



POLITÉCNICA

UNIVERSIDAD  
POLITÉCNICA  
DE MADRID



# **TRABAJO FIN DE GRADO**

## **GRADO EN INGENIERÍA DEL SOFTWARE**

**INTRODUCCIÓN AL MACHINE LEARNING Y  
ANÁLISIS DE SENTIMIENTOS EN TWITTER  
MEDIANTE MODELOS DE APRENDIZAJE  
SUPERVISADO**

**AUTOR: YERAY GRANADA LAYOS**

**TUTOR: FERNANDO ORTEGA REQUENA**

**CURSO 2020-2021**



# Agradecimientos

Quiero agradecer este trabajo de fin de grado a todas las personas que siempre han estado ahí apoyándome y han conseguido que siga luchando hasta el final.

A mi padre, por esperar siempre lo mejor de mí, por intentar que desde pequeño llegara a la excelencia y por decirme siempre “no dejes para mañana lo que puedes hacer hoy”.

A mi madre, por su paciencia, por estudiar conmigo desde pequeño, por sus consejos y por guiarme por el buen camino.

A mis abuelos, por ser unos segundos padres para mí, por cuidarme desde pequeños y por hacerme sentir el nieto más afortunado del mundo.

Al resto de mi familia porque gracias a ellos y a mis padres soy la persona que soy.

A mi amigo Almeida, por ser mi mentor y por ser una persona con unos valores que siempre he querido tener.

A mis amigos Juan, Darna y Kike, que empezamos juntos esta carrera, y gracias a ellos tengo un grupo de amigos que son como una familia para mí.

A Cube y David, porque cuando estamos juntos es imposible no llorar de la risa.

A Gong y Pedro, por las tardes en el museo y la intrusión de nuevas palabras.

A Roberto C., porque siempre le vi como un hermano mayor.

A Truji, por acogerme siempre y hacer los mejores caldos en Islandia.

A Ferni, por tu influencia y gusto musical.

A Tomás, por las inolvidables salidas por León.

A Carlos, porque no conozco un humor igual.

A Alex, porque siempre sabe como comportarse y nos guía por el buen camino.

A Raven, por ser uno de los mejores compañeros de viaje, y aprendí mucho de ti cuando recorrimos el sudeste asiático.

A Pablo, por dejarme estar en su barco de piratas.

A Cano y Bringas, porque el Max no sería nada sin ellos.

A Muri, Marcos, Calabia, Salamanca, Cirvida y Guerra, por las bases de datos que hemos ido creando estos años y los consejos sobre finanzas.

A mis amigos de la universidad Javi, Andrea, Jorge, Iván, Pepe, Dani, Carl y Alex, por todas esas tardes de cafetería y biblioteca.

A Rober y Borja, que entraron un año antes en la carrera y fueron una de las razones por las que entré. Por todo su apoyo y la ayuda que he recibido desde el principio.

A Jorge, por ser amigos desde que casi ni sabíamos andar.

Y a Marta, mi compañera de vida, que me ha apoyado en todo momento, que siempre ha creído en mí, que ha conseguido convertirme en un adulto, y por hacerme la persona más feliz del mundo.

# Índice

## Contenido

<b>1. Introducción .....</b>	<b>6</b>
1.1. Historia de Twitter .....	7
1.2. La expansión de Twitter.....	8
1.3. ¿Qué le hace diferente?.....	8
1.4. Ventajas y desventajas .....	10
<b>2. Objetivos .....</b>	<b>12</b>
<b>3. Machine Learning .....</b>	<b>13</b>
3.1. Introducción al Machine Learning .....	13
3.2. Tipos de aprendizaje.....	16
3.2.1. Aprendizaje supervisado.....	17
3.2.2. Aprendizaje no supervisado .....	23
3.2.3. Aprendizaje por refuerzo .....	25
3.3. Usos prácticos del Machine learning .....	25
3.4.1. Tipos de análisis de sentimiento.....	27
<b>4. Conceptos previos.....</b>	<b>28</b>
4.1. Dataset .....	28
4.2. DataFrame .....	28
4.3. CSV .....	29
<b>5. Herramientas utilizadas.....</b>	<b>30</b>
5.1 Python .....	30
5.1.1. Ventajas de programar en Python.....	30
5.2 Skicit-Learn .....	31
5.3. Numpy .....	33
5.4. Matplotlib .....	34
5.5. Pandas .....	34
5.7. Anaconda .....	35
5.7.1. Jupyter Notebook .....	36
5.7.2. Spyder .....	37
5.8. Kaggle .....	38
<b>6. Análisis de sentimientos en Twitter.....</b>	<b>39</b>

<b>6.1. Dataset usado .....</b>	<b>39</b>
<b>6.2. El preprocessamiento aplicado.....</b>	<b>39</b>
6.2.1. Entender el Problema .....	40
6.2.2. Entender los datos .....	40
6.2.3. Definir un Criterio de Evaluación.....	41
6.2.4. Evaluación de la solución actual .....	41
6.2.5. Preparar los datos.....	41
6.2.6. Construir el modelo .....	42
6.2.7. Análisis de Errores .....	42
6.2.8. Modelo integrado en un Sistema .....	42
<b>6.3. Funciones del modelo.....</b>	<b>42</b>
6.3.1. load_dataset .....	42
6.3.2. remove_unwanted_cols .....	43
6.3.3. clean_text .....	43
6.3.4. create_bag_of_words .....	44
<b>6.4. Entrenamiento del modelo .....</b>	<b>45</b>
6.4.1. ¿Qué es train_test_split?.....	45
<b>7. Modelos utilizados.....</b>	<b>47</b>
<b>8. ANÁLISIS DE RESULTADOS.....</b>	<b>48</b>
<b>8.1 Conceptos sobre métricas.....</b>	<b>48</b>
<b>8.2 Resultados de los modelos .....</b>	<b>51</b>
<b>8.3. Resultados Adicionales .....</b>	<b>60</b>
<b>9. CONCLUSIONES.....</b>	<b>66</b>
<b>10. FUTURO DEL PROYECTO .....</b>	<b>68</b>
<b>11. BIBLIOGRAFÍA.....</b>	<b>69</b>
<b>12. ANEXO.....</b>	<b>73</b>

# 1. Introducción

La idea de red social surgió en la década de 1990 en SixDegrees.com, creada en 1997, considerada por muchos como la primera red social moderna. Permitía a los usuarios tener un perfil y agregar a otros usuarios en un formato similar a como lo conocemos hoy en día. Llegó a tener 3 millones y medio de miembros, y cerró finalmente en 2001.



Figura 1: Las redes sociales

A principios de este milenio surgieron nuevas páginas enfocadas a la interacción entre usuarios: Hi5, Orkut, Friendster y MySpace son algunos de los ejemplos de sitio en auge de la época. En esa misma etapa surgieron dos de las redes sociales más importantes en la actualidad, como lo son LinkedIn y Facebook. Facebook surgió como una plataforma en 2004, como una plataforma para conectar estudiantes de la universidad de Harvard, y hoy se mantiene invicta como la red social más popular del mundo. Twitter nació en 2006, Tumblr en 2007, Instagram EN 2020, Y Snapchat y Google en 2011. Por entonces nadie se imaginaba el impacto tan grande que han tenido, y ha sido el deseo de conectarse con otras personas desde cualquier lugar del planeta el que ha llevado al éxito a las redes sociales donde las personas y las empresas se encuentran inmersas.

Es importante no confundir entre Social Media y Redes Sociales. Las redes sociales son los grupos de conexiones y relaciones que tenemos con otras personas, y los medios sociales son plataformas que garantizan que eso suceda. Se podría decir que las redes sociales son una categoría dentro de los medios sociales.

Podemos clasificar las redes sociales en dos categorías:

- Horizontales: Se dirigen al público en general y no tienen una temática específica. Su objetivo principal es simplemente favorecer las conexiones entre personas. Ejemplos de esta categoría son Facebook y Twitter.
- Verticales: A esta categoría pertenecen todas las redes sociales especializadas, como pueden ser de vídeo, música, conseguir pareja, etc. Algunas de las redes más populares son Youtube o LinkedIn.

## 1.1. Historia de Twitter



Figura 2: La historia de Twitter

Entre todas estas redes sociales hay una que destaca por ser diferente y por todo lo que ofrece. Hablamos de Twitter. Este microblogging nació gracias a Jack Dorsey, Noah Glass, Biz Stone y Evan Williams, siendo estos dos últimos colaboradores de Google. La idea surgió dentro de la compañía Odeo situada en San Francisco, usado en sus inicios por sus empleados. Al inicio la red social tenía de nombre Status. Más adelante pasó a ser Twtrr haciendo referencia al pío de un pájaro. Finalmente empezó a llamarse Twitter porque según los creadores significa “una corta ráfaga de información intrascendente”. El primer tweet fue realizado por Jack Dorsey a las 12:50, que decía “just setting my twtrr”.

En octubre de 2006 Jack Dorsey, Biz Stone y Evan Williams fundan Obvious Corporation, adquiriendo las acciones de Odeo, pero desde abril de 2007 pasó a ser una empresa independiente. En 2008 la empresa estaba formada por solo 18 personas, pero a raíz de 2009 multiplicaron la cantidad de trabajadores debido a su fama y comenzó a nacer una organización mundial. A finales de 2009, Twitter ya contaba con versiones en español, francés, italiano y alemán. En 2010 crean “Promoted tweets”, que consistía en el patrocinio de cualquier empresa que quisiera aparecer como primer resultado de búsqueda dentro de la red social. Fue introduciéndose poco a poco en la publicidad con su servicio “Twitter Ads”, pudiendo incluso medir la métrica y comportamiento de los seguidores con “Twitter Analytics”.

En 2015 incorporó su nueva aplicación Periscope, que permitía transmitir eventos en tiempo real. Pasado un año después de su lanzamiento ya databa de 200 millones de emisiones en directo.

## 1.2. La expansión de Twitter

Desde 2006 se ha ido expandiendo a más de 100 países en el mundo. A finales de 2017, Twitter ya tenía 328 millones de usuarios activos por mes, y un 82% de ellos se conectaban a través de la aplicación de móvil. Se ha llegado a traducir la plataforma en más de 40 idiomas, haciendo posible el acceso a prácticamente todo el mundo.



Figura 3: La expansión de Twitter

En 2006, Twitter solo era conocida en Estados Unidos. En 2007 llegó a Canadá, Sudáfrica, Argentina, Brasil, parte de Europa, Australia y Japón. Para 2009, 2010 y 2011 Twitter ya había incrementado su número de usuarios en nuevos países como Portugal, Nueva Zelanda, Países Bajos, Reino Unido, Noruega, Suecia, Alemania y algunos países de Asia.

## 1.3. ¿Qué le hace diferente?

La característica principal de Twitter y por la que se ganó a millones de usuarios, fueron sus Tweets con caracteres limitados, 140. Las 10 cualidades más destacadas de esta red de microblogging son:

- Sencillez: es una plataforma que no requiere muchos conocimientos y es muy fácil de entender para el usuario.
- Inmediatez: Podemos lanzar mensajes a la red en segundos desde cualquier parte del mundo, solamente necesitamos tener conexión a Internet.
- Brevedad: ésta es una de sus mayores cualidades y que la diferencia de los demás. Tiene un límite máximo de 280 caracteres por tweet, antiguamente el límite era de 140. De esta forma podemos expresar cualquier información de forma resumida de la que se puede extraer una idea general de lo que se está hablando.
- Multiplataforma: se puede utilizar a través de su aplicación móvil o a través de página web y es compatible con muchos de los sistemas operativos y navegadores que utilizamos.

- Universalidad: está disponible en más de 25 idiomas y puede accederse a él desde casi cualquier lugar del planeta.
- Gratuidad: el servicio que proporciona es gratuito para todos los usuarios y organizaciones. Existe la posibilidad de una versión de pago por publicidad enfocado a anunciantes. Aproximadamente desde marzo de 2017, esta plataforma se está planteando realizar una versión Premium de pago por suscripción que estaría enfocada en empresas y profesionales que quieran tener acceso a herramientas adicionales de la gestión de sus perfiles.
- Ilimitado: permite el envío de mensajes sin limitación alguna. Puedes enviar el número de tweets que quieras y en el momento que quieras difundirlos.
- Asimetría: no es necesario que otros usuarios te sigan para poder ver la información que difunden.
- Accesible: cualquier persona u organización que quiera crear una cuenta y utilizarla puede hacerlo, aunque existen algunas restricciones inscritas en las normas y reglas de uso de Twitter.
- Multiformato: permite la inclusión de texto, imágenes, vídeos, gifs y un formato de encuesta que hacen más visual la comunicación de los mensajes.

Es importante saber, que tipo de usuarios hacen uso de esta Red Social, y podemos diferenciarlos a gran escala entre personas físicas y Entidades, empresas o corporaciones.

Pero también los podemos diferenciar de la siguiente manera:

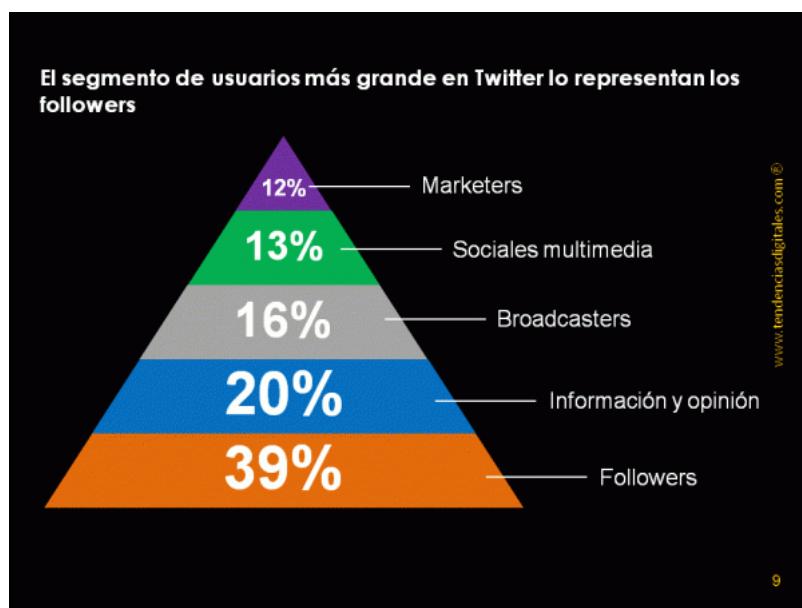


Figura 4: Segmento de usuarios de Twitter

**Followers:** usuarios de la red social que siguen a otros, pero sin generar contenido.

**Información y Opinión:** Personas que generan contenidos informativos y de opinión y siguen a otros que generan este tipo de contenido.

**Broadcasters:** quienes son en su mayoría empresas que colocan contenidos en Twitter que ya han sido publicados por ellos en otros medios.

**Sociales multimedia:** Comparten fotos, generan contenidos, comparten información, difunden videos y aprovechan al máximo las opciones que otorga la red social.

**Marketers:** definidos por los que promueve su negocio y profesión a través de Twitter.

## 1.4. Ventajas y desventajas

Dentro de las cualidades y el tipo de usuario existen ciertas ventajas y desventajas:

### Ventajas a nivel personal

- Conocer las últimas tendencias en el mundo.
- Seguir y estar conectado con personajes públicos, siguiendo sus cuentas oficiales.
- Enlazar tu cuenta de Twitter con tu perfil de Facebook, así cada vez que pubiques en Twitter se podrá conocer tus publicaciones en la otra red social.
- Crear listas dinámicas según tus intereses o preferencias.
- Dar e informar tu opinión en base a un tema.
- Acceder desde distintos dispositivos móviles o computadoras.
- Tener información sobre acontecimientos en tiempo real.
- Crear tweets que interactúen con tus seguidores.
- Mejorar tu imagen como persona.
- Hacer transmisiones en vivo gracias a su servicio de Periscope.
- Conectar con otros profesionales.
- Crear hashtags para resaltar palabras que quieras que resalte en tu publicación.

