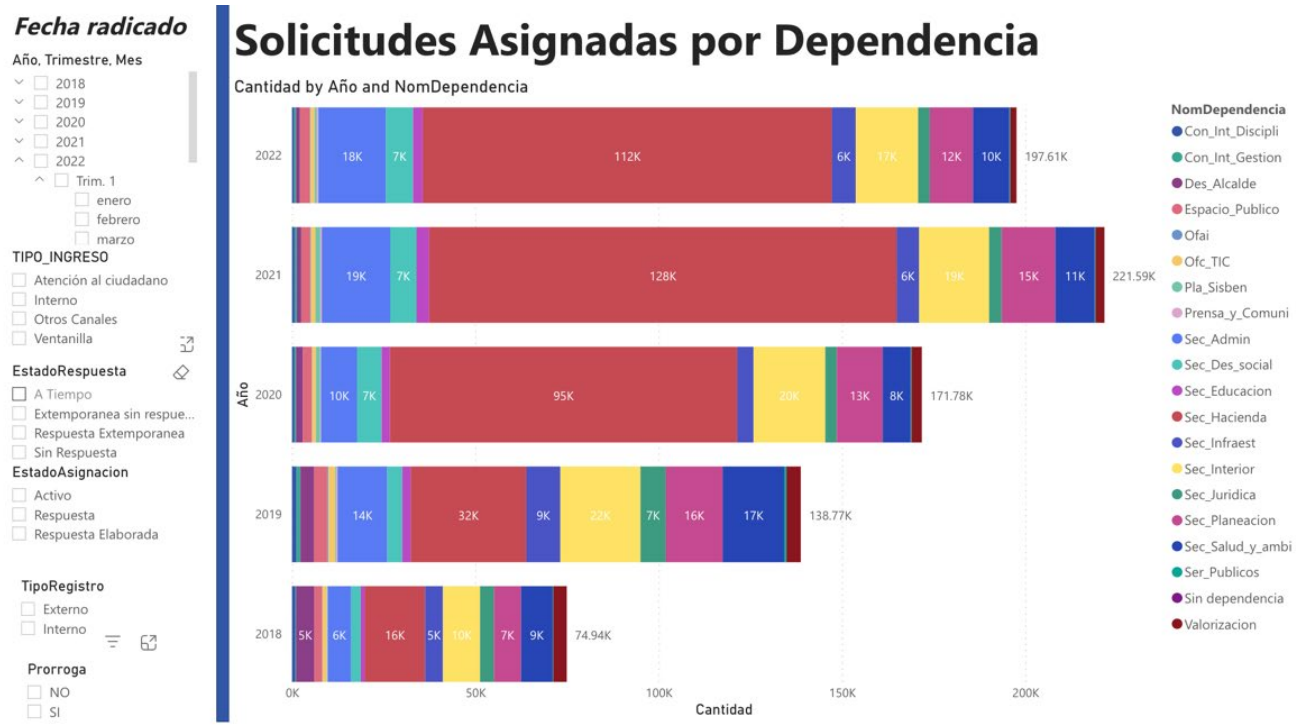
Ilustración 5. ***Tablero de Respuestas a Solicitudes Asignadas***



Fuente: Alcaldía de Bucaramanga

Finalmente, para el entendimiento de las dinámicas propias de las asignaciones, el sistema cuenta con multiples categorías para las designaciones, correspondientes a las dependencias como secretarías u oficinas que tienen atención a la ciudadanía. Los comportamientos de distribución de las designaciones a estas áreas se pueden observar en la gráfica a continuación:

Ilustración 6. ***Tablero de Respuestas a Solicitudes Asignadas***



Fuente: Alcaldía de Bucaramanga

4.2 ADQUISICIÓN Y ENTENDIMIENTO DE LOS DATOS

A partir de las verificaciones y las visualizaciones relacionadas en el punto anterior, el equipo obtuvo una serie de datos preliminar de más de 590.000 solicitudes registradas desde el año 2018 hasta el 2022, que permitiera explorar los datos de manera directa. De esta manera, se realizó un análisis exploratorio de la información a través de un reporte como cumplimiento del primer objetivo específico en el que se observan diferentes comportamientos como, por ejemplo, la distribución entre las categorías

definidas como lo son las dependencias y su significativo desbalance entre la categoría "Sec_Hacienda" que hace referencia a la Secretaría de Hacienda de la Alcaldía, frente a las otras dependencias.

Este hallazgo, responde a la dinámica de solicitudes y requerimientos asociados a los estatutos tributarios, recaudos y presentaciones de declaraciones de impuestos como industria y comercio, entre otros. Particularmente, frente al recaudo del impuesto predial, al año la Alcaldía de Bucaramanga recauda valores por más de 170 mil predios, así como, lo referente al impuesto de Industria y Comercio y Retención de Industria y Comercio, superan transacciones de recaudo por más de 50 mil empresas y establecimientos en la ciudad. Lo anterior, podemos tomar como referencia que desde el año 2018 de operación del sistema GSC, los ciudadanos han creado más 300 mil solicitudes a esta dependencia, solicitando desde recibos de pago, validación de los pagos y solicitudes asociadas a las declaraciones y estados tributarios. Esta condición en los capítulos posteriores tendrá un tratamiento particular para evitar potenciales sesgos para el prototipo de designación elaborado en el presente trabajo.

Ahora bien, desde esa base de datos preliminar, se obtuvieron los siguientes resultados.

Tabla 4. *Resultado de exploración del set de datos por dependencia*

Dependencia	Registros
Sec_Hacienda	344258
Sec_Interior	73104
Sec_Admin	60048
Sec_Planeacion	55707
Sec_Salud_y_ambi	46817
Sec_Infraest	27570
Sec_Des_social	23788
Sec_Juridica	19177

Dependencia	Registros
Valorizacion	13617
Espacio_Publico	12017
Des_Alcalde	11857
Sec_Educacion	10007
Ofc_TIC	5817
Con_Int_Discipli	3880
Pla_Sisben	3435
Con_Int_Gestion	2531
Ser_Publicos	1493
Prensa_y_Comuni	1263
Ofai	740
Sin dependencia	3

Fuente: Elaboración Propia

Esta base, en términos de calidad se observó que había información ausente por procesos de mal diligenciamiento de las solicitudes y ausencias de procesos de validación en las primeras etapas de desarrollo del sistema de PQRSD, lo que también supone un reto para la selección de los atributos que aportan al modelo que se desarrolló y que se mencionan en etapas posteriores frente a las técnicas y modelos seleccionados. A continuación, se presenta un listado de atributos con sus respectivos datos asociados a los valores faltantes, en los que se puede observar ausencias superiores al 80% de los datos totales del set de datos, en cuanto a variables como barrio, oficina, cargo. Así mismo, se evidencia baja o nula presencia de variables asociadas a identificadores de personas o categorías asociadas a las dependencias y/o funcionarios.

Tabla 5. *Consolidado exploración valores nulos en base de datos original*

Campo	Datos nulos	Fuente
tip_persona	0	Entidad
tip_doc	294283	Usuario

Campo	Datos nulos	Fuente
doc_identi	171115	Usuario
asunto	23294	Usuario
comentarios	166687	Usuario
medio_radicado	3	Usuario
numero_pqr	9	Entidad
dependencia	0	Usuario
funcionario	0	Entidad
observ_rechazo_dep	604615	Entidad
observ_resp	407995	Entidad
tipo_solicitud	0	Entidad
sub_tiposolicitud	0	Entidad
observ_asignacion	414235	Entidad
anexos	492642	Usuario
num_folios	8058	Entidad
tip_usu_reg	70869	Entidad
tipo_ingreso	0	Entidad
prioridad	172794	Entidad
ofic_cargo	604677	Entidad
destino	16319	Entidad
funasignado	281657	Entidad
tsolrecla	181182	Entidad
temas	341212	Entidad
estrespuesta	15485	Entidad
solbarrio	538785	Usuario

Fuente: Elaboración Propia

4.2.1 EXPLORACIÓN Y LIMPIEZA DE DATOS

Con base en lo anterior, el equipo solicitó apoyo a la Alcaldía de Bucaramanga, y se toma decisión de realizar una segmentación de la base de datos para iniciar el proceso

de exploración y limpieza, basados en los campos que para el ejercicio tienen un impacto real al objetivo del proyecto y permiten llegar al núcleo de análisis de los textos y la aplicación de modelos que permitieran impactar el proceso de designación. En ese sentido, se obtuvo una base de datos con 590.592 registros llamada "MuestraBD600KPQRSD.csv", con los campos "ID", "Asunto", "Observacion" y "NombreDependencia" esta última, a la cuál fue asignada por el funcionario de la entidad.

*Tabla 4.**Estructura Dataset definitivo***

ID	ASUNTO	OBSERVACION	NombreDepedencia
1	NaN	insatisfaccion por el pan no recibido	Sec_Admin
2	NaN	Queja por contaminación visual en la calle 45 con carrera 13 esquina	Sec_Admin
3	NaN	Prueba PQRS 05/10/2011	Sec_Admin
4	NaN	en la cra 5 N 57 - 59 barrio porvenir hay 2 luminarias fundidas solicita una pronta solución	Sec_Infraest
6	NaN	Esto una solicitud	Sec_Admin
7	NaN	Esto es un reclamo	Sec_Admin
11	NaN	esta es para informar a servicio al cliente que el sistema de PQR se encuentra en producción desde el día de hoy Por favor tenerlo presente para su uso	Sec_Admin
26	NaN	Deseo saber como puedo hacer parte del programa mujeres cabeza de familia busque aqui en la pagina y no	Sec_Des_social

ID	ASUNTO	OBSERVACION	NombreDepedencia
		encontreGracias de antemano	
37	NaN	Prueba de Queja presentada	Sec_Admin
54	NaN	Buenos dias En la cra 35 a con cll 48 la segunda luminaria del anden occidente esta fuera de servicio y esto se presta para que los ladrones hagan de las suyas Esto esta sucediendo hace 2 meses ...	Sec_Infraest
57	NaN	SEÑORES ALCALDIA SOLICITO INFORMACION DE LO ESTIPULADO POR LA COMISION PARA LOS NOMBRADOS PROVISIONALMENTE Y DE ACUERDO A ESTO RATIFIQUEN Y DEN USO DE LA LISTA DE ELEGIBLES DE LA OPEC 3534 DE LA S...	Sec_Educacion

Fuente: Elaboración Propia

Ahora bien, estos campos seleccionados reflejan la baja calidad de la información con la que el sistema está realizando la ingesta de datos por cada registro, una clara deficiencia del sistema, pero una realidad para el análisis de los datos que aumenta la necesidad de explorarlos y definir los esquemas y arquitectura definitivos para la aplicación de los clasificadores y futuros resultados. A continuación, presentamos un consolidados de información que evidencia la baja o nula información en los campos Asunto y observación, claves para el entendimiento de cada petición.

Tabla 5. ***Consolidado Campos Nulos Dataset Definitivo***

Campo	Datos nulos
ID	0
Asunto	9603
Observacion	154010
NombreDependencia	0

Fuente: Elaboración Propia

En este ejercicio de exploración y limpieza, se inicia con el preprocesamiento de los datos. Todo lo concerniente a la limpieza, minúsculas, caracteres válidos, inválidos, eliminación de caracteres que no son una letra o número o un espacio. Posterior a ellos, se establece el proceso de eliminación de “Stop Words” aplicándolas a los campos 'ASUNTO' y 'OBSERVACION'.

Ilustración 7. ***Exploración y limpieza de datos aplicada***

Fuente: Elaboración Propia

Posterior a este escenario, se aplica la corrección ortográfica a todas las palabras únicas en el conjunto “unique_words” que se creó anteriormente. Se realiza utilizando la biblioteca “SpellChecker”, que es una biblioteca en Python para verificar la ortografía de palabras en varios idiomas. Así mismo, Se procesa aún más las correcciones de las palabras desconocidas. Las palabras que se encuentran en la lista de "stop words" son reemplazadas por cadenas vacías, y luego se crea un diccionario para mapear palabras desconocidas a sus correcciones. También se define un nuevo tipo de diccionario que devuelve la clave en lugar de un error si la clave no se encuentra en el diccionario. Con esto, se corrige la ortografía de las palabras en las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame y se combina las columnas 'ASUNTO' y 'OBSERVACION' en una nueva columna llamada 'INPUT'. Al final del proceso, se realiza la creción de un DataFrame nuevo llamado ‘Data_corregido.csv’.

Tabla 6. **Datos PreProcesados ‘Data_corregido.csv’**

ASUNTO	OBSERVACION	NOMBREDEPENDENCIA	INPUT
avaluo catastral	buenos dias pienso avaluo catastral ustedes manifiestan dicen universidad adelanto avaluo catastral predios mencion anteriormente predio nadie vino avaluo subieron 100 justo actuaron	Sec_Hacienda	avaluo catastral buenos dias pienso avaluo catastral ustedes manifiestan dicen universidad adelanto avaluo catastral predios mencion anteriormente predio nadie vino avaluo subieron 100 justo actuaron conviccion propia obedecer estudio debe ser desconozco si convenio chanchullo siempre sucede pais ahora quisiera ver estudios dicen tener firman predio visitados ustedes muchas gracias

ASUNTO	OBSERVACION	NOMBREDEPENDENCIA	INPUT
	conviccion propia obedecer estudio debe ser desconozco si convenio chanchullo siempre sucede pais ahora quisiera ver estudios dicen tener firman predio visitados ustedes muchas gracias		
exhumacion		Sec_Salud_y_ambi	exhumacion
solicitud capacitacion practicas manufactura		Sec_Salud_y_ambi	solicitud capacitacion practicas manufactura
derecho peticion		Sec_Interior	derecho peticion
recibos impuesto predial		Sec_Hacienda	recibos impuesto predial
solicitud ayuda laminas zinc		Sec_Interior	solicitud ayuda laminas zinc
derecho peticion		Sec_Infraest	derecho peticion
informe ultimo trimestre 2018		Sec_Interior	informe ultimo trimestre 2018
remsion facturas		Sec_Salud_y_ambi	remsion facturas
adicion contrato		Sec_Infraest	adicion contrato
despacho comisorio rad 20180058900 jdo 12 civil mpal		Sec_Interior	despacho comisorio rad 20180058900 jdo 12 civil mpal
solicitud copia proceso		Sec_Interior	solicitud copia proceso
exoneracion impuesto predial lote particular		Espacio_Publico	exoneracion impuesto predial lote particular

ASUNTO OBSERVACION	NOMBREDEPENDENCIA	INPUT
invitacion reunion consejo territorial	Sec_Salud_y_ambi	invitacion reunion consejo territorial
acuerdo pago contrato	Sec_Hacienda	acuerdo pago contrato
rta utsp37303122018 solicitud atencion atender queja presentada despacho peticionaria gloria isabel duran galan	Ser_Publicos	rta utsp37303122018 solicitud atencion atender queja presentada despacho peticionaria gloria isabel duran galan

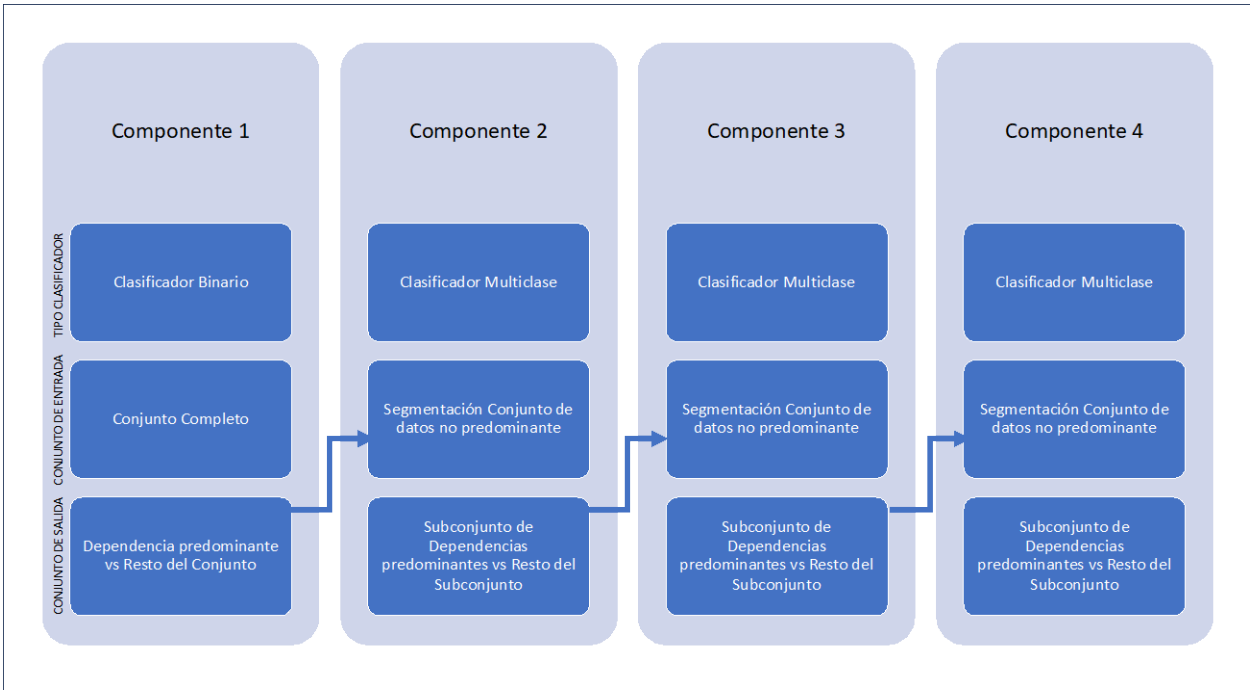
Fuente: Elaboración Propia

4.3 MODELADO

Cómo se mencionó en los apartes anteriores, el proyecto para el cumplimiento de los objetivos planteados realizó el análisis de los datos de la entidad y el entendimiento desde el negocio para entender la realidad estructural y en consecuencia, el desbalanceo que se tiene de la base de datos en especial con todas las solicitudes recibidas para la Secretaría de Hacienda frente a las otras áreas de la Alcaldía de Bucaramanga.

Por lo anterior, se toma la decisión de realizar un diseño que permita la implementación de una arquitectura de clasificación mixta que cierre sesgos y criterios de clasificación erróneos entre las áreas. En consecuencia, el modelo diseñado e implementado, consta de 4 componentes que parten del conjunto completo de los datos y segmenta por cantidad de solicitudes recibidas, los niveles de interacción de los clasificadores.

Ilustración 8. ***Arquitectura de Clasificación Mixta o Híbrida***



Fuente: Elaboración Propia

Para el componente 1, se parte de un clasificador tipo binario que toma un conjunto de entrada para analizar al conjunto completo obtenido de la Alcaldía de Bucaramanga con las 592 mil solicitudes. El conjunto de salida clasifica todas las solicitudes de la dependencia dominante, en este caso “Sec_Hacienda” y reemplaza el valor de las demás áreas como “Otra”.

Para el segundo, tercer y cuarto componente, se realiza una clasificación multiclase, que toma de como conjunto de entrada, el segmento o conjunto no predominante del componente inmediatamente anterior y crea los subconjuntos de dependencias predominantes en ese componente frente al resto de subconjuntos.

Ahora bien, para la implementación de la arquitectura de clasificación mixta o híbrida descrita anteriormente, el equipo inició con cargas de librerías como pandas, numpy, Swifter, pyspellchecker, pandarealle y nltk, entre otras y la importación del dataset ‘Data_corregido.csv’ preprocesado anteriormente con 4 columnas y 590.592 registros, obteniendo este resultado inicial de agrupación por número de solicitudes por dependencia.

Tabla 7. ***Consolidado solicitudes por dependencia***

NombreDependencia	Solicitudes	Porcentaje
Ofc_TIC	1331	**0.23%**
Con_Int_Discipli	1466	**0.25%**
Pla_Sisben	2900	**0.49%**
Extra	3854	**0.65%**
Espacio_Publico	7126	**1.21%**
Sec_Educacion	7238	**1.23%**
Sec_Juridica	10693	**1.81%**
Valorizacion	13238	**2.24%**
Des_Alcalde	15058	**2.55%**
Sec_Infraest	20585	**3.49%**
Sec_Des_social	22173	**3.75%**
Sec_Planeacion	35339	**5.98%**
Sec_Salud_y_ambi	41269	**6.99%**
Sec_Admin	43680	**7.40%**
Sec_Interior	47370	**8.02%**
Sec_Hacienda	317272	**53.72%**

Fuente: Elaboración Propia

También, se toma la decisión de tomar para los análisis y aplicación de los modelos y funcionalidades, las columnas 'INPUT' y 'NombreDependencia' eliminando cualquier fila con valor nulo o 'NaN' que genere error y sesgo al modelo.

4.3.1 MODELOS Y CLASIFICADORES INICIALES

Ya con la base de datos definitiva y depurada para aplicar los modelos, se toma la decisión de realizar un ejercicio de validación del modelo de bolsa de Palabras por Frecuencia, donde se ejecutan las siguientes tareas; creamos un diccionario llamado

`pesos_dict` que tendrá cada nombre único de 'NombreDependencia' como una clave, con un valor inicial de una lista vacía que nos permite definir la longitud del dataset a utilizar y generar un conjunto de datos de entrenamiento aleatorio, `train`, que es el 80% del tamaño del dataframe y el restante es el conjunto `train`. Este ejercicio nos arroja un Accuracy del 58.83% entre las dependencias analizadas.

Por otro lado, se ejecuta el clasificado RandomForest usando TF-IDF en un ejercicio dual con dos valores para el hiperparámetro `n_estimators` el primero 100 y el segundo 200, obteniendo valores 68.43% y 68.66% respectivamente.

También como parte del ejercicio de modelamiento, se obtuvieron Accuracy de 68,4% para el Modelo de SVC con la técnica Kernel RBF y 64,9% con la técnica Kernel Sigmoidal, siendo el primero modelo un score muy competitivo al resto de los modelos sin tener un compromiso de performance importante.

En una siguiente instancia, se obtiene un accuracy de 66,3% (con hiperparámetro `alpha` de 0.8) del Modelo Complement Naive Bayes, un resultado inferior a los clasificadores SVC pero superior a Bolsa de Palabras.

Y finalmente, se aplica el modelo de Regresión Logística, con dos valores para el hiperparámetros `l1_ratio`, el primero de 0.2 y el segundo de 0.5. En este ejercicio, se obtuvo un Accuracy de 68,5% con regresión logística (`l1_ratio` de 0.2), un resultado competitivo con los modelos SVC y RF superior a Bolsa de palabras y Naive Bayes

En la siguiente tabla se consolidan los resultados mencionados anteriormente:

Tabla 8. ***Consolidado Modelos Aplicados Conjunto Completo***

Clasificador	**Hiperparámetro**	**Valor Hiperparámetro**	**Accuracy**

Bolsa de palabras por Frecuencia			0.5883
RandomForest	n_estimators	100	0.6843
RandomForest	n_estimators	200	0.6866
SVC	Kernel	RBF	0.684
SVC	Kernel	Sigmoid	0.649
Multinomial Naive Bayes	Alpha	0.5	0.6529
Multinomial Naive Bayes	Alpha	0.8	0.6468
Complement NB	Alpha	0.5	0.6633
Complement NB	Alpha	0.8	0.6634
Regresión Logística	l1_ratio	0.2	0.6853
Regresión Logística	l1_ratio	0.5	0.6824

Fuente: Elaboración Propia

En ese sentido, se realiza un análisis de los resultados de la clasificación de Regresión Logística, con hiperparámetro “l1_ratio” con valor 0.5, obteniendo los siguientes valores de Accuracy por dependencia:

Tabla 9. ***Accuracy por dependencia***

NombreDependencia	Accuracy
Con_Int_Discipli	**0.675966**
Des_Alcalde	**0.473752**
Espacio_Publico	**0.487431**

NombreDependencia	Accuracy
Extra	0.586545
Ofc_TIC	0.640483
Pla_Sisben	0.839474
Sec_Admin	0.554428
Sec_Des_social	0.609881
Sec_Educacion	0.734370
Sec_Hacienda	0.782466
Sec_Infraest	0.484503
Sec_Interior	0.387147
Sec_Juridica	0.715671
Sec_Planeacion	0.557005
Sec_Salud_y_ambi	0.530110
Valorizacion	0.872365

4.3.2 IMPLEMENTACIÓN ARQUITECTURA Y SELECCIÓN DE HIPERPARÁMETROS

Ahora bien, en el proceso de definición de pasos a seguir, se estableció la necesidad de analizar a profundidad dos escenarios, el primero la aplicación de la arquitectura y en segunda instancia, la selección de los hiperparámetros que mejor se ajusten a los modelos y a las características de los datos. En ese sentido, se llevaron a cabo los ejercicios en el **Anexo 2. Selección de Hiperparámetro**s con las siguientes actividades, que se consolidan a continuación.

a. Primer Componente

Se determina como primer componente de la Arquitectura del Modelo los registros que son Secretaría de Hacienda frente a los de las otras dependencias, caracterizados como "otros".

Tabla 10. ***Primer Componente***

NombreDependencia	INPUT	Porcentaje
Sec_Hacienda	317272	**53.72%**
otro	273320	**46.27%**

Fuente: Elaboración Propia

En este ejercicio, se aplica vectorización TF-IDF al conjunto con una característica especial, se toma el conjunto de entrenamiento el 80% y de prueba el restante del número de registros `otros`, con el objetivo de establecer valores bases de entrenamiento y un parámetro de apoyo como lo es, ignorar palabras y pares de palabras que aparecen en menos de 2 documentos, que no son útiles para la clasificación de las dependencias. Un ejemplo de ello es la siguiente tabla:

Tabla 11. ***Aplicación TF-IDF Primer Componente 37.649***

Palabra	TF-IDF
declaracion	0.537660
periodo	0.402914
ao	0.281163
presentacion	0.277064
registro	0.264828
...	...
penaprosperidadsocial	0.000002
gaseosos	0.000002
pencion	0.000002
becerril	0.000002
zzaingongbike	0.000002

Fuente: Elaboración Propia

Teniendo en cuenta esto, se procede a desarrollar los escenarios para la selección de los mejores hiperparámetros para los clasificadores a utilizar, para ello se utiliza dos estimadores de clasificación, LogisticRegressionCV y GridSearchCV, que son técnicas de validación cruzada incluida en el paquete de `sklearn`, que se ejecuta a través de los diferentes parámetros que se introducen en la cuadrícula de parámetros y extrae los mejores valores y combinaciones de parámetros para los diferentes clasificadores.

Con base en ello, los modelos como Regresión Logística y ComplementNB se aplicaron obteniendo los siguientes resultados:

Tabla 12. ***Selección Parámetros Primer Componente***

Clasificador	Estimadores de clasificación	Hiperparámetros	Rangos Hiperparámetros	Resultados	Accuracy
Regresión Logística	LogisticRegressionCV	C	0.1 - 3	3.55	0.9440
Regresión Logística	LogisticRegressionCV	l1_ratio	0.1 – 0.8	0.55	
ComplementNB	GridSearchCV	alpha	0.1 - 19.0	0.1	0.9355

Fuente: Elaboración Propia

b. Segundo Componente

El segundo componente de la arquitectura contempla el análisis de los datos a partir de dos escenarios, el primero de ellos es que se retoma el dataset `Data_corregido.csv` pero se desestiman los registros valores de `Sec_Hacienda` y se seleccionan los registros de las dependencias que están por encima de 35.000 solicitudes, respondiendo al equilibrio y dependencias dominantes en el dataset, logrando la siguiente distribución:

Tabla 13. ***Segundo Componente***

NombreDependencia	INPUT	Porcentaje
Sec_Planeacion	35339	**13%**

NombreDependencia	INPUT	Porcentaje
Sec_Salud_y_ambi	41269	**15%**
Sec_Admin	43680	**16%**
Sec_Interior	47370	**17%**
Extra	105662	**39%**

Fuente: Elaboración Propia

Al dataset se replica la vectorización TF-IDF con los parámetros aplicados en el primer componente, obteniendo los siguientes resultados.

Tabla 14. ***Aplicación TF-IDF Segundo Componente 306924***

Palabra	TF-IDF
solicitud	0.421835
bucaramanga	0.340867
informacion	0.168444
peticion	0.163073
derecho	0.138262
...	...
ala seora	0.000000
ley 10	0.000000
ala respuesta	0.000000
ley 1042	0.000000
zx ventures	0.000000

Fuente: Elaboración Propia

Tal como se desarrolló para el Primer Componente, se procede a desarrollar los escenarios para la selección de los mejores hiperparámetros para los clasificadores a utilizar, en este caso, se utiliza GridSearchCV, para los modelos como ComplementNB, RandomForest y Multinomial NB obteniendo los siguientes resultados:

Tabla 15. ***Selección Parámetros Segundo Componente***

Clasificador	Estimadores de clasificación	Hiperparámetros	Rangos Hiperparámetros	Resultados	Accuracy
ComplementNB	GridSearchCV	alpha	0.1 - 19.0	0.6000	0.73
MultinomialNB	GridSearchCV	alpha	0.1 - 19.0	0.2	0.729
RandomForest	GridSearchCV	n_estimators	20 - 200	180	0.7507

Fuente: Elaboración Propia

c. Tercer Componente

El tercer componente de la arquitectura contempla el análisis de los datos a partir de dos escenarios, el primero de ellos es que se retoma el dataset `Data_corregido.csv` pero se desestiman los registros valores de `Sec_Hacienda` realizado en el primer componente y las dependencias que están por encima de 35.000 solicitudes realizadas en segundo componente. Seleccionando los registros de las dependencias que están por encima de 7.000 solicitudes, respondiendo al equilibrio y dependencias dominantes en el dataset, logrando la siguiente distribución:

Tabla 16. **Tercer Componente**

NombreDependencia	INPUT	Porcentaje
Espacio_Publico	7126	**7%**
Sec_Educacion	7238	**7%**
Extra	9551	**9%**
Sec_Juridica	10693	**10%**
Valorizacion	13238	**13%**
Des_Alcalde	15058	**14%**
Sec_Infraest	20585	**19%**

NombreDependencia	INPUT	Porcentaje
Sec_Des_social	22173	**21%**

Fuente: Elaboración Propia

Al dataset se replica la vectorización TF-IDF con los parámetros aplicados en el primer componente, obteniendo los siguientes resultados.

Tabla 17. ***Aplicación TF-IDF Tercer Componente 819172***

Palabra	TF-IDF
Bucaramanga	0.326903
solicitud	0.292664
peticion	0.227935
informacion	0.210816
alcalde	0.200544
...	...
desafortunada situacion	0.000000
desafortunadamente afecto	0.000000
desafortunadamente archivo	0.000000
desafortunadamente cambio	0.000000
zykagradezco toda	0.000000

Fuente: Elaboración Propia

Tal como se desarrolló para el Primer y Segundo Componente, se procede a desarrollar los escenarios para la selección de los mejores hiperparámetros para los

clasificadores a utilizar, en este caso, se utiliza GridSearchCV, para los modelos como ComplementNB y Multinomial NB obteniendo los siguientes resultados:

Tabla 18. ***Selección Parámetros Tercer Componente***

Clasificador	Estimadores de clasificación	Hiperparámetros	Rangos Hiperparámetros	Resultados	Accuracy
ComplementNB	GridSearchCV	alpha	0.1 - 19.0	0.2	0.7541
MultinomialNB	GridSearchCV	alpha	0.1 - 19.0	0.1	0.7513

Fuente: Elaboración Propia

d. Cuarto Componente

El tercer componente de la arquitectura contempla el análisis de los datos a partir de dos escenarios, el primero de ellos es que se retoma el dataset `Data_corregido.csv` pero se desestiman los registros que están por encima de 7.000 solicitudes con todo lo trabajado en el primer, segundo y tercer componente, respondiendo al equilibrio y dependencias dominantes en el dataset, logrando la siguiente distribución:

Tabla 19. ***Cuarto Componente***

NombreDependencia	INPUT
Ofc_TIC	1331
Con_Int_Discipli	1466

NombreDependencia	INPUT
Pla_Sisben	2900
Extra	3854

Fuente: Elaboración Propia

Al dataset se replica la vectorización TF-IDF con los parámetros aplicados en el primer componente, obteniendo los siguientes resultados.

Tabla 20. ***Aplicación TF-IDF Cuarto Componente 227277***

Palabra	TF-IDF
Bucaramanga	0.334791
solicitud	0.261035
informacion	0.198850
peticion	0.182219
alcalde	0.164864
...	...
despues mes	0.000000
despues meses	0.000000
despues nia	0.000000
despues notificadas	0.000000
zunilda martirena	0.000000

Fuente: Elaboración Propia

Tal como se desarrolló para el Primer, Segundo y Tercer Componente, se procede a desarrollar los escenarios para la selección de los mejores hiperparámetros para los

clasificadores a utilizar, en este caso, se utiliza GridSearchCV, para los modelos como ComplementNB y Multinomial NB obteniendo los siguientes resultados:

Tabla 21. ***Selección Parámetros Cuarto Componente***

Clasificador	Estimadores de clasificación	Hiperparámetros	Rangos Hiperparámetros	Resultados	Accuracy
ComplementNB	GridSearchCV	alpha	0.1 - 19.0	0.4	0.7162
MultinomialNB	GridSearchCV	alpha	0.1 - 19.0	0.1	0.7147

Fuente: Elaboración Propia

4.3.3 ENTRENAMIENTO Y EVALUACIÓN DEL MODELO

Teniendo en cuenta las anteriores actividades, el equipo realizó el entrenamiento de los modelos dentro de los componentes de la arquitectura mixta, obteniendo los siguientes resultados para los diferentes clasificadores con los valores para los hiperparámetros identificados en el numeral anterior. Para ser precisos, en el Anexo 3. Entrenamiento y Evaluación Modelado Tradicional está todo el despliegue realizado.

En primera instancia, se toma nuevamente el dataset 'Data_corregido.csv' y se inicia el despliegue de los modelos de clasificación para cada componente, descrito a continuación.

a. Primer Componente

Para este componente, se aplican dos clasificadores, Regresión Logística y Complement NB, obteniendo los siguientes resultados.

Tabla 22. ***Resultados Primer Componente***

Clasificador	Hiperparámetros aplicados	Valor Hiperparámetros	Accuracy
Regresión Logística	C	3.55	0.9482
	L1_ratio	0.55	
ComplementNB	alpha	0.1	0.9424

Fuente: Elaboración Propia

b. Segundo Componente

Para este componente, se aplican dos clasificadores ComplementNB, MultinomialNB y Regresión Logística, este último clasificador se decidió aplicarlo y buscar nuevamente los mejores parámetros como medida para mejorar el score de respuesta, para ello, el clasificador fue elaborado con GridSearchCV obteniendo los siguientes resultados.

Tabla 23. ***Resultados Segundo Componente***

Clasificador	Hiperparámetros aplicados	Valor Hiperparámetros	Accuracy
ComplementNB	alpha	0.6	0.7046
MultinomialNB	alpha	0.2	0.7211
Regresión Logística	C	1.5	0.7241
	L1_ratio	0.2	

Fuente: Elaboración Propia

a. Tercer Componente

Para este componente, se aplican dos clasificadores ComplementNB, MultinomialNB, Regresión Logística y SVC, estos últimos clasificadores se decidieron aplicarlos de la misma manera cómo se aplicó para el segundo componente, obteniendo los siguientes resultados.

Tabla 24. ***Resultados Tercer Componente***

Clasificador	Hiperparámetros aplicados	Valor Hiperparámetros	Accuracy
ComplementNB	alpha	0.2	0.7244
MultinomialNB	alpha	0.1	0.7175
Regresión Logística	C	3.5	0.7273
	L1_ratio	0.30000000000000004	
Support Vector Machine (SVM)	Kernel	Linear	0.6681

Fuente: Elaboración Propia

a. Cuarto Componente

Para este componente, se aplican dos clasificadores ComplementNB, MultinomialNB, Regresión Logística y SVC, estos últimos clasificadores se decidieron aplicarlos de la misma manera cómo se aplicó para el segundo componente, obteniendo los siguientes resultados.

Tabla 25. ***Resultados Cuarto Componente***

Clasificador	Hiperparámetros aplicados	Valor Hiperparámetros	Accuracy
ComplementNB	alpha	0.4	0.8431
MultinomialNB	alpha	0.1	0.8416
Regresión Logística	C	3.5	0.7417
	L1_ratio	0.55	
Support Vector Machine (SVM)	Kernel	Linear	0.8406
RandomForest	n_estimators	180	0.7904

Fuente: Elaboración Propia

Con lo anterior, la arquitectura mixta articula los mejores resultados de los clasificadores aplicados por cada componente, permitiendo obtener la siguiente actualización:

Ilustración 9. ***Modelo y Arquitectura Mixta o Híbrida definitiva***

Fuente: Elaboración Propia

4.3.4 VALIDACIÓN DEL MODELO

Teniendo en cuenta las anteriores actividades, el equipo realizó el cargue de los modelos entrenados anteriormente dentro de los componentes de la arquitectura mixta utilizando las funcionalidades `load` de la biblioteca `joblib`. Todo el despliegue realizado, se encuentra en el Anexo 4. Test 60.000 PQRSD.

