Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

UN MODELO DE PREDICCIÓN PARA LA CLASIFICACIÓN DE RECLAMOS DE PRODUCTOS Y SERVICIOS FINANCIEROS

Clasificación de texto multiclase aplicando algoritmos de aprendizaje de máquina supervisado

Proyecto Terminal

Octubre 2018

Alumno: Miguel Enrique Guerra Topete
Tutor del proyecto: Dr. Abel García Nájera

Tabla de Contenido

1.Resumen

Las instituciones bancarias reciben diariamente una alta cantidad de reclamos de sus clientes en base a los servicios que ofrecen. Algunos reclamos pueden ser de alto o bajo grado de urgencia. Este trabajo modela un sistema automático e inteligente que clasifica los reclamos de los clientes en base a las palabras clave que se encuentran en cada reclamo. **Con este modelo, una institución sabría desde antes a qué categoría pertenecería un nuevo reclamo**, y poder atender el reclamo dependiendo de su grado de urgencia.

En este proyecto se resuelve un problema de **clasificación de texto multiclase y el objetivo es encontrar el modelo de aprendizaje de máquina supervisado que sea el más adecuado para predecir la categoría a la que pertenecerían futuros reclamos de clientes.** Se utilizó aprendizaje supervisado porque dentro de los datos se conoce la descripción del reclamo y la categoría a la que pertenece, y con ésto, podremos entrenar nuestro modelo de aprendizaje de máquina.

Para este proyecto se utilizaron datos reales de alrededor de un millón de reclamos de clientes de la base de datos de un banco de Estados Unidos, obtenidos del Bureau of Consumer Financial Protection [1] que previamente fueron subidos a una plataforma llamada Kaggle. Dado la enorme cantidad de datos, se hizo el uso de la misma plataforma Kaggle para obtener los datos y usar su poder computacional para poder analizarlos por medio de la web, utilizando una herramienta llamada Jupiter Notebook. Dentro de esta herramienta se hizo uso de Python, así como existen otros lenguajes de programación que pueden ser usados para el mismo propósito, se seleccionó éste, por su amplia colección de bibliotecas de terceros especial para la visualización y la manipulación de datos e implementación de algoritmos de aprendizaje de máquina. Para este proyecto se utilizaron las siguientes bibliotecas: Numpy, Pandas, Matplotlib, Seaborn y Scikit-Learn.

2.Introducción

Cada vez más información es creada por usuarios de distintas plataformas privadas y públicas, como son las redes sociales, y aplicaciones móviles y web que hoy en día abundan en nuestras computadoras de escritorio, tabletas y dispositivos móviles. Vivimos en la época digital, en donde muchos expertos mencionan que los datos son "*el impulso para la nueva economía*" [2] y otros presumen que *"los datos son el nuevo petróleo"* [3] en la cual, compañías de gran impacto internacional de diferentes industrias como la aeroespacial, la financiera, la automotriz, de transporte, educativa, de entretenimiento, musical, entre muchas otras, apuestan gran suma de su inversión interna en el análisis de datos [4]. Se estima que la demanda de puestos de trabajo de especialistas como: científicos de datos, analistas de datos e ingenieros de datos empezará a escalar exponencialmente, tanto así, que estos puestos son considerados como "*los trabajos más sexies del siglo 21"* [5].

2.1 Motivación

Dado la gran importancia del análisis de datos en diferentes industrias, como mencioné anteriormente, la facilidad que tiene uno de poder obtener datos reales como los que usaremos en este proyecto, las herramientas y plataformas de acceso gratuito que nos ayudan a manipular, visualizar y analizar los datos para resolver problemas, y la alta demanda de empresas en busca de ingenieros en ciencias de la computación, matemáticos y estadísticos con conocimientos prácticos de estas habilidades es casi imposible no encontrar una motivación. Hasta ahora, no encuentro una sola industria en la que no se haga uso del análisis de datos; es por ésto que se seleccionó una industria (financiera) con la que se estuviera poco familiarizada para poder desarrollar un análisis y encontrar una solución que pudiera beneficiar en sus aspectos económicos.

2.2 Situación, Problema y Solución

Los bancos son instituciones financieras con una alta demanda alrededor del mundo, son usados principalmente para resguardar recursos económicos de empresas e individuos, enviar y recibir pagos, pedir préstamos, inversiones, entre muchos otros servicios. Dado su alta demanda de transacciones por día y su constante interacción

con sistemas totalmente diferentes a los suyos de otras instituciones, servicios o terceros, es normal que sucedan errores e imprevistos a los diferentes clientes de éstas instituciones e incluso, dada su vulnerabilidad y valiosa recompensa, son diariamente sujetos de fraudes y estafas protagonizadas por delincuentes cibernéticos.

Por esta situación, estas instituciones cuentan con un departamento de atención al cliente donde reciben diariamente reclamos de clientes con problemas en sus servicios. Según la Comisión Nacional para la Protección y Defensa de los Usuarios de Servicios Financieros (Condusef) de México *"el primer trimestre del 2018 se registraron más de 2 millones de reclamos"*[6] relacionados con fraudes, ataques cibernéticos y quejas tradicionales.

Dado el alto índice de reclamos diarios en estas instituciones es necesario contar con un sistema automático e inteligente que pueda clasificar los reclamos de los clientes, con tan solo usar las palabras claves que se encuentren en cada descripción de cada reclamo. De esta manera, la institución podrá atender el tipo de reclamo con mayor prioridad dependiendo de su gravedad. Es mucho más grave un reclamo por un fraude o robo de tarjeta de crédito que un reclamo por un empleado que no atendió correctamente a un cliente en una sucursal específica, ambos casos son importantes, pero cada uno tiene un grado de urgencia y una solución específica. **Con el modelo que se creará en este proyecto, una institución sabría desde antes a qué categoría pertenecería un nuevo reclamo**. Por lo general este tipo de modelos nunca son 100% exactos, pero de igual manera, se ahorraría mucho tiempo en filtrar el tipo de reclamo hacia el departamento indicado; ésto llevaría a una mejor atención al cliente y retención de usuarios.

2.3 Metodología

Como se explicó anteriormente, para realizar este proyecto se tomarán datos generados por un banco anónimo de Estados Unidos para poder implementar el modelo de predicción.

Primero, se tendrán que analizar los datos y posteriormente descartar todas aquella característica que no tenga importancia para lo que que se quiere realizar. Una vez que se filtren lo datos importantes, se deberán observar las categorías de reclamos con las que cuenta este set de datos y las palabras claves que se encuentran con mayor frecuencia en cada una de la descripciones de cada reclamo. Se probarán diferentes modelos de aprendizaje de máquina que pudieran resolver el problema y se

seleccionará el más adecuado **para predecir la categoría a la que pertenecerían futuros reclamos de clientes.**

3.Tecnologías, algoritmos y terminología

En esta sección se describen brevemente las tecnologías utilizadas en el desarrollo del proyecto, los algoritmos propuestos para resolver el problema y una explicación del término big data que engloba las razones de los usos de las nuevas aplicaciones de aprendizaje de máquina, como lo que se va a realizar en este proyecto.

3.1 Kaggle

Kaggle es una plataforma online que crea competencias para desarrollar modelos de aprendizaje de máquina y análisis de datos. Cuenta con una comunidad de más de un millón de científicos, estudiantes, ingenieros y aficionados de los datos. También, cuenta con una amplia biblioteca de datos reales de diferentes industrias para analizar; este fue el lugar donde se obtuvieron los datos para poder desarrollar éste proyecto. Kaggle cuenta con una plataforma, en la que el usuario puede utilizar lenguajes como Python y R para analizar, compartir y modelar los datos. Esta plataforma está directamente conectada a los servidores de Kaggle para darle al usuario la opción de usar su poder computación para analizar grandes cantidades de datos, que de no ser así, sería casi imposible para muchos usuarios analizar los datos completos en su propio ordenador. Es por esta razón, que se hará uso de esta plataforma para realizar este proyecto.