### Ventajas a nivel de marca

- Mejorar su reputación online como marca.
- Crear oportunidades de ventas en los anuncios de Twitter.
- Generar más leads al integrar Twitter Ads a tu campaña.
- Acceder gratuitamente a Twitter en su servicio 24/7.
- Estar conectados con tus seguidores, que pueden ser o convertirse en clientes...

- Monitorear la competencia, conocer la cantidad de seguidores, y saber si sus publicaciones tienen retweets.
- Ser un excelente canal para ofrecer tus nuevos productos o servicios.
- Promocionar tus eventos, campañas, conferencias, cursos que se lleva a cabo en tu empresa.
- Realizar encuestas a tus seguidores para tener más información sobre un tema.
- Conocer de primera mano las necesidades, dudas o problemas de tus seguidores y/o clientes.
- Tener información en tiempo real, lo que permite tomar acciones.
- Ampliar la cartera de proveedores para mejorar su servicio.
- Crear fidelidad de seguidores agrupándolos en listados de Twitter.

### **Desventajas a nivel personal**

- Tener como seguidores a personas desconocidas o con perfiles falsos como seguidores que conozcan nuestras actualizaciones.
- Tener inconvenientes a la hora de crear un tweet de 140 caracteres.
- Recibir mensajes directos de personas desconocidas
- Abusar de los hashtags puede convertirse en una publicación desagradable para tus seguidores.
- Recibir tweets que agreden la integridad de personas o personajes públicos.
- Tener gran cantidad de mensajes en spam.

### **Desventajas a nivel de marca**

- Ser víctimas de fraude debido a que puede haber cuentas que lleven el nombre de tu marca y confundan a tus seguidores.
- Quedar expuesto a cualquier error, perjudicando la imagen de tu marca.
- Invertir en anuncios de Twitter Ads para posicionar tu marca.
- Contestar después de mucho tiempo al mensaje de tus seguidores, tomando en cuenta que lo que prima en Twitter es la immediatez.
- Tener inconvenientes al momento de brindar información valiosa en los 140 caracteres.
- Aburrir a tus seguidores con mensajes o promociones de tus productos o servicios a cada hora.
- Dar pistas de nuestra estrategia a nuestros competidores.

## **2. Objetivos**

El principal objetivo de este proyecto es aprender de forma autodidacta que es el Machine Learning y el mundo que existe alrededor de él, así como el análisis de sentimientos mediante modelos supervisados, los distintos tipos de algoritmos que existen para el aprendizaje de los datos de entrenamiento y entender las diferencias entre aprendizaje supervisado y no supervisado.

El objetivo final del proyecto es saber entender que significan nuestros resultados y como analizarlos, explicando el porqué de nuestro preprocesamiento y que nos ha llevado a elegir un modelo u otro.

A su vez, que el trabajo de fin de grado sirva para poner a prueba las herramientas y conocimientos aprendidos a lo largo de mi periodo en la universidad, y saber adaptarlos a tecnologías nuevas o lenguajes de programación no usados, como es el caso de Python.

### 3. Machine Learning

#### 3.1. Introducción al Machine Learning

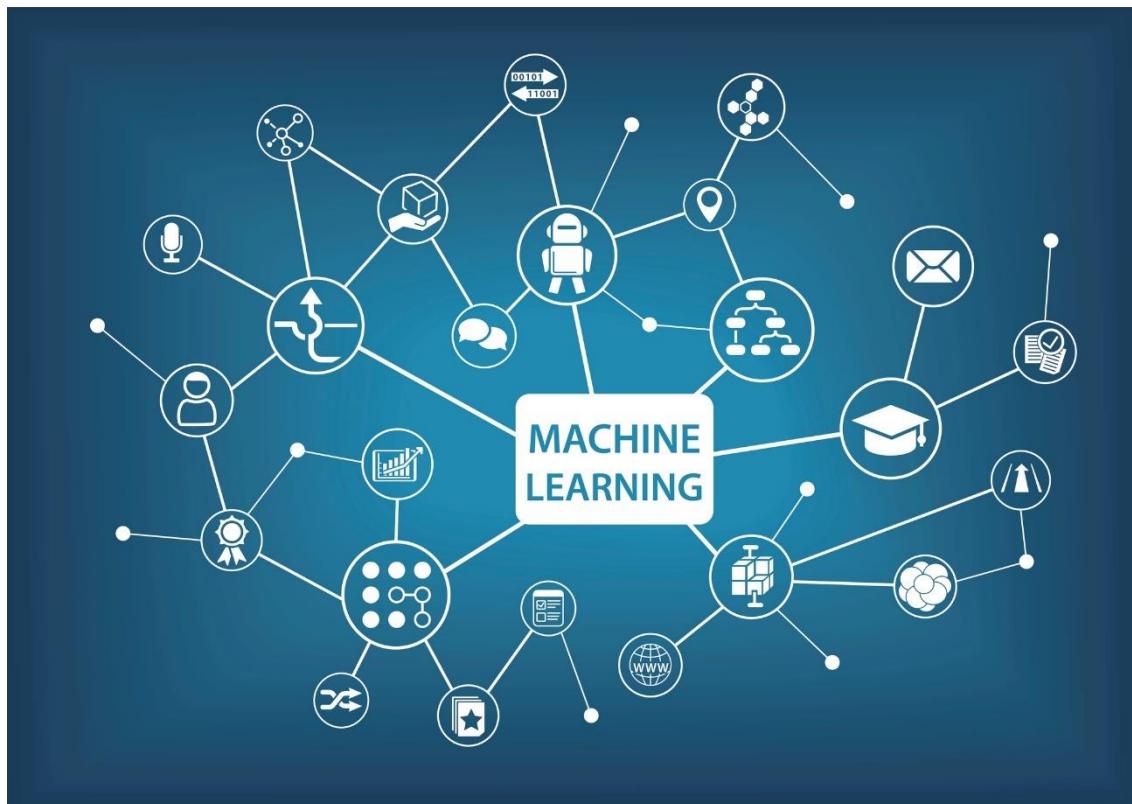


Figura 5: Inteligencia Artificial: Machine Learning

Machine Learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

En la informática clásica, el único modo de conseguir que un sistema informático hiciera algo era escribiendo un algoritmo que definiera el contexto y detalles de cada acción.

Sin embargo, los algoritmos que se usan en el desarrollo del Machine Learning realizan buena parte de estas acciones por su cuenta. Obtienen sus propios cálculos según los datos que se recopilan en el sistema, y cuantos más datos obtienen, mejores y más precisas serán las acciones resultantes.

Las computadoras se programan a sí mismas, hasta cierto punto, usando dichos algoritmos. Estos funcionan como ingenieros que pueden diseñar nuevas respuestas informáticas, como respuesta a la información que se les suministra a través de su interfaz u otros medios. Todo nuevo dato se convierte en un nuevo algoritmo, y a más

datos, mayor complejidad y efectividad de cálculo puede proporcionar el sistema informático.

La clave de la capacidad de un sistema de Aprendizaje Automático se encuentra en la construcción y adaptación de los árboles de decisiones en base a los datos previamente conocidos por el sistema.

Pero también influye la aplicación de fórmulas heurísticas en los nodos que forman el árbol, para el que se elabora un sistema de inferencias.

El sistema de Machine Learning necesita contar con un volumen de datos de relevancia para poder suministrar respuestas realmente válidas. El mínimo que se recomienda es de 6 entradas de datos reales para cada respuesta nueva diseñada, y esto debe repetirse para cada variable que conforman el sistema de trabajo del sistema.

Hay que remontarse al siglo XIX para encontrar algunos de los hitos matemáticos que sentaron las bases de esta tecnología. El teorema de Bayes (1812) definió la probabilidad de que un evento ocurra basándose en el conocimiento de las condiciones previas que pudieran estar relacionadas con dicho evento.

Años después (en la década de 1940) otra serie de científicos sentaron las bases de la programación informática: capaz de traducir una serie de instrucciones en acciones ejecutables por un ordenador. Estos precedentes hicieron posible que en 1950 el matemático Alan Turing plantease por primera vez la pregunta de si es posible que las máquinas puedan pensar, con la que plantó la semilla de la creación de computadoras de ‘inteligencia artificial’. Computadoras capaces de replicar de forma autónoma tareas típicamente humanas, como la escritura o el reconocimiento de imágenes.

Fue un poco más adelante, entre las décadas de 1950 y 1960, cuando distintos científicos empezaron investigar cómo aplicar la biología de las redes neuronales del cerebro humano para tratar de crear las primeras máquinas inteligentes. La idea derivó en la creación de las redes neuronales artificiales, un modelo computacional inspirado en la forma en que las neuronas transmiten la información entre ellas a través de una red de nodos interconectados. Uno de los primeros experimentos en este sentido lo realizaron Marvin Minsky y Dean Edmonds, científicos del Instituto Tecnológico de Massachusetts (MIT). Ambos lograron crear un programa informático capaz de aprender de la experiencia para salir de un laberinto.

Esta fue la primera máquina capaz de aprender por sí misma a resolver una tarea sin haber sido programada para ello de forma explícita, sino que lo hacía tan solo aprendiendo a partir de los ejemplos proporcionados inicialmente. El logro significó un cambio de paradigma respecto al concepto más amplio de inteligencia artificial. “El gran hito del ‘machine learning’ es que permitió pasar de la programación mediante reglas a

dejar que el modelo haga aflorar dichas reglas de manera desasistida gracias a los datos”, explica Juan Murillo, mánager de Estrategia de Datos en BBVA.

A pesar del éxito del experimento, el logro también ponía de manifiesto los límites que la tecnología tenía por entonces: la falta de disponibilidad de datos y la falta de potencia de cómputo de la época hacían que estos sistemas no tuvieran la capacidad suficiente para resolver problemas complejos. Esto derivó en la llegada del llamado ‘primer invierno de la inteligencia artificial’, una serie de décadas durante las cuales la falta de resultados y avances hizo que el mundo académico perdiera esperanza respecto a esta disciplina.

El panorama empezó a cambiar a finales del siglo XX. Con la llegada de internet, las cantidades masivas de información disponibles para entrenar los modelos y el aumento de la potencia de cálculo de los ordenadores. “Ahora podemos hacer lo mismo que antes, pero mil millones de veces más rápido. Los algoritmos son capaces de probar 500.000 millones de veces una misma combinación de datos hasta darnos el resultado óptimo. Y lo hacen en cuestión de horas o minutos, mientras que antes harían falta semanas o meses”, asegura Espinoza.

En 1997 un célebre hito marcó el renacer del aprendizaje automático: el sistema de IBM Deep Blue, entrenado a base de ver miles de partidas exitosas de ajedrez, logró derrotar al máximo campeón mundial de este juego, Garry Kasparov. El logro fue posible gracias al ‘deep learning’ o aprendizaje profundo, una subcategoría del ‘machine learning’ descrita por primera vez en 1960, que permite que los sistemas no solo aprendan de la experiencia, sino que sean capaces de entrenarse a sí mismas para hacerlo cada vez mejor usando los datos. Esto hito fue posible entonces –y no 30 años atrás–, gracias al aumento de la disponibilidad de datos con los que entrenar el modelo: “Lo que hacía este sistema es calcular estadísticamente qué movimiento tiene más probabilidades de hacerle ganar la partida basándose en miles de ejemplos de partidas vistas previamente”.

### 3.2. Tipos de aprendizaje

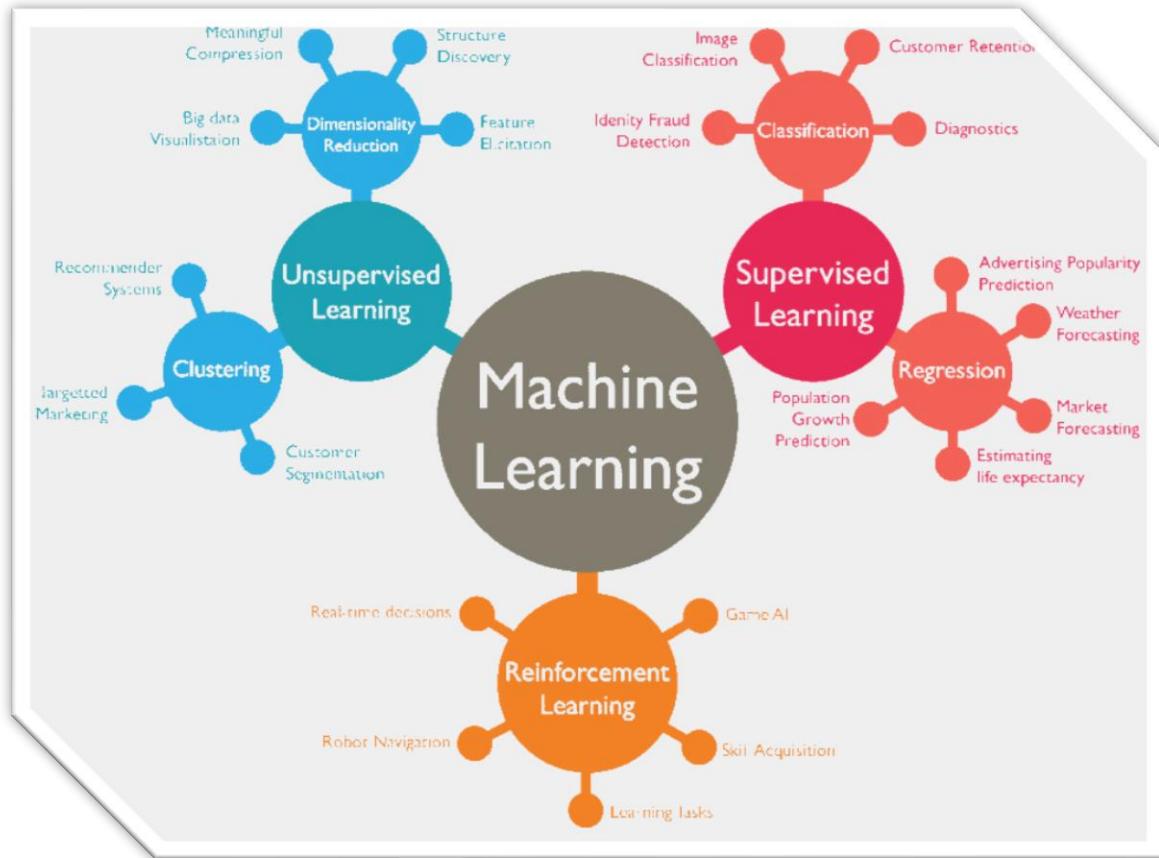


Figura 6: Tipos de aprendizaje en Machine Learning

Los tipos de implementación de machine Learning pueden clasificarse en tres categorías diferentes:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje de refuerzo según la naturaleza de los datos que recibe.