En primera instancia, para cada uno de los componentes de la Arquitectura mixta, se realizan dos actividades, la primera, se aplica vectorización TF-IDF del dataset, se guarda y carga a través de la biblioteca `joblib`. Así mismo, se realiza la carga de los clasificadores con mejor resultado de la siguiente manera;

Tabla 26. ***Clasificadores aplicar en cada componente***

Componente	Clasificador	Valor Hiperparámetros
Primer componente	Regresión Logística	C=3.55, l1_ratio=0.55
Segundo Componente	Regresión Logística	C=1.5, l1_ratio=0.2
Tercer Componente	Regresión Logística	C=3.5, l1_ratio=0.3
Cuarto componente	ComplementNB	alpha=0.4

Fuente: Elaboración Propia

Con lo anterior y en mirar de validar la aplicación de la arquitectura mixta o híbrida, se tomaron 64.632 registros que no fueron parte de los datos de entrenamiento y se analizaron los resultados de Accuracy en ellos, obteniendo un 70.076% total. Incrementando en cada una de las dependencia como se observa en la siguiente tabla:

Tabla 27. ***Validación Modelo Arquitectura 64.632 datos***

NombreDependencia	Accuracy
PLA_SISBEN	0.806264
SEC_ADMIN	0.800532
SEC_DES_SOCIAL	0.670257
SEC_EDUCACION	0.676471
SEC_HACIENDA	0.876558
SEC_INFRAEST	0.596432
SEC_INTERIOR	0.746697
SEC_JURIDICA	0.736342
SEC_SALUD_Y_AMBI	0.747438
VALORIZACION	0.809273

Fuente: Elaboración Propia

Con lo anterior, se valida que el uso de la arquitectura híbrida o mixta en el escenario de desbalance de los datos que tiene la Alcaldía de Bucaramanga, permite incrementar el Accuracy en las dependencias con mayor recurrencia de manera significativa.

Ahora bien, teniendo en cuenta este proceso y como ejercicio de validación, el siguiente paso es cargar los datos de prueba nuevos que consiste en una base de datos de 1.000 PQRSD recibidas posteriormente por la entidad, que no hicieron parte de las pruebas y entrenamiento permitiendo evitar sesgos frente a la evaluación y efectividad del proceso.

En ese sentido, se carga el dataset `test_data.csv` preprocesado y ajustado a las variables y estructuras construidas previamente.

Tabla 28. ***Dataset de pruebas (1.000 PQRSD)***

index	NombreDependencia	input
0	PLA_SISBEN	derecho de petición pla silben derecho de peti...
1	SEC_HACIENDA	solicitud devolución saldo a favor impuesto pr...
2	SEC_DES_SOCIAL	adulto mayor hola buenas tardes mi nombre es b...
3	SEC_INTERIOR	defensora del pueblo remisión de comunicación ...
4	SEC_INTERIOR	querella policia querella policia
...	...	...
995	ESPACIO_PUBLICO	cuenta de cobro mar 2023 centro comercial fegh...
996	SEC_HACIENDA	cobro injustificado en impuesto de industria y...
997	SEC_HACIENDA	solicitud inscripción industria y comercio sol...
998	SEC_INTERIOR	queja anonima peticionario presenta queja cont...
999	SEC_HACIENDA	desembarco de cuentas bancarias desembarco de ...

1000 rows × 2 columns

Fuente: Elaboración Propia

Con esto, se crea la función `eval_test` que realiza las siguientes actividades, primero, carga el vectorizador TF-IDF y el modelo clasificador (ambos ajustados y guardados previamente) para el primer componente. Luego, transforma los datos de entrada usando el vectorizador TF-IDF y predice la 'NombreDependencia' utilizando el clasificador.

Posterior a ello, carga el vectorizador TF-IDF y el clasificador para el segundo componente donde transforma solo las entradas donde la predicción en el primer componente fue 'otro', es decir no es `Sec_Hacienda`. Luego, actualiza esas predicciones utilizando el clasificador del segundo componente.

Para el tercer componente y similar al segundo componente, solo transforma y actualiza las predicciones donde el resultado en el componente anterior fue 'Extra' y por último, transforma y actualiza las predicciones para el cuarto componente.

Finalmente, compara las predicciones resultantes con las verdaderas 'NombreDependencia' del dataset `test_data.csv` y devuelve la media de esta comparación. Esta puntuación representa la precisión del proceso de predicción que en nuestro caso es de 63.4%.

Resultados de clasificación

Para la validación del desempeño real de los datos existentes fuera de la base de prueba y entrenamiento, se determinó realizar una comparación con las designaciones realizadas por un (usuario) funcionario perteneciente a la Oficina Asesora TIC, con más de dos años de experiencia y que dentro de sus actividades, se encuentra el apoyo a la asignación respuesta de PQRSD.

En este ejercicio, de la base de datos aplicadas en el item anterior, se tomó una muestra representativa de 390 registros de peticiones aleatorias del presente año (Confianza 95% y Error 5%), con las que dicho funcionario pudiese realizar las

asignaciones de estas peticiones a las dependencias que a su criterio, deberían ser las responsables de dar respuestas a las mismas, para posteriormente, comparar estos resultados por los arrojados por el modelo de arquitectura híbrida o mixta desarrollado en el presente proyecto.

Tabla 29. ***Set de pruebas – 390 PQRSD***

index	Dependencia Asignada Originalmente	Número de Caracteres	Input	Asignación Manual	Precisión
0	SEC_INTERIOR	93	solicitud ayuda por afectación de ola in…	SEC_HACIENDA	0
1	SEC_DES_SOCIAL	41	petición autorización elecciones atípica…	SEC_INTERIOR	0
2	SEC_INTERIOR	93	solicitud ayuda por afectación de ola in…	SEC_HACIENDA	0
3	PLA_SISBEN	320	solicitud incluir en el silben buen dia …	PLA_SISBEN	1
4	SEC_PLANEACION	47	solicitud vista tecnica solicitud vista …	SEC_SALUD_Y_AMBI	0
5	SEC_HACIENDA	43	solicitud paz y salvo solicitud paz y sa…	SEC_HACIENDA	1
6	SEC_HACIENDA	115	presentación declaración impuesto de ind…	SEC_HACIENDA	1
7	SEC_HACIENDA	63	desembarco de cuentas bancarias desembar…	SEC_HACIENDA	1
8	SEC_INFRAEST	50	solicitud de colaboración para retiro de…	SEC_INFRAEST	1
9	SEC_SALUD_Y_AMBI	39	solicitud de visita solicitud de visita…	SEC_INTERIOR	0
10	SEC_INTERIOR	843	2022 348 despacho comisario senores alca…	SEC_INTERIOR	1
11	EXTRA	2615	el proximo martes continua modernización…	SEC_INFRAEST	0

index	Dependencia Asignada Originalmente	Número de Caracteres	Input	Asignación Manual	Precisión
12	SEC_INFRAEST	567	apoyo de poda y de retiro de frutos coco…	SEC_INFRAEST	1
13	SEC_HACIENDA	612	solicitud de formulario e información pa…	SEC_HACIENDA	1
14	SEC_HACIENDA	335	registro industria y comercio buenas tar…	SEC_HACIENDA	1
15	SEC_HACIENDA	335	registro industria y comercio buenas tar…	SEC_HACIENDA	1
16	SEC_HACIENDA	57	solicitud prescripción deuda solicitud p…	SEC_HACIENDA	1
17	SEC_HACIENDA	57	solicitud prescripción deuda solicitud p…	SEC_HACIENDA	1
18	SEC_HACIENDA	57	solicitud prescripción deuda solicitud p…	SEC_HACIENDA	1
19	SEC_HACIENDA	57	solicitud prescripción deuda solicitud p…	SEC_HACIENDA	1
20	ESPACIO_PUBLICO	5273	derecho de petición solicitud senores al…	ESPACIO_PUBLICO	1

390 rows

Fuente: Elaboración Propia

Para el primer frente de trabajo, presentamos los resultados del usuario humano fueron los siguientes:

Tabla 30. **Resultados asignación usuario humano**

Total PQRSD	**Asignación Manuales correctas**	**Porcentaje de efectividad**
390	231	59.2%

Fuente: Elaboración Propia

Para el segundo frente de trabajo, se realizó la verificación del set de datos de prueba en el modelo aplicado, obteniendo los siguientes resultados:

Tabla 31. ***Resultados Modelo Construido***

Total PQRSD	Asignación por modelo correctas	Porcentaje de efectividad
390	254	65.1%

Fuente: Elaboración Propia

Los resultados que nos arroja este ejercicio son bastante importantes, teniendo en cuenta que al tomar las mismas condiciones de muestra para los dos escenarios, el porcentaje de efectividad evidencia un 6% de mayor confianza el modelo construido frente a la asignación realizada por un usuario humano experto que conoce la entidad, por lo que se hace evidente también que la calidad de la información es primordial y que las peticiones con campos vacíos o con textos muy cortos, generan complicaciones para una asignación efectiva.

Con base en lo anterior, el equipo también desplegó un ejercicio tomando solicitudes que tuviesen más de 256 caracteres que se tienen incluidas dentro de los ejercicios de entrenamiento y pruebas, con el objetivo de verificar el comportamiento del modelo con peticiones de estas características, obteniendo los siguientes resultados frente a los usuarios humanos.

Tabla 32. ***Resultados usuario humano – Peticiones más de 256 caracteres***

Total PQRSD	Asignación Manual correctas	Porcentaje de efectividad
390	275	70.5%

Fuente: Elaboración Propia

Tabla 33. ***Resultados Modelo construido – Peticiones más de 256 caracteres.***

Total PQRSD	Asignación por modelo correctas	Porcentaje de efectividad
390	265	67.9%

Fuente: Elaboración Propia

Tomando este segundo alcance sobre las solicitudes analizadas y comparadas entre el modelo construido y las asignaciones de los usuarios, se infiere que los usuarios humanos funcionales requieren más información para poder asignar de una manera más eficiente las peticiones que llegan a la entidad, y que esto sugiere que se creen mecanismos para que los sistemas de información mejores los criterios de creación y calidad de los datos en cada uno de los registros.

4.4 DESARROLLO Y DESPLIEGUE

4.4.1 Prototipo

Cómo parte de los objetivos del proyecto, se desarrolla un prototipo o herramienta web y móvil a través del framework Flask que permite a los usuarios funcionales de la Alcaldía de Bucaramanga minimizar los tiempos de designación y tener recomendaciones o sugerencias de áreas a designar las PQRSD que se reciban en la entidad.

La aplicación web utiliza los modelos de clasificación previamente entrenados para realizar las predicciones sobre los textos ingresados. Estos modelos serán cargados y utilizados en conjunto con técnicas de vectorización, como TF-IDF, para convertir los textos en representaciones numéricas adecuadas para el procesamiento. El sistema permite a los usuarios ingresar sus textos a través de un formulario en la interfaz web. Una vez que se envíe el texto, se realizará una solicitud POST al servidor Flask, donde se realizará la transformación del texto y se ejecutará el modelo de clasificación correspondiente para obtener la predicción. [10]

Además, se implementó un diseño visual atractivo y responsive para pantallas móviles utilizando HTML, CSS y librerías de estilos como Bootstrap. Se incluye la funcionalidad de modales para mostrar las predicciones en una ventana emergente sin necesidad de recargar la página.

Estas son las interfaces de usuario desarrolladas y desplegadas:

Ilustración 10. ***Interfaz ususario - Home***

Fuente: Elaboración Propia

El usuario tiene la información para seguir los pasos de manera simple y sencilla de la siguiente manera:

a. Inserte el contenido de la PQRSD recibida en el campo de texto.

b. De click en el botón "Analizar".

c. Se desplegará un modal con la recomendación de asignación de la PQRSD.

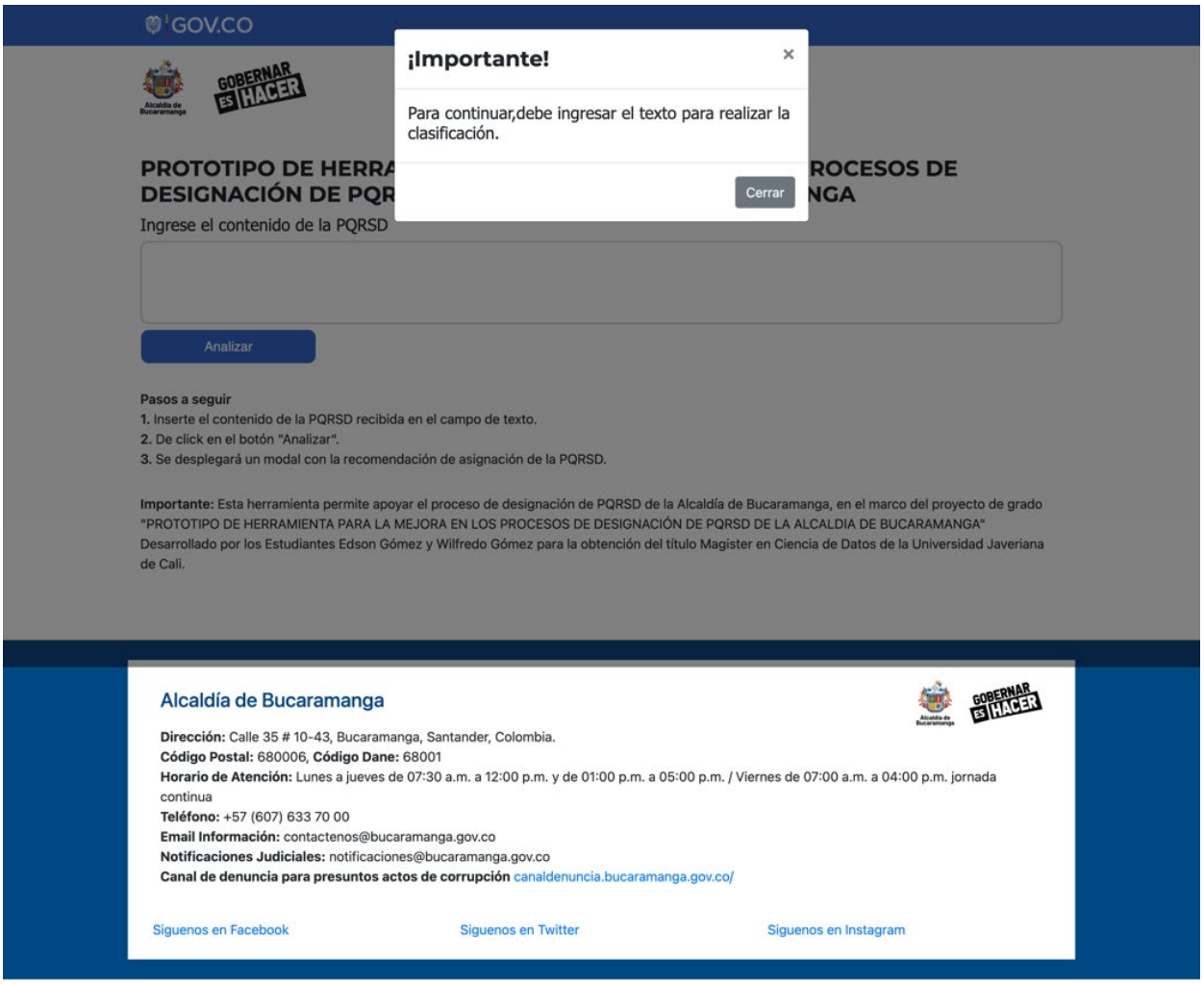
Ilustración 11. ***Interfaz Usuario – Resultado de predicción***

Fuente: Elaboración Propia

Finalmente, al dar click en el botón “Cerrar” del modal, se limpia el campo de texto para consultar otro texto.

En los casos donde el campo de texto se encuentra vacío, la aplicación arroja una alerta a través del modal información que, para continuar, se debe ingresar el texto para realizar la clasificación.

Ilustración 12. ***Alerta de ingreso de texto para clasificar***



Fuente: Elaboración Propia

4.5 ACEPTACIÓN

En el desarrollo de soluciones de software, la usabilidad es un factor determinante para el éxito de la adopción y eficiencia de la herramienta por parte de los usuarios finales. Por esta razón, la evaluación de usabilidad se convierte en un componente esencial en el ciclo de vida del desarrollo de nuestra aplicación.

La usabilidad no se refiere solo a la estética de la interfaz del usuario, sino también a cómo los usuarios interactúan con la aplicación y si su diseño y funcionalidades facilitan

o no la realización de las tareas propuestas. Con una evaluación de usabilidad, podemos identificar áreas de mejora y posibles fallas desde el punto de vista de los usuarios, lo que permite ajustar la herramienta a sus necesidades reales.

En esta sección, se presenta una herramienta de evaluación de usabilidad para la aplicación de designación de PQRSD desarrollada para la Alcaldía de Bucaramanga. Esta herramienta se basa en las recomendaciones del estándar ISO 9241-210 "Ergonomía de la Interacción Humano-Sistema" y las 10 Heurísticas de Nielsen para el diseño de interfaces de usuario, ambas metodologías reconocidas y utilizadas globalmente en la evaluación de la usabilidad de sistemas de software.

La evaluación de usabilidad se plantea como una serie de preguntas en un lenguaje sencillo y amigable, con el fin de que sea accesible y fácil de comprender por nuestros usuarios finales, quienes son funcionarios públicos de la Alcaldía de Bucaramanga encargados de la designación de las PQRSD. Estas preguntas permitirán a los usuarios indicar si ciertos aspectos de la usabilidad de la aplicación se cumplen o no y proporcionar observaciones para cada ítem.

Tabla 34. ***Formulario Validación Usabilidad***

FORMULARIO VALIDACIÓN USABILIDAD

Evaluación prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

No	Pregunta de Evaluación	Cumple	Observaciones
1	¿Recibe una respuesta del sistema cuando realiza una acción (por ejemplo, ingresar texto, hacer clic en "Analizar", etc.)?	Sí / No	
2	¿Se siente cómodo con los términos y el lenguaje utilizado en la aplicación? ¿Son claros y comprensibles?	Sí / No	

No	Pregunta de Evaluación	Cumple	Observaciones
3	Si comete un error (por ejemplo, deja el campo de texto vacío), ¿puede corregirlo fácilmente?	Sí / No	
4	¿La aplicación mantiene un diseño y una disposición similares en todas las pantallas?	Sí / No	
5	¿La aplicación le ayuda a evitar errores? Por ejemplo, ¿le avisa si olvida llenar un campo?	Sí / No	
6	¿Puede encontrar fácilmente las opciones y funciones que necesita usar?	Sí / No	
7	¿La información innecesaria o distractora se mantiene al mínimo en la aplicación?	Sí / No	
8	Si aparece un mensaje de error, ¿es claro para entender cuál es el problema y cómo resolverlo?	Sí / No	
9	Los pasos para usar la aplicación, ¿son claros y fáciles de seguir?	Sí / No	
10	¿Pudo usar la aplicación sin necesidad de formación o ayuda externa?	Sí / No	
11	¿Se siente su opinión tomada en cuenta en el diseño y desarrollo de la aplicación?	Sí / No	
12	¿La aplicación cumple con sus necesidades y expectativas en su tarea de designar PQRSD?	Sí / No	
13	¿Ha notado mejoras en la aplicación a medida que la ha utilizado?	Sí / No	

Fuente: Elaboración Propia

Se realizó validación del prototipo mediante el presente formulario a través de un contratista de la entidad encargado del proceso de designación de las PQRSD, para una de las dependencias de la entidad , los resultados fueron positivos en tanto al entendimiento del uso de la herramienta como de la apropiación de los elementos de la funcionalidad de la misma, a dicho usuario le tomo alrededor de 2 minutos la

comprensión de uso, y su validación con datos reales del aplicativo de PQRSD de la entidad. Los resultados de la aplicación fueron los siguientes:

*Tabla 35**. Formulario diligenciado***

No	Pregunta de Evaluación	Cumple	Observaciones
1	¿Recibe una respuesta del sistema cuando realiza una acción (por ejemplo, ingresar texto, hacer clic en "Analizar", etc.)?	Sí	Evidencia los resultados de la designación
2	¿Se siente cómodo con los términos y el lenguaje utilizado en la aplicación? ¿Son claros y comprensibles?	Sí	Es claro
3	Si comete un error (por ejemplo, deja el campo de texto vacío), ¿puede corregirlo fácilmente?	Sí	Si genera un aviso y puedo borrar el texto que haya ingresado mal
4	¿La aplicación mantiene un diseño y una disposición similares en todas las pantallas?	Sí	No se percibieron modificaciones
5	¿La aplicación le ayuda a evitar errores? Por ejemplo, ¿le avisa si olvida llenar un campo?	Sí	Si genera un aviso
6	¿Puede encontrar fácilmente las opciones y funciones que necesita usar?	Sí	Es muy simple
7	¿La información innecesaria o distractora se mantiene al mínimo en la aplicación?	Sí	No hubo distractores

8	Si aparece un mensaje de error, ¿es claro para entender cuál es el problema y cómo resolverlo?	Sí	Indica que el campo este vacio
9	Los pasos para usar la aplicación, ¿son claros y fáciles de seguir?	Sí	Si muy sencillo
10	¿Pudo usar la aplicación sin necesidad de formación o ayuda externa?	Sí	Si se asemeja a la interfaz del sistema de PQRSD
11	¿Se siente su opinión tomada en cuenta en el diseño y desarrollo de la aplicación?	No	
12	¿La aplicación cumple con sus necesidades y expectativas en su tarea de designar PQRSD?	Sí	En contadas ocasiones tiende a enviar a despacho
13	¿Ha notado mejoras en la aplicación a medida que la ha utilizado?	Sí	Que se conecte directamente al aplicativo de Pqrsd y que tome la información sin tener que copiar y pegar

Fuente: Elaboración Propia

Observaciones adiciones: *La aplicación es muy rápida, efectiva y puede ayudar a las capacitaciones de los funcionarios y contratistas nuevos que no conocen la entidad, sin embargo, debe ser una herramienta de apoyo y no sustituir la labor de los usuarios destinados para tal fin por la responsabilidad del proceso. Se podría automatizar aún mas el proceso si se conectara o se pudiese utilizar dentro del aplicativo de PQRSD pues así no tendría que copiar y pegar. Se puede mejorar aún más si pudiese arrojar más de una respuesta en dado caso que la PQRSD necesite ser remitida a más de una área*.

5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1 CONCLUSIONES

5.1.1 Se lograron los objetivos del proyecto al desarrollar un prototipo de herramienta de analítica de datos que apoya efectivamente los procesos de designación de las PQRSD en la Alcaldía de Bucaramanga. Este prototipo se construyó utilizando un enfoque de clasificación mixta que ha demostrado ser eficaz para gestionar los desafíos específicos que presentan los datos de la Alcaldía.

5.1.2 El análisis de la base de datos de la Alcaldía de Bucaramanga permitió entender mejor la estructura de los datos, su calidad y la lógica del negocio detrás de la situación problema. Además, este análisis inicial fue esencial para identificar la predominancia de la Secretaría de Hacienda en el volumen de solicitudes y la necesidad de un enfoque de clasificación mixta.

5.1.3 La implementación de la Arquitectura de Clasificación Mixta o Híbrida ha demostrado ser una solución eficaz y robusta a los desafíos que presentan los datos desbalanceados en el contexto de la Alcaldía de Bucaramanga. Este enfoque permitió una segmentación y clasificación interesante de las solicitudes, teniendo en cuenta las variaciones significativas en el volumen de estas entre las diferentes dependencias.

5.1.4 La selección de hiperparámetros para los modelos de clasificación como ComplementNB, RandomForest, Regresión Logística, MultinomialNB fue interesante al implementar las librerías de GridSearchCV y LogisticRegressionCV que permitieron analizar las diferentes opciones de los escenarios y dieron los valores de los hiperparámetros que más se ajustaban a la definición dada.

5.1.5 El modelo clasificatorio con mejores resultados es Regresión Logística con hiperparámetros `c` y `l1_ratio` optimizados niveles de efectividad por encima del

90% para el primer componente y por encima del 70.5% para los componentes dos y tres. Para el componente 4, se define ComplementNB con efectiva del 80%, lo que indica un buen ejercicio de clasificación teniendo en cuenta la calidad de la data de la Alcaldía de Bucaramanga.

5.1.6 Se desarrollaron visualizaciones e interfaces intuitivas para la herramienta que facilitan la designación de las PQRSD. Esta interfaz proporciona sugerencias útiles relativas a la dependencia probable que los modelos y técnicas de analítica indican, apoyando a los funcionarios en su labor de asignación de solicitudes.

5.1.7 La herramienta prototipo fue evaluada en entornos controlados de pruebas y preproducción con usuarios internos, validando su impacto positivo en el proceso de designación de PQRSD para la Alcaldía de Bucaramanga. Además, se implementó un cuestionario basado en los principios de usabilidad del estándar ISO 9241-210 y las heurísticas de Nielsen, para asegurar que la interfaz es efectiva, eficiente y satisfactoria para un usuario.

5.1.8 Se concluye que la calidad de los datos es fundamental para realizar las asignaciones de manera efectiva, se evidencio que tanto usuarios humanos como el modelo desarrollado, tuvieron desempeños más bajos en condiciones donde la data provenía de campos vacíos o caracteres limitados.

5.2 TRABAJOS FUTUROS

5.2.1 La capacidad de entendimiento que da la construcción de la arquitectura híbrida en el proyecto permite realizar ejercicio de exploración con diferentes arquitecturas de redes neuronales como Pytorch para mejorar aún más la precisión de los modelos de clasificación.

5.2.2 Se puede profundizar en métodos de balanceo de los datos o dinámicas escaladas para recorrer la clasificación orientadas en arboles binarios que respondan a los retos generados por sesgos con set de datos de estas características

5.2.3 Se pueden utilizar herramientas o técnicas de machine learning para recorrer espacios muestrales para la optimización de hiperparámetros que disminuyan el consumo y costo computacional.

5.2.4 Se pueden considerar diferentes técnicas de preprocesamiento más robustas para mejorar la calidad de los datos de entrenamiento y por lo tanto, mejorar la precisión de los modelos de clasificación.

5.2.5 Se pueden investigar técnicas de interpretación de modelos para comprender mejor cómo se toman las decisiones de clasificación y cómo se pueden mejorar aún más.

5.2.6 Se pueden explorar técnicas de transferencia de aprendizaje para aprovechar los conocimientos previos de modelos reentrenados en otros conjuntos de datos para mejorar la precisión de los modelos de clasificación en entornos específicos.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] Alcaldía de Bucaramanga, «INFORME DEL COMPORTAMIENTO DE LAS PETICIONES, QUEJAS, RECLAMOS, SUGERENCIAS Y DENUNCIAS (PQRSD),» Bucaramanga, 2019.

[2] Ortega Sanabria, Maritza. "La importancia de la evaluación y seguimiento a la atención de solicitudes ciudadanas en la gestión pública de Bogotá DC 2014." (2014).

[3] Gómez Villa, C. M. (2021). Diseño e implementación de mejora en la gestión de peticiones, quejas, reclamos, sugerencias y denuncias, en la Dirección de Bienes y Seguros de la Gobernación de Antioquia.

[4] Aliaga Ancco, J. J., & Chahuara Flores, L. (2019). Clasificación de la Prioridad de Atención a Reclamos Presentados por Clientes Utilizando Machine Learning.

[5] Espinel Farías, J. R., Amórtegui Aros, R., & Rincón Martínez, L. F. (2019). Diseño de una metodología analítica para el tratamiento de quejas y reclamos para el sector hogares y movilidad ETB.

[6] Mora-Ladino, J. A. (2022). Diseño del módulo de PQRS para la recepción de peticiones, quejas, reclamos y sugerencias del portal empleo de la Universidad Católica de Colombia implementando Business Intelligence para optimizar acciones correctivas.

[7] Acevedo Solarte, J. R., & González Sarmiento, C. A. (2018). Transformación digital para el mejoramiento de los servicios al ciudadano beneficiarios de la formalización de tierras en Colombia (Doctoral dissertation, Bogotá: Universidad Externado de Colombia, 2018.).

[8] Luque Sánchez, M. A., & Córtes Díaz, L. F. Análisis etiquetado de textos para predicción de la polaridad, enfoque semi supervisado y etiquetado automático.

[9] Microsoft. (2020). What is the team data science process? Microsoft. https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview

[10] E. Gómez C., "pqrsdbga," GitHub. [Online]. Disponible: https://github.com/edsongomezc/pqrsdbga. [Accessed: 2023].

[11] Alcaldía de Bucaramanga "Asignación Solicitudes BGA" Microsoft Power BI, 2023. [En línea]. Disponible: https://app.powerbi.com/view?r=eyJrIjoiZjU2NGM4NDMtNmQwNi00YjgyLTgzYWMtNDU0ZDdhNWE2ZTgwIiwidCI6Ijc4NjgzZmYyLTBjMjAtNGJkYS1iYzc3LWQ0YjJhODdmMmE2YSIsImMiOjR9&pageName=ReportSection838bc06d7d2171d83f05

Bucaramanga, 24 de Junio de 2023

Señores
PONTIFICIA UNIVERSIDAD JAVERIANA
SEDE CALI
E. S. D.

Asunto: Desarrollo de Pruebas

Respetados señores,

En mi calidad de Contratista a cargo de procesos asistenciales y administrativos de la Oficina TIC del Municipio de Bucaramanga, identificada con C.C. 1096950374 de Málaga me permito confirmar mi participación en el desarrollo de pruebas y validación del prototipo del aplicativo para la facilitación de la designación de PQRSD adelantada dentro del programa de posgrado **MAESTRÍA DE CIENCIA DE DATOS, relacionada al *Desarrollo de un prototipo de herramienta utilizando técnicas y modelos de analítica de datos, para el soporte de los procesos de designación de PQRSD al interior de las dependencias de la alcaldía de Bucaramanga.***

Participe en el proceso de validación de designaciones posibles de una muestra de 390 de PQRSD a partir de los asuntos y observaciones de la mismas, de igual manera en el diligenciamiento de la evaluación de la usabilidad del prototipo en un entorno controlado a través de un acceso remoto a la herramienta, según el formato y las indicaciones compartidas por los Ingenieros Wilfredo Gomez y Edson Gomez.

Se adjuntan los resultados en Excel y el Formato Diligenciado con Observaciones adicionales

Cordialmente,

Jackelin Bonilla.
JACKELINE BONILLA
CC 1096950374
Contratista
Oficina Asesora TIC

Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

Wrangling, Exploration and Cleaning

Estudiantes

Wilfredo Ariel Góme Bueno

Edson Andrés Gómez Cárdenas

In [1]:
```
!pip install swifter
!pip install pyspellchecker
!pip install pandarallel --upgrade
!pip install nltk
```

```
Requirement already satisfied: swifter in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (1.3.4)
Requirement already satisfied: cloudpickle>=0.2.2 in /Users/edsongomez/opt/a
naconda3/lib/python3.9/site-packages (from swifter) (2.0.0)
Requirement already satisfied: parso>0.4.0 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from swifter) (0.8.3)
Requirement already satisfied: psutil>=5.6.6 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from swifter) (5.9.0)
Requirement already satisfied: bleach>=3.1.1 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from swifter) (4.1.0)
Requirement already satisfied: ipywidgets>=7.0.0 in /Users/edsongomez/opt/an
aconda3/lib/python3.9/site-packages (from swifter) (7.6.5)
Requirement already satisfied: pandas>=1.0.0 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from swifter) (1.4.4)
Requirement already satisfied: dask[dataframe]>=2.10.0 in /Users/edsongomez/
opt/anaconda3/lib/python3.9/site-packages (from swifter) (2022.7.0)
Requirement already satisfied: tqdm>=4.33.0 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from swifter) (4.64.1)
Requirement already satisfied: packaging in /Users/edsongomez/opt/anaconda3/
lib/python3.9/site-packages (from bleach>=3.1.1->swifter) (21.3)
Requirement already satisfied: six>=1.9.0 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from bleach>=3.1.1->swifter) (1.16.0)
Requirement already satisfied: webencodings in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from bleach>=3.1.1->swifter) (0.5.1)
Requirement already satisfied: pyyaml>=5.3.1 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from dask[dataframe]>=2.10.0->swifter) (6.
0)
Requirement already satisfied: partd>=0.3.10 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from dask[dataframe]>=2.10.0->swifter) (1.
2.0)
Requirement already satisfied: fsspec>=0.6.0 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from dask[dataframe]>=2.10.0->swifter) (202
2.7.1)
Requirement already satisfied: toolz>=0.8.2 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from dask[dataframe]>=2.10.0->swifter) (0.1
1.2)
Requirement already satisfied: numpy>=1.18 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from dask[dataframe]>=2.10.0->swifter) (1.23.
5)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /Users/edsongome
z/opt/anaconda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifte
r) (1.0.0)
Requirement already satisfied: ipython-genutils~=0.2.0 in /Users/edsongomez/
opt/anaconda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifter)
(0.2.0)
Requirement already satisfied: nbformat>=4.2.0 in /Users/edsongomez/opt/anac
onda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifter) (5.5.0)
Requirement already satisfied: ipython>=4.0.0 in /Users/edsongomez/opt/anaco
nda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifter) (7.31.1)
Requirement already satisfied: widgetsnbextension~=3.5.0 in /Users/edsongome
z/opt/anaconda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifte
r) (3.5.2)
Requirement already satisfied: traitlets>=4.3.1 in /Users/edsongomez/opt/ana
conda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifter) (5.1.1)
Requirement already satisfied: ipykernel>=4.5.1 in /Users/edsongomez/opt/ana
conda3/lib/python3.9/site-packages (from ipywidgets>=7.0.0->swifter) (6.15.
```

```
2)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/edsongomez/o
pt/anaconda3/lib/python3.9/site-packages (from pandas>=1.0.0->swifter) (2.8.
2)
Requirement already satisfied: pytz>=2020.1 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from pandas>=1.0.0->swifter) (2022.1)
Requirement already satisfied: appnope in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->swifte
r) (0.1.2)
Requirement already satisfied: jupyter-client>=6.1.12 in /Users/edsongomez/o
pt/anaconda3/lib/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>
=7.0.0->swifter) (7.3.4)
Requirement already satisfied: matplotlib-inline>=0.1 in /Users/edsongomez/o
pt/anaconda3/lib/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>
=7.0.0->swifter) (0.1.6)
Requirement already satisfied: debugpy>=1.0 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->sw
ifter) (1.5.1)
Requirement already satisfied: nest-asyncio in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->sw
ifter) (1.5.5)
Requirement already satisfied: pyzmq>=17 in /Users/edsongomez/opt/anaconda3/
lib/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->swift
er) (23.2.0)
Requirement already satisfied: tornado>=6.1 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->sw
ifter) (6.1)
Requirement already satisfied: setuptools>=18.5 in /Users/edsongomez/opt/ana
conda3/lib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->
swifter) (67.7.2)
Requirement already satisfied: decorator in /Users/edsongomez/opt/anaconda3/
lib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifte
r) (5.1.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0
in /Users/edsongomez/opt/anaconda3/lib/python3.9/site-packages (from ipython
>=4.0.0->ipywidgets>=7.0.0->swifter) (3.0.20)
Requirement already satisfied: pygments in /Users/edsongomez/opt/anaconda3/l
ib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter)
(2.11.2)
Requirement already satisfied: pexpect>4.3 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swift
er) (4.8.0)
Requirement already satisfied: backcall in /Users/edsongomez/opt/anaconda3/l
ib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter)
(0.2.0)
Requirement already satisfied: jedi>=0.16 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swift
er) (0.18.1)
Requirement already satisfied: pickleshare in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swift
er) (0.7.5)
Requirement already satisfied: jsonschema>=2.6 in /Users/edsongomez/opt/anac
onda3/lib/python3.9/site-packages (from nbformat>=4.2.0->ipywidgets>=7.0.0->
swifter) (4.16.0)
Requirement already satisfied: jupyter_core in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from nbformat>=4.2.0->ipywidgets>=7.0.0->swi
```

```
fter) (4.11.1)
Requirement already satisfied: fastjsonschema in /Users/edsongomez/opt/anaco
nda3/lib/python3.9/site-packages (from nbformat>=4.2.0->ipywidgets>=7.0.0->s
wifter) (2.16.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/edsongome
z/opt/anaconda3/lib/python3.9/site-packages (from packaging->bleach>=3.1.1->
swifter) (3.0.9)
Requirement already satisfied: locket in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (from partd>=0.3.10->dask[dataframe]>=2.10.0->swif
ter) (1.0.0)
Requirement already satisfied: notebook>=4.4.1 in /Users/edsongomez/opt/anac
onda3/lib/python3.9/site-packages (from widgetsnbextension~=3.5.0->ipywidget
s>=7.0.0->swifter) (6.4.12)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0
in /Users/edsongomez/opt/anaconda3/lib/python3.9/site-packages (from jsonsch
ema>=2.6->nbformat>=4.2.0->ipywidgets>=7.0.0->swifter) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from jsonschema>=2.6->nbformat>=4.2.0->ipyw
idgets>=7.0.0->swifter) (21.4.0)
Requirement already satisfied: entrypoints in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from jupyter-client>=6.1.12->ipykernel>=4.5.1
->ipywidgets>=7.0.0->swifter) (0.4)
Requirement already satisfied: prometheus-client in /Users/edsongomez/opt/an
aconda3/lib/python3.9/site-packages (from notebook>=4.4.1->widgetsnbextensio
n~=3.5.0->ipywidgets>=7.0.0->swifter) (0.14.1)
Requirement already satisfied: nbconvert>=5 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from notebook>=4.4.1->widgetsnbextension~=3.
5.0->ipywidgets>=7.0.0->swifter) (6.4.4)
Requirement already satisfied: jinja2 in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->
ipywidgets>=7.0.0->swifter) (2.11.3)
Requirement already satisfied: Send2Trash>=1.8.0 in /Users/edsongomez/opt/an
aconda3/lib/python3.9/site-packages (from notebook>=4.4.1->widgetsnbextensio
n~=3.5.0->ipywidgets>=7.0.0->swifter) (1.8.0)
Requirement already satisfied: argon2-cffi in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from notebook>=4.4.1->widgetsnbextension~=3.
5.0->ipywidgets>=7.0.0->swifter) (21.3.0)
Requirement already satisfied: terminado>=0.8.3 in /Users/edsongomez/opt/ana
conda3/lib/python3.9/site-packages (from notebook>=4.4.1->widgetsnbextension
~=3.5.0->ipywidgets>=7.0.0->swifter) (0.13.1)
Requirement already satisfied: ptyprocess>=0.5 in /Users/edsongomez/opt/anac
onda3/lib/python3.9/site-packages (from pexpect>4.3->ipython>=4.0.0->ipywidg
ets>=7.0.0->swifter) (0.7.0)
Requirement already satisfied: wcwidth in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0
->ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (0.2.5)
Requirement already satisfied: testpath in /Users/edsongomez/opt/anaconda3/l
ib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbext
ension~=3.5.0->ipywidgets>=7.0.0->swifter) (0.6.0)
Requirement already satisfied: mistune<2,>=0.8.1 in /Users/edsongomez/opt/an
aconda3/lib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1->wid
getsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (0.8.4)
Requirement already satisfied: pandocfilters>=1.4.1 in /Users/edsongomez/op
t/anaconda3/lib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (1.5.0)
Requirement already satisfied: jupyterlab-pygments in /Users/edsongomez/opt/
```

```
anaconda3/lib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1->w
idgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (0.1.2)
Requirement already satisfied: defusedxml in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnb
extension~=3.5.0->ipywidgets>=7.0.0->swifter) (0.7.1)
Requirement already satisfied: beautifulsoup4 in /Users/edsongomez/opt/anaco
nda3/lib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1->widget
snbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (4.11.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /Users/edsongomez/o
pt/anaconda3/lib/python3.9/site-packages (from nbconvert>=5->notebook>=4.4.1
->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (0.5.13)
Requirement already satisfied: MarkupSafe>=0.23 in /Users/edsongomez/opt/ana
conda3/lib/python3.9/site-packages (from jinja2->notebook>=4.4.1->widgetsnbe
xtension~=3.5.0->ipywidgets>=7.0.0->swifter) (2.0.1)
Requirement already satisfied: argon2-cffi-bindings in /Users/edsongomez/op
t/anaconda3/lib/python3.9/site-packages (from argon2-cffi->notebook>=4.4.1->
widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (21.2.0)
Requirement already satisfied: cffi>=1.0.1 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from argon2-cffi-bindings->argon2-cffi->noteb
ook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (1.15.1)
Requirement already satisfied: soupsieve>1.2 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from beautifulsoup4->nbconvert>=5->notebook
>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifter) (2.3.1)
Requirement already satisfied: pycparser in /Users/edsongomez/opt/anaconda3/
lib/python3.9/site-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-
cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->swifte
r) (2.21)
Requirement already satisfied: pyspellchecker in /Users/edsongomez/opt/anaco
nda3/lib/python3.9/site-packages (0.7.1)
Requirement already satisfied: pandarallel in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (1.6.5)
Requirement already satisfied: psutil in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (from pandarallel) (5.9.0)
Requirement already satisfied: dill>=0.3.1 in /Users/edsongomez/opt/anaconda
3/lib/python3.9/site-packages (from pandarallel) (0.3.4)
Requirement already satisfied: pandas>=1 in /Users/edsongomez/opt/anaconda3/
lib/python3.9/site-packages (from pandarallel) (1.4.4)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/edsongomez/o
pt/anaconda3/lib/python3.9/site-packages (from pandas>=1->pandarallel) (2.8.
2)
Requirement already satisfied: pytz>=2020.1 in /Users/edsongomez/opt/anacond
a3/lib/python3.9/site-packages (from pandas>=1->pandarallel) (2022.1)
Requirement already satisfied: numpy>=1.18.5 in /Users/edsongomez/opt/anacon
da3/lib/python3.9/site-packages (from pandas>=1->pandarallel) (1.23.5)
Requirement already satisfied: six>=1.5 in /Users/edsongomez/opt/anaconda3/l
ib/python3.9/site-packages (from python-dateutil>=2.8.1->pandas>=1->pandaral
lel) (1.16.0)
Requirement already satisfied: nltk in /Users/edsongomez/opt/anaconda3/lib/p
ython3.9/site-packages (3.8.1)
Requirement already satisfied: tqdm in /Users/edsongomez/opt/anaconda3/lib/p
ython3.9/site-packages (from nltk) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in /Users/edsongomez/opt/anac
onda3/lib/python3.9/site-packages (from nltk) (2022.7.9)
Requirement already satisfied: click in /Users/edsongomez/opt/anaconda3/lib/
python3.9/site-packages (from nltk) (8.0.4)
```

```
Requirement already satisfied: joblib in /Users/edsongomez/opt/anaconda3/li
b/python3.9/site-packages (from nltk) (1.1.0)
```