3.2 Jupyter Notebook

Jupyter Notebook es la aplicación más popular que utilizan los científicos de datos. Es muy popular, porque se pueden utilizar lenguajes como Python y R, correr el código de cada bloque individualmente, escribir párrafos, ecuaciones, figuras, links, etc (como si fuera un editor de texto como Microsoft Word) y compartirlo de manera sencilla descargando el archivo en formato .ipynb o .html para poderlo visualizar en cualquier navegador. Esta aplicación puede ser descargada y ejecutada en cualquier ordenador con el uso de cualquier navegador o bien como mencionamos antes, ligarlo con alguna plataforma para hacer uso de ella, como es el caso de Kaggle.
Por esta razón, se hará uso de esta aplicación para desarrollar el proyecto, ya que también dentro de Kaggle están instaladas todas las bibliotecas que son comúnmente usadas para el análisis de datos y que también se estarán utilizando.

3.3 Python

Python es un lenguaje de programación interpretado y uno de los lenguajes más populares para el análisis de datos, ya que por ser un lenguaje interpretado, en el que no es necesario su compilación para ejecutar el código, hace que el trabajo sea mucho más flexible y rápido a comparación de otros lenguajes. Python, también cuenta con una amplio catálogo de bibliotecas para la manipulación, visualización y el análisis de datos, respaldada por una comunidad enorme de científicos de datos y compañías importantes de tecnología.
Para este proyecto se hará uso de Python versión 3.8, porque es un lenguaje fácil de interpretar para personas con poca o ninguna experiencia en programación y por su colección de bibliotecas; cabe mencionar, que existen otros lenguajes de programación que funcionan muy bien para el mismo propósito como R y Matlab.

3.4 Numpy

Como discutimos anteriormente, Python es un lenguaje interpretado y una de sus desventajas es que es muy lento cuando ejecuta operaciones de matrices y vectores de alto nivel. Numpy es una biblioteca que se puede incorporar a este lenguaje para resolver este problema, ya que está desarrollado con poderosos algoritmos y lenguajes de bajo de nivel como C, C++ y Fortran. Para este proyecto, se usarán varias de sus funciones para encontrar la frecuencia de las palabras clave de cada categoría.

3.5 Pandas

Pandas es una biblioteca y con extensión de Numpy desarrollada para facilitar la manipulación de tablas de datos. Los datos obtenidos para realizar este proyecto están en formato .csv, se usará Pandas para extraerlos en un Dataframe para poder manipular, extraer información, limpiar y utilizar para realizar operaciones con Numpy.

3.6 Matplotlib y Seaborn

Matplotlib y Seaborn son bibliotecas gráficas de Python utilizadas para mostrar gráficas, tablas, mapas, etc. Para el análisis de datos es necesario visualizar de diferentes formas los datos y poder ver los resultados en mapas de calor como se estará haciendo en este proyecto.

3.7 Scikit-Learn

Scikit-Learn es una biblioteca compuesta de algoritmos de aprendizaje de máquina. Incluye algoritmos de clasificación, regresión y agrupación aplicados para diferentes ramas del aprendizaje de máquina como es el aprendizaje supervisado, aprendizaje sin supervisión, aprendizaje profundo y aprendizaje reforzado.
Para este proyecto, probaremos varios algoritmos de aprendizaje supervisado para encontrar el algoritmo adecuado para cumplir el objetivo de lo que deseamos hacer (vea página 4). Más adelante, se explicará la razón por la que se utilizarán los algoritmos que seleccionaremos como posibles candidatos para resolver el problema. Es muy fácil pensar que se pueden probar cada uno de los algoritmos uno por uno hasta encontrar el que mejor convenga, pero cuando se esté analizando una mayor cantidad de datos, es importante entender el problema, para seleccionar el algoritmo que mejor se adecue y encontrar la solución de forma eficiente.

3.7.1 Algoritmos utilizados de Scikit-Learn

Existen diversos algoritmos de aprendizaje de máquina supervisado, para este proyecto es de interés usar algoritmos de clasificación, ya que es necesario **clasificar los reclamos en clusters de cada categoría para después predecir en qué categoría caerá un futuro reclamo**.

A continuación se mostrarán 3 propuestas que se van a considerar para el desarrollo del proyecto, se explicará brevemente su funcionamiento, ventajas y desventajas, aplicaciones de la vida real y porque pueden servir para cumplir el objetivo del proyecto:

- **Regresión Logistica**

Éste algoritmo es muy usado en problemas de clasificación, específicamente para clasificación binaria, pero también **trabaja excelente para la clasificación multiclase**. Algunas de sus aplicaciones son: la predicción del clima, en especial la probabilidad de que llueva en el día; predecir la probabilidad de que un paciente padezca cierta enfermedad en base a sus síntomas.
Para el caso multiclase, que es lo que tratamos de resolver en este proyecto, se utiliza un método llamado uno contra todos (one-vs-all), en el que se tiene, que por cada clase $i = 1, 2, 3, \ldots, k$ (en el caso de este proyecto $k = 14$ categorías) una función de entrenamiento $h_\theta^{(i)}(x)$ para predecir la probabilidad que $y = i$, o sea:

$$h_\theta^{(i)}(x) = P(y = i \mid x\,;\,\theta) \qquad (i = 1, 2, 3, \ldots, k)$$

Fig 14. Función de entrenamiento de Regresión Logística.

Cuando se tiene una nueva entrada (un nuevo reclamo) x, para hacer una predicción, el modelo, previamente entrenado, selecciona la categoría i que tenga mayor probabilidad:

$$max_i\, h_\theta^{(i)}(x)$$

Fig 15. La mayor probabilidad encontrada de la función de entrenamiento de Regresión Logística.

Es por ésto, que algunas de las ventajas de usar Regresión Logística, es que: provee un grado de probabilidad por cada observación, haciéndolo más confiable y es simple y eficiente a la hora de su implementación. Por otro lado, trabaja muy lento cuando se tiene un alto número de elementos (categorías, características y variables).
Dado que en este proyecto tenemos un número mediano de categorías, que estamos buscando el modelo con mejor precisión y podemos utilizar el poder computacional de Kaggle; será interesante observar sus resultados, aunque tome un tiempo considerable para ejecutarse.

- **Naive Bayes**

Naive Bayes o Clasificador Bayesiano Ingenuo es un algoritmo utilizado por varias plataformas de correos para identificar si un correo es spam o no spam. A diferencia de otros algoritmos, NB asume que las variables son independientes y esto ocurre rara vez en la vida real, pero en la práctica a demostrado funcionar asombrosamente bien.
Para este proyecto **se hará uso del Multinomial Bayesiano Ingenuo**, que es una instancia específica del clasificador NB que usa una distribución multinomial por cada

una de sus características, ya que usaremos 14 categorías, que a diferencia del correo spam o no spam que es un caso binomial. Para esto, se tiene que:

$$p(f_1, \ \dots \ , f_n \mid c) \ = \prod_{i=1}^{n} p(f_n \mid c)$$

Fig 16. Función del Clasificador Bayesiano Ingenuo.

En el que se encuentra la probabilidad de cada característica f dada la clase c a la que pertenece, en base al método de máxima similitud de la frecuencia en la que aparece cada elemento en cada una de sus clases, en donde $p(f_n \mid c)$ es una distribución multinomial.

Esto hace que funcione excelente cuando se cuentan palabras, como es el caso del proyecto que se está realizando. NB es simple cuando se implementa y muy útil para una gran cantidad de datos, y es un método que funciona muy bien cuando se tiene un poder computacional limitado. Una de sus principales desventajas es que después de que se entrene el modelo utilizando este algoritmo, si se encuentra con un nuevo valor que no pertenece a una clase en específico, dará una estimación incorrecta.