### 3.2.1. Aprendizaje supervisado

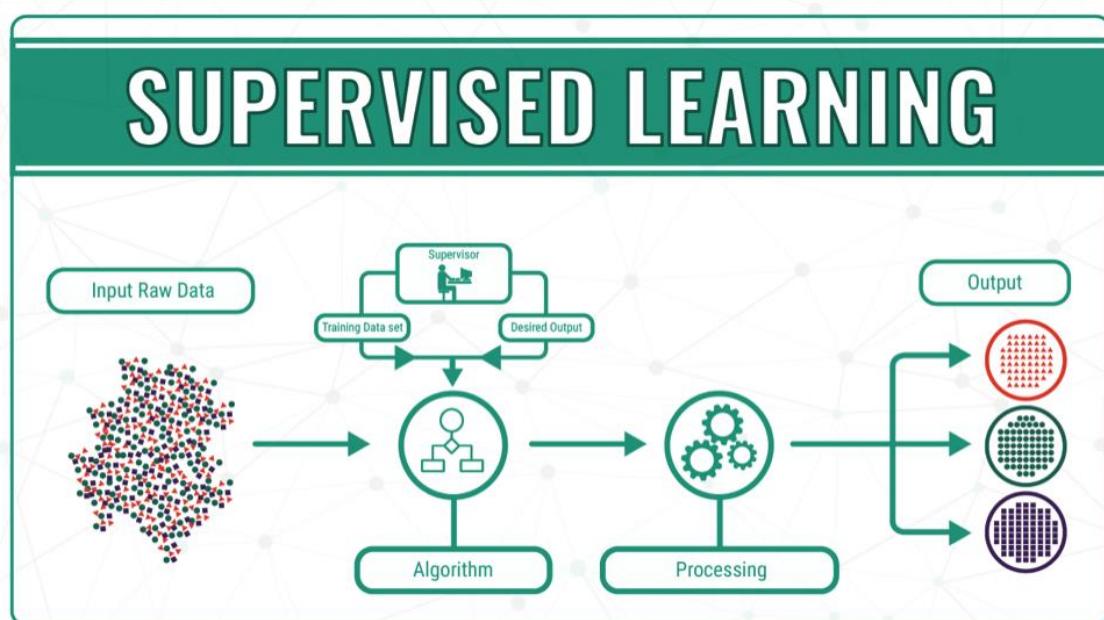


Figura 7: Aprendizaje Supervisado

En el aprendizaje supervisado, los algoritmos trabajan con datos “etiquetados” (*labeled data*), intentando encontrar una función que, dadas las variables de entrada (*input data*), les asigne la etiqueta de salida adecuada. El algoritmo se entrena con un “histórico” de datos y así “aprende” a asignar la etiqueta de salida adecuada a un nuevo valor, es decir, predice el valor de salida.

El aprendizaje supervisado se suele usar en:

- Problemas de *clasificación* (identificación de dígitos, diagnósticos, o detección de fraude de identidad).

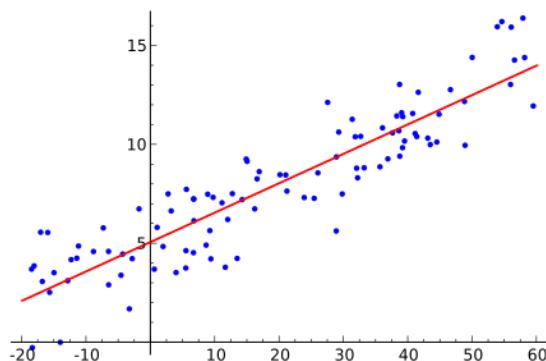


Figura 8: Problemas de clasificación

- Problemas de *regresión* (predicciones meteorológicas, de expectativa de vida, de crecimiento, etc.).

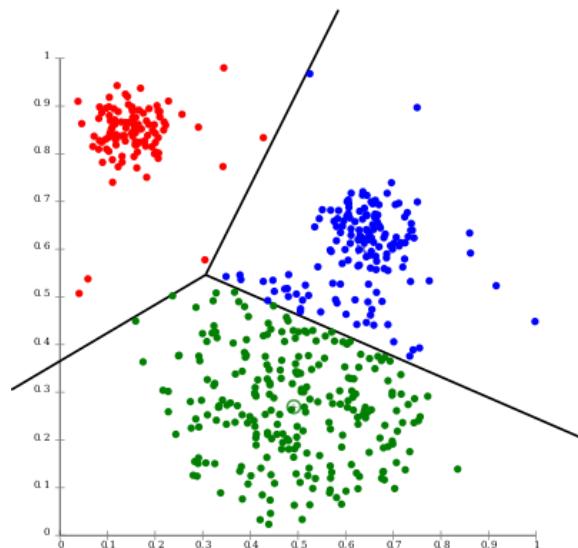


Figura 9: Problemas de regresión

Estos dos tipos principales de aprendizaje supervisado, clasificación y regresión, se distinguen por el tipo de variable objetivo. En los casos de clasificación, es de tipo categórico, mientras que, en los casos de regresión, la variable objetivo es de tipo numérico.

Los algoritmos más habituales que aplican para el aprendizaje supervisado son:

### 3.2.1.1. Algoritmos de regresión

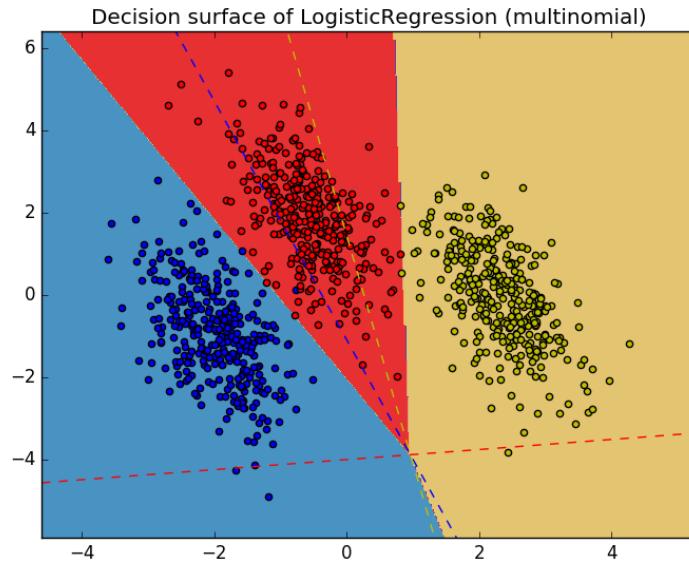


Figura 10: LogisticRegression (multinomial)

En las tareas de regresión, el programa de aprendizaje automático debe estimar y comprender las relaciones entre las variables. El análisis de regresión se enfoca en una variable dependiente y una serie de otras variables cambiantes, lo que lo hace particularmente útil para la predicción y el pronóstico.

### 3.2.1.2. Algoritmos bayesianos

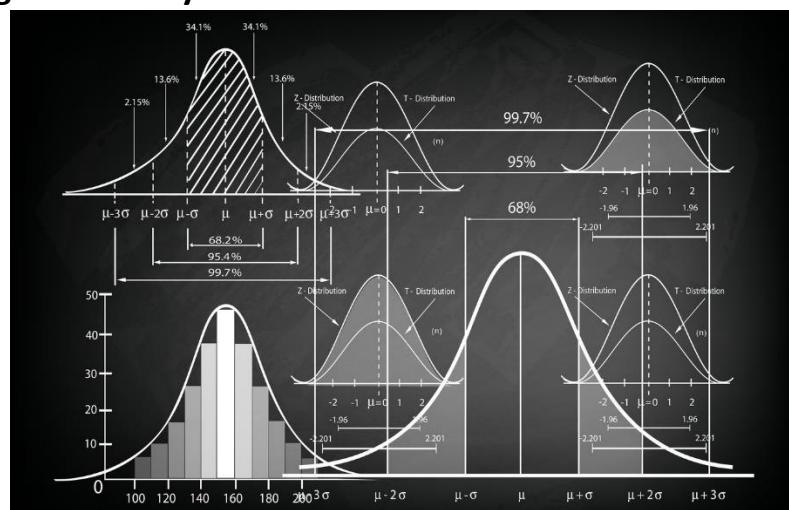


Figura 11: Naive Bayes

Este tipo de algoritmos por clasificación están basados en el teorema de Bayes y clasifican cada valor como independiente de cualquier otro. Lo que permite predecir una clase o categoría en función de un conjunto dado de características, utilizando la probabilidad.

Pese a su simplicidad, el clasificador funciona sorprendentemente bien y se usa a menudo porque supera a los métodos de clasificación más sofisticados.

### 3.2.1.3. Algoritmos de árbol de decisión



Figura 12: Decision Tree

Un árbol de decisión es una estructura de árbol similar a un diagrama de flujo que utiliza un método de bifurcación para ilustrar cada resultado posible de una decisión. Cada nodo dentro del árbol representa una prueba en una variable específica, y cada rama es el resultado de esa prueba.

### 3.2.1.4. Algoritmos de redes neuronales

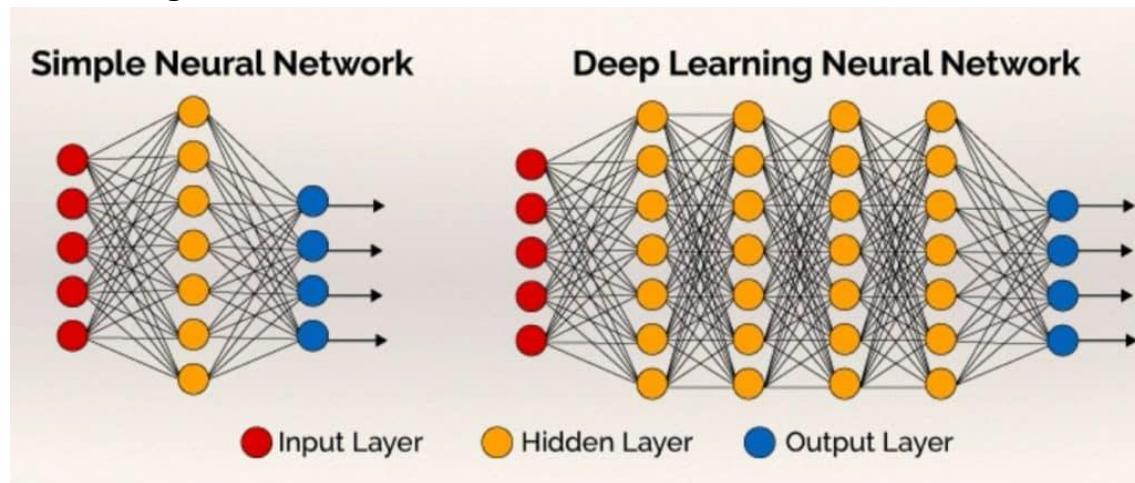


Figura 13: Deep Learning Neural Network

Una red neuronal artificial (RNA) comprende unidades dispuestas en una serie de capas, cada una de las cuales se conecta a las capas anexas. Las RNA se inspiran en los sistemas biológicos, como el cerebro, y en cómo procesan la información.

Por lo tanto, son esencialmente un gran número de elementos de procesamiento interconectados, que trabajan al unísono para resolver problemas específicos.

También aprenden con el ejemplo y la experiencia, y son extremadamente útiles para modelar relaciones no lineales en datos de alta dimensión, o donde la relación entre las variables de entrada es difícil de entender.

### 3.2.1.5. Algoritmos de reducción de dimensión

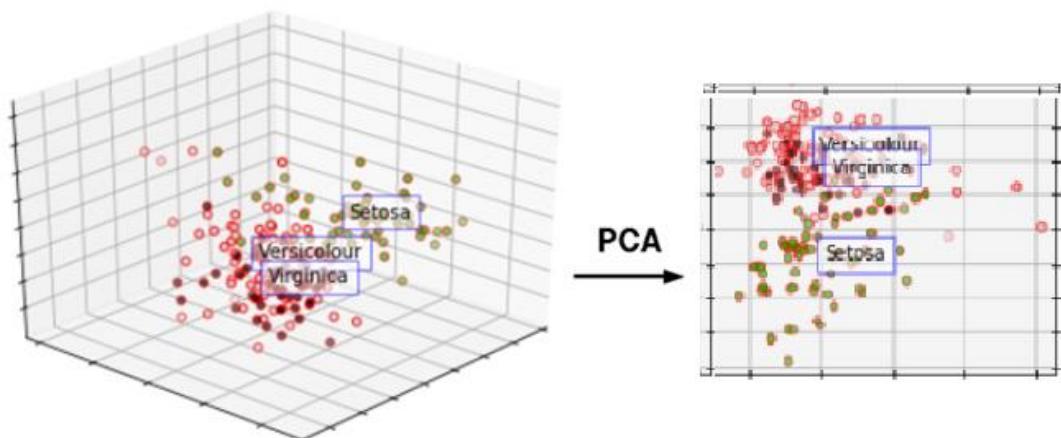


Figura 14: PCA Algorithm

La reducción de dimensión reduce el número de variables que se consideran para encontrar la información exacta requerida.

### 3.2.1.6. Deep Learning

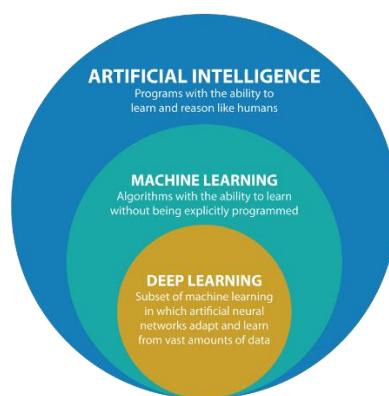


Figura 15: Deep Learning

Los algoritmos de aprendizaje profundo ejecutan datos a través de varias capas de algoritmos de redes neuronales, las cuales pasan a una representación simplificada de los datos a la siguiente capa.

La mayoría funciona bien en conjuntos de datos que tienen hasta unos cientos de características o columnas. Sin embargo, un conjunto de datos no estructurado, como el de una imagen, tiene una cantidad tan grande de características que este proceso se vuelve completamente inviable.

### 3.2.1.7. K-NN más cercanos (K-NN)

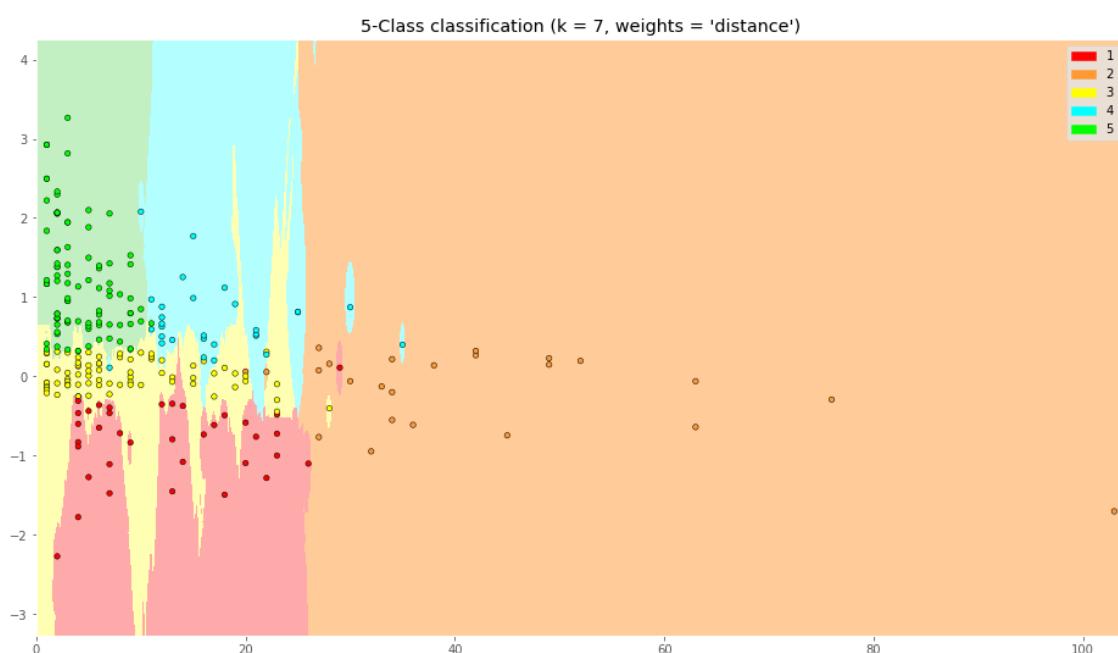


Figura 16: KNN Nearest Neighbors

El algoritmo K-NN es uno de los algoritmos de clasificación más simples y se usa para identificar los puntos de datos que están separados en varias clases para predecir la clasificación de un nuevo punto de muestra. K-NN es un no paramétrico, algoritmo de aprendizaje perezoso. Clasifica los casos nuevos en función de una medida de similitud.