In [ ]:
```
import os
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
os.chdir('/content/drive/My Drive/edson y wilfredo maestria/Proyecto de grad
print(os.getcwd())
print(os.listdir())
```

In [1]:
```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from pandarallel import pandarallel
pandarallel.initialize()
import swifter
from collections import Counter,defaultdict
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from spellchecker import SpellChecker
import multiprocessing
from fractions import Fraction
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
```

```
INFO: Pandarallel will run on 2 workers.
INFO: Pandarallel will use standard multiprocessing data transfer (pipe) to
transfer data between the main process and workers.
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/edsongomez/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

1. Carga de Base de Datos - Muestra de 590.592 Registros

In [2]:
```
data=pd.read_csv('MuestraBD600KPQRSD.csv', sep=";")
data
```

Out[2]:

	ID	ASUNTO	OBSERVACION	NombreDependencia
0	1	NaN	insatisfaccion por el pan no recibido	Sec_Admin
1	2	NaN	Queja por contaminación visual en la calle 45 ...	Sec_Admin
2	3	NaN	Prueba PQRS 05/10/2011	Sec_Admin
3	4	NaN	en la cra 5 Nº 57 - 59 barrio porvenir hay 2 l...	Sec_Infraest
4	6	NaN	Esto una solicitud	Sec_Admin
...	...	...	...	...
590587	696298	Formulario declaración de ICA	Registro de Presentación declaración de ICApar...	Sec_Hacienda
590588	696301	Formulario declaración de ICA	Registro de Presentación declaración de ICApar...	Sec_Hacienda
590589	696304	SOLICITUD DE OFERTA INSTITUCIÓNAL	Señores \r\nSecretaria de Desarrollo\r\nL. C \...	Sec_Des_social
590590	696305	Formulario declaración de ICA	Registro de Presentación declaración de ICApar...	Sec_Hacienda
590591	696306	Incompetentes espacio público	Viernes 9 de la noche e intento pasar con mis ...	Espacio_Publico

590592 rows × 4 columns

In [3]:
```
data.dtypes
```

Out[3]:
```
ID                   int64
ASUNTO              object
OBSERVACION         object
NombreDependencia   object
dtype: object
```

In [4]:
```
max(data.isnull().sum(axis=1))
```

Out[4]:
```
2
```

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
ID                      0
ASUNTO               9603
OBSERVACION        154010
NombreDependencia       0
dtype: int64
```

In [6]:
```
data=pd.read_csv('MuestraBD600KPQRSD.csv', sep=";",usecols=['ASUNTO', 'OBSER
data['ASUNTO']=data['ASUNTO'].str.lower()
data['OBSERVACION']=data['OBSERVACION'].str.lower()
```

2. Pre Procesamiento

2.1 Preprocesamiento sobre dos columnas del DataFrame, 'ASUNTO' y 'OBSERVACION', con el objetivo de limpiar y estandarizar los datos textuales. Este paso es crucial en cualquier proyecto que implique análisis de texto o procesamiento del lenguaje natural.

El código comienza definiendo una función lambda, que es una forma de crear una función pequeña y anónima en Python. En este caso, la función toma una cadena de texto `s` como entrada y devuelve una versión de la cadena en la que todos los espacios múltiples han sido reemplazados por un solo espacio. Esto se logra mediante el método `.split()` que divide la cadena en una lista de palabras y luego `.join()` que une estas palabras con un solo espacio entre ellas. Si `s` no existe (es decir, es None), la función simplemente devuelve una cadena vacía.

Después de definir la función, el código crea una copia del DataFrame original llamada `data_espacios` y establece `data` a None, liberando memoria. La copia se realiza para mantener los datos originales intactos y realizar las operaciones de preprocesamiento en la copia.

A continuación, el código aplica la función lambda previamente definida a las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame `data_espacios`. Esto se logra con el método `.apply()`, que aplica una función a cada elemento de una columna (o a cada fila, dependiendo del eje elegido). En este caso, se está usando para eliminar los espacios extra en las cadenas de estas columnas.

Por último, el código reemplaza los caracteres en las columnas 'ASUNTO' y 'OBSERVACION' que coinciden con los especificados en `chars_validos` usando expresiones regulares, que son patrones de texto que permiten realizar coincidencias y operaciones de sustitución complejas. La función `.replace()` se utiliza con el argumento `regex=True`, lo que significa que `chars_validos` es interpretado como una expresión regular. Dependiendo de lo que se haya definido en `chars_validos`, esta operación podría, por ejemplo, eliminar todos los caracteres no alfabéticos o todos los caracteres no ASCII. Sin embargo, como `chars_validos` no se define en el código proporcionado, su función exacta no se puede determinar.

In [7]:
```
chars_validos={
    '×':'x',
    'ß':'b',
    'à':'a',
    'á':'a',
    'â':'a',
    'ã':'a',
    'ä':'a',
    'å':'a',
    'ç':'c',
    'è':'e',
```

```
    'é':'e',
    'ê':'e',
    'ë':'e',
    'ì':'i',
    'í':'i',
    'î':'i',
    'ï':'i',
    'ò':'o',
    'ó':'o',
    'ô':'o',
    'õ':'o',
    'ö':'o',
    'ø':'o',
    'ù':'u',
    'ú':'u',
    'û':'u',
    'ü':'u',
    'ý':'y',
    'þ':'p',
    'ă':'a',
    'ą':'i',
    'Í':'l',
    'ń':'n',
    'ň':'ñ',
    'ŕ':'r',
    'š':'s',
    '':'l',
    'µ':'u',
    'ω':'w',
    '':'l',
    '':'l',
    '':'l',
    '':'p',
}
```

In [8]:

```
f=lambda s:' '.join(s.split()) if s else ''
data_espacios=data
data=None
data_espacios['ASUNTO']=data_espacios['ASUNTO'].apply(f)
data_espacios['OBSERVACION']=data_espacios['OBSERVACION'].apply(f)

data_espacios['ASUNTO']=data_espacios['ASUNTO'].replace(chars_validos,regex=
data_espacios['OBSERVACION']=data_espacios['OBSERVACION'].replace(chars_vali
```

2.2 Preprocesamiento de los datos, enfocándose en las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame. Su objetivo es eliminar cualquier carácter que no sea una letra (mayúscula o minúscula), un número o un espacio.

Primero, se crea una copia del DataFrame `data_espacios` llamada `data_letras` y luego se establece `data_espacios` a None. Esta es una estrategia de administración de memoria útil en Python para liberar la memoria que el DataFrame `data_espacios` estaba utilizando, dado que ya no es necesario.

Después, el código aplica la función `str.replace()` a las columnas 'ASUNTO' y 'OBSERVACION'. Esta función reemplaza todas las ocurrencias de una subcadena (o un patrón, en este caso) en una cadena. Aquí, la subcadena es una expresión regular, como se indica con el argumento `regex=True`. La expresión regular utilizada, `[^a-zA-Z0-9 ]`, corresponde a cualquier carácter que no sea una letra (mayúscula o minúscula), un número o un espacio. El acento circunflejo (^) al principio de la expresión regular indica la negación, es decir, se selecciona cualquier carácter que no esté en el conjunto. Así, todos los caracteres que no sean letras, números o espacios se reemplazan con la cadena de sustitución proporcionada, que en este caso es una cadena vacía (`''`). Por lo tanto, estos caracteres no deseados se eliminan de las cadenas en las columnas 'ASUNTO' y 'OBSERVACION'.

Como resultado, al final de este fragmento de código, las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame `data_letras` contendrán solo letras, números y espacios, con todos los demás caracteres eliminados.

In [9]:
```
data_letras=data_espacios
data_espacios=None
data_letras['ASUNTO']=data_letras['ASUNTO'].str.replace('[^a-zA-Z0-9 ]', '',
data_letras['OBSERVACION']=data_letras['OBSERVACION'].str.replace('[^a-zA-Z0
```

2.3 Eliminación de las palabras "stop words", que son palabras comunes que generalmente no contribuyen al significado de una frase, al menos en lo que respecta a la tarea de procesamiento de texto. Ejemplos en inglés son "is", "at", "which", y "on". En este caso, se están utilizando las palabras de parada en español.

Primero, se crea una nueva copia del DataFrame `data_letras` llamada `data_checker` y se establece `data_letras` a None para liberar memoria.

A continuación, se obtiene una lista de palabras de parada en español de la biblioteca NLTK (Natural Language Toolkit) con `stopwords.words('spanish')`. NLTK es una biblioteca en Python que proporciona herramientas para el procesamiento del lenguaje natural, incluyendo la identificación y eliminación de palabras de parada.

Luego, se define una función lambda que toma una cadena de texto `s`, la divide en palabras con `s.split()`, y luego une las palabras que no están en la lista de palabras de parada con un espacio. Esta función efectivamente elimina todas las palabras de parada en la cadena.

Por último, se aplica esta función a las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame utilizando el método `.swifter.apply()`. Swifter es una extensión de la biblioteca pandas que permite aplicar funciones a DataFrames de manera más eficiente.

Finalmente, se muestra el DataFrame `data_checker` que contiene las columnas 'ASUNTO' y 'OBSERVACION' con todas las letras convertidas a minúsculas, los caracteres no deseados eliminados, los espacios extra removidos y las palabras de parada eliminadas. Este DataFrame ahora está listo para el análisis de texto o el modelado de lenguaje natural.

In [10]:
```
data_checker=data_letras
data_letras=None
stop_words = stopwords.words('spanish')
stopw=lambda s:' '.join(w for w in s.split() if w not in stop_words)
data_checker['ASUNTO']=data_checker['ASUNTO'].swifter.apply(stopw)
data_checker['OBSERVACION']=data_checker['OBSERVACION'].swifter.apply(stopw)
data_checker
```

Out[10]:

2.4 Cálculo el número de palabras únicas en las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame `data_checker`.

In [ ]:
```
unique_words=set(' '.join(data_checker['ASUNTO'].values).split())
unique_words.update(' '.join(data_checker['OBSERVACION'].values).split())
len(unique_words)
```

Se determina el número de núcleos de CPU disponibles en la máquina donde se está ejecutando el código.

La biblioteca `multiprocessing` en Python proporciona una interfaz para aprovechar múltiples procesadores en una máquina. `multiprocessing.cpu_count()` devuelve el número de CPUs en el sistema.

El código intenta asignar la mitad del número total de CPUs al valor `cpus`. La división por 2 se realiza con el operador `//`, que es una división entera (es decir, descarta cualquier residuo y retorna solo la parte entera de la división). La idea es utilizar la mitad de los recursos disponibles de la CPU para mantener el rendimiento general del sistema y evitar posibles cuellos de botella de recursos.

Sin embargo, en algunos sistemas, `multiprocessing.cpu_count()` podría no ser compatible y arrojar un error `NotImplementedError`. Para manejar esta situación, se usa una estructura de control de excepciones `try/except`. Si `multiprocessing.cpu_count()` arroja un `NotImplementedError`, el código capturará esa excepción y asignará un valor predeterminado de 2 a `cpus`.

Finalmente, el código imprime el valor de `cpus`, que es el número de CPUs que el script pretende utilizar para el procesamiento paralelo.

In [ ]:
```
try:
    cpus = multiprocessing.cpu_count()//2
```

```
except NotImplementedError:
    cpus = 2   # arbitrary default
cpus
```

Este fragmento de código aplica la corrección ortográfica a todas las palabras únicas en el conjunto `unique_words` que se creó anteriormente. Se realiza utilizando la biblioteca SpellChecker, que es una biblioteca en Python para verificar la ortografía de palabras en varios idiomas.

Primero, se crea una instancia de SpellChecker con el idioma configurado como español (`'es'`).

Luego, se define una función llamada `correct()` que toma una cadena `s` y retorna la corrección ortográfica sugerida por SpellChecker para esa cadena.

A continuación, se crea un DataFrame vacío llamado `lookup`, que luego se utilizará para almacenar las palabras originales y sus correcciones ortográficas.

Después, se crea una Serie pandas `unknowns` que contiene todas las palabras en `unique_words` que SpellChecker no reconoce (es decir, palabras que pueden estar mal escritas o que simplemente no están en el diccionario de SpellChecker).

Luego, se aplica la función `correct()` a todas las palabras en `unknowns` utilizando el método `parallel_apply()`. Este método aplica una función a todos los elementos de una Serie o DataFrame de manera paralela, lo que puede acelerar significativamente el proceso si se dispone de múltiples núcleos de CPU.

A continuación, se agregan las palabras desconocidas y sus correcciones al DataFrame `lookup`.

Por último, se guarda el DataFrame `lookup` en un archivo CSV para referencia futura y luego se muestra el DataFrame. Así, este DataFrame `lookup` ahora contiene una correspondencia entre todas las palabras mal escritas o desconocidas en las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame y las correcciones ortográficas sugeridas para esas palabras.

In [ ]:
```
spell=SpellChecker(language='es')
def correct(s:str,spell=SpellChecker(language='es')):
  return spell.correction(s)

lookup =pd.DataFrame([])
unknowns=pd.Series(list(spell.unknown(list(unique_words))))
corrections=unknowns.parallel_apply(correct)
lookup['unknowns']=unknowns
lookup['corrections']=corrections
lookup.to_csv('lookup.csv',index=False)
lookup
```

In [ ]:
```
lookup=pd.read_csv('lookup.csv').dropna().reset_index(drop=True)
lookup
```

Se procesa aún más las correcciones de las palabras desconocidas. Las palabras que se encuentran en la lista de "stop words" son reemplazadas por cadenas vacías, y luego se crea un diccionario para mapear palabras desconocidas a sus correcciones. También se define un nuevo tipo de diccionario que devuelve la clave en lugar de un error si la clave no se encuentra en el diccionario.

Primero, se define una función lambda `stopw_correction` que toma una palabra `w` y devuelve la misma palabra si no está en la lista de "stop words", y una cadena vacía en caso contrario.

Luego, se aplica esta función a la columna 'corrections' del DataFrame `lookup` utilizando el método `swifter.apply()`. Esto elimina las palabras de parada de las correcciones.

Después, se crea un diccionario llamado `lookup_dict` que mapea las palabras desconocidas a sus correcciones. Esto se hace utilizando la función `zip()`, que toma dos listas y las combina en una lista de tuplas, y luego la función `dict()`, que convierte esta lista de tuplas en un diccionario.

A continuación, se define una clase personalizada `ReturnDefaultDict` que hereda de la clase `dict`. Esta clase redefine el método `__missing__()`, que se llama automáticamente cuando se intenta acceder a una clave que no existe en el diccionario. En lugar de lanzar un error, este método personalizado simplemente devuelve la clave.

Finalmente, se crea una instancia de `ReturnDefaultDict` llamada `correct_ddict` usando `lookup_dict` como argumento. Ahora, si intentas acceder a una clave que no está en `correct_ddict`, simplemente devolverá la clave. El objeto `correct_ddict` se muestra al final.

In [ ]:
```
stopw_correction=lambda w:w if w not in stop_words else ''

lookup['corrections']=lookup['corrections'].swifter.apply(stopw_correction)

lookup_dict=dict(zip(lookup['unknowns'].values,lookup['corrections'].values))

class ReturnDefaultDict(dict):
    def __missing__(self, key):
        return key

correct_ddict=ReturnDefaultDict(lookup_dict)
correct_ddict
```

2.5 Se corrige la ortografía de las palabras en las columnas 'ASUNTO' y 'OBSERVACION' del DataFrame `data_checker` utilizando el diccionario `correct_ddict` que se creó

anteriormente. Luego, combina las columnas 'ASUNTO' y 'OBSERVACION' en una nueva columna llamada 'input'.

Primero, se crea una copia del DataFrame `data_checker` y se llama `data_corregido`.

Luego, se define una función lambda `correct_words` que toma una cadena `s`, la divide en palabras con `s.split()`, busca cada palabra en el diccionario `correct_ddict` para obtener la corrección ortográfica, y luego une las palabras corregidas en una cadena con `join()`. Si la cadena `s` está vacía, la función simplemente devuelve una cadena vacía.

Después, se crea una nueva columna en `data_corregido` llamada 'input' que es la concatenación de las columnas 'ASUNTO' y 'OBSERVACION', con un espacio añadido al final.

A continuación, se aplica la función `correct_words` a la columna 'input' utilizando el método `swifter.apply()`. Esto corrige la ortografía de todas las palabras en la columna 'input'.

Las líneas comentadas de código sugieren que originalmente se aplicó la corrección de palabras a las columnas 'ASUNTO' y 'OBSERVACION' individualmente, pero este paso parece haber sido reemplazado por la creación de la columna 'input' y su corrección.

Por último, se muestra el DataFrame `data_corregido`. Ahora contiene una columna 'input' que es la concatenación de las columnas 'ASUNTO' y 'OBSERVACION' con todas las palabras corregidas ortográficamente.

In [ ]:
```
data_corregido=data_checker.copy()
#data_checker=None

correct_words=lambda s:' '.join(correct_ddict[w] for w in s.split()) if s el

data_corregido['input']=data_corregido['ASUNTO']+' '+data_corregido['OBSERVA
data_corregido['input']=data_corregido['input'].swifter.apply(correct_words)
#data_corregido['ASUNTO']=data_corregido['ASUNTO'].swifter.apply(correct_wor
#data_corregido['OBSERVACION']=data_corregido['OBSERVACION'].swifter.apply(c
data_corregido
```

In [ ]:
```
deps=data_corregido.groupby('NombreDependencia').count()['ASUNTO'].sort_valu
deps
```

3. Transformaciones finales en el DataFrame `data_corregido`.

Primero, se crea un diccionario `dep_extra` que asigna ciertos valores en la columna 'NombreDependencia' a la cadena 'Extra'. Esto se hace para todas las dependencias que

aparecen menos de 300 veces en el DataFrame, como indica la expresión `deps[deps<300].index`. Esto puede ser útil para reducir la cantidad de categorías únicas en 'NombreDependencia', agrupando las menos frecuentes en una sola categoría 'Extra'.

Luego, se reemplazan estos nombres de dependencia en la columna 'NombreDependencia' del DataFrame `data_corregido` con 'Extra' utilizando el método `replace()` con `dep_extra` como argumento.

Después, se crea una instancia de la clase `SnowballStemmer` del NLTK con el idioma configurado en español. La derivación es un proceso en el que se reducen las palabras a su raíz o forma base, lo que puede ayudar a consolidar diferentes formas de la misma palabra en una sola entidad.

Luego, se define una función lambda `stemm` que toma una cadena `s` y retorna la forma derivada de la cadena.

A continuación, se recrea la columna 'input' que es la concatenación de las columnas 'ASUNTO' y 'OBSERVACION', con un espacio añadido al final. Parece que se olvidaron de aplicar la función `stemm` en este paso, ya que se define pero nunca se utiliza.

Por último, se muestra la columna 'input' del DataFrame `data_corregido`. En este punto, 'input' contiene las columnas 'ASUNTO' y 'OBSERVACION' concatenadas con todas las palabras corregidas ortográficamente. Sin embargo, parece que se olvidaron de aplicar la derivación.

In [ ]:
```
dep_extra={d:'Extra' for d in deps[deps<300].index}
data_corregido['NombreDependencia']=data_corregido['NombreDependencia'].repl
stemmer = SnowballStemmer("spanish")
stemm=lambda s:stemmer.stem(s)
data_corregido['input']=data_corregido['ASUNTO']+' '+data_corregido['OBSERVA
data_corregido['input']
```

In [ ]:
```
data_corregido.to_csv('data_corregido.csv',sep=';',index=False)
```

4. Pruebas Algoritmo de Derivación "Stem"

Este bloque de código construye un diccionario que mapea cada palabra única en la columna 'input' a su raíz (o forma básica) utilizando el algoritmo de derivación. Luego, se define una función para aplicar esta transformación a cualquier texto.

Primero, se crea una lista vacía `v`. Luego, se recorren todos los valores en la columna 'input' del DataFrame `data_corregido`. Cada valor es una cadena de texto que se divide en palabras individuales con el método `split()`, y estas palabras se agregan a la lista `v`.

A continuación, se crea un diccionario de comprensión que toma el conjunto de palabras en `v` (para eliminar duplicados), aplica la función de derivación `stemm` a cada palabra, y mapea la palabra original a su forma derivada.

Después, se convierte este diccionario en un `ReturnDefaultDict`, que es el tipo de diccionario personalizado que se definió antes. Esto significa que si intentas acceder a una palabra que no está en el diccionario, simplemente devuelve la palabra original. Esto puede ser útil para evitar errores cuando se procesan nuevos textos que pueden contener palabras que no estaban en el conjunto de datos original.

Por último, se define una función lambda `correct_stemm` que toma una cadena de texto, la divide en palabras con `split()`, busca cada palabra en el diccionario `v` para obtener su forma derivada, y luego une las palabras derivadas en una cadena con `join()`. Esta función puede ser utilizada para transformar cualquier texto, reemplazando cada palabra por su forma derivada.

In [ ]:
```
v=[]
for s in data_corregido['input'].values:
  v+=s.split()
v={w:stemm(w) for w in set(v)}
v=ReturnDefaultDict(v)
correct_stemm=lambda s:' '.join(v[w] for w in s.split())
```

In [ ]:
```
data_corregido['input']=data_corregido['input'].swifter.apply(correct_stemm)
data_corregido['input']
```

In [ ]:
```
data_corregido.to_csv('data_corregida_stem.csv',index=False)
```

In [ ]:
```
data_corregido=pd.read_csv('data_corregida_stem.csv')
data_corregido
```

Este bloque de código crea un diccionario que mapea cada dependencia única en la columna 'NombreDependencia' a una lista de palabras ponderadas que aparecen más frecuentemente en las entradas de esa dependencia. Esto se hace utilizando un subconjunto de los datos para entrenamiento, y los pesos se calculan a partir de las 100 palabras más comunes en cada dependencia.

Primero, se crea un diccionario vacío `pesos_dict` donde cada clave es una dependencia única en 'NombreDependencia' y cada valor es una lista vacía.

Luego, se crea un subconjunto `dat` del DataFrame `data_corregido` que solo contiene las columnas 'input' y 'NombreDependencia' y se descartan las filas con valores nulos.

Después, se guarda la longitud del DataFrame `data_corregido` en `n`.

A continuación, se crea un conjunto de entrenamiento `train` que contiene el 80% de las filas de `dat`, seleccionadas al azar, y un conjunto de pruebas `test` que contiene el 20% restante.

Después, se recorren todas las filas en el conjunto de entrenamiento. Para cada fila, se añaden todas las palabras en 'input' a la lista correspondiente en `pesos_dict` para la dependencia de esa fila.

Luego, se recorren todas las dependencias en `pesos_dict`. Para cada dependencia, se calculan las 100 palabras más comunes en la lista de palabras de esa dependencia y se guarda su frecuencia en un objeto Counter `c`. Luego, se calcula la suma total `t` de todas las frecuencias en `c`.

Por último, se reemplaza la lista de palabras de cada dependencia en `pesos_dict` con un defaultdict que mapea cada una de las 100 palabras más comunes a su peso, que es la fracción de la frecuencia de esa palabra sobre la suma total `t`. Un defaultdict es un tipo de diccionario que proporciona un valor predeterminado para las claves que no se encuentran en el diccionario. En este caso, el valor predeterminado es la fracción 0.

Por lo tanto, `pesos_dict` ahora mapea cada dependencia a un diccionario de palabras ponderadas basado en su frecuencia relativa en las entradas de esa dependencia. Este diccionario se muestra al final.

In [ ]:
```
pesos_dict={dep:[] for dep in set(data_corregido['NombreDependencia'].unique
dat=data_corregido[['input','NombreDependencia']].dropna()
n=len(data_corregido)
train=dat.sample(frac=0.8,random_state=7)
test=dat.drop(train.index)
for input,dependencia in train[['input','NombreDependencia']].values:
  pesos_dict[dependencia]+=input.split()
for key in pesos_dict.keys():
  c=Counter(pesos_dict[key]).most_common(100)
  t=sum(val for _,val in c)
  pesos_dict[key]=defaultdict(lambda :Fraction(0),{key:Fraction(val,t) for k

pesos_dict
```

La función `eval` toma una entrada (en forma de texto) y devuelve la dependencia que, según el modelo de frecuencia de palabras ponderadas, es la más probable dada esa entrada.

Primero, divide la entrada en palabras con el método `split()`.

Luego, para cada dependencia en `pesos_dict`, suma los pesos de las palabras en la entrada que corresponden a esa dependencia. Los pesos se obtienen del diccionario de palabras ponderadas en `pesos_dict` para esa dependencia. Esto resulta en un diccionario `deps_score` donde cada clave es la suma total de los pesos de las palabras y cada valor es la dependencia correspondiente.

Finalmente, busca la clave más grande en `deps_score`, que es la suma de pesos más alta, y devuelve la dependencia correspondiente. En otras palabras, la función devuelve la

dependencia que tiene la suma de pesos más alta para las palabras en la entrada.

En resumen, esta función utiliza un modelo de "bolsa de palabras" ponderadas para predecir la dependencia más probable para una entrada dada. La ponderación se realiza de acuerdo a la frecuencia de aparición de cada palabra en las entradas de cada dependencia en el conjunto de entrenamiento. Es un método simple pero efectivo para la clasificación de texto cuando no se dispone de un conjunto de datos de gran tamaño.

In [ ]:
```
def eval(input):
  words=input.split()
  deps_score={sum(pesos_dict[k][w] for w in words):k for k in pesos_dict.key
  return deps_score[max(deps_score.keys())]
```

In [ ]:
```
test['evaluation']=test['input'].swifter.apply(eval)
test
```

In [ ]:
```
accuracy=sum(test['NombreDependencia']==test['evaluation'])/len(test)
accuracy
```

Un accuracy del 57% entre 20 categorias (la probabilidad de acertar aleatoriamente es de 5%)

In [ ]:
```
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(ngram_range=(1,2))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

In [ ]:
```
encoded_train=tfIdfVectorizer.transform(train['input'].values)
encoded_test=tfIdfVectorizer.transform(test['input'].values)
encoded_train
```

In [ ]:
```
clf = RandomForestClassifier(random_state=7)
clf.fit(encoded_train, train['NombreDependencia'].values)
```

In [ ]:
```

```

Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

Modelos técnicas tradicionales

Estudiantes

Wilfredo Ariel Góme Bueno

Edson Andrés Gómez Cárdenas

1. Instalación y carga de Librerias

In [1]:
```
!pip install swifter
!pip install pyspellchecker
!pip install pandarallel --upgrade
!pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab
-wheels/public/simple/
Collecting swifter
  Downloading swifter-1.3.4.tar.gz (830 kB)
     ▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏▏ 830.9/830.9 KB 21.3 MB/s eta 0:0
0:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.9/dis
t-packages (from swifter) (1.4.4)
Requirement already satisfied: psutil>=5.6.6 in /usr/local/lib/python3.9/dis
t-packages (from swifter) (5.9.4)
Requirement already satisfied: dask[dataframe]>=2.10.0 in /usr/local/lib/pyt
hon3.9/dist-packages (from swifter) (2022.12.1)
Requirement already satisfied: tqdm>=4.33.0 in /usr/local/lib/python3.9/dist
-packages (from swifter) (4.65.0)
Requirement already satisfied: ipywidgets>=7.0.0 in /usr/local/lib/python3.
9/dist-packages (from swifter) (7.7.1)
Requirement already satisfied: cloudpickle>=0.2.2 in /usr/local/lib/python3.
9/dist-packages (from swifter) (2.2.1)
Requirement already satisfied: parso>0.4.0 in /usr/local/lib/python3.9/dist-
packages (from swifter) (0.8.3)
Requirement already satisfied: bleach>=3.1.1 in /usr/local/lib/python3.9/dis
t-packages (from swifter) (6.0.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.9/dist-p
ackages (from bleach>=3.1.1->swifter) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist
-packages (from bleach>=3.1.1->swifter) (0.5.1)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.9/dist-p
ackages (from dask[dataframe]>=2.10.0->swifter) (8.1.3)
Requirement already satisfied: partd>=0.3.10 in /usr/local/lib/python3.9/dis
t-packages (from dask[dataframe]>=2.10.0->swifter) (1.3.0)
Requirement already satisfied: fsspec>=0.6.0 in /usr/local/lib/python3.9/dis
t-packages (from dask[dataframe]>=2.10.0->swifter) (2023.3.0)
Requirement already satisfied: toolz>=0.8.2 in /usr/local/lib/python3.9/dist
-packages (from dask[dataframe]>=2.10.0->swifter) (0.12.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.9/dis
t-packages (from dask[dataframe]>=2.10.0->swifter) (6.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/d
ist-packages (from dask[dataframe]>=2.10.0->swifter) (23.0)
Requirement already satisfied: numpy>=1.18 in /usr/local/lib/python3.9/dist-
packages (from dask[dataframe]>=2.10.0->swifter) (1.22.4)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/p
ython3.9/dist-packages (from ipywidgets>=7.0.0->swifter) (3.0.6)
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/p
ython3.9/dist-packages (from ipywidgets>=7.0.0->swifter) (3.6.3)
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/pyt
hon3.9/dist-packages (from ipywidgets>=7.0.0->swifter) (0.2.0)
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.9/di
st-packages (from ipywidgets>=7.0.0->swifter) (7.9.0)
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.9/
dist-packages (from ipywidgets>=7.0.0->swifter) (5.7.1)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.9/
dist-packages (from ipywidgets>=7.0.0->swifter) (5.3.4)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/pyth
on3.9/dist-packages (from pandas>=1.0.0->swifter) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist
```

```
-packages (from pandas>=1.0.0->swifter) (2022.7.1)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.9/di
st-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->swifter) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.9/dist
-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->swifter) (6.2)
Requirement already satisfied: pexpect in /usr/local/lib/python3.9/dist-pack
ages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (4.8.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.9/dist-pa
ckages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (4.4.2)
Collecting jedi>=0.10
  Downloading jedi-0.18.2-py2.py3-none-any.whl (1.6 MB)
     ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ 1.6/1.6 MB 34.6 MB/s eta 0:00:
00
Requirement already satisfied: backcall in /usr/local/lib/python3.9/dist-pac
kages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (0.2.0)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.9/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (0.7.5)
Requirement already satisfied: pygments in /usr/local/lib/python3.9/dist-pac
kages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (2.14.0)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.9/
dist-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (67.6.0)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /usr/local/li
b/python3.9/dist-packages (from ipython>=4.0.0->ipywidgets>=7.0.0->swifter)
(2.0.10)
Requirement already satisfied: locket in /usr/local/lib/python3.9/dist-packa
ges (from partd>=0.3.10->dask[dataframe]>=2.10.0->swifter) (1.0.0)
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.9/d
ist-packages (from widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter)
(6.3.0)
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.9/
dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=
7.0.0->swifter) (0.17.1)
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.
9/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets
>=7.0.0->swifter) (0.16.0)
Requirement already satisfied: Send2Trash>=1.5.0 in /usr/local/lib/python3.
9/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets
>=7.0.0->swifter) (1.8.0)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.9/dist-
packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0
->swifter) (21.3.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-pa
ckages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->
swifter) (6.5.4)
Requirement already satisfied: jupyter-core>=4.6.1 in /usr/local/lib/python
3.9/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidge
ts>=7.0.0->swifter) (5.3.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packa
ges (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swi
fter) (3.1.2)
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.9/dist-pa
ckages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->
swifter) (23.2.1)
Requirement already satisfied: nbformat in /usr/local/lib/python3.9/dist-pac
kages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->s
wifter) (5.8.0)
```

```
Requirement already satisfied: wcwidth in /usr/local/lib/python3.9/dist-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (0.2.6)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.9/dist-packages (from pexpect->ipython>=4.0.0->ipywidgets>=7.0.0->swifter) (0.7.0)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.9/dist-packages (from jupyter-core>=4.6.1->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (3.1.1)
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.9/dist-packages (from argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (21.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (2.1.2)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (1.2.1)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (0.4)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (1.5.0)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (0.2.2)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (0.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (0.7.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (4.11.2)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (4.9.2)
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packages (from nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (2.16.3)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-packages (from nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (4.3.3)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (22.2.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.9/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (0.19.3)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.9/dist-
```

```
packages (from argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1->widgetsnb
extension~=3.6.0->ipywidgets>=7.0.0->swifter) (1.15.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dis
t-packages (from beautifulsoup4->nbconvert->notebook>=4.4.1->widgetsnbextens
ion~=3.6.0->ipywidgets>=7.0.0->swifter) (2.4)
Requirement already satisfied: pycparser in /usr/local/lib/python3.9/dist-pa
ckages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1
->widgetsnbextension~=3.6.0->ipywidgets>=7.0.0->swifter) (2.21)
Building wheels for collected packages: swifter
  Building wheel for swifter (setup.py) ... done
  Created wheel for swifter: filename=swifter-1.3.4-py3-none-any.whl size=16
321 sha256=97c103b88e49dc158349113e276c1637f317da7680a3fc8c3b3d38bfd620b067
  Stored in directory: /root/.cache/pip/wheels/2b/5e/f2/3931524f702ffd03309e
96d35ee2fbf9c61c27377511ee8d4c
Successfully built swifter
Installing collected packages: jedi, swifter
Successfully installed jedi-0.18.2 swifter-1.3.4
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab
-wheels/public/simple/
Collecting pyspellchecker
  Downloading pyspellchecker-0.7.1-py3-none-any.whl (2.5 MB)
     ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ 2.5/2.5 MB 21.7 MB/s eta 0:00:
00
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.7.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab
-wheels/public/simple/
Collecting pandarallel
  Downloading pandarallel-1.6.4.tar.gz (12 kB)
  Preparing metadata (setup.py) ... done
Collecting dill>=0.3.1
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
     ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ 110.5/110.5 KB 6.1 MB/s eta 0:0
0:00
Requirement already satisfied: pandas>=1 in /usr/local/lib/python3.9/dist-pa
ckages (from pandarallel) (1.4.4)
Requirement already satisfied: psutil in /usr/local/lib/python3.9/dist-packa
ges (from pandarallel) (5.9.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist
-packages (from pandas>=1->pandarallel) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/pyth
on3.9/dist-packages (from pandas>=1->pandarallel) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dis
t-packages (from pandas>=1->pandarallel) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-pac
kages (from python-dateutil>=2.8.1->pandas>=1->pandarallel) (1.16.0)
Building wheels for collected packages: pandarallel
  Building wheel for pandarallel (setup.py) ... done
  Created wheel for pandarallel: filename=pandarallel-1.6.4-py3-none-any.whl
size=16677 sha256=df3fd5ed4df0a5ecd4340419a50dd491248adee13742a8385ca0cec275
688f42
  Stored in directory: /root/.cache/pip/wheels/41/01/29/deaa71fe596f8d857e57
c4fb388db8861e23e6ed0b03204dcb
Successfully built pandarallel
Installing collected packages: dill, pandarallel
Successfully installed dill-0.3.6 pandarallel-1.6.4
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab
-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.9/dist-package
s (3.8.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/d
ist-packages (from nltk) (2022.10.31)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packa
ges (from nltk) (1.1.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-package
s (from nltk) (4.65.0)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packag
es (from nltk) (8.1.3)
```

In [2]:
```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from pandarallel import pandarallel
pandarallel.initialize()
import swifter
from collections import Counter,defaultdict
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from spellchecker import SpellChecker
import multiprocessing
from fractions import Fraction
from sklearn.feature_extraction.text import TfidfVectorizer
from joblib import dump, load
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB, ComplementNB
from sklearn.linear_model import LogisticRegression
```

```
INFO: Pandarallel will run on 1 workers.
INFO: Pandarallel will use Memory file system to transfer data between the m
ain process and workers.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

2. Importación Data Preprocesada

In [ ]:
```
data_corregido=pd.read_csv('data_corregida.csv')
data_corregido
```

Out[ ]:

	ASUNTO	OBSERVACION	NombreDependencia	input
0	NaN	insatisfaccion pan recibido	Sec_Admin	insatisfaccion pan recibido
1	NaN	queja contaminacion visual calle 45 carrera 13...	Sec_Admin	queja contaminacion visual calle 45 carrera 1...
2	NaN	prueba pqrs 05102011	Sec_Admin	prueba pqrs 05102011
3	NaN	cra 5 n 57 59 barrio porvenir 2 luminarias fun...	Sec_Infraest	cra 5 n 57 59 barrio porvenir 2 luminarias fu...
4	NaN	solicitud	Sec_Admin	solicitud
...	...	...	...	...
590587	formulario declaracion ica	registro presentacion declaracion icapara ao g...	Sec_Hacienda	formulario declaracion ica registro presentaci...
590588	formulario declaracion ica	registro presentacion declaracion icapara ao g...	Sec_Hacienda	formulario declaracion ica registro presentaci...
590589	solicitud oferta institucional	seores secretaria desarrollo l c reciban cordi...	Sec_Des_social	solicitud oferta institucional seores secretar...
590590	formulario declaracion ica	registro presentacion declaracion icapara ao g...	Sec_Hacienda	formulario declaracion ica registro presentaci...
590591	incompetentes espacio publico	viernes 9 noche intento pasar hijos anden carr...	Espacio_Publico	incompetentes espacio publico viernes 9 noche ...