- **Maquinas de Soporte de Vectores**

SVM, por su siglas en inglés, es un algoritmo que funciona para solucionar problemas de regresión y clasificación, utiliza un enfoque de “calle ancha” para marcar la división que separa cada clase, haciéndolo menos vulnerable al sobreajuste. Para problemas de clasificación no lineales SVM hace uso de una función Kernel, o sea que, construye uno o un conjunto de hiperplanos en un espacio superior de N dimensiones al del espacio donde se haya iniciado, cuando se haya encontrado la función del conjunto de hiperplanos regresa a la dimensión original donde separa las clases utilizando esta misma función. Alguna de sus aplicaciones son: reconocimiento de escritura a mano, especialmente validación de firmas en documentos.

Unas de las fortalezas de SVM es que se pueden resolver eficazmente los problemas de clasificación, como es el caso de este proyecto. Por otro lado, el costo computacional es muy alto, pues encontrar los parámetros necesarios para separar los datos para crear un buen modelo es complicado. Aún así, se intentará utilizar para este proyecto y observar su exactitud.

3.8 Big Data

Se debe comprender a que se refiere el término big data y cual es su relación con este proyecto. En pocas palabras, big data es un conjunto de cantidades grandes de datos, que se basa en el uso de estos 3 importantes atributos:

- **Variedad:** que consiste en saber de ¿Qué forma se encuentran los datos? Estructurados, no estructurados, semiestructurados o la mezcla de todo.
- **Volumen:** ¿Qué cantidad de datos se va a analizar y como son? ¿Terabytes, Petabytes o Exabytes? ¿Son tablas, transacciones o archivos?
- **Velocidad:** ¿A que paso se obtienen los datos? Semanal, mensual, anual o tiempo real.

Es importante entender la relación que tienen las 3 V's [7], ya que al conocer cómo se desarrolla cada una de ellas, dependiendo de el entorno de los datos que se van a analizar, facilita la comprensión de éllos, desarrollo de modelos y la toma de decisiones a futuro.

Big data no es algo nuevo, de hecho se ha venido utilizando desde hace muchos años [8]. La diferencia de los años anteriores al presente, es que ahora la información se puede obtener desde una gran variedad de plataformas tales como: redes sociales, sitios web, blogs, foros, asistentes inteligentes (Alexa, Google Home, Siri), etc. Antes, la única forma de obtener información de los clientes era por medio de encuestas y formularios que se llenaban al momento de interactuar con ellos directamente.

¿Qué hay de las barreras que impiden que se utilice? ¿Que tan dificil es que cualquier empresa sin importar su tamaño haga uso de esta herramienta? Este tema me parece muy importante, ya que se suele hablar de los beneficios que se obtienen al utilizar el big data para resolver problemas, pero pocas veces se habla de las herramientas de hardware y software que se usan para la obtención de los datos. La mayoría de las empresas que utilizan el análisis de big data y creación de modelos usan programas open source o creadas por otras empresas. En la actualidad, existe una cantidad considerable de programas de software que han estado surgiendo para el análisis de datos, en especial software en la nube, en donde muchos usuarios las prefieren por su flexibilidad de uso. También, han surgido diferentes marcos para desarrollar modelos de aprendizaje de máquina, como Scikit-Learn, y esto definitivamente ha ayudado a que los ingenieros se preocupen más por la obtención y limpieza de datos que el

desarrollo desde cero de cada modelo de aprendizaje de máquina. Finalmente, estamos en una época en donde las computadoras, procesadores y GPU's, son económicamente más factibles que en años anteriores, ésto ha beneficiado pequeñas y medianas empresas que antes no contaban con los recursos monetarios para poder realizar análisis y entrenamiento de datos. Todas aquellas empresas que hacen uso del big data, lo ven como una oportunidad para aumentar las posibilidades de obtener mayor número de clientes, mejorar sus servicios, conocer nuevos mercados, nivelar los costos de operación y reducir el costo de sus productos.

4.Desarrollo

En esta sección se describirán los pasos de desarrollo para resolver el problema y llegar a su solución.

4.1 Obtención y manipulación de datos

Lo primero que se deberá hacer es cargar los datos desde el formato .csv en un Dataframe. De esta forma, se podrá manipular utilizando Pandas. Se imprimirán los primeros 5 renglones para observar el contenido.

```
import pandas as pd
df = pd.read_csv('../input/Consumer_Complaints.csv')
df.head()
```

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State
0	3/12/2014	Mortgage	Other mortgage	Loan modification,collection,foreclosure	NaN	NaN	NaN	M&T BANK CORPORATION	MI
1	10/1/2016	Credit reporting	NaN	Incorrect information on credit report	Account status	I have outdated information on my credit repor...	Company has responded to the consumer and the ...	TRANSUNION INTERMEDIATE HOLDINGS, INC.	AL
2	10/17/2016	Consumer Loan	Vehicle loan	Managing the loan or lease	NaN	I purchased a new car on XXXX XXXX. The car de...	NaN	CITIZENS FINANCIAL GROUP, INC.	PA
3	6/8/2014	Credit card	NaN	Bankruptcy	NaN	NaN	NaN	AMERICAN EXPRESS COMPANY	ID
4	9/13/2014	Debt collection	Credit card	Communication tactics	Frequent or repeated calls	NaN	NaN	CITIBANK, N.A.	VA

Fig 1. Se obtienen los datos de un formato .csv.

Ahora se revisarán cuántos renglones y columnas hay en los datos obtenidos.

```
df.shape
```

```
(903983, 18)
```

Fig 2. Número de renglones y columnas del Dataframe obtenido.

Se tienen **903,983 clientes y por cada cliente se observan 18 características**. Se seleccionarán las características que nos interesan, para este caso, es la categoría y la descripción del reclamo, que en la tabla lo llaman “Product” y el “Consumer complaint narrative”. Ya que se conocen las 2 columnas que se necesitan, se eliminaran el resto de las características que no nos interesan, también, se eliminarán los renglones que tengan valor nulo y se cambiará el nombre de cada columna por “Categoria” y “Descripcion”, para detallar mejor lo que hay en cada columna.

```
col = ['Product', 'Consumer complaint narrative']
df = df[col]
df = df[pd.notnull(df['Consumer complaint narrative'])]
df.columns = ['Categoria', 'Descripcion']
df.head()
```

	Categoria	Descripcion
1	Credit reporting	I have outdated information on my credit repor...
2	Consumer Loan	I purchased a new car on XXXX XXXX. The car de...
7	Credit reporting	An account on my credit report has a mistaken ...
12	Debt collection	This company refuses to provide me verificatio...
16	Debt collection	This complaint is in regards to Square Two Fin...

Fig 3. Se filtran aquellas columnas que se van a utilizar para el proyecto.

Se volverá a revisar el tamaño del Dataframe para observar cuantos valores se tienen después de eliminar las características que no nos interesaban y eliminar los valores nulos.

```
df.shape
```

```
(199970, 2)
```

Fig 4. Número de renglones y columnas de la Dataframe filtrado.

El tamaño se ha reducido a 199,970 de 903,983, por lo visto existían muchos clientes que no tenían una descripción del reclamo.

Es importante saber el número y el nombre de las categorías que existen en los datos. A continuación, se visualizarán en una gráfica de barras la cantidad de reclamos que se tienen de cada una de las categorías y se analizará, si es conveniente, eliminar categorías que suenen redundantes.