KNN funciona bien con un pequeño número de variables de entrada ( $p$ ), pero tiene problemas cuando el número de entradas es muy grande.

### 3.2.2. Aprendizaje no supervisado

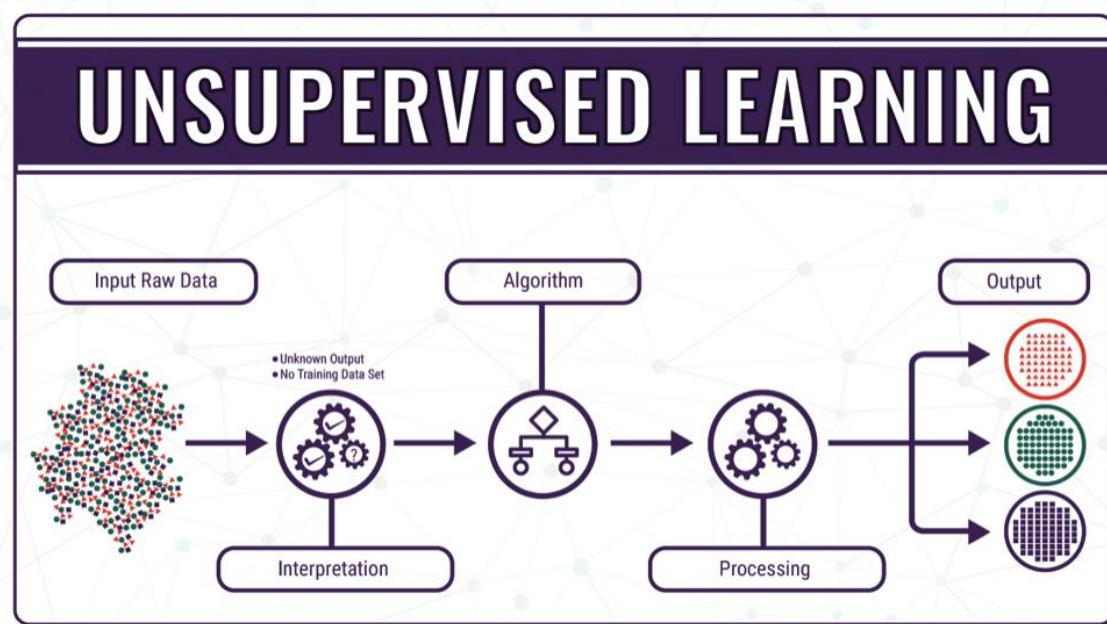


Figura 17: Aprendizaje No Supervisado

El aprendizaje no supervisado tiene lugar cuando no se dispone de datos “etiquetados” para el entrenamiento. Sólo conocemos los datos de entrada, pero no existen datos de salida que correspondan a un determinado *input*. Por tanto, sólo podemos describir la estructura de los datos, para intentar encontrar algún tipo de organización que simplifique el análisis. Por ello, tienen un carácter exploratorio.

El aprendizaje no supervisado se suele usar en:

- Problemas de *clustering*
- Agrupamientos de co-ocurrencias
- Perfilado o *profiling*.

Si embargo, los problemas que implican tareas de encontrar similitud, predicción de enlaces o reducción de datos, pueden ser supervisados o no.

### 3.2.2.1. Algoritmos de clustering

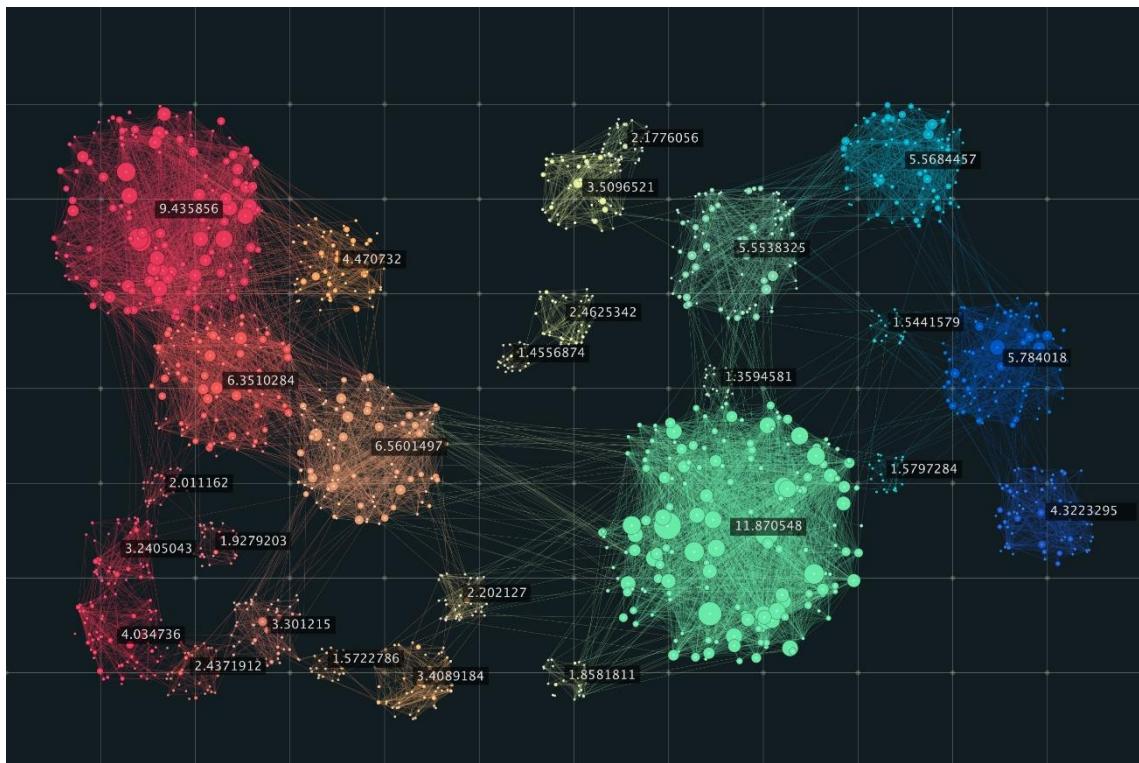


Figura 18: Algoritmos de Clustering

Se utilizan en el aprendizaje no supervisado, y sirven para categorizar datos no etiquetados, es decir, datos sin categorías o grupos definidos.

El algoritmo funciona mediante la búsqueda de grupos dentro de los datos, con el número de grupos representados por la variable K. A continuación, funciona de manera iterativa para asignar cada punto de datos a uno de los K grupos según las características proporcionadas.

### 3.2.3. Aprendizaje por refuerzo

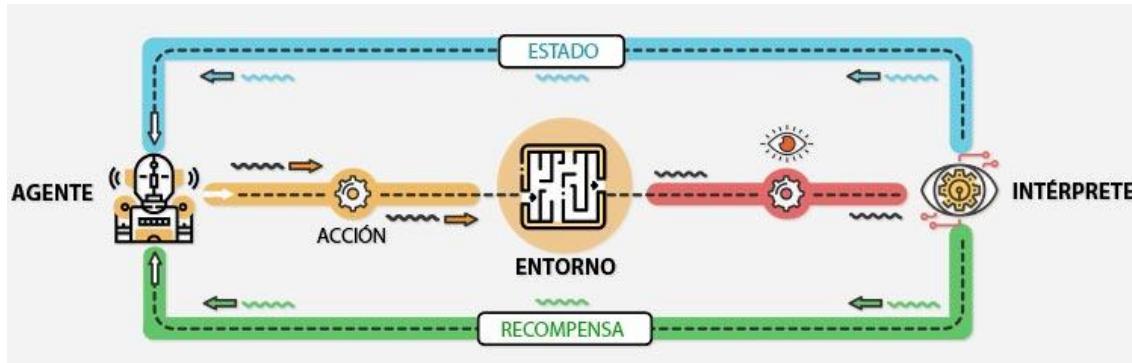


Figura 19: Aprendizaje por refuerzo

Este tipo aprendizaje se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación. El algoritmo aprende observando el mundo que le rodea. Su información de entrada es el *feedback* o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error.

No es un tipo de aprendizaje supervisado, porque no se basa estrictamente en un conjunto de datos etiquetados, sino en la monitorización de la respuesta a las acciones tomadas. Tampoco es un aprendizaje no supervisado, ya que, cuando modelamos a nuestro “aprendiz” sabemos de antemano cuál es la recompensa esperada.

## 3.3. Usos prácticos del Machine learning

- Seguridad informática, diagnóstico de ataques, prevención de fraude online, detección de anomalías, etc.
- Reconocimiento de imágenes o patrones (facial, dactilar, objetos, voz, etc)
- Conducción autónoma, mediante algoritmos *deep learning*: identificación de imágenes en tiempo real, detección de obstáculos y señales de tráfico, prevención de accidentes...
- Salud: evaluación automática de pruebas diagnósticas, robótica médica etc
- Análisis de mercado de valores (predicciones financieras, evolución de mercados etc)
- Motores de recomendación

### 3.4. Análisis de sentimientos

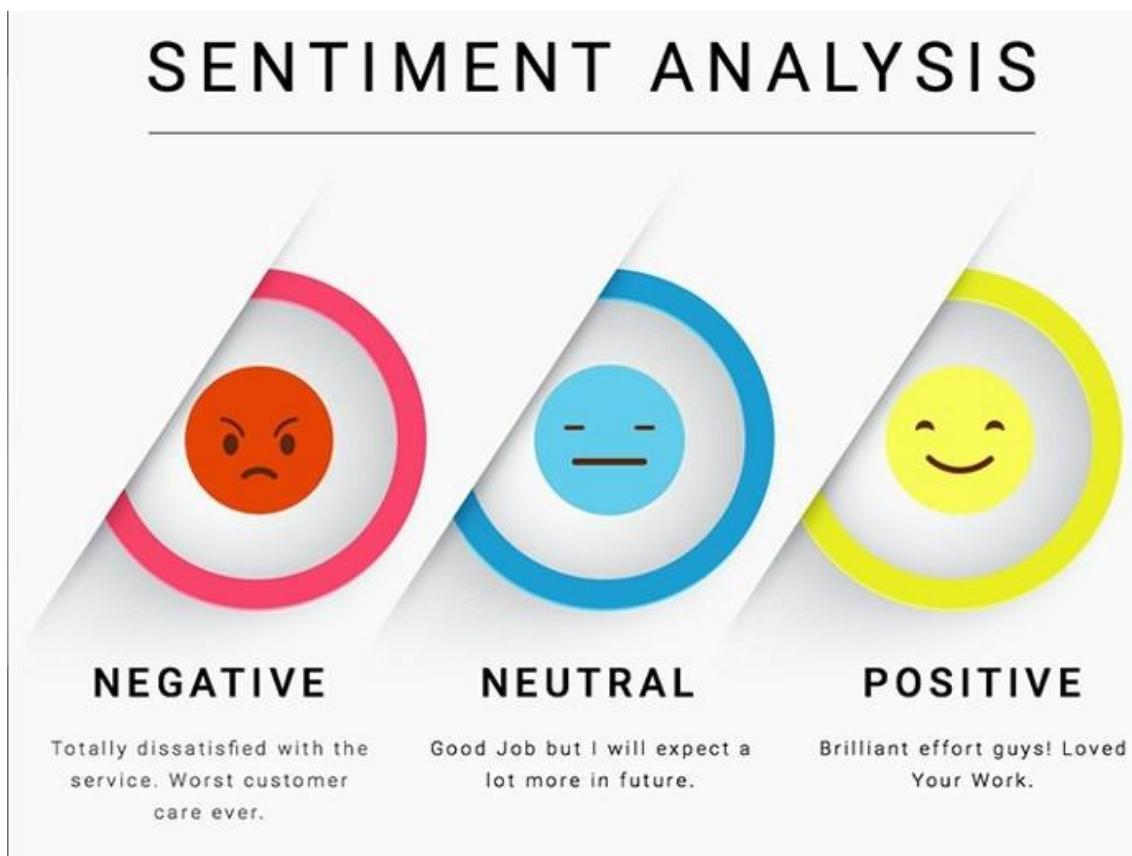


Figura 20: Análisis de sentimientos

El análisis de sentimiento es una área del Procesamiento de Lenguaje Natural ([NLP](#)) que tras el auge del uso de *Machine Learning* ha recobrado mucha fuerza. Y a partir de aquí se han encontrado una gran variedad de aplicaciones que lo han hecho un tema de mucha investigación y debate.

El *social sentiment* es una técnica que pretende encontrar la opinión polarizada de un texto. Vivimos en la época donde se generan más textos que nunca. Las redes sociales son el lugar de internet donde podemos encontrar más opiniones que en ningún otro lado. Dar con las opiniones de las audiencias se ha vuelto una tarea compleja pero muy enriquecedora.

La mayor cantidad de textos que podemos recabar de las redes sociales no tienen una estructura en común. Con la ayuda del análisis de sentimiento puedes estructurar todos estos textos para convertirlos en datos útiles que proyecten la opinión de una audiencia sobre un producto, servicio, un lanzamiento, una marca, un personaje, un político, un evento o cualquier otro tema sobre el que se pueda tener una opinión.

### **3.4.1. Tipos de análisis de sentimiento**

#### **3.4.1.1. Polarizado**

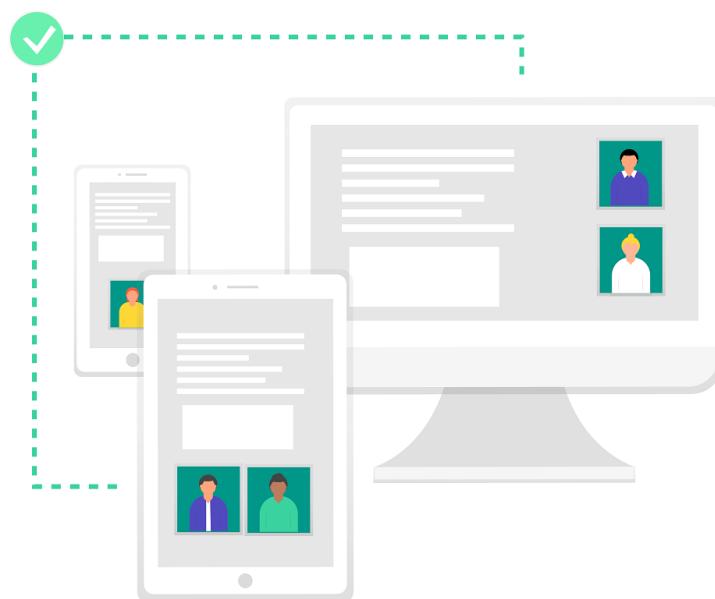
Este es el análisis más común y consiste en polarizar una opinión hacia algo positivo, algo negativo o algo neutral. En algunos casos se opta por hacerlo más granular, incluyendo las categorías de ‘muy positivo’ y ‘muy negativo’. Otro uso es omitir la categoría neutral y ubicar todo texto como algo positivo o negativo para quien lo analiza.