590592 rows × 4 columns

3. Analisis de Distibución de las Categorias

En la primera línea, se está agrupando el DataFrame por la columna 'NombreDependencia' y contando el número de ocurrencias de cada valor único en esa columna. El resultado, extras, es una Serie de pandas donde el índice es el nombre de la dependencia y el valor es el conteo de esa dependencia.

En la segunda línea, se está creando un diccionario donde las claves son los nombres de las dependencias que tienen menos de 1000 ocurrencias en el DataFrame. El valor para cada una de estas claves es la cadena de texto 'Extra'.

Por lo tanto, el resultado final es un diccionario llamado extras donde cada dependencia con menos de 1000 ocurrencias se mapea a la cadena 'Extra'. Esto podría ser útil si estás interesado en agrupar todas las dependencias con menos de 1000 ocurrencias en una categoría 'Extra' para futuros análisis.

In [ ]:
```
extras=data_corregido.groupby('NombreDependencia')['NombreDependencia'].coun
extras={d:'Extra' for d in extras[(extras<1000).values].index}
extras
```

Out[ ]:
```
{'Direccion': 'Extra',
 'Prensa_y_Comuni': 'Extra',
 'ST': 'Extra',
 'Ser_Publicos': 'Extra'}
```

In [ ]:
```
data_corregido['NombreDependencia']=data_corregido['NombreDependencia'].repl
data_corregido.groupby('NombreDependencia')['NombreDependencia'].count().sor
```

Out[ ]:
```
NombreDependencia
Ofc_TIC               1331
Con_Int_Discipli      1466
Pla_Sisben            2900
Extra                 3854
Espacio_Publico       7126
Sec_Educacion         7238
Sec_Juridica         10693
Valorizacion         13238
Des_Alcalde          15058
Sec_Infraest         20585
Sec_Des_social       22173
Sec_Planeacion       35339
Sec_Salud_y_ambi     41269
Sec_Admin            43680
Sec_Interior         47370
Sec_Hacienda        317272
Name: NombreDependencia, dtype: int64
```

Resultado

Se percibe un dataset desbalanceado en la categoria Secretaria de Hacienda con cerca de un 670% de volumen de frecuencias de registros por encima de la segunda categoria

4. Modelo Bolsa de Palabras por Frecuencia

In [ ]:
```
pesos_dict={dep:[] for dep in set(data_corregido['NombreDependencia'].unique
dat=data_corregido[['input','NombreDependencia']].dropna()
n=len(data_corregido)
train=dat.sample(frac=0.8,random_state=7)
test=dat.drop(train.index)
for input,dependencia in train[['input','NombreDependencia']].values:
  pesos_dict[dependencia]+=input.split()
for key in pesos_dict.keys():
  c=Counter(pesos_dict[key]).most_common(100)
  t=sum(val for _,val in c)
  pesos_dict[key]=defaultdict(lambda :Fraction(0),{key:Fraction(val,t) for k

pesos_dict
```

Out[ ]:
```
{'Sec_Salud_y_ambi': defaultdict(<function __main__.<lambda>()>,
                    {'solicitud': Fraction(17529, 268544),
                     'salud': Fraction(12167, 268544),
                     'concepto': Fraction(4787, 134272),
                     'bucaramanga': Fraction(9415, 268544),
                     'visita': Fraction(9059, 268544),
                     'sanitario': Fraction(9043, 268544),
                     'secretaria': Fraction(1603, 67136),
                     'gov': Fraction(4123, 268544),
                     'co': Fraction(4115, 268544),
                     'policia': Fraction(2033, 134272),
                     'respuesta': Fraction(227, 16784),
                     'informacion': Fraction(3625, 268544),
                     'calle': Fraction(3599, 268544),
                     'dia': Fraction(1745, 134272),
                     'establecimiento': Fraction(1707, 134272),
                     'atencion': Fraction(3331, 268544),
                     'peticion': Fraction(1597, 134272),
                     'ppl': Fraction(1549, 134272),
                     'parte': Fraction(3085, 268544),
                     'derecho': Fraction(3083, 268544),
                     'solicito': Fraction(2989, 268544),
                     'personas': Fraction(2951, 268544),
                     'barrio': Fraction(1457, 134272),
                     'correo': Fraction(2777, 268544),
                     'presente': Fraction(171, 16784),
                     'medio': Fraction(85, 8392),
                     'certificado': Fraction(2575, 268544),
                     'inspeccion': Fraction(2565, 268544),
                     'exhumacion': Fraction(637, 67136),
                     '2': Fraction(629, 67136),
                     '1': Fraction(2495, 268544),
                     'dias': Fraction(2455, 268544),
                     'santander': Fraction(1217, 134272),
                     'gracias': Fraction(581, 67136),
                     'nacional': Fraction(2287, 268544),
                     'saludo': Fraction(2247, 268544),
                     'ambiente': Fraction(2243, 268544),
                     'fin': Fraction(2241, 268544),
                     'sanitaria': Fraction(1105, 134272),
                     'estacion': Fraction(2199, 268544),
                     'cordial': Fraction(2197, 268544),
                     'carrera': Fraction(2131, 268544),
                     'condiciones': Fraction(2087, 268544),
                     'si': Fraction(2083, 268544),
                     'mas': Fraction(2073, 268544),
                     'manera': Fraction(251, 33568),
                     'favor': Fraction(2005, 268544),
                     'seores': Fraction(497, 67136),
                     'centro': Fraction(489, 67136),
                     'ubicado': Fraction(243, 33568),
                     'envio': Fraction(1939, 268544),
                     'gestion': Fraction(473, 67136),
                     'ley': Fraction(1879, 268544),
                     'permito': Fraction(925, 134272),
                     'radicado': Fraction(919, 134272),
```

```
                 'caso': Fraction(459, 67136),
                 'municipal': Fraction(111, 16784),
                 'adjunto': Fraction(883, 134272),
                 'alcaldia': Fraction(877, 134272),
                 'direccion': Fraction(1739, 268544),
                 'queja': Fraction(869, 134272),
                 'cordialmente': Fraction(431, 67136),
                 'agradezco': Fraction(1719, 268544),
                 'eps': Fraction(427, 67136),
                 '2021': Fraction(1701, 268544),
                 '3': Fraction(849, 134272),
                 'ruido': Fraction(1691, 268544),
                 'solicitar': Fraction(845, 134272),
                 's': Fraction(419, 67136),
                 'ambiental': Fraction(1667, 268544),
                 'asi': Fraction(831, 134272),
                 'control': Fraction(413, 67136),
                 'cuenta': Fraction(1613, 268544),
                 'cumplimiento': Fraction(1607, 268544),
                 'fecha': Fraction(1563, 268544),
                 'realizar': Fraction(1543, 268544),
                 'aos': Fraction(385, 67136),
                 'servicios': Fraction(1507, 268544),
                 '19': Fraction(1505, 268544),
                 'buenas': Fraction(47, 8392),
                 'entidades': Fraction(749, 134272),
                 'numero': Fraction(747, 134272),
                 'comercial': Fraction(1469, 268544),
                 'c': Fraction(727, 134272),
                 'notificacion': Fraction(1451, 268544),
                 'situacion': Fraction(357, 67136),
                 'libertad': Fraction(1425, 268544),
                 '2020': Fraction(1423, 268544),
                 'custodia': Fraction(349, 67136),
                 'articulo': Fraction(343, 67136),
                 'ciudad': Fraction(1367, 268544),
                 'discapacidad': Fraction(1347, 268544),
                 'covid': Fraction(335, 67136),
                 'atentamente': Fraction(661, 134272),
                 'ustedes': Fraction(1317, 268544),
                 'asunto': Fraction(1313, 268544),
                 'documento': Fraction(653, 134272),
                 'registro': Fraction(651, 134272),
                 'estan': Fraction(1301, 268544),
                 'traslado': Fraction(645, 134272)}),
 'Sec_Admin': defaultdict(<function __main__.<lambda>()>,
             {'s': Fraction(7263, 60583),
              'bucaramanga': Fraction(18935, 242332),
              'empresa': Fraction(30849, 484664),
              'aseo': Fraction(30417, 484664),
              'factura': Fraction(29157, 484664),
              'p': Fraction(14461, 242332),
              'electronica': Fraction(16261, 484664),
              'total': Fraction(15363, 484664),
              '804006674': Fraction(3777, 121166),
              'n': Fraction(15015, 484664),
```

```
'notificacion': Fraction(14847, 484664),
'01': Fraction(3693, 121166),
'expedicion': Fraction(14557, 484664),
'venta': Fraction(7261, 242332),
'generacion': Fraction(1796, 60583),
'solicitud': Fraction(920, 60583),
'informacion': Fraction(512, 60583),
'peticion': Fraction(3419, 484664),
'alcaldia': Fraction(3309, 484664),
'derecho': Fraction(365, 60583),
'respuesta': Fraction(699, 121166),
'correo': Fraction(1129, 242332),
'fecha': Fraction(1129, 242332),
'co': Fraction(2057, 484664),
'dia': Fraction(2023, 484664),
'si': Fraction(1951, 484664),
'favor': Fraction(967, 242332),
'saludo': Fraction(1897, 484664),
'gov': Fraction(943, 242332),
'nota': Fraction(939, 242332),
'00': Fraction(1865, 484664),
'presente': Fraction(1863, 484664),
'dias': Fraction(231, 60583),
'esp': Fraction(923, 242332),
'1': Fraction(915, 242332),
'cordial': Fraction(1821, 484664),
'credito': Fraction(1813, 484664),
'sa': Fraction(449, 121166),
'c': Fraction(1729, 484664),
'calle': Fraction(215, 60583),
'2': Fraction(1697, 484664),
'solicito': Fraction(35, 10312),
'atencion': Fraction(817, 242332),
'siguiente': Fraction(1619, 484664),
'2020': Fraction(1619, 484664),
'gracias': Fraction(403, 121166),
'medio': Fraction(1563, 484664),
'certificacion': Fraction(1531, 484664),
'secretaria': Fraction(1509, 484664),
'servicio': Fraction(749, 242332),
'laboral': Fraction(731, 242332),
'adjunto': Fraction(365, 121166),
'municipio': Fraction(1449, 484664),
'direccion': Fraction(175, 60583),
'articulo': Fraction(345, 121166),
'administrativo': Fraction(1347, 484664),
'manera': Fraction(667, 242332),
'proceso': Fraction(1329, 484664),
'carrera': Fraction(166, 60583),
'certificado': Fraction(331, 121166),
'envio': Fraction(162, 60583),
'municipal': Fraction(1279, 484664),
'codigo': Fraction(1279, 484664),
'ley': Fraction(639, 242332),
'cordialmente': Fraction(637, 242332),
'santander': Fraction(317, 121166),
```

```
                'permito': Fraction(1243, 484664),
                'cuenta': Fraction(619, 242332),
                'agradezco': Fraction(1221, 484664),
                'numero': Fraction(609, 242332),
                'cargo': Fraction(150, 60583),
                'mas': Fraction(591, 242332),
                'documentos': Fraction(147, 60583),
                'nacional': Fraction(1173, 484664),
                'permiso': Fraction(1171, 484664),
                'vida': Fraction(1169, 484664),
                'servicios': Fraction(1143, 484664),
                '2021': Fraction(1139, 484664),
                'entidad': Fraction(1123, 484664),
                'administrativa': Fraction(555, 242332),
                'pago': Fraction(277, 121166),
                'ciudad': Fraction(271, 121166),
                'solicitar': Fraction(23, 10312),
                '91': Fraction(1077, 484664),
                'sindical': Fraction(1069, 484664),
                'barrio': Fraction(1049, 484664),
                'trabajo': Fraction(523, 242332),
                'com': Fraction(261, 121166),
                'publico': Fraction(1041, 484664),
                'parte': Fraction(1037, 484664),
                'seores': Fraction(1023, 484664),
                '3': Fraction(1021, 484664),
                'electronico': Fraction(1017, 484664),
                'oficio': Fraction(1007, 484664),
                'emision': Fraction(123, 60583),
                'fin': Fraction(489, 242332),
                'decreto': Fraction(949, 484664),
                'seor': Fraction(118, 60583),
                'atentamente': Fraction(941, 484664),
                'grado': Fraction(939, 484664)}),
    'Des_Alcalde': defaultdict(<function __main__.<lambda>()>,
                {'bucaramanga': Fraction(655, 13119),
                'solicitud': Fraction(1022, 21865),
                'peticion': Fraction(2341, 65595),
                'informacion': Fraction(2138, 65595),
                'alcalde': Fraction(1997, 65595),
                'derecho': Fraction(1993, 65595),
                'seor': Fraction(230, 13119),
                'respuesta': Fraction(1079, 65595),
                'invitacion': Fraction(215, 13119),
                'envio': Fraction(347, 21865),
                'co': Fraction(1039, 65595),
                'mas': Fraction(1018, 65595),
                'gov': Fraction(991, 65595),
                'alcaldia': Fraction(961, 65595),
                'si': Fraction(316, 21865),
                'dia': Fraction(923, 65595),
                'ciudad': Fraction(175, 13119),
                'santander': Fraction(169, 13119),
                'saludo': Fraction(149, 13119),
                'usted': Fraction(718, 65595),
                'nacional': Fraction(694, 65595),
```

```
'cordial': Fraction(133, 13119),
'atencion': Fraction(44, 4373),
'trabajo': Fraction(644, 65595),
'carlos': Fraction(628, 65595),
'correo': Fraction(203, 21865),
'municipal': Fraction(203, 21865),
'juan': Fraction(604, 65595),
'gracias': Fraction(198, 21865),
'2': Fraction(198, 21865),
'personas': Fraction(193, 21865),
'medio': Fraction(574, 65595),
'reunion': Fraction(569, 65595),
'ley': Fraction(189, 21865),
'direccion': Fraction(562, 65595),
'1': Fraction(183, 21865),
'2020': Fraction(109, 13119),
'publico': Fraction(36, 4373),
'copia': Fraction(536, 65595),
'favor': Fraction(107, 13119),
'manera': Fraction(177, 21865),
'calle': Fraction(523, 65595),
'3': Fraction(518, 65595),
'barrio': Fraction(518, 65595),
'ser': Fraction(517, 65595),
'municipio': Fraction(103, 13119),
'presente': Fraction(34, 4373),
'informe': Fraction(496, 65595),
'colombia': Fraction(491, 65595),
'general': Fraction(484, 65595),
'com': Fraction(161, 21865),
'asunto': Fraction(472, 65595),
'c': Fraction(157, 21865),
'comunicacion': Fraction(469, 65595),
'cardenas': Fraction(469, 65595),
'dias': Fraction(156, 21865),
'parte': Fraction(463, 65595),
'aos': Fraction(461, 65595),
'proyecto': Fraction(152, 21865),
'pico': Fraction(449, 65595),
'vida': Fraction(448, 65595),
'solo': Fraction(446, 65595),
'cuenta': Fraction(437, 65595),
'social': Fraction(433, 65595),
'cordialmente': Fraction(431, 65595),
'desarrollo': Fraction(85, 13119),
'gestion': Fraction(424, 65595),
'carrera': Fraction(422, 65595),
'asi': Fraction(422, 65595),
'transporte': Fraction(28, 4373),
'secretaria': Fraction(139, 21865),
'hacer': Fraction(413, 65595),
'fecha': Fraction(409, 65595),
'articulo': Fraction(136, 21865),
'agradezco': Fraction(407, 65595),
'servicio': Fraction(407, 65595),
'seguridad': Fraction(406, 65595),
```

```
                'oficio': Fraction(403, 65595),
                '2021': Fraction(403, 65595),
                'atentamente': Fraction(133, 21865),
                'salud': Fraction(398, 65595),
                'apoyo': Fraction(394, 65595),
                '2022': Fraction(386, 65595),
                '5': Fraction(127, 21865),
                'doctor': Fraction(127, 21865),
                'proceso': Fraction(379, 65595),
                'ao': Fraction(374, 65595),
                'junta': Fraction(373, 65595),
                's': Fraction(123, 21865),
                'ayuda': Fraction(122, 21865),
                'buenas': Fraction(73, 13119),
                'radicado': Fraction(364, 65595),
                '4': Fraction(121, 21865),
                'poder': Fraction(359, 65595),
                'publica': Fraction(358, 65595),
                'permito': Fraction(358, 65595),
                'adjunto': Fraction(119, 21865),
                'placa': Fraction(352, 65595),
                'puede': Fraction(70, 13119),
                'n': Fraction(349, 65595)}),
 'Con_Int_Discipli': defaultdict(<function __main__.<lambda>()>,
             {'solicitud': Fraction(333, 8779),
              'bucaramanga': Fraction(307, 8779),
              'informacion': Fraction(248, 8779),
              'respuesta': Fraction(233, 8779),
              'queja': Fraction(196, 8779),
              'disciplinaria': Fraction(179, 8779),
              'investigacion': Fraction(178, 8779),
              'disciplinario': Fraction(174, 8779),
              'proceso': Fraction(174, 8779),
              'control': Fraction(168, 8779),
              'rad': Fraction(160, 8779),
              'secretaria': Fraction(130, 8779),
              'alcaldia': Fraction(130, 8779),
              'interno': Fraction(128, 8779),
              'peticion': Fraction(125, 8779),
              'expediente': Fraction(123, 8779),
              'oficio': Fraction(122, 8779),
              'radicado': Fraction(122, 8779),
              'si': Fraction(118, 8779),
              'derecho': Fraction(117, 8779),
              'seor': Fraction(113, 8779),
              'cpa': Fraction(112, 8779),
              'remision': Fraction(107, 8779),
              'oficina': Fraction(103, 8779),
              'solicito': Fraction(101, 8779),
              'poder': Fraction(97, 8779),
              'correo': Fraction(94, 8779),
              'dia': Fraction(88, 8779),
              'parte': Fraction(88, 8779),
              'gov': Fraction(87, 8779),
              'co': Fraction(87, 8779),
              'planeacion': Fraction(85, 8779),
```

```
'competencia': Fraction(84, 8779),
'saludo': Fraction(84, 8779),
'ley': Fraction(83, 8779),
'funcionarios': Fraction(83, 8779),
'fecha': Fraction(81, 8779),
'procuraduria': Fraction(79, 8779),
'cordial': Fraction(77, 8779),
'1': Fraction(77, 8779),
'rdo': Fraction(76, 8779),
'seores': Fraction(74, 8779),
'favor': Fraction(74, 8779),
'2': Fraction(74, 8779),
'entidad': Fraction(72, 8779),
'comunicacion': Fraction(71, 8779),
'manera': Fraction(71, 8779),
'ser': Fraction(70, 8779),
'copia': Fraction(70, 8779),
'denuncia': Fraction(70, 8779),
'funcionario': Fraction(69, 8779),
'cuales': Fraction(69, 8779),
'asunto': Fraction(68, 8779),
'general': Fraction(68, 8779),
'numero': Fraction(68, 8779),
'3': Fraction(67, 8779),
'c': Fraction(67, 8779),
'personeria': Fraction(66, 8779),
'publico': Fraction(65, 8779),
'presente': Fraction(64, 8779),
'asi': Fraction(64, 8779),
'articulo': Fraction(64, 8779),
'adjunto': Fraction(64, 8779),
'dentro': Fraction(63, 8779),
'pruebas': Fraction(62, 8779),
'envio': Fraction(62, 8779),
'planos': Fraction(62, 8779),
'2021': Fraction(61, 8779),
'policia': Fraction(61, 8779),
'referencia': Fraction(59, 8779),
'5': Fraction(58, 8779),
'atentamente': Fraction(58, 8779),
'atencion': Fraction(58, 8779),
'municipal': Fraction(58, 8779),
'calle': Fraction(58, 8779),
'n': Fraction(55, 8779),
'permito': Fraction(54, 8779),
'auto': Fraction(53, 8779),
'anterior': Fraction(53, 8779),
'medidas': Fraction(52, 8779),
'certificado': Fraction(52, 8779),
'hacer': Fraction(52, 8779),
'disciplinarios': Fraction(50, 8779),
'seora': Fraction(50, 8779),
'caso': Fraction(49, 8779),
'electronico': Fraction(47, 8779),
'cumplimiento': Fraction(47, 8779),
'administrativa': Fraction(47, 8779),
```

```
                'ciudadana': Fraction(47, 8779),
                'cuenta': Fraction(47, 8779),
                'conocimiento': Fraction(46, 8779),
                'medio': Fraction(46, 8779),
                'estan': Fraction(45, 8779),
                '4': Fraction(45, 8779),
                'docente': Fraction(44, 8779),
                'agradezco': Fraction(44, 8779),
                'corrupcion': Fraction(44, 8779),
                'administracion': Fraction(44, 8779),
                'fin': Fraction(43, 8779),
                'nacional': Fraction(43, 8779)}),
 'Sec_Des_social': defaultdict(<function __main__.<lambda>()>,
             {'solicitud': Fraction(403, 6112),
              'ayuda': Fraction(1999, 48896),
              'bucaramanga': Fraction(289, 9168),
              'programa': Fraction(3809, 146688),
              'mayor': Fraction(1805, 73344),
              'informacion': Fraction(383, 18336),
              'social': Fraction(1411, 73344),
              'adulto': Fraction(2759, 146688),
              'respuesta': Fraction(2539, 146688),
              'discapacidad': Fraction(2521, 146688),
              'peticion': Fraction(2461, 146688),
              'barrio': Fraction(2309, 146688),
              'desarrollo': Fraction(1105, 73344),
              'accion': Fraction(183, 12224),
              'aos': Fraction(1091, 73344),
              'comunal': Fraction(1997, 146688),
              'mercado': Fraction(485, 36672),
              'derecho': Fraction(1841, 146688),
              'gracias': Fraction(907, 73344),
              'junta': Fraction(1727, 146688),
              'servicio': Fraction(1711, 146688),
              'secretaria': Fraction(71, 6112),
              'jac': Fraction(71, 6112),
              'favor': Fraction(1687, 146688),
              'alcaldia': Fraction(277, 24448),
              'envio': Fraction(545, 48896),
              'si': Fraction(33, 3056),
              'dia': Fraction(65, 6112),
              'solicito': Fraction(1535, 146688),
              'atencion': Fraction(761, 73344),
              'seor': Fraction(725, 73344),
              'agradezco': Fraction(473, 48896),
              'trabajo': Fraction(85, 9168),
              'apoyo': Fraction(113, 12224),
              'presente': Fraction(221, 24448),
              'saludo': Fraction(637, 73344),
              'centro': Fraction(79, 9168),
              'medio': Fraction(311, 36672),
              'institucional': Fraction(103, 12224),
              'buenas': Fraction(411, 48896),
              'vida': Fraction(1225, 146688),
              'calle': Fraction(305, 36672),
              'cordial': Fraction(1193, 146688),
```

```
'oferta': Fraction(393, 48896),
'correo': Fraction(293, 36672),
'numero': Fraction(1163, 146688),
'ayudas': Fraction(385, 48896),
'gobierno': Fraction(377, 48896),
'2': Fraction(1093, 146688),
'1': Fraction(359, 48896),
'parte': Fraction(355, 48896),
'situacion': Fraction(131, 18336),
'familia': Fraction(1039, 146688),
'dias': Fraction(173, 24448),
'radicado': Fraction(1037, 146688),
'documentos': Fraction(1015, 146688),
'colaboracion': Fraction(1007, 146688),
'cuenta': Fraction(1007, 146688),
'santander': Fraction(325, 48896),
'ustedes': Fraction(325, 48896),
'persona': Fraction(313, 48896),
'adjunto': Fraction(935, 146688),
'madre': Fraction(931, 146688),
'solicitar': Fraction(923, 146688),
'manera': Fraction(461, 73344),
'tardes': Fraction(115, 18336),
'edad': Fraction(303, 48896),
'proceso': Fraction(303, 48896),
'mas': Fraction(301, 48896),
'ley': Fraction(299, 48896),
'ser': Fraction(223, 36672),
'gov': Fraction(877, 146688),
'salud': Fraction(877, 146688),
'sisben': Fraction(73, 12224),
'c': Fraction(109, 18336),
'alimentaria': Fraction(145, 24448),
'ninguna': Fraction(9, 1528),
'personas': Fraction(859, 146688),
'caso': Fraction(141, 24448),
'bono': Fraction(141, 24448),
'cupo': Fraction(281, 48896),
'entrega': Fraction(419, 73344),
'co': Fraction(279, 48896),
'seores': Fraction(833, 146688),
'2021': Fraction(137, 24448),
'3': Fraction(409, 73344),
'permito': Fraction(805, 146688),
'bienestar': Fraction(1, 192),
'elecciones': Fraction(253, 48896),
'san': Fraction(757, 146688),
'paciente': Fraction(755, 146688),
'hijos': Fraction(377, 73344),
'recibido': Fraction(749, 146688),
'cordialmente': Fraction(373, 73344),
'dos': Fraction(247, 48896),
'asunto': Fraction(733, 146688),
'rehabilitacion': Fraction(61, 12224),
'ingreso': Fraction(181, 36672),
'basica': Fraction(721, 146688),
```

```
              'cedula': Fraction(179, 36672)}),
 'Ofc_TIC': defaultdict(<function __main__.<lambda>()>,
             {'informacion': Fraction(367, 10751),
              'bucaramanga': Fraction(362, 10751),
              'solicitud': Fraction(302, 10751),
              'alcaldia': Fraction(241, 10751),
              'correo': Fraction(240, 10751),
              'pago': Fraction(216, 10751),
              'gracias': Fraction(205, 10751),
              'com': Fraction(196, 10751),
              'digital': Fraction(194, 10751),
              'si': Fraction(188, 10751),
              'pagina': Fraction(187, 10751),
              'predial': Fraction(186, 10751),
              'impuesto': Fraction(179, 10751),
              'co': Fraction(178, 10751),
              'respuesta': Fraction(172, 10751),
              'favor': Fraction(163, 10751),
              'peticion': Fraction(151, 10751),
              'internet': Fraction(147, 10751),
              'servicio': Fraction(147, 10751),
              'direccion': Fraction(142, 10751),
              'datos': Fraction(140, 10751),
              'dia': Fraction(134, 10751),
              'gov': Fraction(125, 10751),
              'saludo': Fraction(124, 10751),
              'vive': Fraction(122, 10751),
              'cordial': Fraction(118, 10751),
              'realizar': Fraction(113, 10751),
              'pagar': Fraction(112, 10751),
              'atencion': Fraction(109, 10751),
              'derecho': Fraction(109, 10751),
              'agradezco': Fraction(108, 10751),
              'mas': Fraction(106, 10751),
              'c': Fraction(105, 10751),
              'cuenta': Fraction(8, 827),
              'hacer': Fraction(102, 10751),
              'web': Fraction(96, 10751),
              'envio': Fraction(95, 10751),
              'sistema': Fraction(94, 10751),
              'medio': Fraction(93, 10751),
              '1': Fraction(93, 10751),
              'plataforma': Fraction(93, 10751),
              'numero': Fraction(92, 10751),
              'electronico': Fraction(92, 10751),
              'linea': Fraction(7, 827),
              'colombia': Fraction(90, 10751),
              'punto': Fraction(90, 10751),
              'poder': Fraction(90, 10751),
              'presente': Fraction(89, 10751),
              'nacional': Fraction(87, 10751),
              'solicito': Fraction(86, 10751),
              'dias': Fraction(86, 10751),
              'usuario': Fraction(85, 10751),
              'bogota': Fraction(82, 10751),
              'plus': Fraction(82, 10751),
```

```
                 'buen': Fraction(81, 10751),
                 's': Fraction(81, 10751),
                 'empresa': Fraction(80, 10751),
                 '2': Fraction(79, 10751),
                 '3': Fraction(79, 10751),
                 'www': Fraction(79, 10751),
                 'contrasea': Fraction(79, 10751),
                 'cordialmente': Fraction(6, 827),
                 'tic': Fraction(77, 10751),
                 'publico': Fraction(77, 10751),
                 'muchas': Fraction(76, 10751),
                 '2020': Fraction(76, 10751),
                 'acceso': Fraction(76, 10751),
                 'error': Fraction(76, 10751),
                 'parte': Fraction(75, 10751),
                 'wifi': Fraction(74, 10751),
                 'manera': Fraction(73, 10751),
                 'ao': Fraction(73, 10751),
                 'proceso': Fraction(72, 10751),
                 'traves': Fraction(72, 10751),
                 'secretaria': Fraction(70, 10751),
                 'prueba': Fraction(69, 10751),
                 'pvd': Fraction(69, 10751),
                 'seores': Fraction(68, 10751),
                 'fecha': Fraction(68, 10751),
                 'buenas': Fraction(5, 827),
                 'puede': Fraction(5, 827),
                 'ustedes': Fraction(64, 10751),
                 'barrio': Fraction(64, 10751),
                 'colaboracion': Fraction(63, 10751),
                 'adjunto': Fraction(63, 10751),
                 'asunto': Fraction(63, 10751),
                 'servicios': Fraction(63, 10751),
                 'software': Fraction(62, 10751),
                 '30': Fraction(61, 10751),
                 'calle': Fraction(60, 10751),
                 'recibo': Fraction(60, 10751),
                 'tecnologia': Fraction(59, 10751),
                 'computador': Fraction(59, 10751),
                 'clave': Fraction(58, 10751),
                 'n': Fraction(58, 10751),
                 'tecnico': Fraction(58, 10751),
                 'portal': Fraction(58, 10751),
                 'buenos': Fraction(58, 10751),
                 'https': Fraction(57, 10751),
                 'codigo': Fraction(56, 10751)}),
 'Valorizacion': defaultdict(<function __main__.<lambda>()>,
             {'solicitud': Fraction(1070, 13797),
              'levantamiento': Fraction(2306, 32193),
              'valorizacion': Fraction(6508, 96579),
              'inenajenabilidad': Fraction(3833, 96579),
              'bucaramanga': Fraction(1046, 32193),
              'predio': Fraction(139, 4599),
              '0674': Fraction(2696, 96579),
              'resolucion': Fraction(2222, 96579),
              'matricula': Fraction(1, 49),
```

```
'anotacion': Fraction(275, 13797),
'101013': Fraction(1810, 96579),
'numero': Fraction(1678, 96579),
'medida': Fraction(1426, 96579),
'solicito': Fraction(157, 10731),
'cancelacion': Fraction(148, 10731),
'paz': Fraction(443, 32193),
'gravamen': Fraction(1307, 96579),
'oficina': Fraction(1223, 96579),
'inmueble': Fraction(1181, 96579),
'salvo': Fraction(1154, 96579),
'pago': Fraction(1123, 96579),
'predial': Fraction(160, 13797),
'res': Fraction(1112, 96579),
'inmobiliaria': Fraction(1030, 96579),
'certificado': Fraction(977, 96579),
'oficio': Fraction(884, 96579),
'ubicado': Fraction(841, 96579),
'c': Fraction(832, 96579),
'peticion': Fraction(92, 10731),
'tradicion': Fraction(827, 96579),
'calle': Fraction(764, 96579),
'n': Fraction(254, 32193),
'vial': Fraction(106, 13797),
'plan': Fraction(713, 96579),
'creacion': Fraction(704, 96579),
'adjunto': Fraction(691, 96579),
'libertad': Fraction(227, 32193),
'presente': Fraction(680, 96579),
'instrumentos': Fraction(223, 32193),
'publicos': Fraction(95, 13797),
'derecho': Fraction(662, 96579),
'impuesto': Fraction(214, 32193),
'registro': Fraction(641, 96579),
'edificio': Fraction(71, 10731),
'carrera': Fraction(619, 96579),
'respuesta': Fraction(67, 10731),
'agradezco': Fraction(8, 1323),
'direccion': Fraction(193, 32193),
'informacion': Fraction(577, 96579),
'municipal': Fraction(554, 96579),
'solicitar': Fraction(551, 96579),
'barrio': Fraction(530, 96579),
'identificado': Fraction(529, 96579),
'recibo': Fraction(172, 32193),
'cedula': Fraction(172, 32193),
'medio': Fraction(506, 96579),
'apto': Fraction(167, 32193),
'dia': Fraction(499, 96579),
'fecha': Fraction(488, 96579),
'alcaldia': Fraction(23, 4599),
'1': Fraction(475, 96579),
'correo': Fraction(467, 96579),
'gracias': Fraction(458, 96579),
'folio': Fraction(151, 32193),
'favor': Fraction(64, 13797),
```

```
                'competitiva': Fraction(1, 219),
                '2': Fraction(439, 96579),
                '10102013': Fraction(433, 96579),
                'cautelar': Fraction(430, 96579),
                'ciudad': Fraction(61, 13797),
                'propietario': Fraction(422, 96579),
                '10': Fraction(139, 32193),
                'saludo': Fraction(404, 96579),
                'catastral': Fraction(19, 4599),
                '2013': Fraction(8, 1971),
                'refinanciacion': Fraction(382, 96579),
                'cordial': Fraction(127, 32193),
                'realizar': Fraction(377, 96579),
                'atencion': Fraction(377, 96579),
                'notaria': Fraction(125, 32193),
                'tramite': Fraction(374, 96579),
                'nombre': Fraction(121, 32193),
                'radicado': Fraction(361, 96579),
                'apartamento': Fraction(358, 96579),
                'seores': Fraction(356, 96579),
                'cobro': Fraction(355, 96579),
                'colaboracion': Fraction(118, 32193),
                'cc': Fraction(349, 96579),
                'permito': Fraction(116, 32193),
                'encuentra': Fraction(347, 96579),
                'envio': Fraction(346, 96579),
                'municipio': Fraction(346, 96579),
                'manera': Fraction(16, 4599),
                'levantar': Fraction(335, 96579),
                'cuenta': Fraction(328, 96579),
                'mejoramiento': Fraction(109, 32193),
                'octubre': Fraction(46, 13797),
                'movilidad': Fraction(5, 1533),
                'dias': Fraction(314, 96579),
                'predios': Fraction(311, 96579)}),
    'Extra': defaultdict(<function __main__.<lambda>()>,
                {'servicio': Fraction(299, 9575),
                 'bucaramanga': Fraction(808, 28725),
                 'solicita': Fraction(644, 28725),
                 'informacion': Fraction(643, 28725),
                 'solicitud': Fraction(128, 5745),
                 'empresa': Fraction(211, 9575),
                 'essa': Fraction(581, 28725),
                 'energia': Fraction(569, 28725),
                 'electrica': Fraction(556, 28725),
                 'vanti': Fraction(107, 5745),
                 'mes': Fraction(33, 1915),
                 'metros': Fraction(487, 28725),
                 'peticion': Fraction(484, 28725),
                 'dia': Fraction(154, 9575),
                 'prestamo': Fraction(151, 9575),
                 'trabajos': Fraction(437, 28725),
                 'respuesta': Fraction(433, 28725),
                 'cuenta': Fraction(427, 28725),
                 'derecho': Fraction(131, 9575),
                 'responder': Fraction(131, 9575),
```

```
'5': Fraction(379, 28725),
'00': Fraction(337, 28725),
'dias': Fraction(111, 9575),
'grupo': Fraction(331, 28725),
'cubicos': Fraction(108, 9575),
'porfavor': Fraction(316, 28725),
'si': Fraction(21, 1915),
'alcaldia': Fraction(62, 5745),
'direccion': Fraction(292, 28725),
'cancha': Fraction(292, 28725),
'barrio': Fraction(284, 28725),
'calidad': Fraction(92, 9575),
'encarguese': Fraction(11, 1149),
'favor': Fraction(11, 1149),
'ciudad': Fraction(271, 28725),
'fin': Fraction(88, 9575),
'mayo': Fraction(88, 9575),
'2': Fraction(86, 9575),
'30': Fraction(257, 28725),
'cobro': Fraction(17, 1915),
'servicios': Fraction(247, 28725),
'consumo': Fraction(241, 28725),
'lectura': Fraction(241, 28725),
'presenta': Fraction(47, 5745),
'club': Fraction(233, 28725),
'solo': Fraction(233, 28725),
'ley': Fraction(232, 28725),
'facturacion': Fraction(229, 28725),
'factura': Fraction(229, 28725),
'6': Fraction(226, 28725),
'mas': Fraction(221, 28725),
'inderbu': Fraction(221, 28725),
'1': Fraction(44, 5745),
'gracias': Fraction(44, 5745),
'recrear': Fraction(73, 9575),
'apoyo': Fraction(218, 28725),
'acuerdo': Fraction(217, 28725),
'2020': Fraction(72, 9575),
'com': Fraction(43, 5745),
'fecha': Fraction(214, 28725),
'sectores': Fraction(211, 28725),
'3': Fraction(14, 1915),
'4': Fraction(208, 28725),
'personas': Fraction(206, 28725),
'atencion': Fraction(41, 5745),
'calle': Fraction(68, 9575),
'7': Fraction(203, 28725),
'deportivo': Fraction(8, 1149),
'manera': Fraction(197, 28725),
'municipal': Fraction(194, 28725),
'dos': Fraction(194, 28725),
'solicito': Fraction(191, 28725),
'permiso': Fraction(191, 28725),
'puedan': Fraction(63, 9575),
'usuarios': Fraction(63, 9575),
'subsidio': Fraction(37, 5745),
```

```
                'publicos': Fraction(37, 5745),
                'mantenimiento': Fraction(184, 28725),
                'realizar': Fraction(184, 28725),
                'correo': Fraction(179, 28725),
                'promedio': Fraction(179, 28725),
                'saludo': Fraction(178, 28725),
                'cordial': Fraction(176, 28725),
                'oficio': Fraction(176, 28725),
                'seores': Fraction(7, 1149),
                'personal': Fraction(7, 1149),
                '2019': Fraction(7, 1149),
                'abril': Fraction(58, 9575),
                'junio': Fraction(58, 9575),
                'parte': Fraction(173, 28725),
                '12': Fraction(173, 28725),
                'santander': Fraction(56, 9575),
                'meses': Fraction(56, 9575),
                'adjunto': Fraction(166, 28725),
                'transito': Fraction(11, 1915),
                'san': Fraction(164, 28725),
                'cada': Fraction(164, 28725),
                'hacer': Fraction(163, 28725),
                'co': Fraction(163, 28725),
                'carrera': Fraction(54, 9575)}),
 'Sec_Juridica': defaultdict(<function __main__.<lambda>()>,
             {'tutela': Fraction(4741, 85361),
              'bucaramanga': Fraction(4543, 85361),
              'accion': Fraction(4419, 85361),
              'gov': Fraction(3012, 85361),
              'co': Fraction(2934, 85361),
              'rad': Fraction(2678, 85361),
              'juzgado': Fraction(1492, 85361),
              'municipal': Fraction(1485, 85361),
              'correo': Fraction(1438, 85361),
              'civil': Fraction(1322, 85361),
              'presente': Fraction(1274, 85361),
              'notificacion': Fraction(1219, 85361),
              'secretaria': Fraction(1110, 85361),
              'proceso': Fraction(1104, 85361),
              'jzdo': Fraction(1098, 85361),
              'municipio': Fraction(1088, 85361),
              'solicitud': Fraction(1081, 85361),
              'derecho': Fraction(1027, 85361),
              'permito': Fraction(1010, 85361),
              'rdo': Fraction(980, 85361),
              'citacion': Fraction(975, 85361),
              'penal': Fraction(947, 85361),
              'radicado': Fraction(902, 85361),
              'oficio': Fraction(878, 85361),
              'auto': Fraction(873, 85361),
              'electronico': Fraction(872, 85361),
              's': Fraction(855, 85361),
              '2': Fraction(855, 85361),
              '2021': Fraction(850, 85361),
              'santander': Fraction(829, 85361),
              'seores': Fraction(827, 85361),
```

```
'1': Fraction(810, 85361),
'com': Fraction(781, 85361),
'c': Fraction(777, 85361),
'medio': Fraction(772, 85361),
'judicial': Fraction(770, 85361),
'termino': Fraction(769, 85361),
'administrativo': Fraction(769, 85361),
'referencia': Fraction(761, 85361),
'alcaldia': Fraction(737, 85361),
'popular': Fraction(712, 85361),
'conocimiento': Fraction(684, 85361),
'informacion': Fraction(674, 85361),
'fallo': Fraction(665, 85361),
'fecha': Fraction(659, 85361),
'dentro': Fraction(655, 85361),
'despacho': Fraction(652, 85361),
'dias': Fraction(650, 85361),
'ramajudicial': Fraction(638, 85361),
'favor': Fraction(605, 85361),
'2020': Fraction(604, 85361),
'articulo': Fraction(603, 85361),
'cordialmente': Fraction(603, 85361),
'nacional': Fraction(579, 85361),
'n': Fraction(575, 85361),
'mpal': Fraction(568, 85361),
'dia': Fraction(563, 85361),
'4': Fraction(530, 85361),
'ley': Fraction(529, 85361),
'3': Fraction(529, 85361),
'respuesta': Fraction(529, 85361),
'boleta': Fraction(525, 85361),
'peticion': Fraction(516, 85361),
'accionante': Fraction(514, 85361),
'seor': Fraction(508, 85361),
'00': Fraction(505, 85361),
'atentamente': Fraction(502, 85361),
'siguiente': Fraction(501, 85361),
'cto': Fraction(498, 85361),
'traves': Fraction(493, 85361),
'demanda': Fraction(490, 85361),
'conciliacion': Fraction(489, 85361),
'ser': Fraction(477, 85361),
'sentencia': Fraction(455, 85361),
'saludo': Fraction(452, 85361),
'manera': Fraction(443, 85361),
'oficina': Fraction(443, 85361),
'control': Fraction(440, 85361),
'cumplimiento': Fraction(438, 85361),
'desacato': Fraction(417, 85361),
'cordial': Fraction(414, 85361),
'traslado': Fraction(413, 85361),
'providencia': Fraction(410, 85361),
'funciones': Fraction(408, 85361),
'mediante': Fraction(400, 85361),
'parte': Fraction(399, 85361),
'envio': Fraction(397, 85361),
```

```
                'copia': Fraction(396, 85361),
                'decreto': Fraction(392, 85361),
                'asunto': Fraction(387, 85361),
                'circuito': Fraction(385, 85361),
                'general': Fraction(384, 85361),
                'numero': Fraction(384, 85361),
                'nulidad': Fraction(379, 85361),
                'maria': Fraction(379, 85361),
                'notificacionesbucaramanga': Fraction(377, 85361),
                'tramite': Fraction(373, 85361),
                'siguientes': Fraction(371, 85361),
                'contactenosbucaramanga': Fraction(370, 85361),
                'instancia': Fraction(367, 85361)}),
    'Sec_Infraest': defaultdict(<function __main__.<lambda>()>,
                {'solicitud': Fraction(3561, 60238),
                'bucaramanga': Fraction(4781, 120476),
                'peticion': Fraction(3659, 120476),
                'informacion': Fraction(770, 30119),
                'respuesta': Fraction(3053, 120476),
                'derecho': Fraction(2821, 120476),
                'barrio': Fraction(2687, 120476),
                'publico': Fraction(631, 30119),
                'calle': Fraction(2427, 120476),
                'correo': Fraction(2045, 120476),
                'infraestructura': Fraction(2033, 120476),
                'alumbrado': Fraction(497, 30119),
                'obra': Fraction(1887, 120476),
                'contrato': Fraction(911, 60238),
                'carrera': Fraction(417, 30119),
                'parque': Fraction(393, 30119),
                'envio': Fraction(759, 60238),
                'radicado': Fraction(1473, 120476),
                'sector': Fraction(1439, 120476),
                'alcaldia': Fraction(691, 60238),
                'co': Fraction(681, 60238),
                'secretaria': Fraction(1355, 120476),
                'via': Fraction(1315, 120476),
                'gov': Fraction(1309, 120476),
                'oficio': Fraction(1289, 120476),
                'n': Fraction(639, 60238),
                'saludo': Fraction(603, 60238),
                'presente': Fraction(1193, 120476),
                'cordial': Fraction(1169, 120476),
                'si': Fraction(285, 30119),
                '2': Fraction(569, 60238),
                'atencion': Fraction(1127, 120476),
                'gracias': Fraction(275, 30119),
                'cordialmente': Fraction(549, 60238),
                'municipio': Fraction(1095, 120476),
                'obras': Fraction(545, 60238),
                'copia': Fraction(1089, 120476),
                'mantenimiento': Fraction(539, 60238),
                'electronico': Fraction(1069, 120476),
                'adjunto': Fraction(1039, 120476),
                '1': Fraction(519, 60238),
                'mas': Fraction(1007, 120476),
```

```
's': Fraction(979, 120476),
'medio': Fraction(975, 120476),
'poda': Fraction(238, 30119),
'arreglo': Fraction(233, 30119),
'parte': Fraction(919, 120476),
'arboles': Fraction(223, 30119),
'favor': Fraction(445, 60238),
'3': Fraction(443, 60238),
'consorcio': Fraction(221, 30119),
'siguiente': Fraction(221, 30119),
'direccion': Fraction(435, 60238),
'dia': Fraction(217, 30119),
'ciudad': Fraction(216, 30119),
'solicito': Fraction(215, 30119),
'dias': Fraction(214, 30119),
'cdmb': Fraction(425, 60238),
'informe': Fraction(845, 120476),
'2021': Fraction(209, 30119),
'vial': Fraction(417, 60238),
'numero': Fraction(417, 60238),
'estan': Fraction(823, 120476),
'2022': Fraction(823, 120476),
'c': Fraction(817, 120476),
'traves': Fraction(200, 30119),
'manera': Fraction(199, 30119),
'asunto': Fraction(791, 120476),
'nacional': Fraction(779, 120476),
'cuenta': Fraction(757, 120476),
'enviada': Fraction(757, 120476),
'comunidad': Fraction(188, 30119),
'san': Fraction(725, 120476),
'construccion': Fraction(181, 30119),
'encuentra': Fraction(711, 120476),
'permito': Fraction(709, 120476),
'servicio': Fraction(175, 30119),
'proyecto': Fraction(695, 120476),
'cra': Fraction(685, 120476),
'comunicacion': Fraction(685, 120476),
'colaboracion': Fraction(171, 30119),
'municipal': Fraction(679, 120476),
'5': Fraction(335, 60238),
'comuna': Fraction(669, 120476),
'seores': Fraction(655, 120476),
'cualquier': Fraction(651, 120476),
'ley': Fraction(651, 120476),
'gestion': Fraction(315, 60238),
'santander': Fraction(157, 30119),
'fecha': Fraction(157, 30119),
'agradezco': Fraction(157, 30119),
'ser': Fraction(313, 60238),
'luminarias': Fraction(156, 30119),
'2020': Fraction(311, 60238),
'solicitar': Fraction(617, 120476),
'puede': Fraction(615, 120476),
'hace': Fraction(152, 30119),
'frente': Fraction(605, 120476),
```

```
              'seor': Fraction(148, 30119),
              'espacio': Fraction(295, 60238)}),
 'Sec_Planeacion': defaultdict(<function __main__.<lambda>()>,
             {'solicitud': Fraction(16094, 258693),
              'uso': Fraction(3305, 86231),
              'bucaramanga': Fraction(8603, 258693),
              'suelo': Fraction(7511, 258693),
              'planeacion': Fraction(1999, 86231),
              'peticion': Fraction(5276, 258693),
              'calle': Fraction(5227, 258693),
              'barrio': Fraction(4751, 258693),
              'respuesta': Fraction(4484, 258693),
              'derecho': Fraction(4481, 258693),
              'certificado': Fraction(4258, 258693),
              'informacion': Fraction(4253, 258693),
              'predio': Fraction(3998, 258693),
              'comercio': Fraction(3803, 258693),
              'construccion': Fraction(3731, 258693),
              'licencia': Fraction(3616, 258693),
              'secretaria': Fraction(3605, 258693),
              'direccion': Fraction(67, 4881),
              'carrera': Fraction(1176, 86231),
              'establecimiento': Fraction(1124, 86231),
              'publico': Fraction(1068, 86231),
              'solicito': Fraction(3145, 258693),
              'espacio': Fraction(2909, 258693),
              'visita': Fraction(2819, 258693),
              'correo': Fraction(2746, 258693),
              'x': Fraction(904, 86231),
              'si': Fraction(902, 86231),
              'concepto': Fraction(2701, 258693),
              'ubicado': Fraction(2654, 258693),
              '1': Fraction(2609, 258693),
              '2': Fraction(844, 86231),
              'adjunto': Fraction(2500, 258693),
              'presente': Fraction(823, 86231),
              'reconocimiento': Fraction(821, 86231),
              'numero': Fraction(2444, 258693),
              'permiso': Fraction(783, 86231),
              'obra': Fraction(780, 86231),
              'municipal': Fraction(2309, 258693),
              'riesgo': Fraction(760, 86231),
              'radicado': Fraction(2276, 258693),
              'registro': Fraction(2218, 258693),
              'alcaldia': Fraction(738, 86231),
              'comercial': Fraction(2158, 258693),
              'n': Fraction(2125, 258693),
              'solicitar': Fraction(703, 86231),
              'actividad': Fraction(691, 86231),
              'intervencion': Fraction(2068, 258693),
              'co': Fraction(2066, 258693),
              'agradezco': Fraction(670, 86231),
              'suelos': Fraction(1994, 258693),
              'gov': Fraction(1951, 258693),
              'parte': Fraction(645, 86231),
              'medio': Fraction(640, 86231),
```

```
                'dia': Fraction(627, 86231),
                'envio': Fraction(627, 86231),
                'gracias': Fraction(1873, 258693),
                'saludo': Fraction(1853, 258693),
                'notificacion': Fraction(1850, 258693),
                'articulo': Fraction(612, 86231),
                'cordial': Fraction(601, 86231),
                'atencion': Fraction(598, 86231),
                'industria': Fraction(576, 86231),
                'documentos': Fraction(1720, 258693),
                'acto': Fraction(1702, 258693),
                'ley': Fraction(559, 86231),
                'c': Fraction(1648, 258693),
                'seores': Fraction(31, 4881),
                '3': Fraction(1609, 258693),
                'favor': Fraction(1585, 258693),
                'casa': Fraction(1573, 258693),
                'oficio': Fraction(1553, 258693),
                'permito': Fraction(1540, 258693),
                'viabilidad': Fraction(1529, 258693),
                'cambio': Fraction(507, 86231),
                'realizar': Fraction(1510, 258693),
                'cordialmente': Fraction(484, 86231),
                'manera': Fraction(1433, 258693),
                'municipio': Fraction(476, 86231),
                'estan': Fraction(467, 86231),
                'siguiente': Fraction(461, 86231),
                'tramite': Fraction(1379, 258693),
                'dias': Fraction(1364, 258693),
                's': Fraction(454, 86231),
                'piso': Fraction(1297, 258693),
                'planos': Fraction(1280, 258693),
                'colaboracion': Fraction(425, 86231),
                'mas': Fraction(425, 86231),
                'proyecto': Fraction(1273, 258693),
                'zona': Fraction(1255, 258693),
                '2020': Fraction(417, 86231),
                'cuenta': Fraction(1249, 258693),
                'santander': Fraction(413, 86231),
                'copia': Fraction(1234, 258693),
                '10': Fraction(1232, 258693),
                'ciudad': Fraction(1226, 258693),
                'predial': Fraction(408, 86231),
                'alto': Fraction(1208, 258693),
                'acuerdo': Fraction(1204, 258693),
                'ustedes': Fraction(400, 86231),
                'hacer': Fraction(1198, 258693)}),
    'Pla_Sisben': defaultdict(<function __main__.<lambda>()>,
                {'sisben': Fraction(251, 1908),
                 'puntaje': Fraction(57, 1484),
                 'solicitud': Fraction(127, 3339),
                 'bucaramanga': Fraction(793, 26712),
                 'visita': Fraction(103, 4452),
                 'peticion': Fraction(523, 26712),
                 'respuesta': Fraction(1, 53),
                 'encuesta': Fraction(143, 8904),
```

```
'favor': Fraction(1, 63),
'gracias': Fraction(52, 3339),
'derecho': Fraction(29, 1908),
'solicito': Fraction(199, 13356),
'agradezco': Fraction(191, 13356),
'correo': Fraction(169, 13356),
'informacion': Fraction(109, 8904),
'necesito': Fraction(9, 742),
'numero': Fraction(313, 26712),
'nueva': Fraction(311, 26712),
'buenas': Fraction(299, 26712),
'dia': Fraction(295, 26712),
'dias': Fraction(4, 371),
'solicitar': Fraction(47, 4452),
'salud': Fraction(34, 3339),
'si': Fraction(271, 26712),
'datos': Fraction(67, 6678),
'alto': Fraction(131, 13356),
'cedula': Fraction(65, 6678),
'familiar': Fraction(1, 106),
'c': Fraction(247, 26712),
'hacer': Fraction(241, 26712),
'ayuda': Fraction(10, 1113),
'atencion': Fraction(10, 1113),
'aos': Fraction(229, 26712),
'tardes': Fraction(25, 2968),
'buenos': Fraction(223, 26712),
'cc': Fraction(221, 26712),
'nucleo': Fraction(5, 636),
'poder': Fraction(103, 13356),
'presente': Fraction(29, 3816),
'cambio': Fraction(67, 8904),
'colaboracion': Fraction(25, 3339),
'documentos': Fraction(193, 26712),
'hija': Fraction(3, 424),
'asunto': Fraction(47, 6678),
'documento': Fraction(187, 26712),
'adjunto': Fraction(187, 26712),
'saber': Fraction(31, 4452),
'revision': Fraction(31, 4452),
'mas': Fraction(23, 3339),
'pronta': Fraction(181, 26712),
'medio': Fraction(179, 26712),
'urgente': Fraction(22, 3339),
'inclusion': Fraction(22, 3339),
'ustedes': Fraction(25, 3816),
'nombre': Fraction(173, 26712),
'vivo': Fraction(173, 26712),
'cordial': Fraction(19, 2968),
'saludo': Fraction(19, 2968),
'hace': Fraction(169, 26712),
'barrio': Fraction(1, 159),
'alcaldia': Fraction(55, 8904),
'madre': Fraction(41, 6678),
'oficina': Fraction(23, 3816),
'hijo': Fraction(23, 3816),
```

```
                 'actualizacion': Fraction(20, 3339),
                 '2': Fraction(157, 26712),
                 '1': Fraction(13, 2226),
                 'tramite': Fraction(11, 1908),
                 'mayor': Fraction(17, 2968),
                 'ao': Fraction(17, 2968),
                 'familia': Fraction(25, 4452),
                 'manera': Fraction(149, 26712),
                 'ser': Fraction(37, 6678),
                 'q': Fraction(7, 1272),
                 'seores': Fraction(73, 13356),
                 'meses': Fraction(71, 13356),
                 'muchas': Fraction(71, 13356),
                 'direccion': Fraction(47, 8904),
                 'buen': Fraction(47, 8904),
                 'quedo': Fraction(5, 954),
                 'calle': Fraction(137, 26712),
                 'realizar': Fraction(15, 2968),
                 'com': Fraction(67, 13356),
                 'gobierno': Fraction(67, 13356),
                 'radicado': Fraction(67, 13356),
                 'puedo': Fraction(67, 13356),
                 'trabajo': Fraction(19, 3816),
                 'quiero': Fraction(43, 8904),
                 'edad': Fraction(43, 8904),
                 'ciudad': Fraction(16, 3339),
                 'afiliacion': Fraction(16, 3339),
                 'casa': Fraction(16, 3339),
                 'tener': Fraction(127, 26712),
                 'atenta': Fraction(1, 212),
                 'persona': Fraction(125, 26712),
                 'municipio': Fraction(125, 26712),
                 'situacion': Fraction(41, 8904),
                 'servicio': Fraction(61, 13356),
                 'traslado': Fraction(121, 26712),
                 'nivel': Fraction(121, 26712)}),
 'Sec_Educacion': defaultdict(<function __main__.<lambda>()>,
             {'bucaramanga': Fraction(4806, 99359),
                 'educacion': Fraction(3937, 99359),
                 'secretaria': Fraction(2922, 99359),
                 'colegio': Fraction(2595, 99359),
                 'solicitud': Fraction(2386, 99359),
                 'co': Fraction(2334, 99359),
                 'gov': Fraction(2299, 99359),
                 'informacion': Fraction(2074, 99359),
                 'si': Fraction(1828, 99359),
                 'respuesta': Fraction(1779, 99359),
                 'peticion': Fraction(1701, 99359),
                 'correo': Fraction(1644, 99359),
                 'derecho': Fraction(1557, 99359),
                 'santander': Fraction(1354, 99359),
                 'favor': Fraction(1291, 99359),
                 'presente': Fraction(96, 7643),
                 'grado': Fraction(1202, 99359),
                 'gracias': Fraction(1198, 99359),
                 '1': Fraction(1188, 99359),
```