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8, 6))
df.groupby('Categoria').Descripcion.count().plot.bar(ylim = 0)
plt.show()
df.Categoria.unique().shape
```

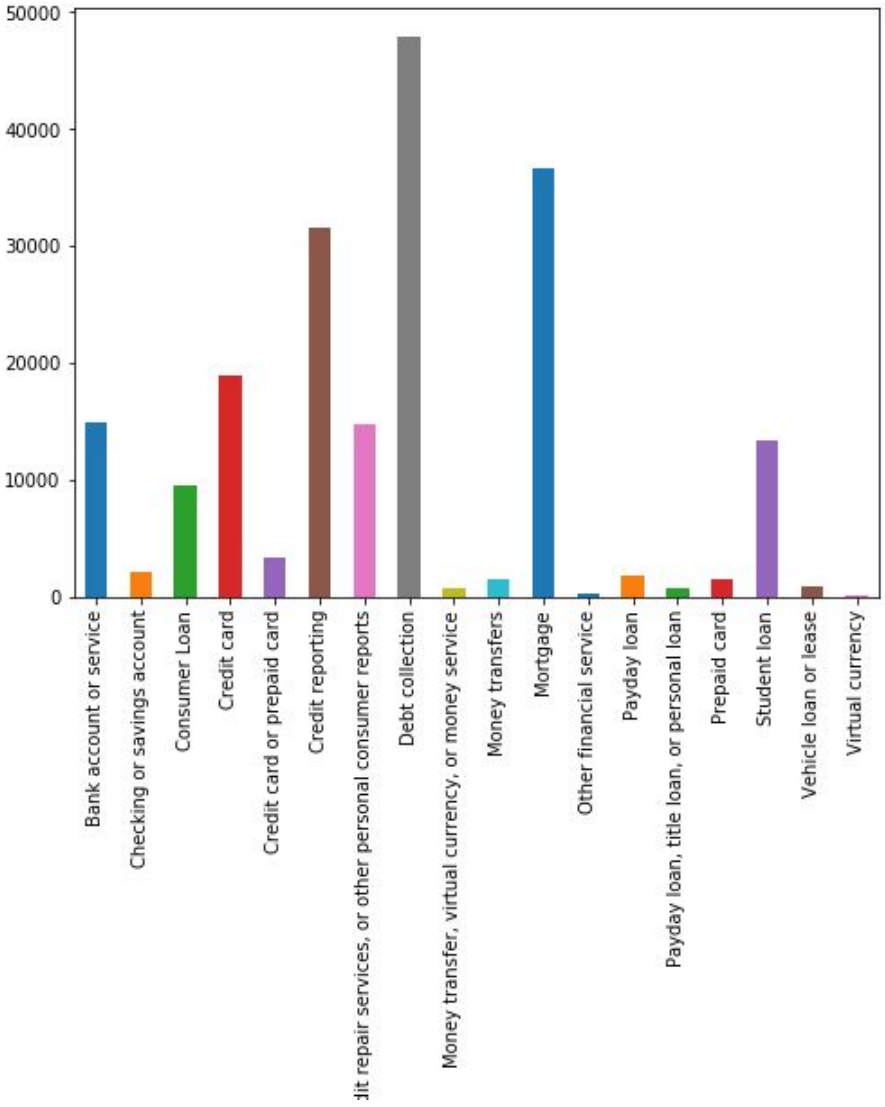

Fig 5. Gráfica de la cantidad de reclamos en cada categoría del Dataframe.

```
(18, )
```

Fig 6. Número de categorías.

Se tienen 18 categorías y se pueden observar varias que tienen un título similar (Fig 5) y que puede confundir el modelo, que que posteriormente se va a entrenar cuando se vayan a categorizar los elementos, ya que pueden tener palabras claves muy parecidas, como es el caso de: “Credit card o prepaid card”, “Credit reporting, credit repair services, or other personal consumer reports”, “Money transfer, virtual currency, or money service”, y “Payday loan, title loan, or personal loan”. Se eliminarán y se observará la nueva gráfica resultante.

```
df = df[(df.Categoria != 'Credit card or prepaid card') &
        (df.Categoria != 'Credit reporting, credit repair services, or other personal consumer reports') &
        (df.Categoria != 'Money transfer, virtual currency, or money service') &
        (df.Categoria != 'Payday loan, title loan, or personal loan')]
```

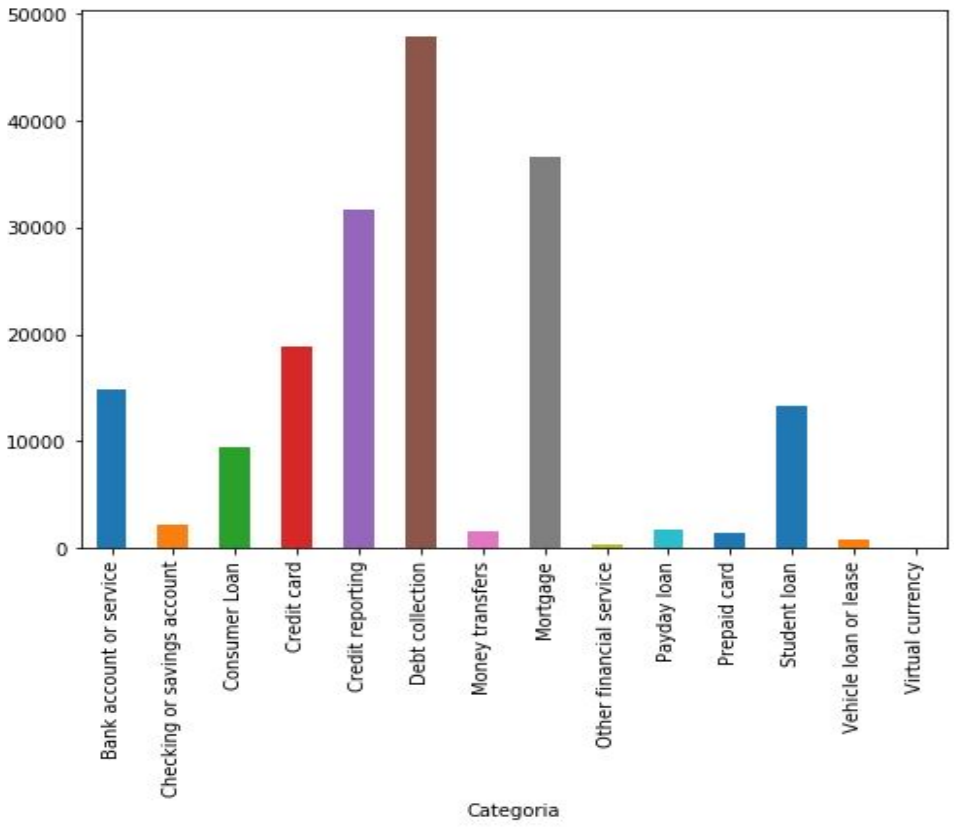

Fig 7. Gráfica de la cantidad de reclamos en cada categoría del Dataframe filtrado.

Ahora luce mejor; son categorías que normalmente se esperarían de un banco y con diferentes títulos. Analizando los datos, se puede observar que “Credit reporting”, “Debt collection”, “Mortgage”, “Bank account or service”, “Credit card” y “Student loan” son las categorías con más de 10,000 reclamos y son categorías que normalmente son muy frecuentes por reclamos de clientes de cualquier banco, aunque esperaba ver muchos más reclamos de “Credit card” por la cantidad de fraudes que ocurren con las tarjetas de crédito. Desafortunadamente, se tienen varias categorías que tienen un rango menor a 5,000 reclamos, que pueden actuar como ruido al momento de probar los modelos de predicción, **pero como es necesario encontrar el modelo que tenga mayor exactitud en base a cada una de las categorías**, ya sea con mayor reclamos y menor reclamos, se dejarán intactas.

4.2 Manipulación de texto

Ya que se tienen ordenados los datos con los que se va a trabajar, lo siguiente es **convertir el texto en valores numéricos**, ya que los algoritmos de aprendizaje de máquina trabajan de ésta manera. Primero, se creará otra columna en el Dataframe que represente cada categoría con números únicos y se llamará “categoria_id”.