#### **3.4.1.2. Intención**

Este tipo de clasificación se enfoca en lo que los usuarios quieren hacer con lo que dice y no tanto en lo que dicen. Suele tener etiquetas como: interesado y no interesado.

#### **3.4.1.3. Emoción**

Una clasificación más especializada puede ir hacia la detección puntual de la emoción que se transmite, siendo las más comunes: Alegría, Felicidad, Frustración, Enojo, Tristeza. Este tipo de clasificación de sentimiento es muy utilizado en lanzamientos/campañas de producto para medir que tan alineadas están las expectativas contra la opinión de la audiencia.



*Figura 21: Relevancia del análisis de sentimientos*

## 4. Conceptos previos

### 4.1. Dataset

Un conjunto de datos o dataset corresponde a los contenidos de una única tabla de base de datos o una única matriz de datos de estadística, donde cada columna de la tabla representa una variable en particular, y cada fila representa a un miembro determinado del conjunto de datos que estamos tratando.

### 4.2. DataFrame

Los data frames son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas. Esta estructura de datos es la más usada para realizar análisis de datos y seguro te resultará familiar si has trabajado con otros paquetes estadísticos.

Podemos entender a los data frames como una versión más flexible de una matriz. Mientras que en una matriz todas las celdas deben contener datos del mismo tipo, los renglones de un data frame admiten datos de distintos tipos, pero sus columnas conservan la restricción de contener datos de un sólo tipo.

Está compuesto por vectores.

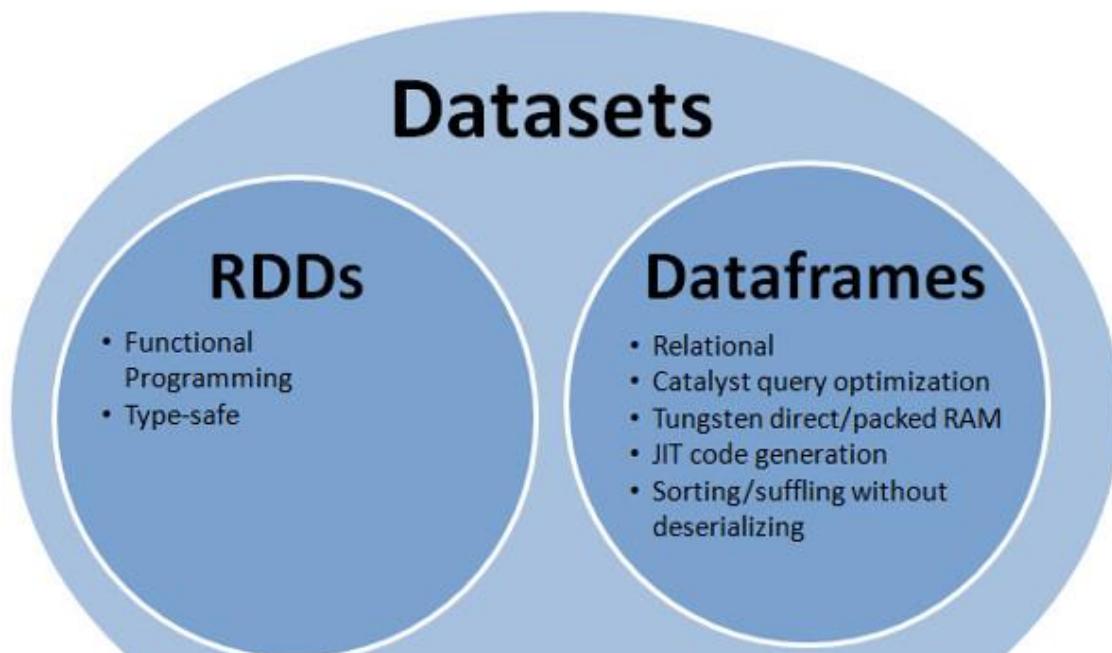


Figura 22: Diferencias entre Datasets y DataFrames

## 4.3. CSV

The screenshot shows a Microsoft Excel spreadsheet with the following details:

- Title Bar:** Shows "Autoguardado" and "training.1600000.processed.noemotion.csv".
- Menu Bar:** Includes Archivo, Inicio, Insertar, Disposición de página, Fórmulas, Datos, Revisar, Vista, Ayuda.
- Toolbar:** Includes Cortar, Copiar, Pegar, Combinar y centrar, Ajustar texto, General, Normal, Bueno, Incorrecto, Celda de co..., Celda vincul..., Entrada.
- Cells:** Column A contains 38 numbered entries from 1 to 38, each followed by a timestamp and a multi-line text message. For example, entry 1 starts with "1 target," and entry 38 starts with "38 ,146781374", both followed by their respective messages.
- Bottom Status Bar:** Shows "training.1600000.processed.noemotion" and a small plus sign icon.

Figura 23: Training 1600000 proceseed neomoticon csv

Las siglas CSV vienen del inglés “Comma Separated Values”, o lo que es lo mismo, valores separados por comas. Un archivo CSV es cualquier archivo de texto en el cual los caracteres están separados por comas, creando una tabla formada por filas y columnas. Las columnas quedan definidas por cada punto y coma (;), mientras que cada fila se define mediante una línea adicional en el texto. Este tipo de archivos están asociados directamente a la creación de tablas de contenido. Sirven para manejar una gran cantidad de datos en formato tabla, sin que ello lleve sobrecoste computacional alguno.

## 5. Herramientas utilizadas

### 5.1 Python

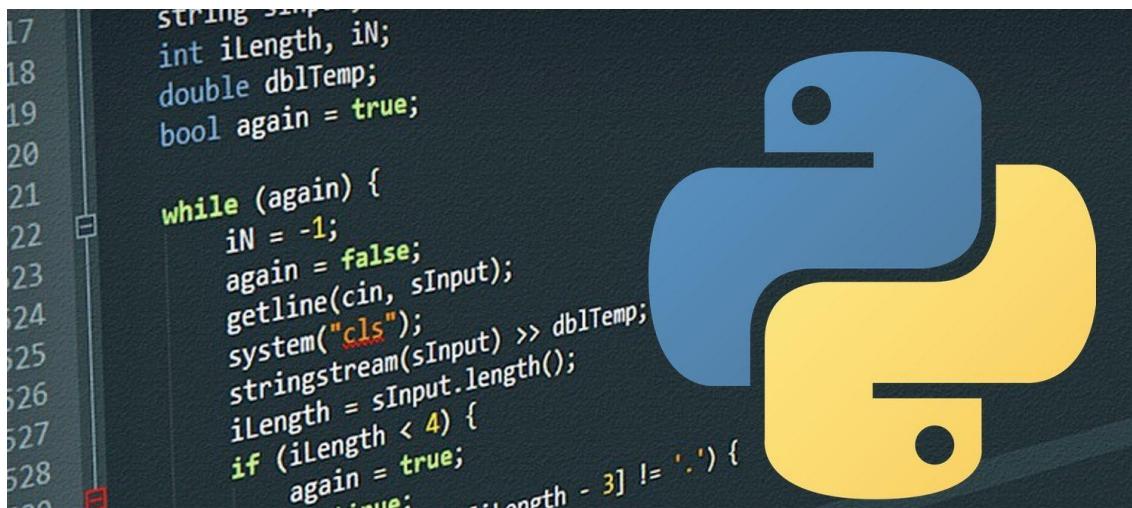


Figura 24: Python

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Es un lenguaje de programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. Una de las razones de su éxito es que cuenta con una licencia de código abierto que permite su utilización en cualquier escenario. Esto hace que sea uno de los lenguajes de iniciación de muchos programadores siendo impartido en escuelas y universidades de todo el mundo. Sumado a esto cuenta con grandes compañías que hacen de este un uso intensivo.

Python es ideal para trabajar con grandes volúmenes de datos ya que, el ser multiplataforma, favorece su extracción y procesamiento, por eso lo eligen las empresas de Big Data. A nivel científico, tiene una gran biblioteca de recursos con especial énfasis en las matemáticas para aspirantes a programadores en áreas especializadas

#### 5.1.1. Ventajas de programar en Python

- **Simplificado y rápido:** Este lenguaje simplifica mucho la programación, es un gran lenguaje para scripting.
- **Elegante y flexible:** El lenguaje ofrece muchas facilidades al programador al ser fácilmente legible e interpretable.
- **Programación sana y productiva:** Es sencillo de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y fomenta la productividad.

- **Ordenado y limpio:** es muy legible y sus módulos están bien organizados.
- **Portable:** Es un lenguaje muy portable. Podemos usarlo en prácticamente cualquier sistema de la actualidad.
- **Comunidad:** Cuenta con un gran número de usuarios. Su comunidad participa activamente en el desarrollo del lenguaje.

## 5.2 Skicit-Learn

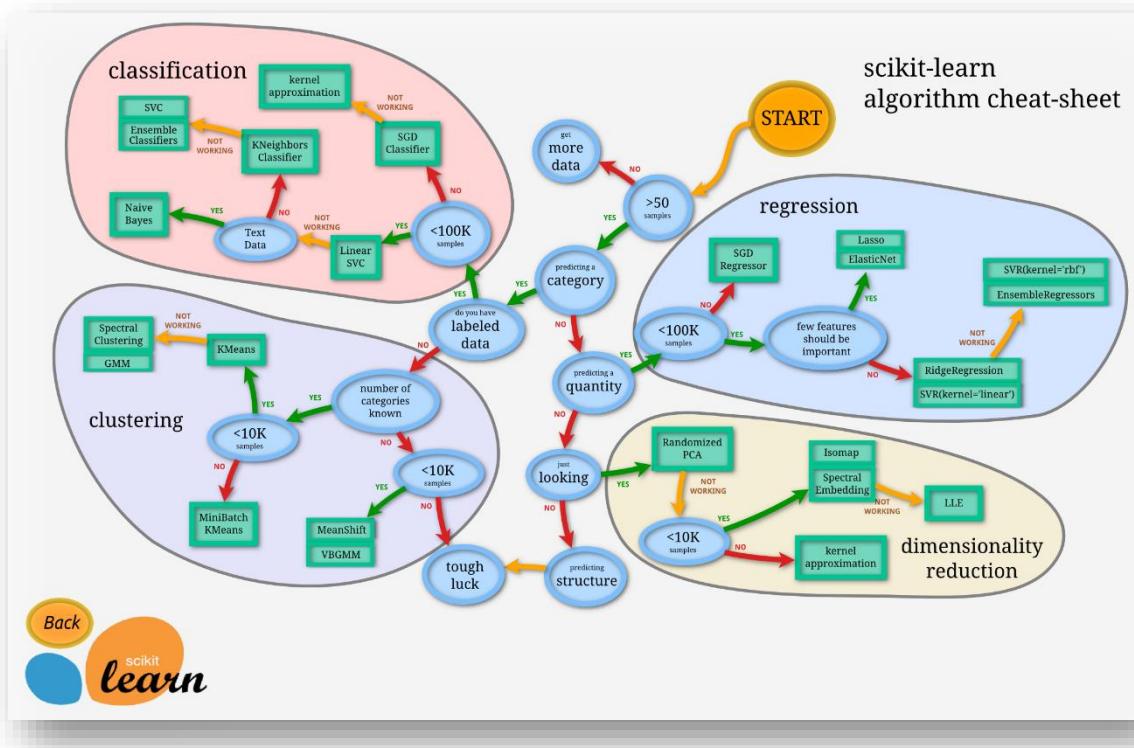


Figura 25: Scikit-learn algoithm cheat-sheet

Scikit-Learn es una de estas librerías gratuitas para Python. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Además, presenta la compatibilidad con otras librerías de Python como NumPy, SciPy y matplotlib.

La gran variedad de algoritmos y utilidades de Scikit-learn la convierten en la herramienta básica para empezar a programar y estructurar los sistemas de análisis datos y modelado estadístico. Los algoritmos de Scikit-Learn se combinan y depuran con otras estructuras de datos y aplicaciones externas como Pandas o PyBrain.

La ventaja de la programación en Python, y Scikit-Learn en concreto, es la variedad de módulos y algoritmos que facilitan el aprendizaje y trabajo del científico de datos en las primeras fases de su desarrollo.

Esta librería está construida sobre SciPy (Scientific Python) e incluye las siguientes librerías o paquetes:

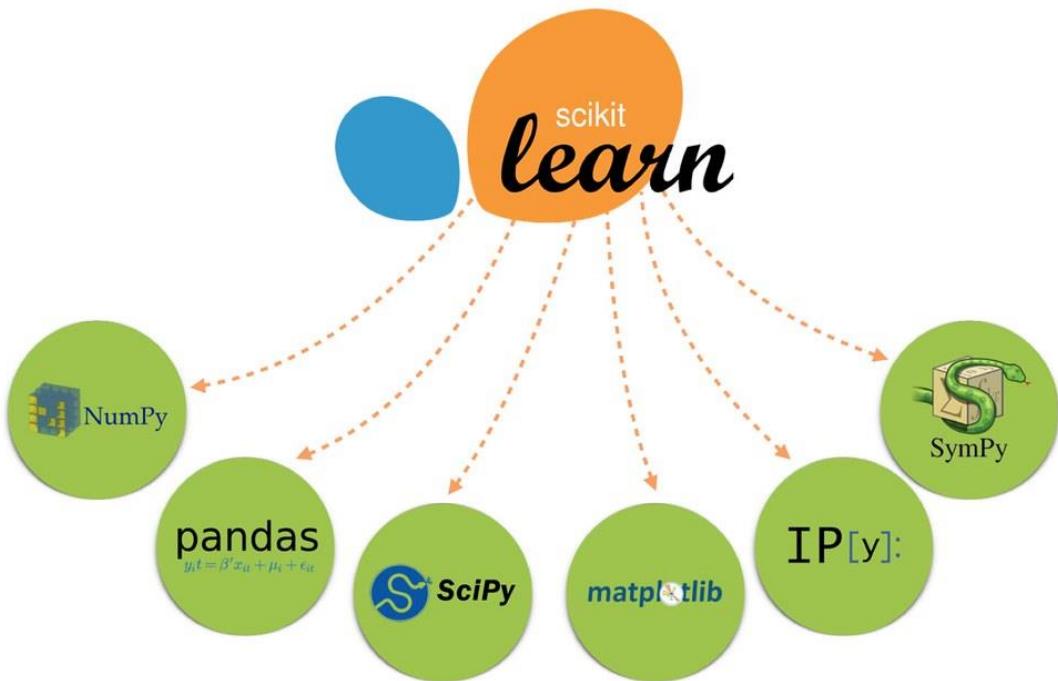


Figura 26: librerías Skicit-learn

- NumPy: librería de matriz n-dimensional base
- Pandas: estructura de datos y análisis
- SciPy: librería fundamental para la informática científica
- Matplotlib: trazado completo 2D
- Ipython: consola interactiva mejorada
- SymPy: matemática simbólica

Para implementar scikit-learn, primero se debe importar los paquetes anteriores, para poderlos implementar en la programación. Se pueden descargar estos paquetes usando las líneas de comando o si estás usando Jupyter estos ya vienen instalados en este IDE.

La gran mayoría de los algoritmos de Machine Learning que entran en la clasificación de aprendizaje supervisado forman parte de scikit-learn, desde los modelos lineales generalizados, como lo es regresión lineal, como las máquinas de vectores de soporte (SVM), árboles de decisión y hasta los métodos bayesianos, todos ellos son parte de las herramientas de scikit-learn.

Existen varios métodos para verificar la precisión de los modelos supervisados y esta librería cuenta con las instrucciones necesarios para poder implementar estos métodos sin tanto esfuerzo.