'ao': Fraction(90, 7643),
'proceso': Fraction(1155, 99359),
'c': Fraction(1115, 99359),
'dia': Fraction(1108, 99359),
'atencion': Fraction(1093, 99359),
'saludo': Fraction(1086, 99359),
'alcaldia': Fraction(1083, 99359),
'institucion': Fraction(1076, 99359),
'medio': Fraction(1070, 99359),
'cordial': Fraction(80, 7643),
'dias': Fraction(1022, 99359),
'2': Fraction(1009, 99359),
'cupo': Fraction(77, 7643),
'numero': Fraction(985, 99359),
'radicado': Fraction(74, 7643),
'solicito': Fraction(906, 99359),
'agradezco': Fraction(904, 99359),
'permito': Fraction(891, 99359),
'judicial': Fraction(855, 99359),
'documentos': Fraction(837, 99359),
'mas': Fraction(835, 99359),
'asunto': Fraction(809, 99359),
'electronico': Fraction(796, 99359),
'docente': Fraction(786, 99359),
'2021': Fraction(781, 99359),
'ley': Fraction(777, 99359),
'municipal': Fraction(776, 99359),
'manera': Fraction(776, 99359),
'cordialmente': Fraction(771, 99359),
'com': Fraction(765, 99359),
'nacional': Fraction(758, 99359),
'fecha': Fraction(744, 99359),
'adjunto': Fraction(740, 99359),
'buenas': Fraction(739, 99359),
'nombre': Fraction(726, 99359),
'5': Fraction(720, 99359),
'seora': Fraction(718, 99359),
'cuenta': Fraction(55, 7643),
'parte': Fraction(710, 99359),
'asi': Fraction(685, 99359),
'2020': Fraction(677, 99359),
'seores': Fraction(675, 99359),
'educativa': Fraction(675, 99359),
'nios': Fraction(674, 99359),
'docentes': Fraction(668, 99359),
'poder': Fraction(662, 99359),
'puede': Fraction(655, 99359),
'ser': Fraction(654, 99359),
'hijo': Fraction(654, 99359),
'aos': Fraction(647, 99359),
'municipio': Fraction(634, 99359),
'articulo': Fraction(631, 99359),
'atentamente': Fraction(631, 99359),
'oficio': Fraction(630, 99359),
'saber': Fraction(629, 99359),
'notificacion': Fraction(629, 99359),

```
                'pago': Fraction(597, 99359),
                'cucuta': Fraction(592, 99359),
                '3': Fraction(590, 99359),
                'hacer': Fraction(588, 99359),
                'estudiantes': Fraction(587, 99359),
                'tardes': Fraction(582, 99359),
                'siguiente': Fraction(577, 99359),
                'direccion': Fraction(577, 99359),
                'colombia': Fraction(576, 99359),
                'escolar': Fraction(576, 99359),
                'solicitar': Fraction(575, 99359),
                'seor': Fraction(571, 99359),
                'superior': Fraction(567, 99359),
                'fin': Fraction(567, 99359),
                'comunicacion': Fraction(562, 99359),
                'familia': Fraction(43, 7643),
                'buenos': Fraction(43, 7643),
                'calle': Fraction(549, 99359),
                'traves': Fraction(42, 7643),
                'servicio': Fraction(42, 7643),
                'caso': Fraction(536, 99359),
                'copia': Fraction(41, 7643),
                'envio': Fraction(531, 99359),
                'colaboracion': Fraction(527, 99359),
                'ustedes': Fraction(526, 99359)}),
    'Sec_Interior': defaultdict(<function __main__.<lambda>()>,
                {'policia': Fraction(21233, 498759),
                 'ppl': Fraction(20126, 498759),
                 'bucaramanga': Fraction(19855, 498759),
                 'solicitud': Fraction(15187, 498759),
                 'estacion': Fraction(12418, 498759),
                 'parte': Fraction(2807, 166253),
                 'despacho': Fraction(7985, 498759),
                 'informacion': Fraction(7805, 498759),
                 'personas': Fraction(2601, 166253),
                 '1': Fraction(250, 16089),
                 'ley': Fraction(6907, 498759),
                 'articulo': Fraction(6902, 498759),
                 'presente': Fraction(6730, 498759),
                 'respuesta': Fraction(6508, 498759),
                 'atencion': Fraction(6431, 498759),
                 'dia': Fraction(2136, 166253),
                 'peticion': Fraction(6407, 498759),
                 'nacional': Fraction(6245, 498759),
                 'correo': Fraction(6239, 498759),
                 '2': Fraction(6052, 498759),
                 'derecho': Fraction(6046, 498759),
                 'custodia': Fraction(5965, 498759),
                 'proceso': Fraction(5834, 498759),
                 'gov': Fraction(5801, 498759),
                 'medida': Fraction(1894, 166253),
                 'co': Fraction(5489, 498759),
                 'asi': Fraction(5423, 498759),
                 'comisorio': Fraction(1756, 166253),
                 'permito': Fraction(1752, 166253),
                 'publico': Fraction(5228, 498759),
```

```
'salud': Fraction(1697, 166253),
'entidades': Fraction(5078, 498759),
'radicado': Fraction(5066, 498759),
'libertad': Fraction(1671, 166253),
'calle': Fraction(4859, 498759),
'fin': Fraction(4780, 498759),
'medio': Fraction(4481, 498759),
'hacinamiento': Fraction(4480, 498759),
'aseguramiento': Fraction(4430, 498759),
'municipal': Fraction(4384, 498759),
'3': Fraction(4288, 498759),
'secretaria': Fraction(4286, 498759),
'dias': Fraction(4183, 498759),
'inpec': Fraction(4166, 498759),
'si': Fraction(4051, 498759),
'alcaldia': Fraction(1348, 166253),
'manera': Fraction(1344, 166253),
'judicial': Fraction(4001, 498759),
'barrio': Fraction(3985, 498759),
'c': Fraction(3980, 498759),
'saludo': Fraction(1323, 166253),
'fuga': Fraction(1298, 166253),
'estaciones': Fraction(3856, 498759),
'centro': Fraction(3770, 498759),
'codigo': Fraction(1245, 166253),
'cuenta': Fraction(1232, 166253),
'cordial': Fraction(3655, 498759),
'comunicacion': Fraction(1166, 166253),
'situacion': Fraction(3487, 498759),
'2020': Fraction(3470, 498759),
'mismo': Fraction(3452, 498759),
'adjunto': Fraction(3377, 498759),
'seor': Fraction(3371, 498759),
'competencia': Fraction(3347, 498759),
'fecha': Fraction(3289, 498759),
'hacer': Fraction(3269, 498759),
'favor': Fraction(1089, 166253),
'envio': Fraction(3253, 498759),
'carrera': Fraction(3253, 498759),
'espacio': Fraction(3247, 498759),
'interior': Fraction(3206, 498759),
'metropolitana': Fraction(3202, 498759),
'territoriales': Fraction(1067, 166253),
'numero': Fraction(3188, 498759),
'electronico': Fraction(3173, 498759),
'solicito': Fraction(3155, 498759),
'privadas': Fraction(3154, 498759),
'n': Fraction(1049, 166253),
'atentamente': Fraction(3140, 498759),
'asunto': Fraction(3127, 498759),
'cumplimiento': Fraction(1040, 166253),
'derechos': Fraction(1029, 166253),
'dentro': Fraction(1026, 166253),
'control': Fraction(3065, 498759),
'diferentes': Fraction(3062, 498759),
'caso': Fraction(3005, 498759),
```

```
                'permiso': Fraction(3001, 498759),
                'detencion': Fraction(2992, 498759),
                'tramite': Fraction(2953, 498759),
                '10': Fraction(981, 166253),
                'civil': Fraction(2932, 498759),
                'primera': Fraction(970, 166253),
                'ciudad': Fraction(2906, 498759),
                'oficio': Fraction(2897, 498759),
                'cordialmente': Fraction(2860, 498759),
                'mas': Fraction(2848, 498759),
                'seores': Fraction(942, 166253),
                'juzgado': Fraction(2797, 498759),
                'mediante': Fraction(2774, 498759),
                'santander': Fraction(2773, 498759)}),
 'Espacio_Publico': defaultdict(<function __main__.<lambda>()>,
             {'solicitud': Fraction(829, 15711),
              'espacio': Fraction(2296, 47133),
              'publico': Fraction(2215, 47133),
              'bucaramanga': Fraction(616, 15711),
              'peticion': Fraction(1208, 47133),
              'calle': Fraction(338, 15711),
              'derecho': Fraction(964, 47133),
              'informacion': Fraction(896, 47133),
              'comercial': Fraction(280, 15711),
              'barrio': Fraction(263, 15711),
              'plaza': Fraction(781, 47133),
              'municipio': Fraction(745, 47133),
              'edificio': Fraction(695, 47133),
              'local': Fraction(76, 5237),
              'respuesta': Fraction(682, 47133),
              'factura': Fraction(649, 47133),
              'cuenta': Fraction(611, 47133),
              'alcaldia': Fraction(608, 47133),
              'permiso': Fraction(179, 15711),
              'carrera': Fraction(533, 47133),
              'administracion': Fraction(175, 15711),
              'centro': Fraction(506, 47133),
              'central': Fraction(494, 47133),
              'numero': Fraction(493, 47133),
              'parte': Fraction(164, 15711),
              'secretaria': Fraction(52, 5237),
              'solicita': Fraction(155, 15711),
              'correo': Fraction(437, 47133),
              'si': Fraction(48, 5237),
              'visita': Fraction(427, 47133),
              'dia': Fraction(421, 47133),
              '00': Fraction(419, 47133),
              'adjunto': Fraction(44, 5237),
              'cobro': Fraction(131, 15711),
              'c': Fraction(392, 47133),
              'dadep': Fraction(389, 47133),
              'invasion': Fraction(43, 5237),
              'siguiente': Fraction(376, 47133),
              'sector': Fraction(124, 15711),
              'envio': Fraction(370, 47133),
              'san': Fraction(368, 47133),
```

```
'predio': Fraction(122, 15711),
'atencion': Fraction(40, 5237),
'1': Fraction(359, 47133),
'acropolis': Fraction(356, 47133),
'solicito': Fraction(350, 47133),
'mas': Fraction(349, 47133),
'medio': Fraction(344, 47133),
'supercentro': Fraction(38, 5237),
'dias': Fraction(341, 47133),
'presente': Fraction(338, 47133),
'nombre': Fraction(338, 47133),
'ciudad': Fraction(334, 47133),
'parque': Fraction(37, 5237),
'defensoria': Fraction(37, 5237),
'cordialmente': Fraction(106, 15711),
'vendedores': Fraction(316, 47133),
'venta': Fraction(313, 47133),
'administrativo': Fraction(310, 47133),
'2': Fraction(34, 5237),
'municipal': Fraction(305, 47133),
'seor': Fraction(305, 47133),
'manifiesta': Fraction(302, 47133),
'saludo': Fraction(299, 47133),
'personas': Fraction(295, 47133),
'favor': Fraction(293, 47133),
'agradezco': Fraction(32, 5237),
'cordial': Fraction(287, 47133),
'co': Fraction(286, 47133),
'queja': Fraction(281, 47133),
'gracias': Fraction(281, 47133),
'oficio': Fraction(280, 47133),
'gov': Fraction(278, 47133),
'3': Fraction(92, 15711),
'01': Fraction(272, 47133),
'radicado': Fraction(271, 47133),
'valor': Fraction(30, 5237),
'uso': Fraction(269, 47133),
'via': Fraction(88, 15711),
'manera': Fraction(263, 47133),
'ubicado': Fraction(29, 5237),
'estan': Fraction(260, 47133),
'colaboracion': Fraction(260, 47133),
'ser': Fraction(86, 15711),
'locales': Fraction(86, 15711),
'copia': Fraction(254, 47133),
'hace': Fraction(254, 47133),
'mes': Fraction(254, 47133),
'2021': Fraction(28, 5237),
'total': Fraction(251, 47133),
'seores': Fraction(247, 47133),
'p': Fraction(247, 47133),
'permito': Fraction(82, 15711),
'cra': Fraction(27, 5237),
'buenas': Fraction(79, 15711),
'pago': Fraction(236, 47133),
'com': Fraction(236, 47133),
```

```
              'reunion': Fraction(236, 47133),
              '2022': Fraction(235, 47133),
              'sr': Fraction(233, 47133)}),
 'Sec_Hacienda': defaultdict(<function __main__.<lambda>()>,
             {'declaracion': Fraction(327459, 3033974),
              'periodo': Fraction(122803, 1516987),
              'ao': Fraction(85584, 1516987),
              'presentacion': Fraction(84385, 1516987),
              'registro': Fraction(161253, 3033974),
              'formulario': Fraction(159605, 3033974),
              'gravable': Fraction(77357, 1516987),
              'vig': Fraction(74578, 1516987),
              'reteica': Fraction(107995, 3033974),
              'reteicapara': Fraction(92397, 3033974),
              '2021': Fraction(84387, 3033974),
              'ica': Fraction(74491, 3033974),
              '2020': Fraction(36941, 1516987),
              'icapara': Fraction(56727, 3033974),
              'comercio': Fraction(21864, 1516987),
              'industria': Fraction(37599, 3033974),
              '2022': Fraction(18420, 1516987),
              'solicitud': Fraction(35145, 3033974),
              'impuesto': Fraction(14992, 1516987),
              'bucaramanga': Fraction(29285, 3033974),
              'pago': Fraction(12975, 1516987),
              'informacion': Fraction(23295, 3033974),
              'respuesta': Fraction(19795, 3033974),
              'predial': Fraction(8557, 1516987),
              'mes': Fraction(8362, 1516987),
              'correo': Fraction(15889, 3033974),
              'favor': Fraction(7371, 1516987),
              'adjunto': Fraction(14713, 3033974),
              'retencion': Fraction(14503, 3033974),
              'nit': Fraction(7235, 1516987),
              '1': Fraction(7109, 1516987),
              'dia': Fraction(13955, 3033974),
              '2': Fraction(6850, 1516987),
              '2019': Fraction(13217, 3033974),
              'envio': Fraction(13097, 3033974),
              'cuenta': Fraction(6304, 1516987),
              'alcaldia': Fraction(6201, 1516987),
              'solicito': Fraction(6052, 1516987),
              's': Fraction(11989, 3033974),
              '3': Fraction(5952, 1516987),
              'medio': Fraction(5802, 1516987),
              '4': Fraction(5776, 1516987),
              '8': Fraction(11443, 3033974),
              '7': Fraction(5708, 1516987),
              'gracias': Fraction(10715, 3033974),
              'presente': Fraction(5249, 1516987),
              'municipio': Fraction(10213, 3033974),
              'peticion': Fraction(10179, 3033974),
              '5': Fraction(10083, 3033974),
              'cordialmente': Fraction(10071, 3033974),
              'si': Fraction(9907, 3033974),
              'com': Fraction(9685, 3033974),
```

```
'agradezco': Fraction(4831, 1516987),
'6': Fraction(9519, 3033974),
'co': Fraction(9517, 3033974),
'hacienda': Fraction(9315, 3033974),
'secretaria': Fraction(9081, 3033974),
'c': Fraction(4533, 1516987),
'saludo': Fraction(9063, 3033974),
'cordial': Fraction(4499, 1516987),
'oficio': Fraction(4368, 1516987),
'11': Fraction(8665, 3033974),
'buenas': Fraction(4295, 1516987),
'10': Fraction(8587, 3033974),
'empresa': Fraction(8491, 3033974),
'derecho': Fraction(8355, 3033974),
'dias': Fraction(8177, 3033974),
'12': Fraction(8149, 3033974),
'seores': Fraction(8127, 3033974),
'colaboracion': Fraction(8031, 3033974),
'cancelacion': Fraction(3998, 1516987),
'tardes': Fraction(3967, 1516987),
'inscripcion': Fraction(7911, 3033974),
'sas': Fraction(7757, 3033974),
'atencion': Fraction(7595, 3033974),
'9': Fraction(3684, 1516987),
'recibo': Fraction(3657, 1516987),
'numero': Fraction(7151, 3033974),
'marzo': Fraction(3571, 1516987),
'buen': Fraction(7041, 3033974),
'fecha': Fraction(3442, 1516987),
'colombia': Fraction(3413, 1516987),
'buenos': Fraction(3374, 1516987),
'realizar': Fraction(3364, 1516987),
'bogota': Fraction(3343, 1516987),
'proceso': Fraction(6639, 3033974),
'valor': Fraction(6569, 3033974),
'electronico': Fraction(3268, 1516987),
'presentar': Fraction(3239, 1516987),
'requerimiento': Fraction(3229, 1516987),
'junio': Fraction(6415, 3033974),
'mayo': Fraction(6231, 3033974),
'nombre': Fraction(6221, 3033974),
'abril': Fraction(6153, 3033974),
'correspondiente': Fraction(6107, 3033974),
'atenta': Fraction(2969, 1516987),
'quedo': Fraction(2951, 1516987),
'permito': Fraction(2910, 1516987),
'pagina': Fraction(2881, 1516987),
'devolucion': Fraction(5589, 3033974)})}
```