```
df['categoria_id'] = df['Categoria'].factorize()[0]
df.head()
```

	Categoria	Descripcion	categoria_id
1	Credit reporting	I have outdated information on my credit repor...	0
2	Consumer Loan	I purchased a new car on XXXX XXXX. The car de...	1
7	Credit reporting	An account on my credit report has a mistaken ...	0
12	Debt collection	This company refuses to provide me verificatio...	2
16	Debt collection	This complaint is in regards to Square Two Fin...	2

Fig 8. Se le asigna a cada categoría un número único y se agrega al Dataframe.

Ahora es necesario encontrar una forma de **convertir la descripción de los reclamos en valores numéricos**; para esto se usará un método llamado “bolsa de palabras” [9]. Es un método utilizado en el procesamiento de texto, en el que se extraen la cantidad y la frecuencia de palabras encontradas en el texto que se esté procesando sin importar su orden. Aquí es donde se empezará a usar la biblioteca de Scikit-Learn, que cuenta con una función llamada TfidfVectorizer [10], esta función utiliza el método de “bolsa de

palabras” y nos regresa una matriz con el total y la frecuencia en la que aparecen las palabras en cada una de las descripciones de cada cliente. Para inicializar TfidfVectorizer se necesitan especificar los siguientes parámetros:

- **sublinear_tf**: toma en cuenta la frecuencia en la que aparece cada palabra. Para este caso, se dará el valor de “True” ya que es importante conocer la frecuencia.
- **min_df**: es el mínimo de palabras que se toman en cuenta en cada descripción. Para este caso, se le asignará el valor “5” ya que podremos limitar a que la descripción sea mayor a 5 palabras, si no, no cuenta como tal (Nótese que este parámetro haría que se disminuyera el número de reclamos en la matriz resultante, ya que es posible que existan descripciones con menor o igual a 5 palabras).
- **norm**: para normalizar. Se usará “L2” para normalizar, que es mejor conocido como el principio de mínimos cuadrados [11]; este principio ayudará a normalizar la matriz resultante, que de no ser usado, causaría valores atípicos.
- **encouding**: decodifica el lenguaje que se esté utilizando. Se le dará valor de “latin-1” ya que que es el lenguaje que se está usando.
- **ngram_range**: este parámetro se usa para tomar en cuenta secuencias n-gramas. Para este caso se le dará un rango de “(1, 2)” ya que nos interesan secuencias unigramas y bigramas (se podría tomar en cuenta también los trigramas, pero se sugiere que no se use, por la baja posibilidad que existieran y porque confundiría el algoritmo que descartaría secuencias que pueden pasar por unigramas y bigramas).
- **stop_words**: este parámetro ignora los pronombres comunes en la lengua. Ya que la descripcion esta en ingles, se usará “english” como valor, no es de interés contabilizar los pronombres y solo causará ruido en los resultados.

```
from sklearn.feature_extraction.text import TfidfVectorizer

bolsa_de_palabras = TfidfVectorizer(sublinear_tf = True,
                        min_df = 5,
                        norm = 'l2',
                        encoding = 'latin-1',
                        ngram_range = (1, 2),
                        stop_words = 'english')

val_descripcion = bolsa_de_palabras.fit_transform(df.Descripcion)
val_descripcion.shape
```

```
(180563, 389084)
```

Fig 9. Se crea la bolsa de palabras de la descripción de los reclamos.

El resultado es una matriz de 180,563 clientes representado por 389,084 secuencias de palabras unigramas y bigramas de cada uno. **Como se puede observar, el número de clientes disminuyó de 199,970 a 180,563**, seguramente porque existían descripciones de menor o igual a 5 palabras que se habían configurado con el parámetro **min_df**.

Ya que se tiene la bolsa de palabras en una matriz contenida en una variable "**val_descripcion**", o sea los valores "X" o de entrada, de igual forma también, se guardará la columna categoria_id en un vector que se llamará "**val_categoria**", o sea los valores "y" o de salida, y que anteriormente se habían transformado a valores numéricos.

```
val_categoria = df.categoria_id
```

Fig 10. Se asignan los valores numéricos de las categorías a un vector.

Estas variables se usarán más adelante para entrenar y probar el modelo que mejor convenga utilizar.

A continuación, se crearán 2 diccionarios que facilitaran las iteraciones que se tengan que hacer más adelante. Se hará uno de id a nombre de categoría y otro de nombre de categoría a id.

```
categoria_id_df = df[['Categoria', 'categoria_id']].drop_duplicates().sort_values('categoria_id')
categoria_a_id = dict(categoria_id_df.values)
id_a_categoria = dict(categoria_id_df[['categoria_id', 'Categoria']].values)
```

Fig 11. Creación de diccionarios de las categorías.

Antes de continuar, se hará un poco más de análisis, se visualizarán los 3 unigramas y bigramas más populares de cada una de las categorías, con esto también, se revisará que la matriz que obtuvimos funcione y tenga sentido de acuerdo a los elementos que se encontraron con más frecuencia en cada una de las categorías. Se utilizará una función de Scikit-Learn llamada chi2 [12] para ayudar a encontrar los valores más correlacionados y Numpy para ayudar a desarrollar operaciones matriciales.

```
from sklearn.feature_selection import chi2
import numpy as np

arreglo_temp = []

for Categoria, categoria_id in sorted(categoria_a_id.items()):
    palabras_chi2 = chi2(val_descripcion, val_categoria == categoria_id)
    indices = np.argsort(palabras_chi2[0])
    nombres_palabras = np.array(bolsa_de_palabras.get_feature_names())[indices]
    unigramas = [v for v in nombres_palabras if len(v.split(' ')) == 1]
    bigramas = [v for v in nombres_palabras if len(v.split(' ')) == 2]

    arreglo_temp.append((Categoria, unigramas[-3:], bigramas[-3:]))

uni_bi_categoria = pd.DataFrame(arreglo_temp, columns = ('categoria', 'unigrama', 'bigrama'))
uni_bi_categoria
```

	categoria	unigrama	bigrama
0	Bank account or service	[checking, bank, overdraft]	[debit card, overdraft fees, checking account]
1	Checking or savings account	[atm, deposit, app]	[active hours, use app, xxxx app]
2	Consumer Loan	[dealership, car, vehicle]	[vehicle xxxx, car loan, auto loan]
3	Credit card	[express, macy, card]	[annual fee, american express, credit card]
4	Credit reporting	[transunion, experian, equifax]	[equifax xxxx, trans union, credit report]
5	Debt collection	[collect, collection, debt]	[debt collection, collect debt, collection age...
6	Money transfers	[paypal, moneygram, western]	[money gram, money transfer, western union]
7	Mortgage	[escrow, modification, mortgage]	[short sale, mortgage company, loan modification]
8	Other financial service	[fedloanhelp, certegy, lexington]	[client solutions, global client, lexington law]
9	Payday loan	[castle, ace, payday]	[picture loans, big picture, payday loan]
10	Prepaid card	[rush, prepaid, rushcard]	[gift card, prepaid card, rush card]
11	Student loan	[loans, student, navient]	[income based, student loans, student loan]
12	Vehicle loan or lease	[santander, car, vehicle]	[credit acceptance, vehicle issues, consumer usa]
13	Virtual currency	[signup, bitcoins, coinbase]	[com 75, bonus changed, coinbase com]

Fig 12.Tabla que muestra los unigramas y bigramas más frecuentes de cada categoría.

Estos fueron los 3 unigramas y bigramas más frecuentes en cada una de las categorías (Fig 12). Como se puede observar, los valores encontrados tienen sentido de acuerdo con la categoría a la que pertenecen y también, se puede dar una idea de que si se encuentran elementos con estas características en una descripción se tendrá una percepción de a qué categoría pertenece.

4.3 Modelos

Se ha llegado a la parte del proyecto en la que habrá que **encontrar el modelo de predicción más adecuado**, utilizando los datos que se han limpiado, transformado y normalizado. Antes de proceder, se utilizará otra función de Scikit-Learn llamada train_test_split [13], esta función ayudará a separar los datos en datos de entrenamiento y datos de prueba. **Se usarán los datos de entrenamiento estrictamente para encontrar y finalmente entrenar el modelo y se utilizarán los datos de prueba para probar el modelo que finalmente seleccionemos**. Es importante realizar este paso desde ahora, y no después de encontrar el modelo adecuado, porque nunca se deben mezclar los datos de entrenamiento con los datos de prueba, para así, cerciorarnos que el modelo funcione eficazmente.
Para este proyecto se separan los datos: 70% de entrenamiento y 30% de prueba.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(val_descripcion,
                                                    val_categoria,
                                                    test_size = 0.3,
                                                    random_state = 0)
```