Igual que los algoritmos de aprendizaje supervisados, esta librería cuenta con una gran variedad de algoritmos disponibles para aprendizaje no supervisado, comenzando por la agrupación, el análisis factorial, el análisis de componentes principales y las redes neuronales no supervisadas.

### 5.3. Numpy

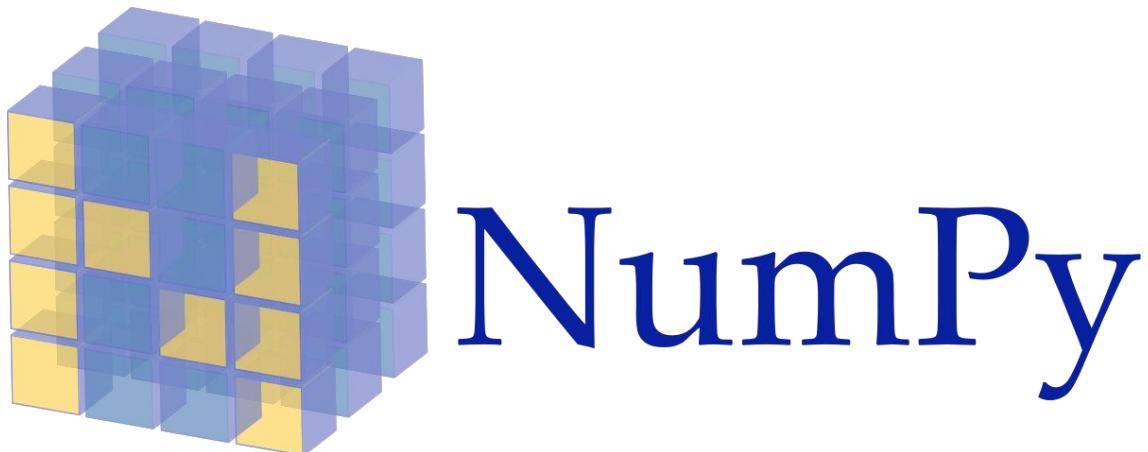


Figura 27: Numpy

Numpy es la librería para computación científica en Python. Trae integradas muchas funciones de cálculo matricial de  $N$  dimensiones, así como la transformada de Fourier, múltiples funciones de álgebra lineal y varias funciones de aleatoriedad.

Es la librería de cálculo más popular debido a su facilidad de uso y la rapidez de sus cálculos, ya que gran parte de la librería está escrita directamente en lenguaje C.

La funcionalidad principal de NumPy es su estructura de datos "ndarray", para una matriz de  $n$  dimensiones. Estas matrices son vistas escalonadas de la memoria.<sup>6</sup> A diferencia de la estructura de datos de lista incorporada de Python, estas matrices se escriben de forma homogénea: todos los elementos de una única matriz deben ser del mismo tipo.

## 5.4. Matplotlib

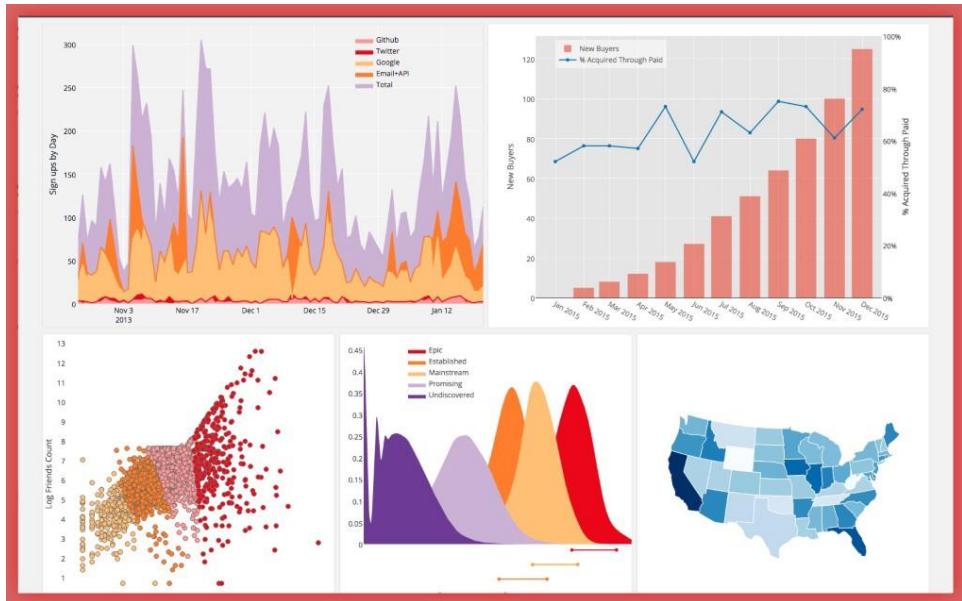


Figura 28: Matplotlib

Matplotlib es una librería de Python que permite realizar gráficas 2D de excelente calidad. Es multiplataforma y puede ser usada desde scripts o desde la consola de Python.

## 5.5. Pandas

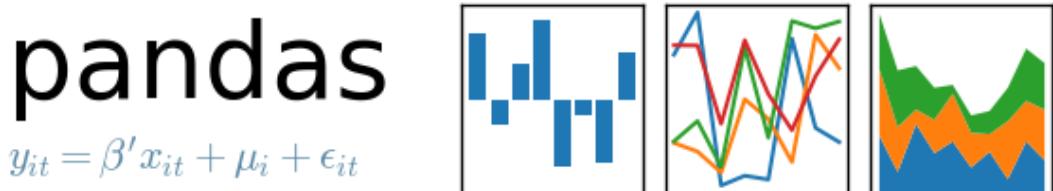


Figura 29: Pandas

Pandas es un paquete de Python que proporciona estructuras de datos similares a los dataframes de R. Pandas depende de Numpy, la librería que añade un potente tipo matricial a Python. Los principales tipos de datos que pueden representarse con pandas son:

- Datos tabulares con columnas de tipo heterogéneo con etiquetas en columnas y filas.
- Series temporales.

Pandas proporciona herramientas que permiten:

- leer y escribir datos en diferentes formatos: CSV, Microsoft Excel, bases SQL y formato HDF5
- seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas
- fusionar y unir datos
- transformar datos aplicando funciones tanto en global como por ventanas
- manipulación de series temporales
- hacer gráficas

En pandas existen tres tipos básicos de objetos todos ellos basados a su vez en Numpy:

- Series (listas, 1D),
- DataFrame (tablas, 2D) y
- Panels (tablas 3D).

## 5.7. Anaconda

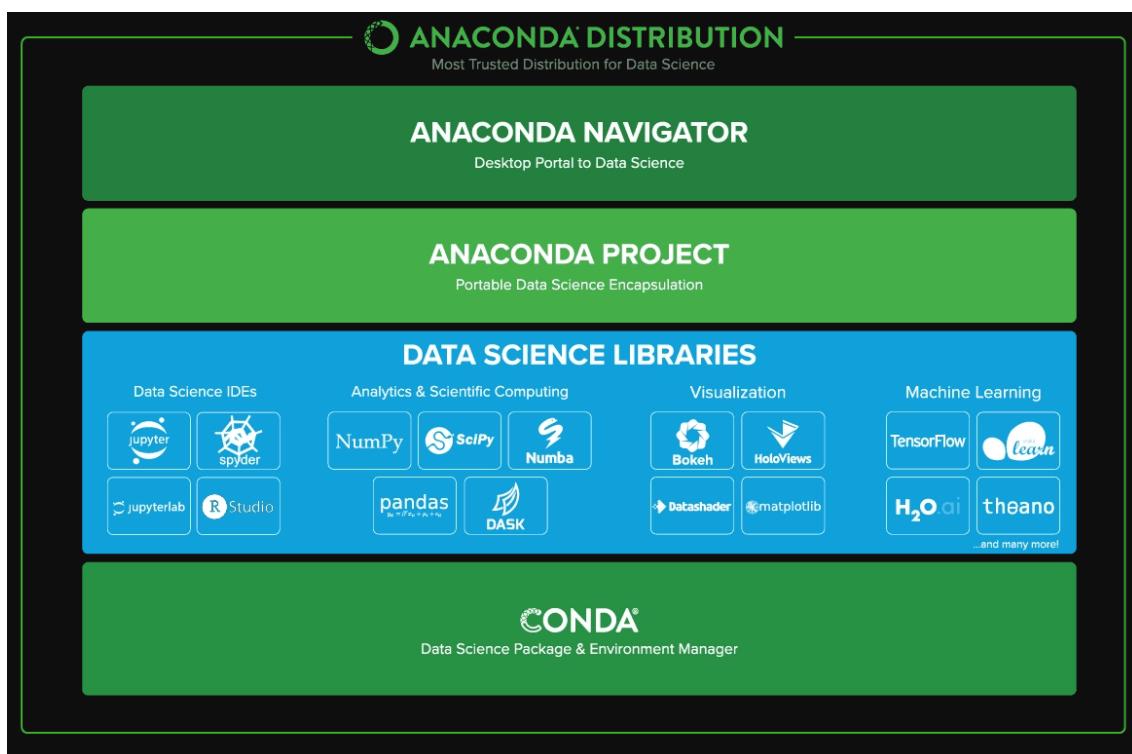


Figura 30: Anaconda

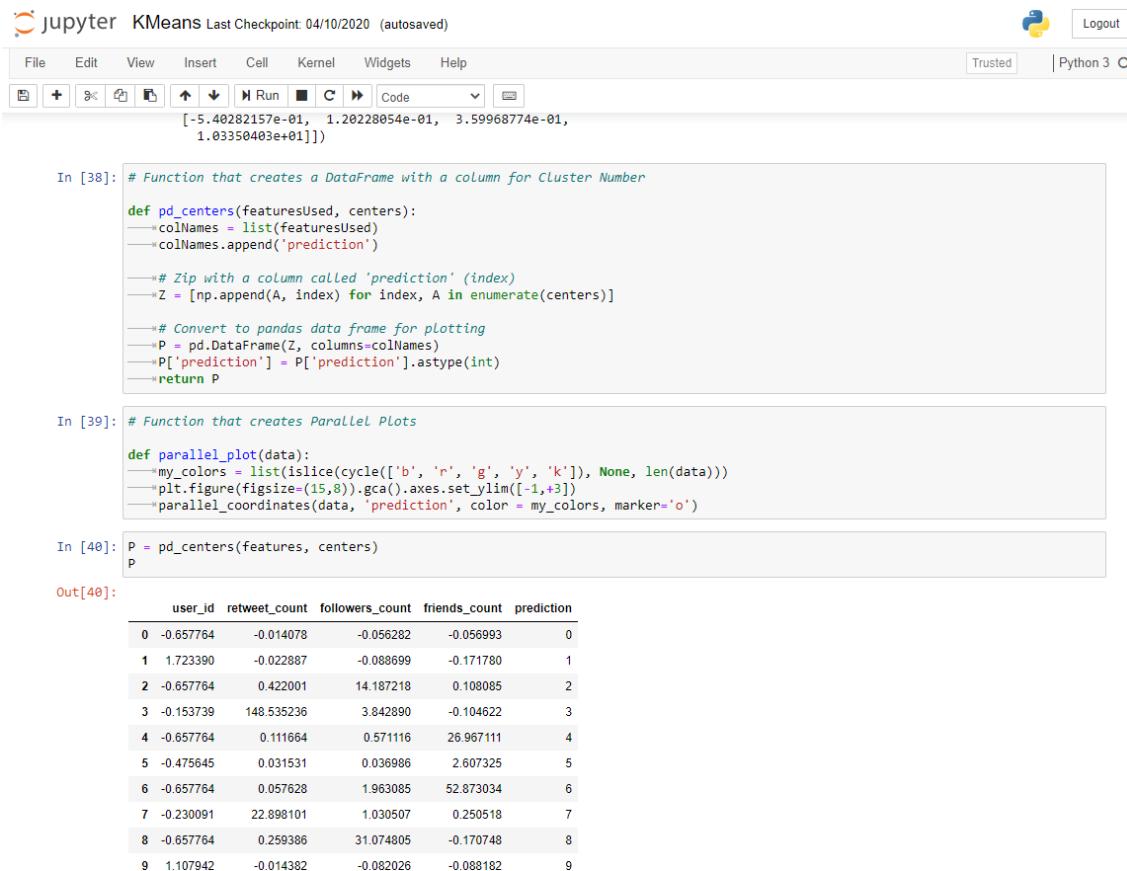
Anaconda es una Suite de código abierto que abarca una serie de aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos con Python. En líneas generales Anaconda Distribution es una distribución de Python que funciona como un

gestor de entorno, un gestor de paquetes y que posee una colección de más de 720 paquetes de código abierto.

Anaconda Distribution se agrupa en 4 sectores o soluciones tecnológicas, Anaconda Navigator, Anaconda Project, Las librerías de Ciencia de datos y Conda.

Durante el proyecto vamos a utilizar Jupyter Notebook y Spyder.

### 5.7.1. Jupyter Notebook



The screenshot shows the Jupyter Notebook interface. At the top, there's a toolbar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Python 3 logo. On the right, there are Logout and Trusted buttons. Below the toolbar, there are several icons for file operations like Open, Save, and Run. The main area contains three code cells:

- In [38]:** # Function that creates a DataFrame with a column for Cluster Number
- In [39]:** # Function that creates Parallel Plots
- In [40]:** P = pd\_centers(features, centers)

Below the cells, the **Out[40]:** section displays a table of data:

	user_id	retweet_count	followers_count	friends_count	prediction
0	-0.657764	-0.014078	-0.056282	-0.056993	0
1	1.723390	-0.022887	-0.088699	-0.171780	1
2	-0.657764	0.422001	14.187218	0.108085	2
3	-0.153739	148.535236	3.842890	-0.104622	3
4	-0.657764	0.111664	0.571116	26.967111	4
5	-0.475645	0.031531	0.036986	2.607325	5
6	-0.657764	0.057628	1.963085	52.873034	6
7	-0.230091	22.898101	1.030507	0.250518	7
8	-0.657764	0.259386	31.074805	-0.170748	8
9	1.107942	-0.014382	-0.082026	-0.088182	9

Figura 31: Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Los usos incluyen: limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos, aprendizaje automático...

## 5.7.2. Spyder

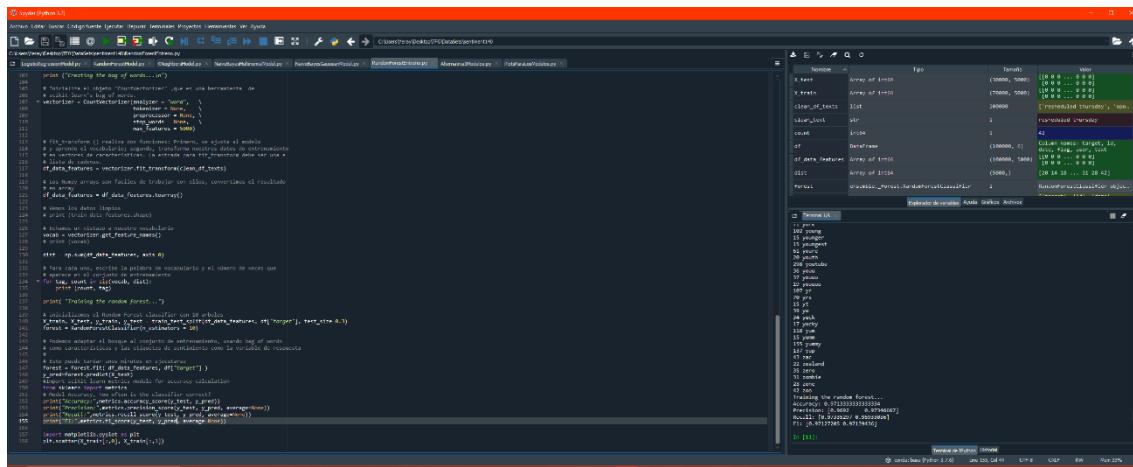


Figura 32: Spyder

Este es un potente entorno de desarrollo interactivo para el lenguaje Python. Tiene funciones avanzadas de edición, pruebas interactivas, depuración e introspección y un entorno informático numérico. Gracias al soporte de IPython (*intérprete interactivo mejorado de Python*) y bibliotecas populares de Python como NumPy, SciPy o matplotlib (*trazado interactivo 2D / 3D*).