In [ ]:
```
def eval(input):
  words=input.split()
  deps_score={sum(pesos_dict[k][w] for w in words):k for k in pesos_dict.key
  return deps_score[max(deps_score.keys())]
```

In [ ]:
```
test['evaluation']=test['input'].swifter.apply(eval)
```

```
test
```

Out[ ]:

In [ ]:
```
accuracy=sum(test['NombreDependencia']==test['evaluation'])/len(test)
accuracy
```

Out[ ]:
```
0.5883438595303002
```

Resultado

Un accuracy del 58.8% para el Modelo de Bolsa de palabras desde frecuencias entre 16 categorias (la probabilidad de acertar aleatoriamente es de 6%)

In [ ]:
```
n=1000 #para training y dejar al menos 300 para test en la categoria mas baj
dat=data_corregido[['input','NombreDependencia']].dropna()
train=dat.groupby('NombreDependencia',as_index=False).sample(n=n)
test=dat.drop(train.index)
train
```

Out[ ]:

	input	NombreDependencia
74833	traslado hallazgos presunto alcance disciplina...	Con_Int_Discipli
146711	e201911125 radicado cpa 091019	Con_Int_Discipli
120	doctoralejandro ordoez maldonadoprocurador ge...	Con_Int_Discipli
27451	investigacion disciplinaria cpa 09117	Con_Int_Discipli
45127	apertura investigacion	Con_Int_Discipli
...	...	...
49004	levantamiento medida inenajenabilidad	Valorizacion
131774	recibo valorizacion beneficio local constancia...	Valorizacion
461425	solicitud levantamiento inenajenabilidad solic...	Valorizacion
45479	restablecer direccion correspondencia	Valorizacion
24816	solicitud levantamiento inenajenabilidad resol...	Valorizacion

16000 rows × 2 columns

In [ ]:
```
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(min_df=7,ngram_range=(1,2))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[ ]:

	TF-IDF
solicitud	0.270861
bucaramanga	0.248211
informacion	0.211404
disciplinaria	0.176099
disciplinario	0.175603
...	...
inconvenientes	0.000000
incremento	0.000000
independencia	0.000000
independiente	0.000000
zonas verdes	0.000000

6259 rows × 1 columns

In [ ]:
```
encoded_train=tfIdfVectorizer.transform(train['input'].values)
encoded_test=tfIdfVectorizer.transform(test['input'].values)
encoded_train
```

Out[ ]:
```
<16000x6259 sparse matrix of type '<class 'numpy.float64'>'
        with 376571 stored elements in Compressed Sparse Row format>
```

5. Modelo Random Forest

Se establecieron dos configuraciones diferentes en función de los hiperparámetros de los estimadores, correspondientes a 100 y 200

In [ ]:
```
rfc = RandomForestClassifier(n_estimators=100,random_state=7,n_jobs=-1,verbo
rfc.fit(encoded_train, train['NombreDependencia'].values)
```

Out[ ]:
```
RandomForestClassifier
```

In [ ]:
```
rfc2 = RandomForestClassifier(n_estimators=200,random_state=7,n_jobs=-1,verb
rfc2.fit(encoded_train, train['NombreDependencia'].values)
```

Out[ ]:
```
RandomForestClassifier
```

Usando TF-IDF con Random Forest Classifier con 100 estimadores

In [ ]:
```
rfc.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6843447176431277
```

Usando TF-IDF con Random Forest Classifier con 200 estimadores

In [ ]:
```
rfc2.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6866176347738917
```

Resultado

Un accuracy de 68.6% para el Modelo Random Forest (200 Estimadores), supero el rendimiento de la bolsa de palabras bajo frecuencias

In [ ]:
```
dump(rfc, 'random_forest_classifier.joblib')
rfc = load('random_forest_classifier.joblib')
```

6. Modelo SVC

In [ ]:
```
svc=SVC(verbose=1,random_state=7)#kernel rbf
svc.fit(encoded_train, train['NombreDependencia'])
```

```
[LibSVM]
```

Out[ ]:
```
 SVC
```

In [ ]:
```
svc2=SVC(kernel='sigmoid',verbose=1,random_state=7)#kernel sigmoid
svc2.fit(encoded_train, train['NombreDependencia'])
```

```
[LibSVM]
```

Out[ ]:
```
 SVC
```

In [ ]:
```
svc.score(encoded_test,test['NombreDependencia'].values)
```

In [ ]:
```
svc2.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6490013783693473
```

Resultado

Un Accuracy de 68,4% para el Modelo de SVC con la técnica Kernel RBF y 64,9% con la técnica Kernel Sigmoidal, siendo el primero modelo un score muy competitivo al resto de los modelos sin tener un compromiso de performance importante.

In [ ]:
```
dump(svc, 'support_vector_machine.joblib')
svc = load('support_vector_machine.joblib')
```

7. Modelo Multinomial Naive Bayes

In [ ]:
```
mnb = MultinomialNB(alpha=0.5,force_alpha=True)
mnb.fit(encoded_train, train['NombreDependencia'])
```

Out[ ]:
```
MultinomialNB
```

In [ ]:
```
mnb2 = MultinomialNB(alpha=0.8,force_alpha=True)
mnb2.fit(encoded_train, train['NombreDependencia'])
```

Out[ ]:
```
MultinomialNB
```

Con Multinomial Naive Bayes se obtiene un 65%

In [ ]:
```
mnb.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6529172003787035
```

In [ ]:
```
mnb2.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6468172198707953
```

In [ ]:
```
cnb = ComplementNB(alpha=0.5,force_alpha=True)
cnb.fit(encoded_train, train['NombreDependencia'])
```

Out[ ]:
```
ComplementNB
```

In [ ]:
```
cnb2 = ComplementNB(alpha=0.8,force_alpha=True)
cnb2.fit(encoded_train, train['NombreDependencia'])
```

Out[ ]:
```
ComplementNB
```

In [ ]:
```
cnb.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6633002199821787
```

In [ ]:
```
cnb2.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6634063822677656
```

Resultado

Un accuracy de 66,3% (alpha 0.8) del Modelo Complement Naive Bayes, un resultado inferior a los clasificadores RF SV pero superior a Bolsa de Palabras

8. Modelo de Regresión Logística

In [ ]:
```
lr=LogisticRegression(random_state=7,n_jobs=-1,penalty='elasticnet',solver='
lr.fit(encoded_train, train['NombreDependencia'])
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concurrent work
ers.
convergence after 20 epochs took 45 seconds
[Parallel(n_jobs=-1)]: Done   1 out of   1 | elapsed:   45.1s finished
```

Out[ ]:
```
 LogisticRegression
```

In [ ]:
```
lr2=LogisticRegression(random_state=7,n_jobs=-1,penalty='elasticnet',solver=
lr2.fit(encoded_train, train['NombreDependencia'])
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concurrent work
ers.
convergence after 20 epochs took 16 seconds
[Parallel(n_jobs=-1)]: Done   1 out of   1 | elapsed:   15.8s finished
```

Out[ ]:
```
 LogisticRegression
```

In [ ]:
```
lr.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6853941579416352
```

In [ ]:
```
lr2.score(encoded_test,test['NombreDependencia'].values)
```

Out[ ]:
```
0.6824216139451994
```

Resultado

Un accuracy de 68,5% con regresion logistica (ratio 0.2), un resultado compertitvo con los modelos SVC y RF superior a Bolsa de palabras y Naive Bayes

Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

Selección de hiperparámetros

Estudiantes

Wilfredo Ariel Góme Bueno

Edson Andrés Gómez Cárdenas

In [ ]:
```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from pandarallel import pandarallel
pandarallel.initialize(progress_bar=True)
import swifter
from collections import Counter,defaultdict
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from spellchecker import SpellChecker
import multiprocessing
from fractions import Fraction
from sklearn.feature_extraction.text import TfidfVectorizer
from joblib import dump, load
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB, ComplementNB,GaussianNB
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import GridSearchCV
import optuna
```

```
INFO: Pandarallel will run on 6 workers.
INFO: Pandarallel will use Memory file system to transfer data between the m
ain process and workers.
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/n00b1337/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

1. Del cuaderno 0. Wrangling, Exploration and Clenaning, se trae el dataseet preprocesado y organizado.

In [ ]:
```
data_corregido=pd.read_csv('data_corregida.csv')
data_corregido
```

Out[ ]:

	ASUNTO	OBSERVACION	NombreDependencia	input
0	NaN	insatisfaccion pan recibido	Sec_Admin	insatisfaccion pan recibido
1	NaN	queja contaminacion visual calle 45 carrera 13...	Sec_Admin	queja contaminacion visual calle 45 carrera 1...
2	NaN	prueba pqrs 05102011	Sec_Admin	prueba pqrs 05102011
3	NaN	cra 5 n 57 59 barrio porvenir 2 luminarias fun...	Sec_Infraest	cra 5 n 57 59 barrio porvenir 2 luminarias fu...
4	NaN	solicitud	Sec_Admin	solicitud
...	...	...	...	...
590587	formulario declaracion ica	registro presentacion declaracion icapara ao g...	Sec_Hacienda	formulario declaracion ica registro presentaci...
590588	formulario declaracion ica	registro presentacion declaracion icapara ao g...	Sec_Hacienda	formulario declaracion ica registro presentaci...
590589	solicitud oferta institucional	seores secretaria desarrollo l c reciban cordi...	Sec_Des_social	solicitud oferta institucional seores secretar...
590590	formulario declaracion ica	registro presentacion declaracion icapara ao g...	Sec_Hacienda	formulario declaracion ica registro presentaci...
590591	incompetentes espacio publico	viernes 9 noche intento pasar hijos anden carr...	Espacio_Publico	incompetentes espacio publico viernes 9 noche ...

590592 rows × 4 columns

1.1 Primer componente

Se determina como primer componente de la Arquitectura del Modelo los registros que son Secretaría de Hacienda frente a los demás, caracterizados como "otros".

In [ ]:
```
data_corregido['NombreDependencia'].unique()
```

Out[ ]:
```
array(['Sec_Admin', 'Sec_Infraest', 'Sec_Des_social', 'Sec_Educacion',
       'Sec_Hacienda', 'Des_Alcalde', 'Sec_Planeacion', 'Ofc_TIC',
       'Con_Int_Discipli', 'Ser_Publicos', 'Sec_Interior', 'Extra',
       'Sec_Salud_y_ambi', 'Espacio_Publico', 'Valorizacion',
       'Sec_Juridica', 'Pla_Sisben', 'Prensa_y_Comuni', 'Direccion', 'ST'],
      dtype=object)
```

In [ ]:
```
primer_componente=data_corregido[['input','NombreDependencia']].dropna().cop
primer_componente.loc[primer_componente['NombreDependencia']!='Sec_Hacienda'
```

```
primer_componente.groupby('NombreDependencia').count()
```

Out[ ]:

NombreDependencia	input
Sec_Hacienda	317272
otro	273320

Este bloque de código está realizando un proceso similar al bloque anterior en el que se usaba el algoritmo TF-IDF, pero con algunos cambios.

Primero, está seleccionando un conjunto de entrenamiento y un conjunto de prueba a partir del DataFrame `primer_componente`, pero en este caso, se está asegurando de que la muestra de entrenamiento para cada dependencia tenga exactamente `n` registros. Para ello, se usa el método `sample()` en lugar de `sample(frac=0.8)`. Esto puede ser útil si se sabe que se requiere un número específico de registros de entrenamiento.

A continuación, se repite el proceso de agrupar las entradas por dependencia, entrenar el vectorizador TF-IDF en el conjunto de datos resultante y convertir los pesos TF-IDF para la primera dependencia en un DataFrame.

Sin embargo, en este caso, se configura el vectorizador TF-IDF para ignorar las palabras y pares de palabras que aparecen en menos de 2 documentos con `min_df=2`. Esto puede ser útil para eliminar las palabras y pares de palabras menos comunes que pueden no ser útiles para clasificar las dependencias.

En resumen, este bloque de código está preparando un nuevo conjunto de entrenamiento y prueba, entrenando un nuevo vectorizador TF-IDF en el conjunto de entrenamiento y mostrando las palabras y pares de palabras más importantes para la primera dependencia en el conjunto de entrenamiento, según el algoritmo TF-IDF.

In [ ]:
```
n=int(273320*0.8) #para training el resto para test
train=primer_componente.groupby('NombreDependencia',as_index=False).sample(n
test=primer_componente.drop(train.index)
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(min_df=2,ngram_range=(1,2))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[ ]:

	TF-IDF
declaracion	0.537660
periodo	0.402914
ao	0.281163
presentacion	0.277064
registro	0.264828
...	...
penaprosperidadsocial	0.000002
gaseosos	0.000002
pencion	0.000002
becerril	0.000002
zzaingongbike	0.000002

37649 rows × 1 columns

In [ ]:
```
dump(tfIdfVectorizer, 'tfidf_primera_capa.joblib')
tfIdfVectorizer = load('tfidf_primera_capa.joblib')
```

1.1.1 Logistic Regressor Classifier

En resumen, este bloque de código prepara los datos de entrenamiento, configurando un clasificador de regresión logística con validación cruzada y regularización Elastic Net, entrenando el clasificador, y guardando el modelo entrenado para su uso posterior.

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
l1_ratios=[i*0.1 for i in range(2,9)]#dejar al menos un poco de regularizaci
Cs=[i*0.1 for i in range(8,40,2)]#reguralizadores razonables
lrc=LogisticRegressionCV(Cs=Cs,cv=3,penalty='elasticnet',solver='saga',n_job
lrc.fit(X_train,y_train)
dump(lrc, 'logistic_regression_classifier.joblib')
lrc
```

Se calcula el rendimiento del modelo en estos datos de prueba. Se utiliza la función score() para hacer esto. Esta función da la precisión de las predicciones del modelo, que es la proporción de entradas de prueba para las que mi modelo predijo correctamente la etiqueta.

In [ ]:
```
print('Best C:',float(lrc.C_))
print('Best l1_ratio:',float(lrc.l1_ratio_))
X_test=tfIdfVectorizer.transform(test['input'])
y_test=test['NombreDependencia'].values
print('Score:',lrc.score(X_test,y_test))
lrc
```

```
Best C: 3.6
Best l1_ratio: 0.6000000000000001
Score: 0.9440762004175365
```

Out[ ]:
```
LogisticRegressionCV
```

1.1.2 Logistic Regressor Classifier - Segundo round

En esta parte del código, se afina el modelo de regresión logística basado en la información que se obtuvo en los pasos anteriores.

Primero, se transforma los datos de entrenamiento en vectores TF-IDF usando la función `transform()` del objeto `tfIdfVectorizer`. Se almacena los vectores resultantes en `X_train`. Además, se extrae las etiquetas correspondientes de los datos de entrenamiento y se guardan en `y_train`.

El objetivo es ajustar los parámetros de regularización `l1_ratios` y `Cs` de mi modelo para obtener un mejor rendimiento. Basados en los resultados que se obtuvieron en la búsqueda de parámetros anterior, se toma la decisión de probar un rango más pequeño y específico de valores para estos parámetros. Así que se establecen los siguientes: `l1_ratios` en `[0.55, 0.6, 0.65]` y `Cs` en `[3.55, 3.6, 3.65]`.

Luego, se crea un nuevo modelo de regresión logística con validación cruzada utilizando los parámetros de regularización afinados. Se utilizan los mismos parámetros que antes, pero esta vez con los nuevos rangos de `l1_ratios` y `Cs`.

Entonces, se ajusta este nuevo modelo a los datos de entrenamiento usando la función `fit()`. Una vez que el modelo se entrenó, se guarda en un archivo usando la función `dump()` de joblib. Esto me permitirá cargar el modelo más tarde sin tener que volver a entrenarlo.

Finalmente, se imprime el objeto del modelo para ver su configuración y los parámetros que aprendió durante el entrenamiento.

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
l1_ratios=[0.55,0.6,0.65]#l1 basados en la anterior busqueda
Cs=[3.55,3.6,3.65]#reguralizadores razonables basados en la anterior busqued
lrc=LogisticRegressionCV(Cs=Cs,cv=3,penalty='elasticnet',solver='saga',n_job
lrc.fit(X_train,y_train)
dump(lrc, 'logistic_regression_classifier.joblib')
lrc
```

In [ ]:
```
print('Best C:',float(lrc.C_))
print('Best l1_ratio:',float(lrc.l1_ratio_))
X_test=tfIdfVectorizer.transform(test['input'])
y_test=test['NombreDependencia'].values
```

```
print('Score:',lrc.score(X_test,y_test))
lrc
```

```
Best C: 3.55
Best l1_ratio: 0.55
Score: 0.9440827244258873
```

Out[ ]: LogisticRegressionCV

1.1.3 ComplementNB

Se decide afinar el parámetro alpha del modelo Complement Naive Bayes (CNB), que controla el suavizado de Laplace utilizado por el modelo. Para ello, se establece un rango de posibles valores de alpha desde 0.1 hasta 19.0, con un paso de 1.0, y se almacenan en la lista alphas.

Además, se tiene interés en probar el modelo tanto con como sin normalización, por lo que se define norm como una lista con los valores False y True.

Luego, se crea un nuevo modelo Complement Naive Bayes, obligándolo a usar el suavizado de Laplace incluso si alpha es 0.

Para encontrar la mejor combinación de parámetros alpha y norm, se utiliza una búsqueda en malla (GridSearchCV). Esta herramienta prueba todas las combinaciones posibles de los parámetros proporcionados y selecciona la que da el mejor rendimiento en la validación cruzada. Se entrena la búsqueda en malla con los datos de entrenamiento.

Finalmente, se guarda el modelo resultante en un archivo utilizando dump() y se imprime para ver la combinación de parámetros seleccionada.

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
alphas=[i*0.1 for i in range(1,200,10)]
parameters = {'alpha':alphas, 'norm':[False, True]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier.joblib')
clf
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
```

Out[ ]: GridSearchCV
estimator: ComplementNB
ComplementNB

In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.1, 'norm': True}
0.9355791748546586
```

1.1.4 ComplementNB segundo round

En este fragmento de código, se realiza un ajuste más fino del modelo Complement Naive Bayes (CNB).

Se especifica un nuevo conjunto de posibles valores para `alpha`, el parámetro de suavizado. En lugar de incrementar de 1.0 en 1.0 como antes, ahora se incrementa de 0.01 en 0.01, desde 0.01 hasta 1.99. Este nuevo rango se almacena en la lista `alphas`.

El modelo CNB se inicializa de nuevo, obligando al uso del suavizado de Laplace incluso si `alpha` es 0.

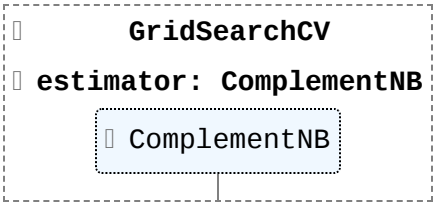
Se realiza la búsqueda en malla (`GridSearchCV`) de nuevo, pero esta vez con el nuevo conjunto de alphas. Esto significa que se está probando una gama más amplia y precisa de posibles valores de `alpha` que antes.

De nuevo, se entrena la búsqueda en malla con los datos de entrenamiento, y se guarda el modelo resultante en un archivo utilizando `dump()`. Finalmente, se imprime el modelo para visualizar la combinación de parámetros seleccionada.

In [ ]:
```
alphas=[i*0.01 for i in range(1,200)]
parameters = {'alpha':alphas, 'norm':[False, True]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier.joblib')
clf
```

```
Fitting 3 folds for each of 398 candidates, totalling 1194 fits
```

Out[ ]:



In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.1, 'norm': True}
0.9355791748546586
```

1.2 Segundo componente

Se crea un nuevo conjunto de datos llamado `segundo_componente` a partir del conjunto de datos previamente limpio y procesado, `data_corregido`. Se seleccionan las

columnas 'input' y 'NombreDependencia' y se eliminan los registros que contienen valores faltantes con el método `dropna()`.

A continuación, se realiza un filtrado de datos donde se excluyen los registros que tienen 'Sec_Hacienda' como 'NombreDependencia'. Este filtrado se realiza con el método `loc`, que permite seleccionar filas basadas en un criterio booleano. En este caso, el criterio es que la 'NombreDependencia' no sea 'Sec_Hacienda'.

Finalmente, se agrupan los datos por 'NombreDependencia' utilizando el método `groupby()`, y se cuenta el número de registros para cada 'NombreDependencia' utilizando el método `count()`. El resultado se ordena en función del número de 'input' usando `sort_values()`. Este último paso proporciona un resumen de cuántos registros hay para cada 'NombreDependencia', ordenados de menor a mayor.

In [ ]:
```
segundo_componente=data_corregido[['input','NombreDependencia']].dropna().co
segundo_componente=segundo_componente.loc[segundo_componente['NombreDependen
segundo_componente.groupby('NombreDependencia').count().sort_values(by='inpu
```

Out[ ]:

NombreDependencia	input
Prensa_y_Comuni	416
Ser_Publicos	577
ST	760
Direccion	763
Ofc_TIC	1331
Extra	1338
Con_Int_Discipli	1466
Pla_Sisben	2900
Espacio_Publico	7126
Sec_Educacion	7238
Sec_Juridica	10693
Valorizacion	13238
Des_Alcalde	15058
Sec_Infraest	20585
Sec_Des_social	22173
Sec_Planeacion	35339
Sec_Salud_y_ambi	41269
Sec_Admin	43680
Sec_Interior	47370

In [ ]:
```
deps=segundo_componente.groupby('NombreDependencia',as_index=False).count().
deps['input']=deps['input']<35000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
segundo_componente['NombreDependencia']=segundo_componente['NombreDependenci
segundo_componente.groupby('NombreDependencia').count().sort_values(by='inpu
```

Out[ ]:

NombreDependencia	input
Sec_Planeacion	35339
Sec_Salud_y_ambi	41269
Sec_Admin	43680
Sec_Interior	47370
Extra	105662

Se prepara un conjunto de datos para entrenar un modelo basado en el esquema TF-IDF (Term Frequency-Inverse Document Frequency). A continuación, explicaré paso a paso lo que hace este código:

1. `n=35339`: Se establece el número de muestras que se desea tomar de cada grupo en el conjunto de datos.
2. `train_cv=segundo_componente.groupby('NombreDependencia',as_index=Fa` Aquí se agrupa el dataframe `segundo_componente` por la columna 'NombreDependencia', y se toma una muestra aleatoria de `n` filas de cada grupo. El resultado es un nuevo dataframe `train_cv`, que contiene una cantidad equitativa de muestras de cada 'NombreDependencia'.
3. `dataset=train_cv.groupby('NombreDependencia', as_index=False).agg({'input':' '.join})['input'].values`: De nuevo, se agrupa el dataframe `train_cv` por 'NombreDependencia', pero esta vez se concatenan todas las entradas de 'input' en una única cadena de texto para cada 'NombreDependencia'. El resultado es un array de textos, uno para cada 'NombreDependencia', que se almacena en la variable `dataset`.
4. `tfIdfVectorizer=TfidfVectorizer(min_df=2,ngram_range=(1,2),)`: Se inicializa un objeto `TfidfVectorizer` con ciertos parámetros. El parámetro `min_df=2` significa que una palabra debe aparecer al menos en dos documentos para ser considerada en el vocabulario. El parámetro `ngram_range=(1,2)` indica que el vectorizador debe considerar tanto palabras individuales como bigramas (pares de palabras consecutivas).
5. `tfIdf = tfIdfVectorizer.fit_transform(dataset)`: Se ajusta el vectorizador TF-IDF al conjunto de datos y se transforma este en una matriz de términos y sus frecuencias.

6. `df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_names_out(), columns=["TF-IDF"])` : Se crea un dataframe a partir de la matriz TF-IDF. Las filas representan las palabras del vocabulario y la columna "TF-IDF" contiene los valores TF-IDF correspondientes.

7. `df = df.sort_values('TF-IDF', ascending=False)` : Finalmente, se ordena el dataframe por valores TF-IDF descendentes. Las palabras en la parte superior del dataframe son las que tienen los valores TF-IDF más altos, es decir, las que son más representativas de los documentos en el conjunto de datos.

In [ ]:
```
n=35339
train_cv=segundo_componente.groupby('NombreDependencia',as_index=False).samp
dataset=train_cv.groupby('NombreDependencia', as_index=False).agg({'input':'
tfIdfVectorizer=TfidfVectorizer(min_df=2,ngram_range=(1,2),)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[ ]:

	TF-IDF
solicitud	0.421835
bucaramanga	0.340867
informacion	0.168444
peticion	0.163073
derecho	0.138262
...	...
ala seora	0.000000
ley 10	0.000000
ala respuesta	0.000000
ley 1042	0.000000
zx ventures	0.000000

306924 rows × 1 columns

In [ ]:
```
dump(tfIdfVectorizer, 'tfidf_segundo_componente.joblib')
tfIdfVectorizer = load('tfidf_segundo_componente.joblib')
```

ComplementNB

Se entrena un clasificador basado en el modelo Naive Bayes Complementario, utilizando una búsqueda de cuadrícula para optimizar los parámetros. Aquí está el desglose paso a paso de lo que está ocurriendo:

1. `X_train=tfIdfVectorizer.transform(train_cv['input'])` : Se transforman los datos de entrada del conjunto de entrenamiento en vectores TF-IDF utilizando el vectorizador TF-IDF que se ajustó previamente.

2. `y_train=train_cv['NombreDependencia'].values` : Se almacenan los valores objetivo del conjunto de entrenamiento en `y_train` . Estos son los valores que el clasificador intentará predecir.

3. `alphas=[i*0.1 for i in range(1,200,10)]` : Se establece una lista de posibles valores para el parámetro `alpha` del clasificador ComplementNB. `alpha` es un parámetro de suavizado que ayuda a evitar el subajuste y el sobreajuste.

4. `parameters = {'alpha':alphas, 'norm':[False, True]}` : Se establece un diccionario con los parámetros que la búsqueda de cuadrícula probará. `norm` es otro parámetro del clasificador ComplementNB que, si es verdadero, normalizará las probabilidades a priori.

5. `cnb=ComplementNB(force_alpha=True)` : Se inicializa un objeto ComplementNB, forzando a que se tenga en cuenta el parámetro `alpha` .

6. `clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)` : Se inicializa un objeto GridSearchCV que realizará una búsqueda de cuadrícula para encontrar la combinación óptima de parámetros para el clasificador ComplementNB. La validación cruzada se realiza con tres particiones (`cv=3`).

7. `clf.fit(X_train,y_train)` : Se entrena el clasificador con los datos de entrada y salida.

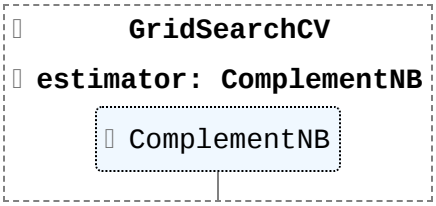
8. `dump(clf, 'complementNB_classifier_segundo_componente.joblib')` : Se guarda el modelo entrenado en un archivo joblib para su uso futuro.

9. `clf` : Finalmente, se imprime la representación del clasificador para tener una visión general de sus parámetros y estado.

In [ ]:
```
X_train=tfIdfVectorizer.transform(train_cv['input'])
y_train=train_cv['NombreDependencia'].values
alphas=[i*0.1 for i in range(1,200,10)]
parameters = {'alpha':alphas, 'norm':[False, True]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier_segundo_componente.joblib')
clf
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
```

Out[ ]:



In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 1.1, 'norm': True}
0.7313110194716335
```

ComplementNB segundo round

Se afina aún más el clasificador basado en el modelo Naive Bayes Complementario, utilizando una búsqueda de cuadrícula para optimizar los parámetros. Aquí está el desglose paso a paso de lo que está ocurriendo:

alphas=[i*0.1 for i in range(4,20)]: Se establece una nueva lista de posibles valores para el parámetro alpha del clasificador ComplementNB. En este caso, se ha reducido el rango de valores de alpha para afinar más la optimización basándose en los resultados obtenidos en la ejecución anterior.

parameters = {'alpha':alphas, 'norm':[False, True]}: Se establece un diccionario con los parámetros que la búsqueda de cuadrícula probará.

cnb=ComplementNB(force_alpha=True): Se inicializa un nuevo objeto ComplementNB.

clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1): Se inicializa un objeto GridSearchCV que realizará una búsqueda de cuadrícula para encontrar la combinación óptima de parámetros para el clasificador ComplementNB.

clf.fit(X_train,y_train): Se entrena el clasificador con los datos de entrada y salida.

dump(clf, 'complementNB_classifier_segundo_componente.joblib'): Se guarda el modelo entrenado en un archivo joblib para su uso futuro.

clf: Finalmente, se imprime la representación del clasificador para tener una visión general de sus parámetros y estado.

Esta repetición de entrenamiento con un rango de parámetros ajustado permite obtener un modelo más optimizado, ya que se está realizando una búsqueda más detallada en la región de parámetros que ha producido buenos resultados en la primera ejecución.

In [ ]:
```
alphas=[i*0.1 for i in range(4,20)]
parameters = {'alpha':alphas, 'norm':[False, True]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
```

```
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier_segundo_componente.joblib')
clf
```

In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.6000000000000001, 'norm': True}
0.7331729824752582
```

MultinomialNB

Se configura y entrena un clasificador de Naive Bayes Multinomial utilizando una búsqueda de cuadrícula para encontrar el mejor valor del parámetro alpha.

In [ ]:
```
alphas=[i*0.1 for i in range(0,200,10)]
parameters = {'alpha':alphas}
mnb=MultinomialNB()
clf = GridSearchCV(mnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'multinomialNB_classifier_segundo_componente.joblib')
clf
```

In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 1.0}
0.7214522149125605
```

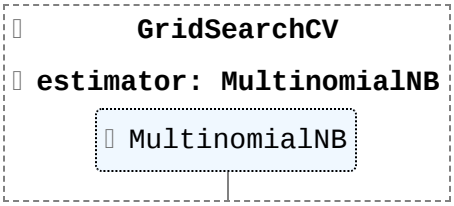
MultinomialNB segundo round

En este caso, se afina el clasificador basado en los datos dados anteriormente para este modelo.

In [ ]:
```
alphas=[i*0.1 for i in range(1,20)]
parameters = {'alpha':alphas}
mnb=MultinomialNB(force_alpha=True)
clf = GridSearchCV(mnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'multinomialNB_classifier_segundo_componente.joblib')
clf
```

```
Fitting 3 folds for each of 19 candidates, totalling 57 fits
```

Out[ ]:



In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.2}
0.7290698607232146
```

Random Forest Classifier

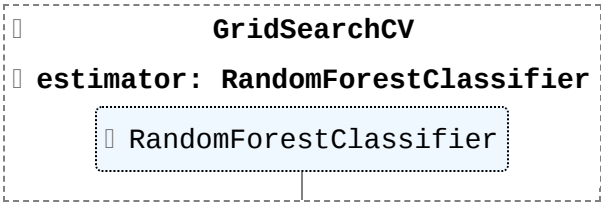
Se configura y entrena un clasificador de Random Forest mediante una búsqueda de cuadrícula para optimizar los hiperparámetros. Inicialmente, se define un rango de posibles valores para `n_estimators`, que es el número de árboles a procesar. Luego, se crea un diccionario con los parámetros que la búsqueda de cuadrícula probará, en esta instancia, se inicia con 'criterion' y 'max_features'.

In [ ]:
```
n_estimators=[i for i in range(20,21)]
parameters = {'n_estimators':n_estimators,
              'criterion':['gini', 'entropy'],
              'max_features':['sqrt', 'log2']}
rfc=RandomForestClassifier(verbose=1)
clf = GridSearchCV(rfc, parameters,cv=3,n_jobs=6,verbose=2)
clf.fit(X_train,y_train)
dump(clf, 'randomForest_classifier_segunda_capa.joblib')
clf
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 42.7min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    2.5s finished
[CV] END .criterion=gini, max_features=sqrt, n_estimators=20; total time=42.
8min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 43.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    2.6s finished
[CV] END .criterion=gini, max_features=sqrt, n_estimators=20; total time=43.
1min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 43.1min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    3.1s finished
[CV] END .criterion=gini, max_features=sqrt, n_estimators=20; total time=43.
2min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 55.1min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:   10.3s finished
[CV] END .criterion=gini, max_features=log2, n_estimators=20; total time=55.
3min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 55.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 55.5min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:   10.5s finished
[CV] END .criterion=gini, max_features=log2, n_estimators=20; total time=55.
6min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
```

```
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:   10.4s finished
[CV] END .criterion=gini, max_features=log2, n_estimators=20; total time=55.
7min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 41.3min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    2.0s finished
[CV] END criterion=entropy, max_features=sqrt, n_estimators=20; total time=4
1.4min
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 41.3min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    1.9s finished
[CV] END criterion=entropy, max_features=sqrt, n_estimators=20; total time=4
1.4min
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 41.7min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    1.7s finished
[CV] END criterion=entropy, max_features=sqrt, n_estimators=20; total time=4
1.8min
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 41.7min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    8.6s finished
[CV] END criterion=entropy, max_features=log2, n_estimators=20; total time=4
1.8min
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 43.5min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    7.4s finished
[CV] END criterion=entropy, max_features=log2, n_estimators=20; total time=4
3.6min
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 43.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    6.9s finished
[CV] END criterion=entropy, max_features=log2, n_estimators=20; total time=4
3.9min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 38.7min finished
```

Out[ ]:



In [ ]:

```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 20}
0.7363762543090168
```

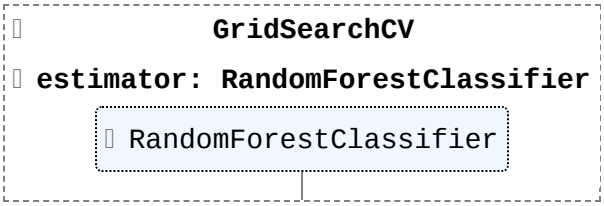
Ahora, se realiza el proceso para determinar el parámetro "n_estimators".

In [ ]:
```
n_estimators=[i for i in range(20,200,20)]
parameters = {'n_estimators':n_estimators}
rfc=RandomForestClassifier(verbose=1,criterion='gini',max_features='sqrt')
clf = GridSearchCV(rfc, parameters,cv=3,n_jobs=6,verbose=2)
clf.fit(X_train,y_train)
dump(clf, 'randomForest_classifier_segunda_capa.joblib')
clf
```

```
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed: 289.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:   31.9s finished
[CV] END ................................n_estimators=140; total time=289.
9min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed: 291.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:   30.7s finished
[CV] END ................................n_estimators=140; total time=291.
5min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed: 296.9min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:   31.3s finished
[CV] END ................................n_estimators=140; total time=297.
4min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed: 377.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:   45.2s finished
[CV] END ................................n_estimators=160; total time=378.
2min
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed: 381.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:   41.9s finished
[CV] END ................................n_estimators=160; total time=382.
6min
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed: 382.6min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:   36.5s finished
[CV] END ................................n_estimators=160; total time=383.
2min
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed: 357.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed: 355.5min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:   28.4s finished
[CV] END ................................n_estimators=180; total time=357.
5min
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:   28.1s finished
[CV] END ................................n_estimators=180; total time=356.
0min
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed: 353.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
```

```
ers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:   23.8s finished
[CV] END ..............................n_estimators=180; total time=353.
8min
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed: 350.0min finished
```

Out[ ]:



In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'n_estimators': 180}
0.7507060293997719
```

1.3 Tercer componente

En la primera línea, se crea una copia del DataFrame data_corregido, seleccionando sólo las columnas 'NombreDependencia' e 'input' y se almacena en tercer_componente.

A continuación, se calcula la cantidad de entradas por cada 'NombreDependencia' en el DataFrame tercer_componente y se ordena el resultado en función del recuento de 'input'. Se almacena este resultado en deps.

Luego, se selecciona del DataFrame deps solo aquellas filas donde la columna 'input' tiene un valor menor que 35000 y posteriormente, se filtra por las que son mayores a 7000.

Por último, se actualiza tercer_componente para incluir solo las filas cuyo 'NombreDependencia' está presente en el DataFrame actualizado deps.

Finalmente, devuelve el DataFrame deps que contiene los recuentos de 'input' por 'NombreDependencia' que son menos de 35000. Este es un paso de preprocesamiento típico en el análisis de datos, donde se podrían estar eliminando 'dependencias' con muy pocas entradas para simplificar el análisis o la modelización.

In [ ]:
```
tercer_componente=data_corregido[['NombreDependencia','input']].copy()
deps=tercer_componente[['NombreDependencia','input']].groupby('NombreDepende
deps=deps[deps['input']<35000]
tercer_componente=tercer_componente[tercer_componente['NombreDependencia'].i
deps
```

Out[ ]:

	NombreDependencia	input
7	Prensa_y_Comuni	416
18	Ser_Publicos	577
8	ST	760
2	Direccion	763
5	Ofc_TIC	1331
4	Extra	1338
0	Con_Int_Discipli	1466
6	Pla_Sisben	2900
3	Espacio_Publico	7126
11	Sec_Educacion	7238
15	Sec_Juridica	10693
19	Valorizacion	13238
1	Des_Alcalde	15058
13	Sec_Infraest	20585
10	Sec_Des_social	22173

In [ ]:
```
deps['input']=deps['input']<7000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
tercera_capa['NombreDependencia']=tercera_capa['NombreDependencia'].replace(
tercera_capa.groupby('NombreDependencia').count().sort_values(by='input')[['
```

Out[ ]:

	input
NombreDependencia	
Espacio_Publico	7126
Sec_Educacion	7238
Extra	9551
Sec_Juridica	10693
Valorizacion	13238
Des_Alcalde	15058
Sec_Infraest	20585
Sec_Des_social	22173

En este fragmento de código, estamos generando un conjunto de entrenamiento y un modelo TF-IDF para este conjunto.