Fig 13. Se separan 70% en datos de entrenamiento y 30% en datos de prueba.

4.4 Buscando el modelo de predicción

Ya que se conocen los algoritmos que se van a probar (página 8), a continuación, se ejecutará el siguiente código para encontrar el **algoritmo que arroje el mejor grado de exactitud con los datos de entrenamiento** (X_train, y_train). De la biblioteca de Scikit-Learn se usarán los algoritmos que discutimos anteriormente: logisticRegression [14], MultinomialNB [15] y SVC [16] (Support Vector Classification). Una vez que se consiga el mejor modelo, se entrenará, se corroborarán con los datos de prueba observando las métricas de precisión, recall y f1-score para saber si el modelo funciona adecuadamente. Para realizar ésto, se usará una función de Scikit-Learn llamada cross_val_score [17], esta función regresa un arreglo de puntuaciones de exactitud de cada modelo.

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
import time

modelos = [
    LogisticRegression(),
    MultinomialNB(),
    LinearSVC()
]

cross_val = 5
cross_val_df = pd.DataFrame(index = range(cross_val * len(modelos)))
arreglo_temp = []

for modelo in modelos:
    nombre_modelo = modelo.__class__.__name__
    start_time = time.time()
    exactitudes = cross_val_score(modelo, X_train, y_train, scoring = 'accuracy', cv = cross_val)
    print("'{}' - {} seg".format(modelo.__class__.__name__, round((time.time() - start_time),2)))
    for fold_idx, accuracy in enumerate(exactitudes):
        arreglo_temp.append((nombre_modelo, fold_idx, accuracy))

cross_val_df = pd.DataFrame(arreglo_temp, columns = ['nombre_modelo', 'fold_idx', 'exactitud'])
```

```
'LogisticRegression' - 475.09 seg
'MultinomialNB' - 7.95 seg
'LinearSVC' - 114.65 seg
```

Fig 17. Se obtienen las precisiones de cada modelo y sus tiempos de ejecución.

Como se había comentado, Regresión Logística tomó más tiempo en procesar los datos con casi 8 minutos, después Máquina de Soporte de Vectores le tomó casi 2 minutos en procesar, tomando en cuenta, que se utilizó el algoritmo de método lineal, si se hubiera utilizado el método con polinomios hubiera tomado mucho más tiempo. Por último, el Clasificador Bayesiano Ingenuo le tomó el menor de los tiempos con casi 8 segundos de procesamiento. Ahora se observarán cuáles fueron los resultados de cada modelo y se comparara con sus tiempos de ejecución para tomar una decisión.

A continuación, se visualizarán los resultados en una diagrama de cajas de acuerdo a las puntuaciones obtenidas de cada modelo, de esta forma, se podrá observar qué tan dispersos se encuentran los puntos unos de otros. Se utilizará la biblioteca Seaborn con una combinación de las funciones boxplot [18] y stripplot [19] para graficar la información.

```
import seaborn as sns

sns.set(rc={'figure.figsize':(10,10)})
sns.boxplot(x = 'nombre_modelo', y = 'exactitud', data = cross_val_df)
sns.stripplot(x = 'nombre_modelo', y = 'exactitud', data = cross_val_df, size = 8, jitter = True, edgecolor = "gray", linewidth = 2)

plt.show()
```

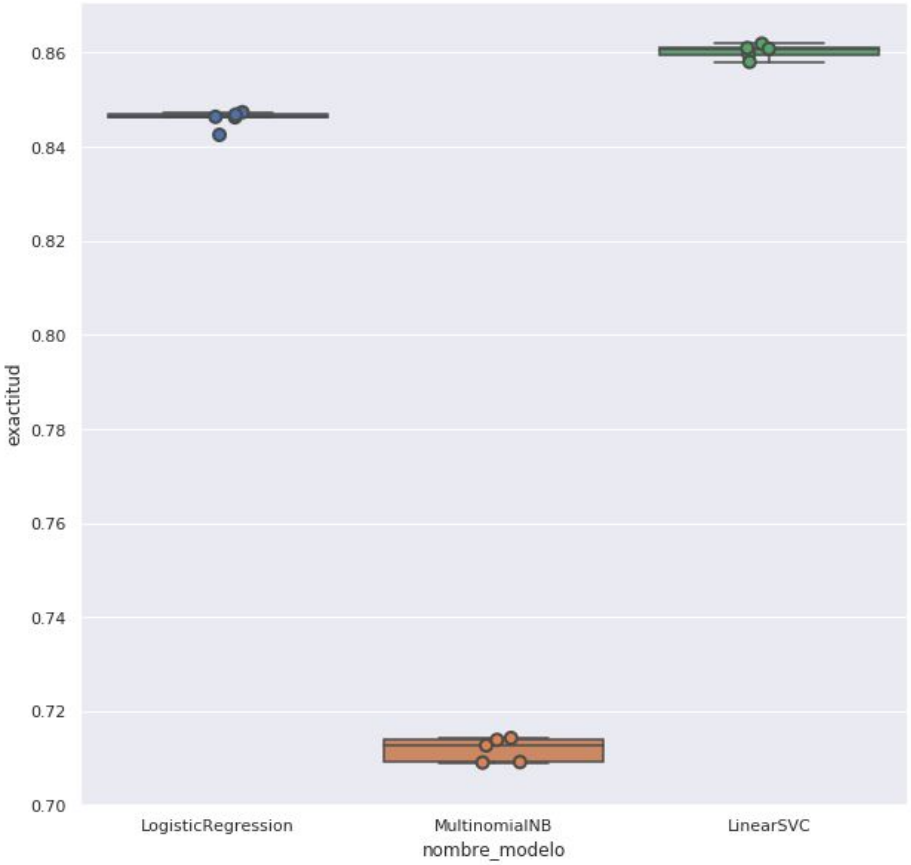

Fig 18. Diagrama de cajas que muestra las precisiones de cada modelo.

Como se puede observar (Fig. 18), **Regresión Logística y SVM se encuentran en el rango 83% a 88%, mientras que Clasificación Bayesiana Ingenua se sitúa de 70% a 72%**, sin embargo, NB fue el modelo con mayor varianza, o sea, que genero puntuación más dispersa entre cada punto. De acuerdo con los tiempos de ejecución, los que mejor exactitud obtuvieron fueron los que tomaron más tiempo en ejecutarse, Regresión Logística y SVM, sin embargo, para volúmenes más grandes de datos NB podría ser una gran opción por su rápida ejecución.
Para asegurarse de que se seleccione el mejor modelo se obtendrá el promedio de las puntuaciones de cada uno de los modelos.