Spyder es un entorno de desarrollo integrado y multiplataforma de código abierto (*IDE*) para programación científica en el lenguaje Python. Este IDE se liberó bajo la licencia de MIT.

Posé un visor de documentación. El programa nos va a poder mostrar documentación para cualquier llamada de clase o función realizada en el Editor o en una consola. Vamos a poder explorar las variables creadas durante la ejecución de un archivo. Va a ser posible editarlas con varios editores basados en GUI, como un diccionario y los de matriz Numpy.

## 5.8. Kaggle



Figura 33: Kaggle Platform

Kaggle es una plataforma gratuita que pose a disposición de los usuarios una serie de problemas para solucionar con temáticas como la ciencia de datos, el análisis predictivo y lo machine learning. Los usuarios de Kaggle son de todo el mundo, actualmente tiene más de un millón de miembros, conocidos como "Kagglers" donde más de 536.000 miembros son activos.

El creador de la competición es el encargado de preparar lo dataset y dar una descripción detallada de problema que se tiene que resolver. Una vez el problema se publica los participantes de la competición envían sus modelos que se compartirán a la plataforma, serán puntuados inmediatamente y se resumen a una tabla de mandos en directo. Cuando finaliza el plazo de una competición, el anfitrión tendrá licencia libre de derechos de autor de la entrada ganadora.

## 6. Análisis de sentimientos en Twitter

### 6.1. Dataset usado

Nuestro dataset ha sido descargado de la plataforma de Kaggle. Contiene 1.600.000 tweets extraídos usando el api de Twitter. Los tweets han sido anotados (0 negativo, 4 positivo) y se pueden utilizar para detectar el sentimiento.

### 6.2. El preprocessamiento aplicado

#### Bag of Words (BoW)

Un modelo de bolsa de palabras es una forma de extraer entidades del texto para su uso en el modelado, como con algoritmos de aprendizaje automático.

El enfoque es muy simple y flexible, y se puede utilizar en una gran variedad de maneras para extraer características de documentos.

Una bolsa de palabras es una representación de texto que describe la aparición de palabras dentro de un documento. Se trata de:

- Un vocabulario de palabras conocidas.
- Una medida de la presencia de palabras conocidas.

Se llama bolsa de palabras, porque cualquier información sobre el orden o estructura de palabras en el documento se descarta. El modelo sólo se refiere a si se producen palabras conocidas en el documento, no en el lugar del documento.

La intuición es que los documentos son similares si tienen contenido similar. Además, sólo desde el contenido podemos aprender algo sobre el significado del documento.

La bolsa de palabras puede ser tan simple o compleja como quieras. La complejidad viene tanto al decidir cómo diseñar el vocabulario de las palabras conocidas (o tokens) como en cómo puntuar la presencia de palabras conocidas.

#### TF-IDF

TF-IDF es una abreviatura de fórmula de frecuencia inversa de documento cuyo objetivo es definir la importancia de una palabra clave o de una frase en un documento o en una página web.

La frecuencia de un término (TF) señala la frecuencia con la que un término específico aparece en un documento. Se puede calcular dividiendo el número de veces que se utiliza la palabra clave en el documento por el número total de palabras de dicho documento.

La frecuencia inversa del documento (IDF) indica la frecuencia con la que se utiliza una palabra clave en un conjunto de documentos. Se trata de un parámetro correctivo cuyo objetivo es compensar la repetición de palabras como artículos o palabras con funciones similares y darles más prominencia a las palabras significativas.

El conjunto de estos dos parámetros es la fórmula TF-IDF que indica la relevancia de una palabra clave en un documento. Los motores de búsqueda utilizan mucho esta fórmula para determinar el corpus de páginas web relevantes para una consulta. Cuanto mayor sea el valor TF-IDF, más relevante (importante) es la palabra clave para el documento.

### El proceso del aprendizaje automático en el Machine Learning

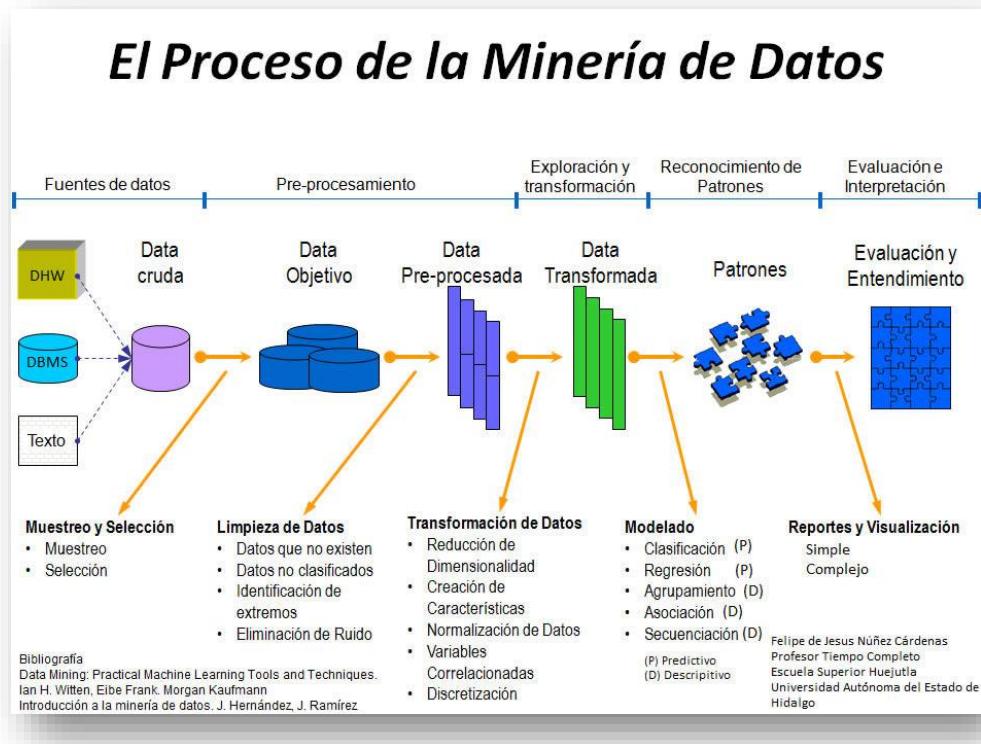


Figura 34: Fase del proceso de Machine Learning

### 6.2.1. Entender el Problema

Es muy importante entender el problema que tenemos que resolver. Entender el problema normalmente lleva bastante tiempo. En nuestro dataset se especifica que son tweets con una etiqueta con el sentimiento de negativo o positivo.

### 6.2.2. Entender los datos

Tan importante como entender el problema es entender los datos que tenemos disponibles. Es común hacer un análisis exploratorio de datos para familiarizarnos con

ellos. En el análisis exploratorio se suelen hacer gráficos, correlaciones y estadísticas descriptivas para comprender mejor qué historia nos están contando los datos.

### **6.2.3. Definir un Criterio de Evaluación.**

El criterio de evaluación se trata normalmente de una medida de error. Para problemas de clasificación con 2 clases podemos utilizar otras medidas tales como la precisión y la exhaustividad.

### **6.2.4. Evaluación de la solución actual**

En esta fase plantearemos la forma que van a tener nuestros datos para luego ser procesados por el modelo.

### **6.2.5. Preparar los datos**

La preparación de datos es una de las fases del machine learning que supone un mayor esfuerzo. Los principales desafíos a los que nos vamos a encontrar en esta fase son los siguientes:

#### **Datos Incompletos**

- Eliminándolos: una opción fácil es sólo quedarnos con los datos completos. Esto puede ser una opción si se trata de una minoría. Pero casi nunca es la mejor opción porque habría muchos datos que estaríamos obligados a desechar.
- Imputarlos con un valor razonable: cuando falte un valor, pondremos automáticamente un valor que tenga sentido.
- Imputarlos con un modelo de aprendizaje automático: podemos construir un modelo de machine learning que prediga cuál es el valor que nos falta aprendiendo de los casos en los que sí tenemos datos.

#### **Combinar datos de varias fuentes**

Tenemos que combinar los datos de forma que los algoritmos de machine learning puedan considerar toda la información.

#### **Dar el formato adecuado a los datos**

Si queremos usar librerías de machine learning que ya están disponibles, tenemos que darles formato a nuestros datos. En general, estas librerías esperan que los datos tengan forma de matriz o de tensor. Un tensor es una generalización de una matriz. Si la matriz tiene 2 dimensiones, el tensor tiene un número «n» de dimensiones.

#### **Calcular características relevantes (features)**

Los algoritmos de machine learning funcionan mucho mejor si le ofrecemos características relevantes en vez de los datos puros.

## 6.2.6. Construir el modelo

Durante esta fase, tenemos que elegir qué tipo de técnica de machine learning queremos usar. El algoritmo de machine learning aprenderá automáticamente a obtener los resultados adecuados con los datos históricos que hemos preparado.

## 6.2.7. Análisis de Errores

En la fase de análisis de errores intentaremos asegurarnos de que nuestro modelo es capaz de generalizar. La generalización es la capacidad que tienen los modelos de machine learning de producir buenos resultados cuando usan datos nuevos.

Si queremos obtener resultados realmente buenos, deberemos iterar sobre las fases anteriores varias veces. Con cada iteración, nuestro entendimiento del problema y de los datos será cada vez mayor. Esto hará que podamos diseñar mejores características relevantes y reducir el error de generalización. Un mayor entendimiento también nos ofrecerá la posibilidad de elegir con más criterio la técnica de machine learning que más se ajuste al problema. Casi siempre, tener más datos ayuda. En la práctica, más datos y un modelo simple tienden a funcionar mejor que un modelo complejo con menos datos.

## 6.2.8. Modelo integrado en un Sistema

Una vez que estemos satisfechos con el error, tenemos que compararlo con el error de la solución actual. Si es lo suficientemente mejor, integraremos el modelo del machine learning en nuestro sistema.

## 6.3. Funciones del modelo

### 6.3.1. load\_dataset

Para leer el dataset se ha creado la siguiente función:

➤ def load\_dataset (filename, cols):

La función recibe dos parámetros, que son “filename” y “cols”.

Cuando abrimos en Excel el CSV (explicar que es un CSV), nos encontramos que no existe en la primera columna identificadores de las mismas, para ellos utilizaremos “header\_names”, indicando el nombre de cada columna:

- target: sentimiento del tweet (0 negativo, 4 positivo).
- id: el identificador del tweet (ej: 2087).
- date: la fecha en la que se escribió el tweet, con el formato del siguiente ejemplo (sáb 16 de mayo 23:58:44 UTC 2009)

- flag: la consulta que se ha realizado. Si no hay ninguna consulta, toma de valor NO\_QUERY.
- user: el usuario que escribió el tweet (ej: yeraygranada94)
- text: el contenido del tweet (ej: Hoy presento mi TFG)

Cuando leemos este CSV en concreto, nos damos cuenta de que las primeras 798623 (hacer referencia figura) filas corresponden a tweets con un target igual a 0. Para ello pandas ofrece dos funcionalidades a la hora de leer un csv:

- skiprows: selecciona desde que fila empieza a leer datos, y los anteriores los descarta.
- nrows: selecciona el número de filas que quieras que lea del csv.

Con el ejemplo de la captura estamos seleccionando desde la fila 750000 y leeré las 200000 siguientes.

### **6.3.2. remove\_unwanted\_cols**

➤ def remove\_unwanted\_cols (df, cols):

Con esta función hacemos referencia a las columnas que hemos creado en la variable de "header\_names", y seleccionaremos aquellas que no nos sean de utilidad. En nuestro caso solo nos interesa el sentimiento del tweet y el contenido de este, por lo que las columnas 'id', 'date', 'flag' y 'user' no nos son de utilidad.

### **6.3.3. clean\_text**

Funcionará con una cadena de texto, y sus objetivos serán:

- cambios en minúsculas
- tokenizar (se descompone en palabras)
- elimina la puntuación y el texto que no es de la palabra
- encuentra Word Stems
- elimina las palabras de detención
- se une a las palabras significativas del Stem Words



Figura 35: Preprocesamiento del modelo

Tenemos que decidir cómo tratar con palabras que ocurren con frecuencia y que no tienen mucho significado. Estas palabras son las denominadas “stopwords”. Podemos seleccionar el idioma deseado, en este caso el inglés, que incluyen palabras como “a”, “y”, “is”, “the” ... Existen librerías de Python que vienen ya con una lista de palabras integrada. Importaremos la lista mediante “from nltk.corpus import stopwords”.

#### **6.3.4. create\_bag\_of\_words**

La manera de lograr que podamos interpretar el sentimiento de las palabras se llama vectorización del texto y consiste en convertir las oraciones en una matriz

Bag of Words es la palabra vector para cada revisión. Esto puede ser un simple recuento de palabras para cada revisión donde cada posición del vector representa una palabra (devuelta en la lista 'vocab') y el valor de esa posición representa el número de veces que la palabra se utiliza en los tweets.

La función siguiente también devuelve un tf-idf (frecuencia de documento inverso de frecuencia) que ajusta los valores de acuerdo con el número de revisiones fo que utilizan la palabra. Las palabras que se producen en muchas revisiones pueden ser menos discriminatorias que las palabras que ocurren más raramente. La transformación tf-idf reduce el valor de una palabra determinada en proporción al número de documentos en los que aparece.

La función devuelve lo siguiente:

- vectorizer: esto se puede aplicar a cualquier nueva revisión para convertir el tweet en el mismo vector de palabras que el conjunto de entrenamiento.
- Vocab: la lista de palabras a las que se refieren los vectores de palabras.
- train\_data\_features: vectores de recuento de palabras en bruto para cada revisión
- tfidf\_features: vectores de palabras transformadas tf-idf
- tfidf: la transformación tf-idf que se puede aplicar a las nuevas revisiones para convertir los recuentos de palabras sin procesar en los recuentos de palabras transformadas de la misma manera que los datos de entrenamiento.

Nuestro vectorizador tiene un argumento llamado 'ngram\_range'. Una simple bolsa de palabras divide las reseñas en palabras individuales. Si tenemos una ngram\_range de (1,2) significa que la revisión se divide en palabras individuales y también tuplas de palabras consecutivas. El argumento max\_features limita el tamaño de la palabra vector, en este caso a un máximo de 10.000 palabras (o 10.000 ngramas de palabras si un ngram puede ser más de una palabra).

## 6.4. Entrenamiento del modelo

### 6.4.1. ¿Qué es train\_test\_split?

Nos permite dividir un dataset en dos bloques, típicamente bloques destinados al entrenamiento y validación del modelo (llamemos a estos bloques "bloque de entrenamiento" y "bloque de pruebas" para mantener la coherencia con el nombre de la función). Es una operación que es común en todos los modelos de aprendizaje supervisado es la división de nuestro conjunto de datos en -al menos- dos partes: una parte *Train*, de entrenamiento, que corresponderá a la mayor parte de nuestro dataset y que usaremos para entrenar nuestro modelo y un parte *Test*, de menor tamaño, sobre la que evaluaremos nuestro modelo entrenado.