Primero, se seleccionan muestras aleatorias de cada "NombreDependencia" en el dataframe `tercera_capa` con un tamaño de muestra igual a `n` (7126 en este caso). Este conjunto de datos se almacena en `train_cv`.

Después, se agrupa el conjunto de entrenamiento por "NombreDependencia" y se concatena todas las entradas correspondientes a cada "NombreDependencia" en una sola cadena de texto. Este paso es importante para entrenar nuestro modelo TF-IDF, ya que cada "NombreDependencia" se representará como un solo documento.

Se crea un objeto `TfidfVectorizer`, que se utilizará para transformar las entradas en vectores TF-IDF. El parámetro `min_df` se establece en 1, lo que significa que se incluirán en el modelo las palabras que aparecen al menos en un documento. El parámetro `ngram_range` se establece en (1,2), lo que significa que el vectorizador considerará tanto unigramas como bigramas.

Entonces, se entrena el vectorizador TF-IDF con el conjunto de datos transformado y se crea una matriz TF-IDF.

Finalmente, se convierte esta matriz TF-IDF en un dataframe y se ordena en orden descendente por la columna "TF-IDF". Este dataframe muestra el peso TF-IDF de cada término para el primer documento en el conjunto de datos (la primera "NombreDependencia").

In [ ]:
```
n=7126
train_cv=tercera_capa.groupby('NombreDependencia',as_index=False).sample(n=n
dataset=train_cv.groupby('NombreDependencia', as_index=False).agg({'input':'
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,2),)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[ ]:

	TF-IDF
bucaramanga	0.326903
solicitud	0.292664
peticion	0.227935
informacion	0.210816
alcalde	0.200544
...	...
desafortunada situacion	0.000000
desafortunadamente afecto	0.000000
desafortunadamente archivo	0.000000
desafortunadamente cambio	0.000000
zykagradezco toda	0.000000

819172 rows × 1 columns

Complement NB

Se establece un conjunto de hiperparámetros potenciales para el modelo Complement Naive Bayes: una serie de valores de alfa, que es el parámetro de suavizado, y dos opciones para el parámetro 'norm', que decide si se debe aplicar o no la normalización.

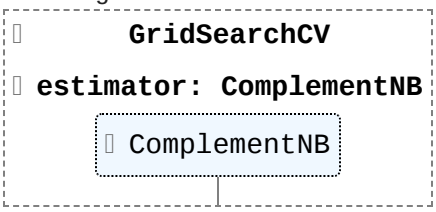
Se crea una instancia del modelo ComplementNB con force_alpha=True, lo que significa que el valor de alfa que se utilizará no es el óptimo estimado por máxima verosimilitud, sino el que se establecerá manualmente.

Se crea una instancia de GridSearchCV, que realizará una búsqueda en cuadrícula para encontrar los mejores hiperparámetros para el modelo ComplementNB, utilizando validación cruzada de 3 divisiones (cv=3) para evaluar el rendimiento de cada combinación de hiperparámetros.

In [ ]:
```
X_train=tfIdfVectorizer.transform(train_cv['input'])
y_train=train_cv['NombreDependencia'].values
alphas=[i*0.1 for i in range(1,200,10)]
parameters = {'alpha':alphas, 'norm':[False, True]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier_tercer_componente.joblib')
clf
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
```

Out[ ]:



In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.1, 'norm': False}
0.7539292646249818
```

Complement NB segundo round

En este caso, se afina el clasificador basado en los datos dados anteriormente para este modelo.

In [ ]:
```
alphas=[i*0.1 for i in range(11)]
parameters = {'alpha':alphas, 'norm':[False]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
```

```
dump(clf, 'complementNB_classifier_tercer_componente.joblib')
clf
```

```
Fitting 3 folds for each of 11 candidates, totalling 33 fits
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:1052: RuntimeWarning: divide by zero encountered in log
  logged = np.log(comp_count / comp_count.sum(axis=1, keepdims=True))
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:1052: RuntimeWarning: divide by zero encountered in log
  logged = np.log(comp_count / comp_count.sum(axis=1, keepdims=True))
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:1052: RuntimeWarning: divide by zero encountered in log
  logged = np.log(comp_count / comp_count.sum(axis=1, keepdims=True))
```

Out[ ]:

GridSearchCV

estimator: ComplementNB

ComplementNB

In [ ]:

```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.2, 'norm': False}
0.7541748389619102
```

Multinomial NB

Se crea una instancia de MultinomialNB llamada mnb. y una instancia de GridSearchCV llamada clf, que realizará una búsqueda en cuadrícula para encontrar el mejor valor de alpha para el modelo, utilizando validación cruzada con 3 divisiones (cv=3) para evaluar el rendimiento de cada posible valor de alpha. El parámetro n_jobs=6 indica que se deben usar 6 núcleos de procesador para el cálculo en paralelo, y verbose=1 hace que la función imprima información detallada sobre el progreso del ajuste del modelo.

In [ ]:

```
alphas=[i*0.1 for i in range(0,200,10)]
parameters = {'alpha':alphas}
mnb=MultinomialNB()
clf = GridSearchCV(mnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'multinomialNB_classifier_tercer_componente.joblib')
clf
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:629: FutureWarning: The default value for `force_alpha` will c
hange to `True` in 1.4. To suppress this warning, manually set the value of
`force_alpha`.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:629: FutureWarning: The default value for `force_alpha` will c
hange to `True` in 1.4. To suppress this warning, manually set the value of
`force_alpha`.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:629: FutureWarning: The default value for `force_alpha` will c
hange to `True` in 1.4. To suppress this warning, manually set the value of
`force_alpha`.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
```

Out[ ]:

GridSearchCV

estimator: MultinomialNB

MultinomialNB

In [ ]:

```python
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.0}
0.7335286123194589
```

Multinomial NB segundo round

En este caso, se afina el clasificador basado en los datos dados anteriormente para este modelo.

In [ ]:

```python
alphas=[i*0.1 for i in range(10)]
parameters = {'alpha':alphas}
mnb=MultinomialNB(force_alpha=True)
clf = GridSearchCV(mnb, parameters,cv=3,n_jobs=6,verbose=1)
```

```
clf.fit(X_train,y_train)
dump(clf, 'multinomialNB_classifier_tercer_componente.joblib')
clf
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:907: RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:907: RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:907: RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
```

Out[ ]:

```
GridSearchCV
estimator: MultinomialNB
MultinomialNB
```

In [ ]:

```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.1}
0.751368214713
```

1.4 Cuarto componente

Se crea una copia de las columnas NombreDependencia e input del dataframe data_corregido y se almacena en cuarto_componente.

Luego se agrupan los datos por NombreDependencia y se cuenta el número de instancias para cada dependencia. Esta información se almacena en deps.

Posteriormente, se actualiza deps para conservar solo las filas donde el recuento de input es menor que 7000.

Finalmente, se actualiza cuarto_componente para conservar solo las filas donde NombreDependencia está en la lista de dependencias con menos de 7000 instancias. deps es luego impreso, probablemente para verificar el resultado de estas operaciones.

In [ ]:

```
cuarto_componente=data_corregido[['NombreDependencia','input']].copy()
deps=cuarto_componente[['NombreDependencia','input']].groupby('NombreDepende
deps=deps[deps['input']<7000]
cuarto_componente=cuarto_componente[cuarto_componente['NombreDependencia'].i
deps
```

Out[ ]:

	NombreDependencia	input
7	Prensa_y_Comuni	416
18	Ser_Publicos	577
8	ST	760
2	Direccion	763
5	Ofc_TIC	1331
4	Extra	1338
0	Con_Int_Discipli	1466
6	Pla_Sisben	2900

In [ ]:
```
deps['input']=deps['input']<1000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
cuarto_componente['NombreDependencia']=cuarto_componente['NombreDependencia'
cuarto_componente.groupby('NombreDependencia').count().sort_values(by='input
```

Out[ ]:

NombreDependencia	input
Ofc_TIC	1331
Con_Int_Discipli	1466
Pla_Sisben	2900
Extra	3854

Crea un conjunto de entrenamiento (train_cv) al seleccionar una muestra de cada NombreDependencia en cuarto_componente. El número de muestras seleccionadas para cada NombreDependencia es de 1331.

Crea un 'corpus' de textos para cada NombreDependencia concatenando todas las entradas de la columna input en train_cv y almacenándolo en dataset.

Inicializa un TfidfVectorizer con un mínimo de frecuencia de documento de 1 y un rango de n-gramas de (1,2). Esto significa que se considerarán tanto unigramas como bigramas en el análisis y que todas las palabras serán incluidas, sin importar cuán infrecuentes sean.

Usa el TfidfVectorizer para ajustar y transformar el 'corpus' de texto en dataset, creando una matriz TF-IDF (tfIdf).

Crea un dataframe pandas (df) a partir de la matriz TF-IDF para la primera dependencia, con las palabras como índices y las puntuaciones TF-IDF como valores.

Ordena el dataframe df en función de las puntuaciones TF-IDF en orden descendente, lo que significa que las palabras con la mayor puntuación TF-IDF estarán en la parte superior.

Finalmente, imprime df para mostrar las palabras más relevantes para la primera dependencia en train_cv.

In [ ]:
```
n=1331
train_cv=cuarto_componente.groupby('NombreDependencia',as_index=False).sampl
dataset=train_cv.groupby('NombreDependencia', as_index=False).agg({'input':'
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,2),)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[ ]:

	TF-IDF
bucaramanga	0.334791
solicitud	0.261035
informacion	0.198850
peticion	0.182219
alcalde	0.164864
...	...
despues mes	0.000000
despues meses	0.000000
despues nia	0.000000
despues notificadas	0.000000
zunilda martirena	0.000000

227277 rows × 1 columns

Complement NB

Define una lista de valores alfa para explorar en la búsqueda de cuadrícula. Estos son los valores de regularización que se aplicarán al modelo.

De allí, se define un diccionario de parámetros para la búsqueda de cuadrícula. Estos son los parámetros y sus posibles valores que se explorarán para encontrar la mejor configuración para el modelo.

Inicializa un modelo Complement Naive Bayes y una Búsqueda de Cuadrícula para explorar todas las combinaciones de los parámetros definidos anteriormente.

Ajusta la Búsqueda de Cuadrícula a los datos de entrenamiento. Esto entrenará un modelo Complement Naive Bayes para cada combinación de parámetros y seleccionará el modelo con el mejor rendimiento.

Guarda el modelo entrenado en un archivo joblib, lo que permite cargar y utilizar el modelo en el futuro sin tener que volver a entrenarlo.

In [ ]:
```
X_train=tfIdfVectorizer.transform(train_cv['input'])
y_train=train_cv['NombreDependencia'].values
alphas=[i*0.1 for i in range(1,200,10)]
parameters = {'alpha':alphas, 'norm':[False, True]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier_cuarto_componente.joblib')
clf
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
```

Out[ ]:
```
GridSearchCV
estimator: ComplementNB
ComplementNB
```

In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.1, 'norm': False}
0.7138440372676559
```

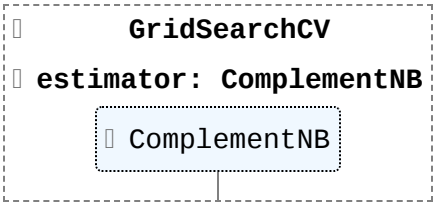
Complement NB segundo round

Se afina el clasificador a partir de los resultados dados anteriormente

In [ ]:
```
alphas=[i*0.1 for i in range(11)]
parameters = {'alpha':alphas, 'norm':[False]}
cnb=ComplementNB(force_alpha=True)
clf = GridSearchCV(cnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'complementNB_classifier_cuarto_componente.joblib')
clf
```

```
Fitting 3 folds for each of 11 candidates, totalling 33 fits
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:1052: RuntimeWarning: divide by zero encountered in log
  logged = np.log(comp_count / comp_count.sum(axis=1, keepdims=True))
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:1052: RuntimeWarning: divide by zero encountered in log
  logged = np.log(comp_count / comp_count.sum(axis=1, keepdims=True))
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:1052: RuntimeWarning: divide by zero encountered in log
  logged = np.log(comp_count / comp_count.sum(axis=1, keepdims=True))
```

Out[ ]:



In [ ]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.4, 'norm': False}
0.7162852724499529
```

Multinomial NB

Se crea una instancia de MultinomialNB llamada mnb. y una instancia de GridSearchCV llamada clf, que realizará una búsqueda en cuadrícula para encontrar el mejor valor de alpha para el modelo, utilizando validación cruzada con 3 divisiones (cv=3) para evaluar el rendimiento de cada posible valor de alpha. El parámetro n_jobs=6 indica que se deben usar 6 núcleos de procesador para el cálculo en paralelo, y verbose=1 hace que la función imprima información detallada sobre el progreso del ajuste del modelo.

In [ ]:
```
alphas=[i*0.1 for i in range(0,200,10)]
parameters = {'alpha':alphas}
mnb=MultinomialNB()
clf = GridSearchCV(mnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
dump(clf, 'multinomialNB_classifier_cuarta_capa.joblib')
clf
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:629: FutureWarning: The default value for `force_alpha` will c
hange to `True` in 1.4. To suppress this warning, manually set the value of
`force_alpha`.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:629: FutureWarning: The default value for `force_alpha` will c
hange to `True` in 1.4. To suppress this warning, manually set the value of
`force_alpha`.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:629: FutureWarning: The default value for `force_alpha` will c
hange to `True` in 1.4. To suppress this warning, manually set the value of
`force_alpha`.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:635: UserWarning: alpha too small will result in numeric error
s, setting alpha = 1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
```

Out[ ]:

GridSearchCV

estimator: MultinomialNB

MultinomialNB

In [ ]:

```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.0}
0.6846363651997455
```

Multinomial NB segundo round

Se afina el clasificador a partir de los resultados anteriores

In [ ]:

```
alphas=[i*0.1 for i in range(10)]
parameters = {'alpha':alphas}
mnb=MultinomialNB(force_alpha=True)
clf = GridSearchCV(mnb, parameters,cv=3,n_jobs=6,verbose=1)
clf.fit(X_train,y_train)
```

```
dump(clf, 'multinomialNB_classifier_cuarta_capa.joblib')
clf
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:907: RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:907: RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/home/n00b1337/Documents/Max/Wil/venv/lib/python3.10/site-packages/sklearn/n
aive_bayes.py:907: RuntimeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
```

Out[ ]:

GridSearchCV

estimator: MultinomialNB

MultinomialNB

In [ ]:

```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'alpha': 0.1}
0.7147825810881066
```

Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

Modelos técnicas tradicionales

Entrenamiento con un diccionario mas grande con los hiperparametros encontrados en el notebook anterior y aplicación de hiperparámetros seleccionados

Estudiantes

Wilfredo Ariel Góme Bueno

Edson Andrés Gómez Cárdenas

In [1]:
```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from pandarallel import pandarallel
pandarallel.initialize(progress_bar=True)
from sklearn.feature_extraction.text import TfidfVectorizer
from joblib import dump, load
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB, ComplementNB,GaussianNB
from sklearn.linear_model import LogisticRegression,LogisticRegressionCV
from sklearn.model_selection import GridSearchCV
```

```
INFO: Pandarallel will run on 8 workers.
INFO: Pandarallel will use Memory file system to transfer data between the m
ain process and workers.
```

Del cuaderno 0. Wrangling, Exploration and Clenaning, se trae el dataseet preprocesado y organizado.

In [13]:
```
data_corregido=pd.read_csv('data_corregida.csv',usecols=['NombreDependencia'
data_corregido
```

Out[13]:

	NombreDependencia	input
0	Sec_Admin	insatisfaccion pan recibido
1	Sec_Admin	queja contaminacion visual calle 45 carrera 1...
2	Sec_Admin	prueba pqrs 05102011
3	Sec_Infraest	cra 5 n 57 59 barrio porvenir 2 luminarias fu...
4	Sec_Admin	solicitud
...	...	...
590587	Sec_Hacienda	formulario declaracion ica registro presentaci...
590588	Sec_Hacienda	formulario declaracion ica registro presentaci...
590589	Sec_Des_social	solicitud oferta institucional seores secretar...
590590	Sec_Hacienda	formulario declaracion ica registro presentaci...
590591	Espacio_Publico	incompetentes espacio publico viernes 9 noche ...

590592 rows × 2 columns

1. Primer Componente

Para el primer componente, se aplica los hiperparámetros detectados en el cuaderno 2. Selección Hiperparámetros.

In [14]:
```
primer_componente=data_corregido.copy()
primer_componente.loc[primera_capa['NombreDependencia']!='Sec_Hacienda','Nom
primer_componente.groupby('NombreDependencia').count()
```

Out[14]:

	input
NombreDependencia	
Sec_Hacienda	317272
otro	273320

In [ ]:
```
n=int(273320*0.8) #para training el resto para test
train=primer_componente.groupby('NombreDependencia',as_index=False).sample(n
test=primer_componente.drop(train.index)
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,2))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

In [6]:
```
dump(tfIdfVectorizer, 'tfidf_primera_capa.joblib')
tfIdfVectorizer = load('tfidf_primera_capa.joblib')
```

1.1 Logistic Regressor Classifier

In [17]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
l1_best=0.55
C_best=3.55
lrc=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=7,verbose=1,max_iter
lrc.fit(X_train,y_train)
dump(lrc, 'logistic_regression_classifier_best.joblib')
lrc
```

```
Epoch 42, change: 0.00029442
Epoch 52, change: 0.00014719
Epoch 43, change: 0.00026449
Epoch 53, change: 0.00013056
Epoch 44, change: 0.00023315
Epoch 54, change: 0.00011719
Epoch 45, change: 0.00020566
Epoch 55, change: 0.00010455
Epoch 46, change: 0.00018461
convergence after 56 epochs took 2864 seconds
Epoch 47, change: 0.00016280
Epoch 48, change: 0.00014541
Epoch 49, change: 0.00013068
Epoch 50, change: 0.00011503
Epoch 51, change: 0.00010133
convergence after 52 epochs took 2598 seconds
[Parallel(n_jobs=7)]: Done   1 out of   1 | elapsed: 43.3min finished
```

Out[17]:
```
 LogisticRegression
```

In [18]:
```
X_test=tfIdfVectorizer.transform(test['input'])
y_test=test['NombreDependencia'].values
print('Score:',lrc.score(X_test,y_test))
```

```
Score: 0.9482972338204593
```

Resultado

Score 98%

1.2 ComplementNB

In [13]:
```
alpha_best=0.1
norm_best = True
cnb=ComplementNB(force_alpha=True,alpha=alpha_best,norm=norm_best)
cnb.fit(X_train,y_train)
dump(cnb, 'complementNB_classifier_best.joblib')
cnb
```

Out[13]:
```
 ComplementNB
```

In [14]:
```
print('Score:',cnb.score(X_test,y_test))
```

```
Score: 0.9424908663883089
```

Resultado

Score 94.2%

2. Segunda Componente

Se entrena clasificadores con los hiperparámetros seleccionados en el cuaderno 2. Selección de Hiperparámetros

In [3]:
```
segundo_componente=data_corregido.copy()
segundo_componente=segundo_componente.loc[segundo_componente['NombreDependen
segundo_componente.groupby('NombreDependencia').count().sort_values(by='inpu
deps=segundo_componente.groupby('NombreDependencia',as_index=False).count().
deps['input']=deps['input']<35000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
segundo_componente['NombreDependencia']=segundo_componentea['NombreDependenc
segundo_componente.groupby('NombreDependencia').count().sort_values(by='inpu
```

Out[3]:

NombreDependencia	input
Sec_Planeacion	35339
Sec_Salud_y_ambi	41269
Sec_Admin	43680
Sec_Interior	47370
Extra	105662

In [4]:
```
n=int(35339*0.8)
train=segundo_componente.groupby('NombreDependencia',as_index=False).sample(
test=segundo_componente.drop(train.index)
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(min_df=2,ngram_range=(1,3))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[4]:

	TF-IDF
solicitud	0.413317
bucaramanga	0.345057
informacion	0.170304
peticion	0.163075
respuesta	0.139058
...	...
fotos recibo	0.000000
fotos seor	0.000000
fotos solicito	0.000000
fotos solicitud informacion	0.000000
zx ventures colombia	0.000000

405255 rows × 1 columns

In [52]:
```
dump(tfIdfVectorizer, 'tfidf_segundo_componente.joblib')
tfIdfVectorizer = load('tfidf_segundo_componente.joblib')
```

2.1 ComplementNB - Resultado

In [144…
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
alpha_best=0.6
norm_best = True
cnb=ComplementNB(force_alpha=True,alpha=alpha_best,norm=norm_best)
cnb.fit(X_train,y_train)
dump(cnb, 'complementNB_classifier_segundo_componente_best.joblib')
print(cnb)
X_test=tfIdfVectorizer.transform(test['input'])
y_test=test['NombreDependencia'].values
print('Score:',cnb.score(X_test,y_test))
```

```
ComplementNB(alpha=0.6, force_alpha=True, norm=True)
Score: 0.7046034933505096
```

2.2 MultinomialNB - Resultado

In [55]:
```
alpha_best=0.2
mnb=MultinomialNB(alpha=alpha_best)
mnb.fit(X_train,y_train)
dump(mnb, 'multinomialNB_classifier_segundo_componente_best.joblib')
print(mnb)
print('Score:',mnb.score(X_test,y_test))
```

```
MultinomialNB(alpha=0.2)
Score: 0.7211988027128405
```

2.3 Logistic Regressor - Resultado

In [8]:
```
l1s_=[0.1*i for i in range(2,9)]
Cs_=[0.5*i for i in range(2,8)]
parameters = {'l1_ratio':l1s_, 'C':Cs_}
lrc=LogisticRegression(penalty='elasticnet',solver='saga')
clf = GridSearchCV(lrc, parameters,cv=3,n_jobs=8,verbose=2)
x=segunda_capa

X_train=tfIdfVectorizer.transform(x['input'].values)
y_train=segunda_capa['NombreDependencia'].values
clf.fit(X_train,y_train)
dump(clf, 'logistic_regression_classifier_segundo_componente_best.joblib')
print(clf)
print(clf.best_params_)
print(clf.best_score_)
```

```
Fitting 3 folds for each of 42 candidates, totalling 126 fits
[CV] END ..............................C=1.0, l1_ratio=0.4; total time=12.
6min
[CV] END ..............................C=1.0, l1_ratio=0.4; total time=14.
2min
[CV] END ..............C=1.0, l1_ratio=0.30000000000000004; total time=19.
3min
[CV] END ..............C=1.0, l1_ratio=0.30000000000000004; total time=20.
3min
[CV] END ..............C=1.0, l1_ratio=0.30000000000000004; total time=21.
5min
[CV] END ..............................C=1.0, l1_ratio=0.5; total time= 9.
7min
[CV] END ..............................C=1.0, l1_ratio=0.4; total time=13.
8min
[CV] END ..............................C=1.0, l1_ratio=0.5; total time= 9.
0min
[CV] END ..............C=1.0, l1_ratio=0.6000000000000001; total time= 6.
8min
[CV] END ..............................C=1.0, l1_ratio=0.5; total time= 9.
7min
[CV] END ..............C=1.0, l1_ratio=0.6000000000000001; total time= 8.
3min
[CV] END ..............................C=1.0, l1_ratio=0.2; total time=33.
5min
[CV] END ..............C=1.0, l1_ratio=0.6000000000000001; total time= 7.
3min
[CV] END ..............C=1.0, l1_ratio=0.7000000000000001; total time= 6.
9min
[CV] END ..............C=1.0, l1_ratio=0.7000000000000001; total time= 6.
9min
[CV] END ..............C=1.0, l1_ratio=0.7000000000000001; total time= 6.
5min
[CV] END ..............................C=1.0, l1_ratio=0.2; total time=36.
5min
[CV] END ..............................C=1.0, l1_ratio=0.2; total time=37.
1min
[CV] END ..............................C=1.0, l1_ratio=0.8; total time= 6.
9min
[CV] END ..............................C=1.0, l1_ratio=0.8; total time= 7.
1min
[CV] END ..............................C=1.0, l1_ratio=0.8; total time= 7.
7min
[CV] END ..............................C=1.5, l1_ratio=0.4; total time=21.
8min
[CV] END ..............................C=1.5, l1_ratio=0.4; total time=24.
5min
[CV] END ..............C=1.5, l1_ratio=0.30000000000000004; total time=34.
1min
[CV] END ..............C=1.5, l1_ratio=0.30000000000000004; total time=35.
9min
[CV] END ..............C=1.5, l1_ratio=0.30000000000000004; total time=33.
6min
[CV] END ..............................C=1.5, l1_ratio=0.5; total time=17.
6min
[CV] END ..............................C=1.5, l1_ratio=0.4; total time=21.
```

```
9min
[CV] END .................C=1.5, l1_ratio=0.6000000000000001; total time=15.
3min
[CV] END ..............................C=1.5, l1_ratio=0.2; total time=53.
6min
[CV] END ..............................C=1.5, l1_ratio=0.5; total time=17.
0min
[CV] END ..............................C=1.5, l1_ratio=0.5; total time=18.
4min
[CV] END ..............................C=1.5, l1_ratio=0.2; total time=56.
2min
[CV] END ..............................C=1.5, l1_ratio=0.2; total time=55.
3min
[CV] END .................C=1.5, l1_ratio=0.6000000000000001; total time=15.
7min
[CV] END .................C=1.5, l1_ratio=0.6000000000000001; total time=14.
5min
[CV] END .................C=1.5, l1_ratio=0.7000000000000001; total time=13.
5min
[CV] END ..............................C=1.5, l1_ratio=0.8; total time=13.
1min
[CV] END .................C=1.5, l1_ratio=0.7000000000000001; total time=13.
5min
[CV] END .................C=1.5, l1_ratio=0.7000000000000001; total time=14.
1min
[CV] END ..............................C=1.5, l1_ratio=0.8; total time=14.
0min
[CV] END ..............................C=1.5, l1_ratio=0.8; total time=14.
9min
[CV] END ..............................C=2.0, l1_ratio=0.4; total time=35.
4min
[CV] END ..............................C=2.0, l1_ratio=0.4; total time=37.
6min
[CV] END .................C=2.0, l1_ratio=0.30000000000000004; total time=51.
4min
[CV] END .................C=2.0, l1_ratio=0.30000000000000004; total time=53.
3min
[CV] END .................C=2.0, l1_ratio=0.30000000000000004; total time=54.
0min
[CV] END ..............................C=2.0, l1_ratio=0.5; total time=29.
3min
[CV] END ..............................C=2.0, l1_ratio=0.2; total time=78.
1min
[CV] END ..............................C=2.0, l1_ratio=0.4; total time=36.
9min
[CV] END .................C=2.0, l1_ratio=0.6000000000000001; total time=23.
5min
[CV] END ..............................C=2.0, l1_ratio=0.2; total time=79.
3min
[CV] END ..............................C=2.0, l1_ratio=0.2; total time=83.
7min
[CV] END ..............................C=2.0, l1_ratio=0.5; total time=27.
6min
[CV] END ..............................C=2.0, l1_ratio=0.5; total time=31.
4min
[CV] END .................C=2.0, l1_ratio=0.6000000000000001; total time=25.
```

```
4min
[CV] END .................C=2.0, l1_ratio=0.7000000000000001; total time=21.
1min
[CV] END .................C=2.0, l1_ratio=0.6000000000000001; total time=23.
2min
[CV] END .................C=2.0, l1_ratio=0.7000000000000001; total time=22.
7min
[CV] END ...............................C=2.0, l1_ratio=0.8; total time=21.
2min
[CV] END .................C=2.0, l1_ratio=0.7000000000000001; total time=23.
3min
[CV] END ...............................C=2.0, l1_ratio=0.8; total time=24.
6min
[CV] END ...............................C=2.0, l1_ratio=0.8; total time=24.
1min
[CV] END ...............................C=2.5, l1_ratio=0.4; total time=53.
8min
[CV] END ...............................C=2.5, l1_ratio=0.4; total time=56.
3min
[CV] END .................C=2.5, l1_ratio=0.30000000000000004; total time=70.
4min
[CV] END .................C=2.5, l1_ratio=0.30000000000000004; total time=73.
6min
[CV] END .................C=2.5, l1_ratio=0.30000000000000004; total time=74.
3min
[CV] END ...............................C=2.5, l1_ratio=0.5; total time=40.
7min
[CV] END ...............................C=2.5, l1_ratio=0.2; total time=108.
9min
[CV] END ...............................C=2.5, l1_ratio=0.2; total time=109.
2min
[CV] END .................C=2.5, l1_ratio=0.6000000000000001; total time=34.
2min
[CV] END ...............................C=2.5, l1_ratio=0.5; total time=40.
4min
[CV] END ...............................C=2.5, l1_ratio=0.4; total time=55.
7min
[CV] END ...............................C=2.5, l1_ratio=0.5; total time=45.
2min
[CV] END ...............................C=2.5, l1_ratio=0.2; total time=120.
7min
[CV] END .................C=2.5, l1_ratio=0.7000000000000001; total time=31.
4min
[CV] END .................C=2.5, l1_ratio=0.6000000000000001; total time=33.
9min
[CV] END .................C=2.5, l1_ratio=0.6000000000000001; total time=39.
1min
[CV] END .................C=2.5, l1_ratio=0.7000000000000001; total time=34.
5min
[CV] END ...............................C=2.5, l1_ratio=0.8; total time=32.
5min
[CV] END .................C=2.5, l1_ratio=0.7000000000000001; total time=33.
5min
[CV] END ...............................C=2.5, l1_ratio=0.8; total time=36.
5min
[CV] END ...............................C=2.5, l1_ratio=0.8; total time=36.
```

```
8min
[CV] END ..............................C=3.0, l1_ratio=0.4; total time=68.
4min
[CV] END ..............................C=3.0, l1_ratio=0.4; total time=73.
3min
[CV] END ..............C=3.0, l1_ratio=0.30000000000000004; total time=95.
3min
[CV] END ..............C=3.0, l1_ratio=0.30000000000000004; total time=98.
8min
[CV] END ..............C=3.0, l1_ratio=0.30000000000000004; total time=98.
8min
[CV] END ..............................C=3.0, l1_ratio=0.2; total time=138.
3min
[CV] END ..............................C=3.0, l1_ratio=0.2; total time=135.
1min
[CV] END ..............................C=3.0, l1_ratio=0.5; total time=54.
5min
[CV] END ..............................C=3.0, l1_ratio=0.2; total time=146.
9min
[CV] END ..............................C=3.0, l1_ratio=0.4; total time=68.
1min
[CV] END ..............C=3.0, l1_ratio=0.6000000000000001; total time=46.
3min
[CV] END ..............................C=3.0, l1_ratio=0.5; total time=61.
1min
[CV] END ..............................C=3.0, l1_ratio=0.5; total time=55.
0min
[CV] END ..............C=3.0, l1_ratio=0.7000000000000001; total time=42.
7min
[CV] END ..............C=3.0, l1_ratio=0.6000000000000001; total time=48.
7min
[CV] END ..............C=3.0, l1_ratio=0.6000000000000001; total time=52.
2min
[CV] END ..............C=3.0, l1_ratio=0.7000000000000001; total time=48.
7min
[CV] END ..............C=3.0, l1_ratio=0.7000000000000001; total time=46.
1min
[CV] END ..............................C=3.0, l1_ratio=0.8; total time=45.
2min
[CV] END ..............................C=3.0, l1_ratio=0.8; total time=50.
5min
[CV] END ..............................C=3.0, l1_ratio=0.8; total time=52.
2min
[CV] END ..............................C=3.5, l1_ratio=0.4; total time=88.
0min
[CV] END ..............................C=3.5, l1_ratio=0.4; total time=100.
1min
[CV] END ..............C=3.5, l1_ratio=0.30000000000000004; total time=118.
2min
[CV] END ..............C=3.5, l1_ratio=0.30000000000000004; total time=116.
2min
[CV] END ..............C=3.5, l1_ratio=0.30000000000000004; total time=122.
4min
[CV] END ..............................C=3.5, l1_ratio=0.2; total time=160.
4min
[CV] END ..............................C=3.5, l1_ratio=0.2; total time=178.
```

```
7min
[CV] END ..............................C=3.5, l1_ratio=0.2; total time=179.
2min
[CV] END .................C=3.5, l1_ratio=0.6000000000000001; total time=63.
1min
[CV] END ..............................C=3.5, l1_ratio=0.5; total time=69.
1min
[CV] END ..............................C=3.5, l1_ratio=0.5; total time=72.
6min
[CV] END ..............................C=3.5, l1_ratio=0.4; total time=88.
4min
[CV] END ..............................C=3.5, l1_ratio=0.5; total time=81.
8min
[CV] END .................C=3.5, l1_ratio=0.6000000000000001; total time=71.
4min
[CV] END .................C=3.5, l1_ratio=0.7000000000000001; total time=59.
3min
[CV] END .................C=3.5, l1_ratio=0.6000000000000001; total time=64.
1min
[CV] END .................C=3.5, l1_ratio=0.7000000000000001; total time=61.
5min
[CV] END ..............................C=3.5, l1_ratio=0.8; total time=56.
5min
[CV] END .................C=3.5, l1_ratio=0.7000000000000001; total time=63.
4min
[CV] END ..............................C=3.5, l1_ratio=0.8; total time=62.
8min
[CV] END ..............................C=3.5, l1_ratio=0.8; total time=58.
3min
GridSearchCV(cv=3,
             estimator=LogisticRegression(penalty='elasticnet', solver='sag
a'),
             n_jobs=8,
             param_grid={'C': [1.0, 1.5, 2.0, 2.5, 3.0, 3.5],
                         'l1_ratio': [0.2, 0.30000000000000004, 0.4, 0.5,
                                      0.6000000000000001, 0.700000000000000
1,
                                      0.8]},
             verbose=2)
{'C': 1.5, 'l1_ratio': 0.2}
0.72419526832002
```

3. Tercera componente

Se entrena clasificadores con los hiperparámetros seleccionados en el cuaderno 2.

Selección de Hiperparámetros

In [19]:
```
tercer_componente=data_corregido.copy()
deps=tercer_componente[['NombreDependencia','input']].groupby('NombreDepende
deps=deps[deps['input']<35000]
tercer_componente=tercer_componente[tercer_componente['NombreDependencia'].i
deps['input']=deps['input']<7000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
tercer_componente['NombreDependencia']=tercer_componente['NombreDependencia'
```

```
n=int(7126*0.9)
train=tercer_componente.groupby('NombreDependencia',as_index=False).sample(n
test=tercer_componente.drop(train.index)
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,3))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[19]:

	TF-IDF
bucaramanga	0.310523
solicitud	0.293271
peticion	0.215465
informacion	0.210184
alcalde	0.193109
...	...
dia 23 mayo	0.000000
dia 23 maria	0.000000
dia 23 junio21	0.000000
dia 23 junio	0.000000
zunilda martirena janiot	0.000000

1802300 rows × 1 columns

In [106…

```
dump(tfIdfVectorizer, 'tfidf_tercera_capa.joblib')
tfIdfVectorizer = load('tfidf_tercera_capa.joblib')
```

3.1 Complement NB - Resultado

In [5]:

```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
alpha_best=0.2
norm_best = False
cnb=ComplementNB(force_alpha=True,alpha=alpha_best,norm=norm_best)
cnb.fit(X_train,y_train)
dump(cnb, 'complementNB_classifier_tercera_capa_best.joblib')
print(cnb)
X_test=tfIdfVectorizer.transform(test['input'])
y_test=test['NombreDependencia'].values
print('Score:',cnb.score(X_test,y_test))
```

```
ComplementNB(alpha=0.2, force_alpha=True)
Score: 0.7244563817653336
```

3.2 Multinomial NB - Resultado

In [108…

```
alpha_best=0.1
mnb=MultinomialNB(alpha=alpha_best)
mnb.fit(X_train,y_train)
dump(mnb, 'multinomialNB_classifier_tercera_capa_best.joblib')
print(mnb)
print('Score:',mnb.score(X_test,y_test))
```

```
MultinomialNB(alpha=0.1)
Score: 0.7175392766474116
```

3.3 Logistic Regressor - Resultado

In [23]:

```
l1s_=[0.1*i for i in range(2,9)]
Cs_=[0.5*i for i in range(2,8)]
parameters = {'l1_ratio':l1s_, 'C':Cs_}
lrc=LogisticRegression(penalty='elasticnet',solver='saga')
clf = GridSearchCV(lrc, parameters,cv=3,n_jobs=8,verbose=2)
x=tercera_capa

X_train=tfIdfVectorizer.transform(x['input'].values)
y_train=tercera_capa['NombreDependencia'].values
clf.fit(X_train,y_train)
dump(clf, 'logistic_regression_classifier_tercera_capa_best.joblib')
print(clf)
print(clf.best_params_)
print(clf.best_score_)
```

```
[CV] END ................C=3.5, l1_ratio=0.6000000000000001; total time=14.
8min
[CV] END ..............................C=3.5, l1_ratio=0.2; total time=60.
1min
[CV] END ................C=3.5, l1_ratio=0.7000000000000001; total time=13.
0min
[CV] END ................C=3.5, l1_ratio=0.6000000000000001; total time=14.
9min
[CV] END ................C=3.5, l1_ratio=0.6000000000000001; total time=16.
4min
[CV] END ..............................C=3.5, l1_ratio=0.8; total time=12.
8min
[CV] END ................C=3.5, l1_ratio=0.7000000000000001; total time=15.
0min
[CV] END ................C=3.5, l1_ratio=0.7000000000000001; total time=13.
4min
[CV] END ..............................C=3.5, l1_ratio=0.2; total time=72.
6min
[CV] END ..............................C=3.5, l1_ratio=0.8; total time=13.
4min
[CV] END ..............................C=3.5, l1_ratio=0.8; total time= 9.
5min
GridSearchCV(cv=3,
             estimator=LogisticRegression(penalty='elasticnet', solver='sag
a'),
             n_jobs=8,
             param_grid={'C': [1.0, 1.5, 2.0, 2.5, 3.0, 3.5],
                         'l1_ratio': [0.2, 0.30000000000000004, 0.4, 0.5,
                                      0.6000000000000001, 0.700000000000000
1,
                                      0.8]},
             verbose=2)
{'C': 3.5, 'l1_ratio': 0.30000000000000004}
0.6768752784229966
```