```
cross_val_df.groupby('nombre_modelo').exactitud.mean()
```

```
nombre_modelo
LinearSVC              0.860294
LogisticRegression     0.845879
MultinomialNB          0.711862
Name: precision, dtype: float64
```

Fig 19. Media de precisiones de cada modelo.

Máquina de Soporte de Vectores obtuvo 86.02%, Regresión Logística obtuvo 84.59% y clasificación Bayesiana Ingenua obtuvo 71.19% de promedio de exactitud. Según los resultados de los 3 modelos, el mejor modelo para predecir la categoría de futuros reclamos es **SVM**. A continuación entrenaremos y probaremos si el modelo funciona eficazmente con los datos de prueba.

4.5 Entrenando y probando el modelo

Ahora que se conoce el mejor modelo, se entrenará con los datos de entrenamiento (X_train, y_train), se utilizarán los datos de entrada de prueba (X_test) para obtener predicciones y posteriormente se probarán con los datos de prueba de salida para observar sus resultados.

```
model = LinearSVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Fig 20. Se entrena el SVM con los datos de entrenamiento y se predicen resultados con los datos de prueba de entrada.

Ya que se tiene un vector de predicción obtenido de los datos de prueba de entrada aplicando el modelo entrenado, lo siguiente, es compararlo con los los datos de prueba de salida (y_test vs y_pred) para visualizar que tan bien predíjo el modelo cada reclamo de cada categoría. Para esto, se utilizará una función de Scikit-Learn llamada confusión_matrix [20], ésta permitirá, observar el resultado en un mapa de calor, o sea, una matriz que muestra en qué categoría cae cada reclamo de acuerdo a la predicción que arrojó el modelo entrenado contra los datos de salida que ya se conocían.

```
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, y_pred)
plt.subplots(figsize = (12, 5))
sns.heatmap(conf_mat,
            annot = True,
            fmt = 'd',
            xticklabels = categoria_id_df.Categoria.values,
            yticklabels = categoria_id_df.Categoria.values)
plt.ylabel('valores de categoria de prueba')
plt.xlabel('Predición')
plt.show()
```

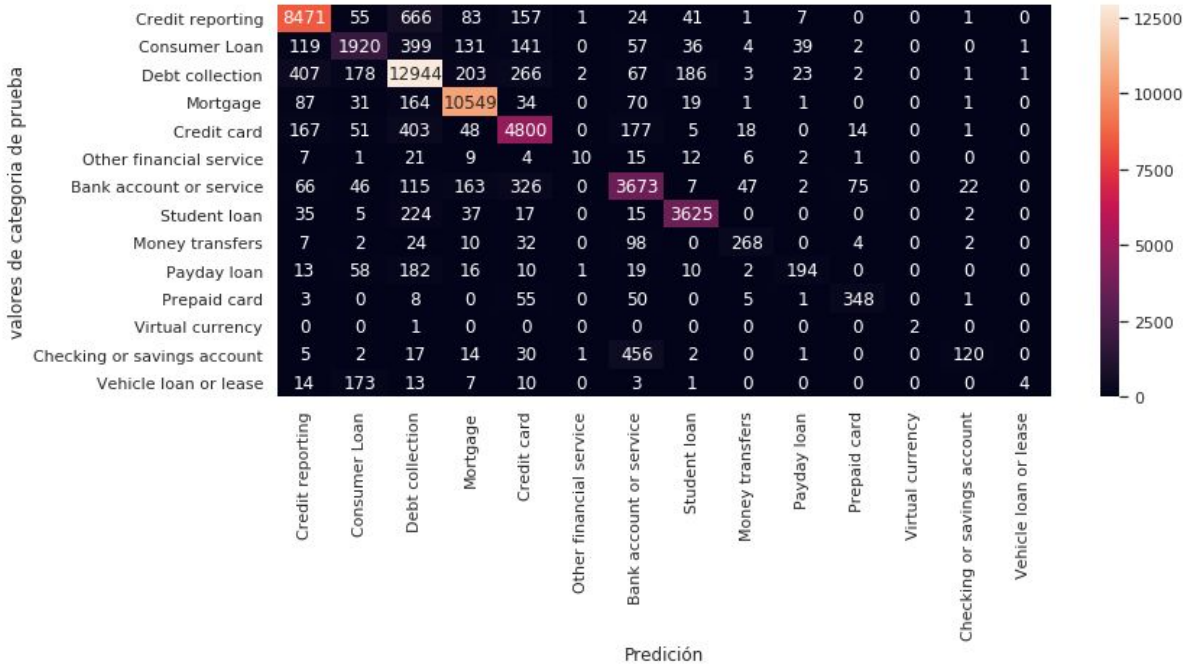

Fig 21. Matriz de confusión que muestra las predicciones comparadas con los datos de prueba de salida.

Ahora, se tiene una clara explicación el resultado de la predicción. La diagonal central de la matriz (Fig 21), donde se intercepta cada categoría con ella misma, muestra los valores que se predijeron correctamente y el resto, los valores que se predijeron incorrectamente; lo cual no es de sorprenderse que existan valores que se predijeron erróneamente, pues existen reclamos en el que frecuentan palabras que quizás el modelo interpreta como una categoría que no es (parecido a lo que sucede cuando recibes un correo que no es es spam pero el algoritmo lo coloca en la bandeja de

spam). Tambíen, se sabe que el modelo no es 100% exacto y si lo fuera, tendería a sobre ajustarse. Si se recuerda, la gráfica donde se mostraba la cantidad de reclamos de cada categoría (Fig. 7), observamos con esta matriz, que los elementos de colores pertenecen a las categorías con más de 10,000 reclamos, el resto, pertenecen a las categorías que tenían menos de 5,000 reclamos. Otra cosa que se visualiza, es que el número de predicciones correctas es mucho más grande que el número de predicciones incorrectas y que la única categoría con que no tuvo predicciones incorrectas fue “virtual currency”, sin embargo, se debe a que solo se probaron 2 reclamos que probablemente contenian palabras muy frecuentes en esa categoria, posiblemente, si se tuvieran más datos de ésta categoría hubiera clasificado algunos elementos incorrectamente.

A continuación, se imprimirá el reporte de cada una de las categorías y el promedio. Como se tienen datos no balanceados, o sea que, existen diferentes números de datos en cada categoría, no es recomendable fiarse sólo del promedio de los valores de exactitud del modelo; es necesario utilizar una métrica llamada f1-score, que es recomendable usarla cuando se tienen datos no balanceados, ya que toma en cuenta la precisión y el recall. Para esto, se usará otra función de Scikit-Learn llamada classification_report [21], esta función regresa un reporte completo de (precision, recall y f1-score) de cada una de las categorías y el promedio de todas.

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=df['Categoria'].unique()))
```

	precision	recall	f1-score	support
Credit reporting	0.90	0.89	0.90	9507
Consumer Loan	0.76	0.67	0.71	2849
Debt collection	0.85	0.91	0.88	14283
Mortgage	0.94	0.96	0.95	10957
Credit card	0.82	0.84	0.83	5684
Other financial service	0.67	0.11	0.19	88
Bank account or service	0.78	0.81	0.79	4542
Student loan	0.92	0.92	0.92	3960
Money transfers	0.75	0.60	0.67	447
Payday loan	0.72	0.38	0.50	505
Prepaid card	0.78	0.74	0.76	471
Virtual currency	1.00	0.67	0.80	3
Checking or savings account	0.79	0.19	0.30	648
Vehicle loan or lease	0.67	0.02	0.03	225
micro avg	0.87	0.87	0.87	54169
macro avg	0.81	0.62	0.66	54169
weighted avg	0.86	0.87	0.86	54169

Fig 22. Reporte de cada una de las categorías y el promedio.

Finalmente, se tienen los datos para evaluar si SVM realizó eficazmente el trabajo, para poder explicar mejor, las métricas de evaluación son:

- **Precisión** nos dice, qué proporción de descripción de reclamos clasificamos en su categoría que en realidad son de su categoría. Es decir:

$$precisión = \frac{verdad\ positiva}{verdad\ positiva + falso\ positivo}$$

Fig 23. Fórmula para encontrar la precisión.

- **Recall** nos dice, qué proporción de descripción de reclamos son en realidad de su categoría y que se clasificó en su categoría. Es decir:

$$recall = \frac{verdad\ positiva}{verdad\ positiva + falso\ negativo}$$

Fig 24. Fórmula para encontrar el recall.

- **F1-score** es una métrica que considera ambos precisión y recall. Es decir:

$$F_1 = 2 * \frac{precisión * recall}{precisión + recall}$$

Fig 25. Fórmula para encontrar el recall.

Con el conocimiento de estas métricas, **ahora se sabe que SVM si realizó un buen trabajo prediciendo las categorías de futuros reclamos**. Como se puede observar, tomando en cuenta la precisión y el recall se calculó el f1-score que arrojó un **promedio de 86%**, que es igual al promedio de exactitud y que rara vez ocurre como lo es en este caso. Este resultado, nos da un modelo que puede realizar un buen trabajo para predecir reclamos nuevos que obtenga esta institución bancaria.

5.Conclusiones

En conclusión, **utilizando el algoritmo de Clasificación de Soporte de Vectores se obtuvo un promedio f1-score de 86%, que resulta adecuado para poder predecir la categoría de futuros reclamos de clientes de este banco**.

Se recolectaron los datos desde una fuente confiable, asegurando que los datos son de clientes reales; se manipularon y limpiaron para poder extraer los elementos que importaba utilizar para cumplir el objetivo; se filtraron categorías que no fueran de mucha ayuda para el modelo; se transformó cada categoría a valor numérico; se utilizó el método de bolsa de palabras para obtener la frecuencia de cada palabra por cada descripción de cada reclamo para cada cliente de los datos y se obtuvo una matriz con valores numéricos de este proceso; se separaron los datos de entrenamiento con los datos de prueba para después encontrar el mejor modelo; y por último se seleccionó SVM como mejor candidato, se entrenó, se probó y se demostró que funciona como **un modelo de predicción para clasificación de reclamos de productos y servicios financieros** del set de datos de este banco anónimo.

Este es un ejemplo de lo que es posible resolver utilizando este set de datos. Sin embargo, no podemos generalizar; este modelo funciona solo para este banco en específico, no es posible utilizar estos mismos datos para predecir reclamos de otras instituciones, muchas de las categorías podrían ser diferentes, los tipos de reclamos más frecuentados posiblemente cambiarían, podría ser otro idioma, entre muchas otras cosas, que se tendrían que analizar de acuerdo a los datos de cada institución.

Es importante recalcar que sin el uso de las herramientas de hoy en día, este tipo de proyectos no se podría llevar a cabo tan fácilmente, pues sin lenguajes de programación como Python en conjunto con el catálogo de bibliotecas especializadas en la ciencia de datos y una comunidad como Kaggle donde se puede explorar diferentes tipos de datos, que por otro lado sería mucho más difícil encontrarlos y mucho más complicado analizarlos, pues no es muy probable que algún estudiante cualquiera cuente con un poder computacional como esta plataforma posee.

6.Referencias

1. Consumer Complaint Database - Data.gov. (n.d.). https://catalog.data.gov/dataset/consumer-complaint-database. Accessed 3 October 2018
2. Data is giving rise to a new economy. (2017, May 6). *The Economist*. https://www.economist.com/briefing/2017/05/06/data-is-giving-rise-to-a-new-economy. Accessed 3 October 2018
3. The world's most valuable resource is no longer oil, but data. (2017, May 6). *The Economist*. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data. Accessed 3 October 2018
4. Winig, L. (n.d.). GE's Big Bet on Data and Analytics, 21.
5. Davenport, T. H., & Patil, D. J. (2012, October 1). Data Scientist: The Sexiest Job of the 21st Century. *Harvard Business Review*. https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century. Accessed 3 October 2018
6. Juárez, E. (n.d.). Sumaron más de 2.2 millones los reclamos a bancos, en el primer trimestre: Condusef. *El Economista*. https://www.eleconomista.com.mx/sectorfinanciero/Sumaron-mas-de-2.2-millones-los-reclamos-a-bancos-en-el-primer-trimestre-Condusef-20180715-0063.html. Accessed 3 October 2018
7. Einav, Liran, and Jonathan Levin. "Economics in the Age of Big Data." *Science*, vol. 346, no. 6210, Nov. 2014, p. 1243089. *science.sciencemag.org*, doi:10.1126/science.1243089.
8. Cerchiello, Paola, and Paolo Giudici. "Big Data Analysis for Financial Risk Management." *Journal of Big Data*, vol. 3, Dec. 2016. *ResearchGate*, doi:10.1186/s40537-016-0053-4.
9. What is the bag-of-words algorithm? - Quora. (n.d.). https://www.quora.com/What-is-the-bag-of-words-algorithm. Accessed 10 October 2018
10. sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. Accessed 6 October 2018
11. Becke, A. D., & Sprung, D. W. L. (1977). Least square fits with normalization parameters and linear constraints. *Nuclear Physics A*, *284*(3), 425–428. doi:10.1016/0375-9474(77)90394-3
12. sklearn.feature_selection.chi2 — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html. Accessed 5 October 2018
13. sklearn.model_selection.train_test_split — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Accessed 7 October 2018

14. sklearn.linear_model.LogisticRegression — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 8 October 2018
15. sklearn.naive_bayes.MultinomialNB — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. Accessed 8 October 2018
16. sklearn.svm.SVC — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC. Accessed 8 October 2018
17. sklearn.model_selection.cross_val_score — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html. Accessed 8 October 2018
18. seaborn.boxplot — seaborn 0.9.0 documentation. (n.d.). https://seaborn.pydata.org/generated/seaborn.boxplot.html. Accessed 9 October 2018
19. seaborn.stripplot — seaborn 0.9.0 documentation. (n.d.). https://seaborn.pydata.org/generated/seaborn.stripplot.html. Accessed 10 October 2018
20. sklearn.metrics.confusion_matrix — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. Accessed 10 October 2018
21. sklearn.metrics.classification_report — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html. Accessed 10 October 2018
22. Kaggle: Your Home for Data Science. (n.d.). https://www.kaggle.com/. Accessed 10 October 2018
23. Project Jupyter. (n.d.). http://www.jupyter.org. Accessed 10 October 2018
24. Welcome to Python.org. (n.d.). *Python.org*. https://www.python.org/. Accessed 10 October 2018
25. NumPy — NumPy. (n.d.). http://www.numpy.org/. Accessed 10 October 2018
26. Python Data Analysis Library — pandas: Python Data Analysis Library. (n.d.). https://pandas.pydata.org/. Accessed 10 October 2018
27. Matplotlib: Python plotting — Matplotlib 3.0.0 documentation. (n.d.). https://matplotlib.org/. Accessed 10 October 2018
28. seaborn: statistical data visualization — seaborn 0.9.0 documentation. (n.d.). https://seaborn.pydata.org/. Accessed 10 October 2018
29. scikit-learn: machine learning in Python — scikit-learn 0.20.0 documentation. (n.d.). http://scikit-learn.org/stable/. Accessed 10 October 2018

30. bayesian - Difference between naive Bayes & multinomial naive Bayes. (n.d.). *Cross Validated*. https://stats.stackexchange.com/questions/33185/difference-between-naive-bayes-multinomial-naive-bayes. Accessed 8 October 2018

31. Raschka, S. (2016). *Python machine learning: unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Birmingham Mumbai: Packt Publishing open source