In [30]:

```
n=int(7126*0.9)
train=tercera_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
test=tercera_capa.drop(train.index)
lrc=LogisticRegression(C=3.5,l1_ratio=0.3,penalty='elasticnet',solver='saga'
X_train=tfIdfVectorizer.transform(train['input'].values)
y_train=train['NombreDependencia'].values
lrc.fit(X_train,y_train)

X_test=tfIdfVectorizer.transform(test['input'].values)
y_test=test['NombreDependencia'].values
lrc.score(X_test,y_test)
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
Epoch 1, change: 1.00000000
Epoch 2, change: 0.11609439
Epoch 3, change: 0.07756278
Epoch 4, change: 0.07159479
Epoch 5, change: 0.04478771
Epoch 6, change: 0.04981822
Epoch 7, change: 0.04077620
Epoch 8, change: 0.02839304
Epoch 9, change: 0.03605028
Epoch 10, change: 0.01817341
Epoch 11, change: 0.02108932
Epoch 12, change: 0.00824103
Epoch 13, change: 0.01695385
Epoch 14, change: 0.00613609
Epoch 15, change: 0.00453604
Epoch 16, change: 0.00404247
Epoch 17, change: 0.00312860
Epoch 18, change: 0.00272989
Epoch 19, change: 0.00230623
Epoch 20, change: 0.00182542
Epoch 21, change: 0.00167675
Epoch 22, change: 0.00122430
Epoch 23, change: 0.00105439
Epoch 24, change: 0.00088653
Epoch 25, change: 0.00075462
Epoch 26, change: 0.00066331
Epoch 27, change: 0.00047746
Epoch 28, change: 0.00045205
Epoch 29, change: 0.00039233
Epoch 30, change: 0.00026064
Epoch 31, change: 0.00025031
Epoch 32, change: 0.00020870
Epoch 33, change: 0.00017580
Epoch 34, change: 0.00014400
Epoch 35, change: 0.00012577
Epoch 36, change: 0.00010448
convergence after 37 epochs took 389 seconds
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed:  6.5min finished
```

Out[30]: `0.7273262445270245`

3.4 SVC - Resultado

In [24]:
```
kernels=['linear', 'poly', 'rbf', 'sigmoid']
parameters = {'kernel':kernels}
lrc=SVC()
clf = GridSearchCV(lrc, parameters,cv=3,n_jobs=8,verbose=2)
x=tercera_capa

X_train=tfIdfVectorizer.transform(x['input'].values)
y_train=tercera_capa['NombreDependencia'].values
clf.fit(X_train,y_train)
dump(clf, 'svm_classifier_tercera_capa_best.joblib')
```

```
print(clf)
print(clf.best_params_)
print(clf.best_score_)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV] END ....................................kernel=linear; total time=60.
3min
[CV] END ....................................kernel=linear; total time=77.
5min
[CV] END ....................................kernel=linear; total time=86.
1min
[CV] END ...................................kernel=sigmoid; total time=37.
5min
[CV] END ...................................kernel=sigmoid; total time=43.
5min
[CV] END .....................................kernel=rbf; total time=131.
9min
[CV] END .....................................kernel=rbf; total time=132.
1min
[CV] END ....................................kernel=poly; total time=136.
8min
[CV] END ...................................kernel=sigmoid; total time=30.
2min
[CV] END .....................................kernel=rbf; total time=88.
0min
[CV] END ....................................kernel=poly; total time=169.
4min
[CV] END ....................................kernel=poly; total time=177.
9min
GridSearchCV(cv=3, estimator=SVC(), n_jobs=8,
             param_grid={'kernel': ['linear', 'poly', 'rbf', 'sigmoid']},
             verbose=2)
{'kernel': 'linear'}
0.6681398840236197
```

In [31]:
```
n=int(7126*0.9)
train=tercera_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
test=tercera_capa.drop(train.index)
svc=SVC(kernel='linear',)
X_train=tfIdfVectorizer.transform(train['input'].values)
y_train=train['NombreDependencia'].values
svc.fit(X_train,y_train)

X_test=tfIdfVectorizer.transform(test['input'].values)
y_test=test['NombreDependencia'].values
svc.score(X_test,y_test)
```

Out[31]:
```
0.7264064167187902
```

4. Cuarta Componente

Se entrena clasificador con hiperparámetros seleccionado en el cuaderno 2. Selección Hiperparámetros

In [32]:
```
cuarta_capa=data_corregido.copy()
deps=cuarta_capa[['NombreDependencia','input']].groupby('NombreDependencia',
deps=deps[deps['input']<7000]
cuarta_capa=cuarta_capa[cuarta_capa['NombreDependencia'].isin(deps['NombreDe
deps['input']=deps['input']<1000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
cuarta_capa['NombreDependencia']=cuarta_capa['NombreDependencia'].replace(de
cuarta_capa.groupby('NombreDependencia').count().sort_values(by='input')[['i
```

Out[32]:

NombreDependencia	input
Ofc_TIC	1331
Con_Int_Discipli	1466
Pla_Sisben	2900
Extra	3854

In [33]:
```
n=int(1331*0.9)
train=cuarta_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
test=cuarta_capa.drop(train.index)
dataset=train.groupby('NombreDependencia', as_index=False).agg({'input':' '.
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,3),)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(), index=tfIdfVectorizer.get_feature_na
df = df.sort_values('TF-IDF', ascending=False)
df
```

Out[33]:

	TF-IDF
solicitud	0.229979
bucaramanga	0.209609
informacion	0.188582
disciplinaria	0.167772
respuesta	0.158357
...	...
electrica region	0.000000
electrica region garantizar	0.000000
electrica region proximo	0.000000
electrica region solicita	0.000000
zykagradezco toda ayuda	0.000000

241033 rows × 1 columns

4.1 Complement NB - Resultado

In [124…
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
alpha_best=0.4
norm_best = False
cnb=ComplementNB(force_alpha=True,alpha=alpha_best,norm=norm_best)
cnb.fit(X_train,y_train)
dump(cnb, 'complementNB_classifier_cuarta_capa_best.joblib')
print(cnb)
X_test=tfIdfVectorizer.transform(test['input'])
y_test=test['NombreDependencia'].values
print('Score:',cnb.score(X_test,y_test))
```

```
ComplementNB(alpha=0.4, force_alpha=True)
Score: 0.8431660718034852
```

4.2 Multinomial NB - Resultado

In [125…
```
alpha_best=0.1
mnb=MultinomialNB(alpha=alpha_best)
mnb.fit(X_train,y_train)
dump(mnb, 'multinomialNB_classifier_cuarta_capa_best.joblib')
print(mnb)
print('Score:',mnb.score(X_test,y_test))
```

```
MultinomialNB(alpha=0.1)
Score: 0.8416964098257401
```

4.3 Logistic Regressor - Resultado

In [123…
```
l1_best=0.55
C_best=3.55
lrc=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_iter
lrc.fit(X_train,y_train)
dump(lrc, 'logistic_regression_classifier_cuarta_capa_best.joblib')
print(lrc)
print('Score:',lrc.score(X_test,y_test))
```

```
/home/maxi/Documents/Wil/venv/lib/python3.10/site-packages/sklearn/linear_mo
del/_logistic.py:1165: UserWarning: l1_ratio parameter is only used when pen
alty is 'elasticnet'. Got (penalty=l2)
  warnings.warn(
[Parallel(n_jobs=8)]: Using backend LokyBackend with 8 concurrent workers.
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =     14380640     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.06684D+05    |proj g|=  5.45975D+02
 This problem is unconstrained.

At iterate   50    f=  4.46057D+04    |proj g|=  5.49684D+02

At iterate  100    f=  3.46443D+04    |proj g|=  1.86068D+02

At iterate  150    f=  3.27570D+04    |proj g|=  9.38441D+01

At iterate  200    f=  3.24576D+04    |proj g|=  6.59163D+01

At iterate  250    f=  3.24112D+04    |proj g|=  1.01036D+01

At iterate  300    f=  3.24023D+04    |proj g|=  6.01104D+00

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
*****    300    316      1     0     0   6.011D+00   3.240D+04
  F =   32402.315748094195

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT
/home/maxi/Documents/Wil/venv/lib/python3.10/site-packages/sklearn/linear_mo
del/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
```

```
  n_iter_i = _check_optimize_result(
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed:  8.3min finished
LogisticRegression(C=3.55, l1_ratio=0.55, max_iter=300, n_jobs=8, verbose=1)
Score: 0.741712351447809
```

4.4 SVC - Resultado

In [27]:
```
kernels=['linear', 'poly', 'rbf', 'sigmoid']
parameters = {'kernel':kernels}
lrc=SVC()
clf = GridSearchCV(lrc, parameters,cv=3,n_jobs=8,verbose=2)
x=cuarta_capa

X_train=tfIdfVectorizer.transform(x['input'].values)
y_train=cuarta_capa['NombreDependencia'].values
clf.fit(X_train,y_train)
dump(clf, 'svm_classifier_tercera_capa_best.joblib')
print(clf)
print(clf.best_params_)
print(clf.best_score_)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV] END .....................................kernel=linear; total time=  1
6.8s
[CV] END .....................................kernel=linear; total time=  1
7.0s
[CV] END .....................................kernel=linear; total time=  2
1.1s
[CV] END .......................................kernel=rbf; total time=  2
4.3s
[CV] END .......................................kernel=rbf; total time=  2
9.5s
[CV] END .....................................kernel=poly; total time=  3
0.3s
[CV] END .....................................kernel=poly; total time=  3
2.0s
[CV] END ..................................kernel=sigmoid; total time=  1
5.0s
[CV] END .....................................kernel=poly; total time=  3
5.1s
[CV] END ..................................kernel=sigmoid; total time=  1
3.2s
[CV] END ..................................kernel=sigmoid; total time=  1
6.8s
[CV] END .......................................kernel=rbf; total time=  2
2.9s
GridSearchCV(cv=3, estimator=SVC(), n_jobs=8,
             param_grid={'kernel': ['linear', 'poly', 'rbf', 'sigmoid']},
             verbose=2)
{'kernel': 'linear'}
0.7555269014461774
```

In [34]:
```
n=int(1331*0.9)
train=cuarta_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
test=cuarta_capa.drop(train.index)
```

```
svc=SVC(kernel='linear',)
X_train=tfIdfVectorizer.transform(train['input'].values)
y_train=train['NombreDependencia'].values
svc.fit(X_train,y_train)

X_test=tfIdfVectorizer.transform(test['input'].values)
y_test=test['NombreDependencia'].values
svc.score(X_test,y_test)
```

Out[34]: 0.8406466512702079

4.5 Random Forest Classifier - Resultado

In [128…

```
n_estimators_best=180
criterion_best='gini'
max_features_best='sqrt'
rfc=RandomForestClassifier(n_estimators=n_estimators_best,criterion=criterio
rfc.fit(X_train,y_train)
dump(rfc, 'randomForest_classifier_cuarta_capa_best.joblib')
print(rfc)
print('Score:',rfc.score(X_test,y_test))
```

```
building tree 176 of 180
building tree 177 of 180
building tree 178 of 180
building tree 179 of 180
building tree 180 of 180
[Parallel(n_jobs=8)]: Done 180 out of 180 | elapsed:   41.4s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
[Parallel(n_jobs=8)]: Done  25 tasks      | elapsed:    0.0s
RandomForestClassifier(n_estimators=180, n_jobs=8, verbose=2)
Score: 0.7904681923157674
[Parallel(n_jobs=8)]: Done 146 tasks      | elapsed:    0.2s
[Parallel(n_jobs=8)]: Done 180 out of 180 | elapsed:    0.2s finished
```

Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

Modelos técnicas tradicionales

Pruebas a 64.632 solicitudes no utilizadas en el conjunto de entrenamiento

Estudiantes

Wilfredo Ariel Góme Bueno

Edson Andrés Gómez Cárdenas

In [1]:
```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from pandarallel import pandarallel
pandarallel.initialize(progress_bar=True)
from sklearn.feature_extraction.text import TfidfVectorizer
from joblib import dump, load
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB, ComplementNB,GaussianNB
from sklearn.linear_model import LogisticRegression,LogisticRegressionCV
from sklearn.model_selection import GridSearchCV
```

```
INFO: Pandarallel will run on 4 workers.
INFO: Pandarallel will use standard multiprocessing data transfer (pipe) to
transfer data between the main process and workers.

WARNING: You are on Windows. If you detect any issue with pandarallel, be su
re you checked out the Troubleshooting page:
https://nalepae.github.io/pandarallel/troubleshooting/
```

Se importa los datos ya preprocesados

In [6]:
```
data_corregido=pd.read_csv('data_corregida.csv', sep= ';' ,usecols=['NombreD
data_corregido
```

Out[6]:

	NombreDependencia	input
0	Sec_Admin	insatisfaccion pan recibido
1	Sec_Admin	queja contaminacion visual calle 45 carrera 1...
2	Sec_Admin	prueba pqrs 05102011
3	Sec_Infraest	cra 5 n 57 59 barrio porvenir 2 luminarias fu...
4	Sec_Admin	solicitud
...	...	...
590587	Sec_Hacienda	formulario declaracion ica registro presentaci...
590588	Sec_Hacienda	formulario declaracion ica registro presentaci...
590589	Sec_Des_social	solicitud oferta institucional seores secretar...
590590	Sec_Hacienda	formulario declaracion ica registro presentaci...
590591	Espacio_Publico	incompetentes espacio publico viernes 9 noche ...

590592 rows × 2 columns

1. Primera capa

Se cargan los modelos de la capa

In [3]:
```
primera_capa=data_corregido.copy()
primera_capa.loc[primera_capa['NombreDependencia']!='Sec_Hacienda','NombreDe
n=int(273320)
train1=primera_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=primera_capa.groupby('NombreDependencia', as_index=False).agg({'inpu
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,2))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_primera_capa_final.joblib')
tfIdfVectorizer = load('tfidf_primera_capa_final.joblib')
```

In [4]:
```
train1.to_csv('train_primera_capa', index=False)
```

In [5]:
```
X_train=tfIdfVectorizer.transform(train1['input'])
y_train=train1['NombreDependencia'].values
l1_best=0.55
C_best=3.55
capa1=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_it
capa1.fit(X_train,y_train)
dump(capa1, 'capa1_final.joblib')
capa1 = load('capa1_final.joblib')
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
convergence after 53 epochs took 5680 seconds
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed: 94.7min finished
```

2. Segunda capa

Se cargan los modelos de la capa

In [6]:
```
segunda_capa=data_corregido.copy()
segunda_capa=segunda_capa.loc[segunda_capa['NombreDependencia']!='Sec_Hacien
segunda_capa.groupby('NombreDependencia').count().sort_values(by='input')
deps=segunda_capa.groupby('NombreDependencia',as_index=False).count().sort_v
deps['input']=deps['input']<35000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
segunda_capa['NombreDependencia']=segunda_capa['NombreDependencia'].replace(
n=int(35339)
train2=segunda_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=segunda_capa.groupby('NombreDependencia', as_index=False).agg({'inpu
tfIdfVectorizer=TfidfVectorizer(min_df=2,ngram_range=(1,3))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_segunda_capa_final.joblib')
tfIdfVectorizer = load('tfidf_segunda_capa_final.joblib')
```

In [7]:
```
train2.to_csv('train_segunda_capa', index=False)
```

In [8]:
```
X_train=tfIdfVectorizer.transform(train2['input'])
y_train=train2['NombreDependencia'].values
l1_best=0.2
C_best=1.5
capa2=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_it
capa2.fit(X_train,y_train)
dump(capa2, 'capa2_final.joblib')
capa2 = load('capa2_final.joblib')
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
convergence after 24 epochs took 3994 seconds
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed: 66.6min finished
```

3. Tercera Capa

Se cargan los modelos de la capa

In [9]:
```
tercera_capa=data_corregido.copy()
deps=tercera_capa[['NombreDependencia','input']].groupby('NombreDependencia'
deps=deps[deps['input']<35000]
tercera_capa=tercera_capa[tercera_capa['NombreDependencia'].isin(deps['Nombr
deps['input']=deps['input']<7000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
tercera_capa['NombreDependencia']=tercera_capa['NombreDependencia'].replace(
n=int(7126)
train3=tercera_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=tercera_capa.groupby('NombreDependencia', as_index=False).agg({'inpu
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,3))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_tercera_capa_final.joblib')
tfIdfVectorizer = load('tfidf_tercera_capa_final.joblib')
```

In [10]:
```
train3.to_csv('train_tercera_capa', index=False)
```

In [ ]:
```
X_train=tfIdfVectorizer.transform(train3['input'])
y_train=train3['NombreDependencia'].values
l1_best=0.3
C_best=3.5
capa3=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_it
capa3.fit(X_train,y_train)
dump(capa3, 'capa3_final.joblib')
capa3 = load('capa3_final.joblib')
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
```

4. Cuarta Capa

Se cargan los modelos de la capa

In [12]:
```
cuarta_capa=data_corregido.copy()
deps=cuarta_capa[['NombreDependencia','input']].groupby('NombreDependencia',
deps=deps[deps['input']<7000]
cuarta_capa=cuarta_capa[cuarta_capa['NombreDependencia'].isin(deps['NombreDe
deps['input']=deps['input']<1000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
cuarta_capa['NombreDependencia']=cuarta_capa['NombreDependencia'].replace(de
n=int(1331)
train4=cuarta_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=cuarta_capa.groupby('NombreDependencia', as_index=False).agg({'input
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,3),)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_cuarta_capa_final.joblib')
tfIdfVectorizer = load('tfidf_cuarta_capa_final.joblib')
```

In [13]:
```
train4.to_csv('train_cuarta_capa', index=False)
```

In [14]:
```
X_train=tfIdfVectorizer.transform(train4['input'])
y_train=train4['NombreDependencia'].values
alpha_best=0.4
norm_best = False
capa4=ComplementNB(force_alpha=True,alpha=alpha_best,norm=norm_best)
capa4.fit(X_train,y_train)
dump(capa4, 'capa4_final.joblib')
capa4=load('capa4_final.joblib')
```

Carga de nuevo set de datos de pruebas

In [7]:
```
test_data=pd.read_csv('data_test_final.csv', sep=';')
deps={'Prensa_y_Comuni'.upper(): 'Extra'.upper(),
    'Ser_Publicos'.upper(): 'Extra'.upper(),
    'Direccion'.upper(): 'Extra'.upper()}
test_data['NombreDependencia']=test_data['NombreDependencia'].replace(' ','_
```

```
test_data['NombreDependencia']=test_data['NombreDependencia'].replace(deps,r
test_data
```

Out[7]:

	NombreDependencia	input
0	SEC_INFRAEST	buenos dias cra 35 cll 48 segunda luminaria a...
1	SEC_INFRAEST	bucaramanga enero 13 2012 dr luis fransco boh...
2	SEC_INFRAEST	19 dic radico derecho peticion 18 17 72573 di...
3	SEC_INFRAEST	secretaria infraestructura dado respuesta der...
4	SEC_INFRAEST	cordial saludo permito informar hoy consecuen...
...	...	...
64627	SEC_SALUD_Y_AMBI	solicitud visita restaurante cafeteria superm...
64628	SEC_INFRAEST	pendientes actividad demolicion contrato 167 p...
64629	SEC_INFRAEST	traslado 31002022093843 traslado 31002022093843
64630	SEC_HACIENDA	respuesta requerimiento ordinario n 006451 res...
64631	SEC_DES_SOCIAL	solicitud oferta institucional seores secretar...

64632 rows × 2 columns

Se procede a aplicar los modelos cargados al set de datos cargado

In [35]:
```
def eval_test(data:pd.DataFrame):
    #capa1
    tfIdfVectorizer = load('tfidf_primera_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'])
    clf=load('capa1_final.joblib')
    result=clf.predict(x)
    #capa2
    tfIdfVectorizer = load('tfidf_segunda_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'][result=='otro'])
    clf=load('capa2_final.joblib')
    result[result=='otro']=clf.predict(x)
    #capa3
    tfIdfVectorizer = load('tfidf_tercera_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'][result=='Extra'])
    clf=load('capa3_final.joblib')
    result[result=='Extra']=clf.predict(x)
    #capa4
    tfIdfVectorizer = load('tfidf_cuarta_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'][result=='Extra'])
    clf=load('capa4_final.joblib')
    result[result=='Extra']=clf.predict(x)
    #score
    return np.mean([i==k for i,k in zip(map(str.upper,result),data['NombreDe
```

In [36]:
```
print('El score es:',eval_test(test_data))
```

```
El score es: 0.7007674217106078
```

Resultado

Se obtiene un 70,076% de precisión y confianza

In [44]:

```python
def eval_test2(data: pd.DataFrame):
    # capa1
    tfIdfVectorizer = load('tfidf_primera_capa_final.joblib')
    x = tfIdfVectorizer.transform(data['input'])
    clf = load('capa1_final.joblib')
    result = clf.predict(x)
    # capa2
    tfIdfVectorizer = load('tfidf_segunda_capa_final.joblib')
    x = tfIdfVectorizer.transform(data['input'][result == 'otro'])
    clf = load('capa2_final.joblib')
    result[result == 'otro'] = clf.predict(x)
    # capa3
    tfIdfVectorizer = load('tfidf_tercera_capa_final.joblib')
    x = tfIdfVectorizer.transform(data['input'][result == 'Extra'])
    clf = load('capa3_final.joblib')
    result[result == 'Extra'] = clf.predict(x)
    # capa4
    tfIdfVectorizer = load('tfidf_cuarta_capa_final.joblib')
    x = tfIdfVectorizer.transform(data['input'][result == 'Extra'])
    clf = load('capa4_final.joblib')
    result[result == 'Extra'] = clf.predict(x)
    # score
    score = np.mean([i == k for i, k in zip(map(str.upper, result), data['No

    # Create a DataFrame with predictions and actual labels
    results_df = pd.DataFrame({
        'predictions': result,
        'true_labels': data['NombreDependencia'].str.upper()
    })

    return score, results_df

# Utilizando la función para obtener el score y las predicciones
score, results_df = eval_test2(test_data)

# Verificando si la predicción coincide con la etiqueta verdadera
results_df['correct'] = results_df['true_labels'] == results_df['predictions

# Agrupando por la etiqueta verdadera y calculando la media de la columna 'c
accuracy_per_category = results_df.groupby('true_labels')['correct'].mean()

print('El score general es:', score)
print(accuracy_per_category)
```

```
El score general es: 0.7007674217106078
true_labels
CON_INT_DISCIPLI    0.490566
DES_ALCALDE         0.349539
EXTRA               0.568387
PLA_SISBEN          0.806264
SEC_ADMIN           0.800532
SEC_DES_SOCIAL      0.670257
SEC_EDUCACION       0.676471
SEC_HACIENDA        0.876558
SEC_INFRAEST        0.596432
SEC_INTERIOR        0.746697
SEC_JURIDICA        0.736342
SEC_SALUD_Y_AMBI    0.747438
ST                  0.000000
VALORIZACION        0.809273
Name: correct, dtype: float64
```

In [45]:
```
results_df
```

Out[45]:

	predictions	true_labels	correct
0	Sec_Infraest	SEC_INFRAEST	True
1	Des_Alcalde	SEC_INFRAEST	False
2	Sec_Infraest	SEC_INFRAEST	True
3	Sec_Infraest	SEC_INFRAEST	True
4	Sec_Infraest	SEC_INFRAEST	True
...	...	...	...
64627	Sec_Salud_y_ambi	SEC_SALUD_Y_AMBI	True
64628	Sec_Infraest	SEC_INFRAEST	True
64629	Sec_Salud_y_ambi	SEC_INFRAEST	False
64630	Sec_Hacienda	SEC_HACIENDA	True
64631	Sec_Des_social	SEC_DES_SOCIAL	True

64632 rows × 3 columns

In [46]:
```
accuracy_per_category
```

Out[46]:
```
true_labels
CON_INT_DISCIPLI    0.490566
DES_ALCALDE         0.349539
EXTRA               0.568387
PLA_SISBEN          0.806264
SEC_ADMIN           0.800532
SEC_DES_SOCIAL      0.670257
SEC_EDUCACION       0.676471
SEC_HACIENDA        0.876558
SEC_INFRAEST        0.596432
SEC_INTERIOR        0.746697
SEC_JURIDICA        0.736342
SEC_SALUD_Y_AMBI    0.747438
ST                  0.000000
VALORIZACION        0.809273
Name: correct, dtype: float64
```

In [ ]:

Prototipo de herramienta para la mejora en los procesos de designación de PQRSD de la Alcaldía de Bucaramanga

Modelos técnicas tradicionales

Pruebas a 1000 nuevas solicitudes

Estudiantes

Wilfredo Ariel Góme Bueno

Edson Andrés Gómez Cárdenas

In [ ]:
```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from pandarallel import pandarallel
pandarallel.initialize(progress_bar=True)
from sklearn.feature_extraction.text import TfidfVectorizer
from joblib import dump, load
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB, ComplementNB,GaussianNB
from sklearn.linear_model import LogisticRegression,LogisticRegressionCV
from sklearn.model_selection import GridSearchCV
```

```
INFO: Pandarallel will run on 8 workers.
INFO: Pandarallel will use Memory file system to transfer data between the m
ain process and workers.
```

Se importa los datos ya preprocesados

In [ ]:
```
data_corregido=pd.read_csv('data_corregida.csv',usecols=['NombreDependencia'
data_corregido
```

Out[ ]:

	NombreDependencia	input
0	Sec_Admin	insatisfaccion pan recibido
1	Sec_Admin	queja contaminacion visual calle 45 carrera 1...
2	Sec_Admin	prueba pqrs 05102011
3	Sec_Infraest	cra 5 n 57 59 barrio porvenir 2 luminarias fu...
4	Sec_Admin	solicitud
...	...	...
590587	Sec_Hacienda	formulario declaracion ica registro presentaci...
590588	Sec_Hacienda	formulario declaracion ica registro presentaci...
590589	Sec_Des_social	solicitud oferta institucional seores secretar...
590590	Sec_Hacienda	formulario declaracion ica registro presentaci...
590591	Espacio_Publico	incompetentes espacio publico viernes 9 noche ...

590592 rows × 2 columns

1. Primera capa

Se cargan los modelos de la capa

In [ ]:
```
primera_capa=data_corregido.copy()
primera_capa.loc[primera_capa['NombreDependencia']!='Sec_Hacienda','NombreDe
n=int(273320)
train=primera_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=primera_capa.groupby('NombreDependencia', as_index=False).agg({'inpu
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,2))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_primera_capa_final.joblib')
tfIdfVectorizer = load('tfidf_primera_capa_final.joblib')
```

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
l1_best=0.55
C_best=3.55
capa1=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_it
capa1.fit(X_train,y_train)
dump(capa1, 'capa1_final.joblib')
capa1 = load('capa1_final.joblib')
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
Epoch 1, change: 1.00000000
Epoch 2, change: 0.15908511
Epoch 3, change: 0.14081662
Epoch 4, change: 0.08037008
Epoch 5, change: 0.06304148
Epoch 6, change: 0.05288985
Epoch 7, change: 0.03718222
Epoch 8, change: 0.02832479
Epoch 9, change: 0.01565397
Epoch 10, change: 0.02644116
Epoch 11, change: 0.01317326
Epoch 12, change: 0.01017501
Epoch 13, change: 0.00893920
Epoch 14, change: 0.00814743
Epoch 15, change: 0.00706169
Epoch 16, change: 0.00628665
Epoch 17, change: 0.00581704
Epoch 18, change: 0.00496701
Epoch 19, change: 0.00439242
Epoch 20, change: 0.00394489
Epoch 21, change: 0.00339607
Epoch 22, change: 0.00311158
Epoch 23, change: 0.00260570
Epoch 24, change: 0.00238880
Epoch 25, change: 0.00207799
Epoch 26, change: 0.00185929
Epoch 27, change: 0.00166361
Epoch 28, change: 0.00142912
Epoch 29, change: 0.00131473
Epoch 30, change: 0.00116169
Epoch 31, change: 0.00103862
Epoch 32, change: 0.00087156
Epoch 33, change: 0.00076883
Epoch 34, change: 0.00068198
Epoch 35, change: 0.00060899
Epoch 36, change: 0.00053695
Epoch 37, change: 0.00048054
Epoch 38, change: 0.00043547
Epoch 39, change: 0.00038980
Epoch 40, change: 0.00035378
Epoch 41, change: 0.00031748
Epoch 42, change: 0.00028788
Epoch 43, change: 0.00025828
Epoch 44, change: 0.00023430
Epoch 45, change: 0.00020974
Epoch 46, change: 0.00019039
Epoch 47, change: 0.00017049
Epoch 48, change: 0.00015412
Epoch 49, change: 0.00013898
Epoch 50, change: 0.00012541
Epoch 51, change: 0.00011343
Epoch 52, change: 0.00010172
convergence after 53 epochs took 3107 seconds
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed: 51.8min finished
```

2. Segunda capa

Se cargan los modelos de la capa

In [ ]:
```
segunda_capa=data_corregido.copy()
segunda_capa=segunda_capa.loc[segunda_capa['NombreDependencia']!='Sec_Hacien
segunda_capa.groupby('NombreDependencia').count().sort_values(by='input')
deps=segunda_capa.groupby('NombreDependencia',as_index=False).count().sort_v
deps['input']=deps['input']<35000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
segunda_capa['NombreDependencia']=segunda_capa['NombreDependencia'].replace(
n=int(35339)
train=segunda_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=segunda_capa.groupby('NombreDependencia', as_index=False).agg({'inpu
tfIdfVectorizer=TfidfVectorizer(min_df=2,ngram_range=(1,3))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_segunda_capa_final.joblib')
tfIdfVectorizer = load('tfidf_segunda_capa_final.joblib')
```

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
l1_best=0.2
C_best=1.5
capa2=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_it
capa2.fit(X_train,y_train)
dump(capa2, 'capa2_final.joblib')
capa2 = load('capa2_final.joblib')
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
Epoch 1, change: 1.00000000
Epoch 2, change: 0.17563340
Epoch 3, change: 0.09393052
Epoch 4, change: 0.05956055
Epoch 5, change: 0.06389785
Epoch 6, change: 0.04193017
Epoch 7, change: 0.04243141
Epoch 8, change: 0.03471635
Epoch 9, change: 0.02264910
Epoch 10, change: 0.01831648
Epoch 11, change: 0.01284021
Epoch 12, change: 0.00947415
Epoch 13, change: 0.00454371
Epoch 14, change: 0.01545534
Epoch 15, change: 0.00427827
Epoch 16, change: 0.00300471
Epoch 17, change: 0.00177620
Epoch 18, change: 0.00111584
Epoch 19, change: 0.00055782
Epoch 20, change: 0.00041550
Epoch 21, change: 0.00045502
Epoch 22, change: 0.00014563
Epoch 23, change: 0.00011818
convergence after 24 epochs took 2236 seconds
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed: 37.3min finished
```

3. Tercera Capa

Se cargan los modelos de la capa

In [ ]:
```
tercera_capa=data_corregido.copy()
deps=tercera_capa[['NombreDependencia','input']].groupby('NombreDependencia'
deps=deps[deps['input']<35000]
tercera_capa=tercera_capa[tercera_capa['NombreDependencia'].isin(deps['Nombr
deps['input']=deps['input']<7000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
tercera_capa['NombreDependencia']=tercera_capa['NombreDependencia'].replace(
n=int(7126)
train=tercera_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=tercera_capa.groupby('NombreDependencia', as_index=False).agg({'inpu
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,3))
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_tercera_capa_final.joblib')
tfIdfVectorizer = load('tfidf_tercera_capa_final.joblib')
```

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
l1_best=0.3
C_best=3.5
capa3=LogisticRegression(C=C_best,l1_ratio=l1_best,n_jobs=8,verbose=1,max_it
capa3.fit(X_train,y_train)
```

```
dump(capa3, 'capa3_final.joblib')
capa3 = load('capa3_final.joblib')
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent worke
rs.
Epoch 1, change: 1.00000000
Epoch 2, change: 0.08872941
Epoch 3, change: 0.03853008
Epoch 4, change: 0.03564792
Epoch 5, change: 0.02659975
Epoch 6, change: 0.02650278
Epoch 7, change: 0.02856518
Epoch 8, change: 0.02341340
Epoch 9, change: 0.01624578
Epoch 10, change: 0.02079931
Epoch 11, change: 0.00780774
Epoch 12, change: 0.00605259
Epoch 13, change: 0.00325813
Epoch 14, change: 0.00322094
Epoch 15, change: 0.00244158
Epoch 16, change: 0.00605675
Epoch 17, change: 0.00248515
Epoch 18, change: 0.00152521
Epoch 19, change: 0.00131333
Epoch 20, change: 0.00113093
Epoch 21, change: 0.00097027
Epoch 22, change: 0.00083984
Epoch 23, change: 0.00065996
Epoch 24, change: 0.00047684
Epoch 25, change: 0.00041164
Epoch 26, change: 0.00035676
Epoch 27, change: 0.00030568
Epoch 28, change: 0.00026473
Epoch 29, change: 0.00022879
Epoch 30, change: 0.00019777
Epoch 31, change: 0.00017135
Epoch 32, change: 0.00014833
Epoch 33, change: 0.00012735
Epoch 34, change: 0.00010999
convergence after 35 epochs took 1056 seconds
[Parallel(n_jobs=8)]: Done   1 out of   1 | elapsed: 17.6min finished
```

4. Cuarta Capa

Se cargan los modelos de la capa

In [ ]:
```
cuarta_capa=data_corregido.copy()
deps=cuarta_capa[['NombreDependencia','input']].groupby('NombreDependencia',
deps=deps[deps['input']<7000]
cuarta_capa=cuarta_capa[cuarta_capa['NombreDependencia'].isin(deps['NombreDe
deps['input']=deps['input']<1000
deps={dep[0]:'Extra' for dep in deps.values if dep[1]}
cuarta_capa['NombreDependencia']=cuarta_capa['NombreDependencia'].replace(de
n=int(1331)
```

```
train=cuarta_capa.groupby('NombreDependencia',as_index=False).sample(n=n)
dataset=cuarta_capa.groupby('NombreDependencia', as_index=False).agg({'input
tfIdfVectorizer=TfidfVectorizer(min_df=1,ngram_range=(1,3),)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
dump(tfIdfVectorizer, 'tfidf_cuarta_capa_final.joblib')
tfIdfVectorizer = load('tfidf_cuarta_capa_final.joblib')
```

In [ ]:
```
X_train=tfIdfVectorizer.transform(train['input'])
y_train=train['NombreDependencia'].values
alpha_best=0.4
norm_best = False
capa4=ComplementNB(force_alpha=True,alpha=alpha_best,norm=norm_best)
capa4.fit(X_train,y_train)
dump(capa4, 'capa4_final.joblib')
capa4=load('capa4_final.joblib')
```

Carga de nuevo set de datos de pruebas

In [ ]:
```
test_data=pd.read_csv('test_data.csv')
deps={'Prensa_y_Comuni'.upper(): 'Extra'.upper(),
    'Ser_Publicos'.upper(): 'Extra'.upper(),
    'Direccion'.upper(): 'Extra'.upper()}
test_data['NombreDependencia']=test_data['NombreDependencia'].replace(' ','_
test_data['NombreDependencia']=test_data['NombreDependencia'].replace(deps,r
test_data
```

Out[ ]:

	NombreDependencia	input
0	PLA_SISBEN	derecho de petición pla silben derecho de peti...
1	SEC_HACIENDA	solicitud devolución saldo a favor impuesto pr...
2	SEC_DES_SOCIAL	adulto mayor hola buenas tardes mi nombre es b...
3	SEC_INTERIOR	defensora del pueblo remisión de comunicación ...
4	SEC_INTERIOR	querella policia querella policia
...	...	...
995	ESPACIO_PUBLICO	cuenta de cobro mar 2023 centro comercial fegh...
996	SEC_HACIENDA	cobro injustificado en impuesto de industria y...
997	SEC_HACIENDA	solicitud inscripción industria y comercio sol...
998	SEC_INTERIOR	queja anonima peticionario presenta queja cont...
999	SEC_HACIENDA	desembarco de cuentas bancarias desembarco de ...

1000 rows × 2 columns

Se procede a aplicar los modelos cargados al set de datos cargado

In [ ]:
```
def eval_test(data:pd.DataFrame):
    #capa1
    tfIdfVectorizer = load('tfidf_primera_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'])
```

```
    clf=load('capa1_final.joblib')
    result=clf.predict(x)
    #capa2
    tfIdfVectorizer = load('tfidf_segunda_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'][result=='otro'])
    clf=load('capa2_final.joblib')
    result[result=='otro']=clf.predict(x)
    #capa3
    tfIdfVectorizer = load('tfidf_tercera_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'][result=='Extra'])
    clf=load('capa3_final.joblib')
    result[result=='Extra']=clf.predict(x)
    #capa4
    tfIdfVectorizer = load('tfidf_cuarta_capa_final.joblib')
    x=tfIdfVectorizer.transform(data['input'][result=='Extra'])
    clf=load('capa4_final.joblib')
    result[result=='Extra']=clf.predict(x)
    #score
    return np.mean([i==k for i,k in zip(map(str.upper,result),data['NombreDe
```

In [ ]:
```
print('El score es:',eval_test(test_data))
```

```
El score es: 0.634
```

Resultado

Se obtiene un 63.4% de precisión y confianza