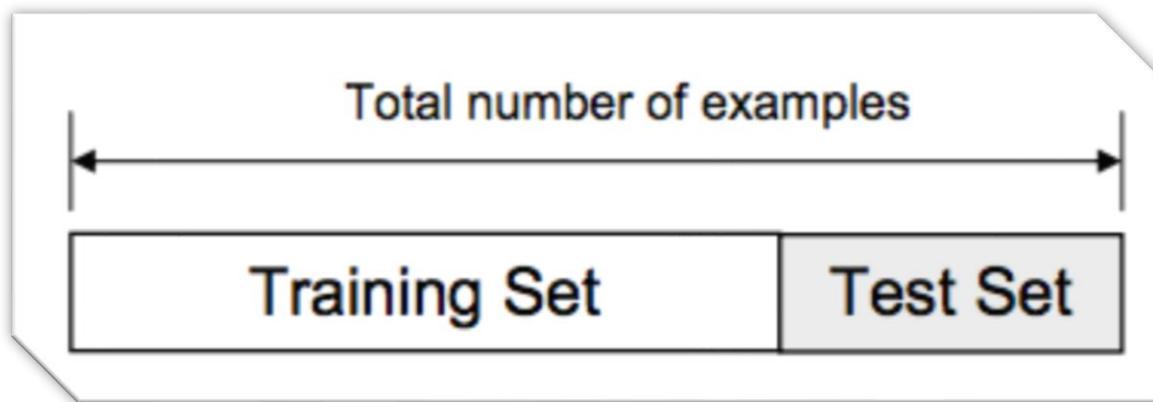


Figura 36: Tarining Set vs Test Set

Nuestro set de entrenamiento quedará con la siguiente estructura:

- X = list(df['cleaned\_text'])
- y = list(df['target'])
- X\_train, X\_test, y\_train, y\_test = train\_test\_split (X,y, test\_size = 0.3)

Dedicaremos un 30% al test, y un 70% al Training Set.

Finalmente aplicamos nuestro Bag of Words a nuestro set de entrenamiento.

- vectorizer, vocab, train\_data\_features, tfidf\_features, tfidf = \
- create\_bag\_of\_words(X\_train)

Ahora podemos crear un DataFrame con nuestras palabras y el número de veces que aparecen. Debido a que el número de palabras es muy grande seleccionaremos las 10 palabras que aparecen más veces.

- bag\_dictionary = pd.DataFrame()
- bag\_dictionary['ngram'] = vocab
- bag\_dictionary['count'] = train\_data\_features[0]
- bag\_dictionary['tfidf\_features'] = tfidf\_features[0]
- bag\_dictionary.sort\_values(by=['count'], ascending=False, inplace=True)
- print(bag\_dictionary.head(10))

## 7. Modelos utilizados

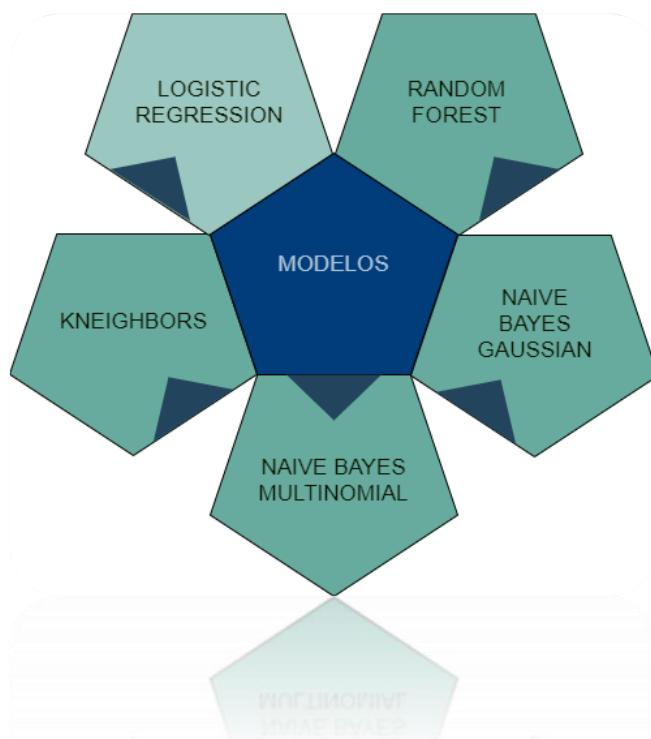


Figura 37: Modelos del proyecto

Para tener una visión distinta y abarcar distintas formas de entrenar nuestro modelo se han usado 5 técnicas distintas para posteriormente ver la diferencia de resultados y ver cuál es más conveniente en el caso de nuestro DataSet.

La estructura para todos es la misma, y aplican las funciones explicadas con anterioridad, la única diferencia entre ellos es a la hora de entrenar el modelo.

Ahora hemos transformado nuestras revisiones de texto libre en vectores de números (que representan palabras) podemos aplicar muchas técnicas de aprendizaje automático diferentes. Aquí usará una regresión logística relativamente simple.

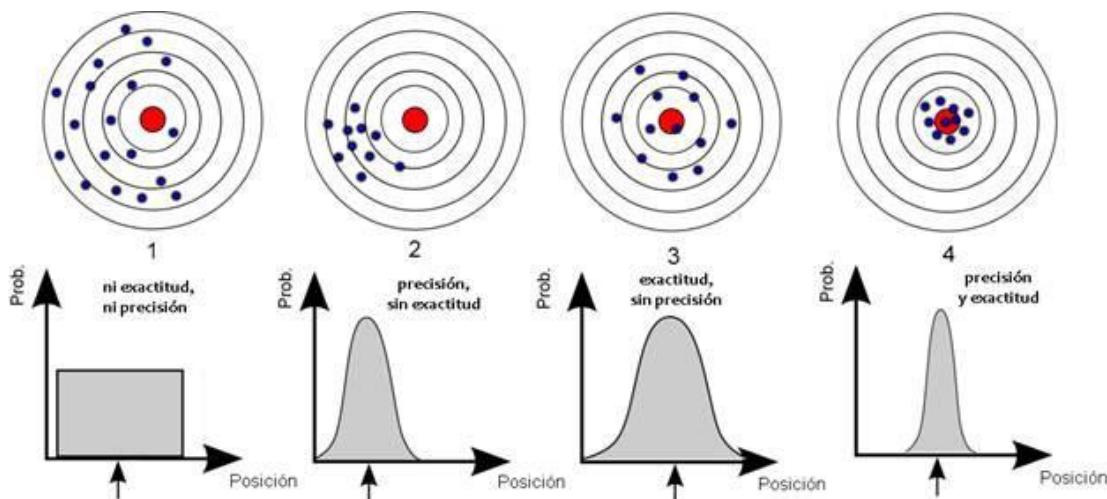
1. Usaremos los vectores de palabras transformadas del tipo tf-idf para entrenar el modelo. Tomaremos tanto los tweets como la etiqueta correspondiente al sentimiento (“target”).
2. Ahora aplicaremos el modelo de bag of words para probar los tweets y evaluar la precisión.
3. Primero aplicaremos nuestro vectorizador para crear un vector de palabras para los tweets en el conjunto de datos de prueba.
4. Como estamos usando la transformación tf-idf, aplicaremos el transformador tfidf para que los vectores de palabras se transformen de la misma manera que el conjunto de datos de entrenamiento.
5. Ahora vamos a predecir el sentimiento de todos los tweets de prueba y veremos si el sentimiento es positivo o negativo.

## 8. ANÁLISIS DE RESULTADOS

### 8.1 Conceptos sobre métricas

Antes de mostrar los resultados es importante tener en cuenta los siguientes aspectos:

- Accuracy: mide el porcentaje de casos que el modelo ha acertado. Se refiere a lo cerca que ha estado el resultado de una medición del valor verdadero. A nivel estadístico, el accuracy está relacionado con el sesgo de una estimación. Es también conocido como Verdadero Positivo o True Positive Rate. Resulta la proporción entre los positivos reales deducidos por el algoritmo y todos los casos positivos.
- Precision: mide la calidad del modelo de machine learning en tareas de clasificación. Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor sea la dispersión mayor será la precisión. Resulta la proporción entre el número de predicciones correctas, tanto positivas como negativas, y el total de predicciones.
- Recall: La métrica de exhaustividad nos va a informar sobre la cantidad que el modelo de machine learning es capaz de identificar. Es también conocida como la Tasa de Verdaderos Negativos o “True Negative Rate”. Resulta los casos negativos que el algoritmo ha clasificado correctamente. Muestra como de bien puede el modelo detectar esa clase.
- F1: El valor F1 se utiliza para combinar las medidas de precision y recall en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones. Es de gran utilidad cuando la distribución de las clases es desigual. Teniendo en cuenta estas métricas podemos obtener cuatro posibles casos para cada clase:
  - Alta precisión y alto recall: el modelo maneja perfectamente esa clase.
  - Alta precisión y bajo recall: el modelo no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
  - Baja precisión y alto recall: El modelo detecta bien la clase, pero también incluye muestras de la otra clase.
  - Baja precisión y bajo recall: El modelo no logra clasificar la clase correctamente.



Si medimos la efectividad de nuestro modelo por la cantidad de aciertos que ha obtenido y teniendo solamente en cuenta a la clase mayoritaria podemos estar teniendo una falsa sensación de que el modelo esta funcionando correctamente. Esto suele ocurrir en los problemas de clasificación donde en nuestro conjunto de entrenamiento contamos con que algunas de las clases sean minoritarias, provocando un desbalanceo en los datos.

Para entender como de bien esta funcionando nuestro modelo utilizaremos una matriz de confusión. La matriz de confusión de un problema de clase n es una *matriz nXn* en la que las filas se nombran según las clases reales y las columnas, según las clases previstas por el modelo. Sirve para mostrar de forma explícita cuándo una clase es confundida con otra. Por eso, permite trabajar de forma separada con distintos tipos de error.

VALORES PREDICCIÓN	
VALORES REALES	Verdaderos positivos
	Falsos Positivos
	Falsos Negativos
	Verdaderos Negativos

Figura 39: Matriz de confusión

De esta forma, la diagonal principal contiene la suma de todas las predicciones correctas. La otra diagonal refleja los errores del clasificador: los falsos positivos o “true positives”, o los falsos negativos o “false negatives”

Para entender de otra forma la matriz de confusión la estudiaremos con otras métricas de evaluación:

Matriz de confusión		Estimado por el modelo				
		Negativo ( <i>N</i> )	Positivo ( <i>P</i> )			
Real	Negativo	a: (TN)	b: (FP)	<b>Precisión ("precision")</b> Porcentaje predicciones positivas correctas:	$d/(b+d)$	
	Positivo	c: (FN)	d: (TP)			
		<b>Sensibilidad, exhaustividad ("Recall")</b> Porcentaje casos positivos detectados	<b>Especificidad (Specificity)</b> Porcentaje casos negativos detectados	<b>Exactitud ("accuracy")</b> Porcentaje de predicciones correctas <i>(No sirve en datasets poco equilibrados)</i>		
		$d/(d+c)$	$a/(a+b)$	$(a+d)/(a+b+c+d)$		

Figura 40: Matriz de confusión y otras métricas

- **a** es el número de predicciones correctas de clase negativa (negativos reales)
- **b** es el número de predicciones incorrectas de clase positiva (falsos positivos)
- **c** es el número de predicciones incorrectas de clase negativa (falsos negativos)
- **d** es el número de predicciones correctas de clase positiva (positivos reales).

Ahora detallamos las fórmulas de las métricas:

$$\begin{aligned}
 \text{Precision} &= \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\
 \text{Recall} &= \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\
 \text{Accuracy} &= \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}
 \end{aligned}$$

Figura 41: Fórmulas de las métricas

## 8.2 Resultados de los modelos

Se han hecho tres tipos de pruebas distintas con cada uno de los modelos. A continuación, se detallan los resultados y el análisis para cada una de las entradas.

(nrows=100000)								
	Classifier	Accuracy	Precision		Recall		F1	
			P	N	P	N	P	N
1	Logistic Regression	0.764	0.76781452	0.76035214	0.75393371	0.77397942	0.76081081	0.76710526
2	Random Forest	0.74136	0.7322298	0.75182419	0.77176916	0.7101273	0.75147974	0.73038112
3	KNeighbors	0.525423	0.54322	0.342139	0.535243	0.242545	0.54252	0.5132
4	Naive Bayes Multinomial	0.774133333	0.77947333	0.76892693	0.76683938	0.78147999	0.77310474	0.77515264
5	Naive Bayes Gaussian	0.703866666	0.76094101	0.6680768	0.58975734	0.81673518	0.66450151	0.7349642

(nrows=200000)								
	Classifier	Accuracy	Precision		Recall		F1	
			P	N	P	N	P	N
1	Logistic Regression	0.82965	0.69954338	0.86232927	0.56068672	0.91952771	0.62246519	0.89001044
2	Random Forest	0.807833333	0.65251559	0.84548485	0.50586054	0.90939665	0.56990451	0.87627693
3	KNeighbors	0.537645634	0.5256389	0.352839	0.51329487	0.37129884	0.55578	0.5449837
4	Naive Bayes Multinomial	0.82025	0.81511372	0.82090815	0.3683687	0.97195013	0.50742179	0.89006677
5	Naive Bayes Gaussian	0.5008	0.32211688	0.91355437	0.89591498	0.36845033	0.47386172	0.52511416

(nrows=300000)								
	Classifier	Accuracy	Precision		Recall		F1	
			P	N	P	N	P	N
1	Logistic Regression	0.871822222	0.68163567	0.89417155	0.43081341	0.95984059	0.52794828	0.92584307
2	Random Forest	0.856088888	0.61944751	0.88254478	0.37091438	0.9540101	0.46399603	0.91688698
3	KNeighbors	0.55809234	0.57983298	0.4673899	0.5143798	0.45329804	0.65938	0.61387924
4	Naive Bayes Multinomial	0.8621111112	0.83133733	0.86354545	0.22116023	0.99097873	0.34937611	0.92288381
5	Naive Bayes Gaussian	0.436366666	0.21365227	0.94474231	0.89822652	0.34483585	0.34519614	0.50525207

Figura 42: Resultados para n entradas

En esta tabla se han añadidos los 5 modelos que se han utilizado, Logistic Regression, Random Forest, Kneighbors, Naive Bayes Multinomial y Naive Bayes Gaussian. Para cada uno de ellos se detallan el Accuracy, Precision, Recall y F1-Score. Para estas tres últimas métricas se distinguen los resultados entre P (Positivos) y N (Negativos). La fila con mayor porcentaje se ha señalado con un recuadro y el resultado sombreado. Vamos a estudiar a fondo para los 3 nrows analizados.

(nrows=100000)								
	Classifier	Accuracy	Precision		Recall		F1	
			P	N	P	N	P	N
1	Logistic Regression	0.764	0.76781452	0.76035214	0.75393371	0.77397942	0.76081081	0.76710526
2	Random Forest	0.74136	0.7322298	0.75182419	0.77176916	0.7101273	0.75147974	0.73038112
3	KNeighbors	0.525423	0.54322	0.342139	0.535243	0.242545	0.54252	0.5132
4	Naive Bayes Multinomial	0.774133333	0.77947333	0.76892693	0.76683938	0.78147999	0.77310474	0.77515264
5	Naive Bayes Gaussian	0.703866666	0.76094101	0.6680768	0.58975734	0.81673518	0.66450151	0.7349642

Figura 43: Resultados para 100000 datos de entrada

Para nrows=100000 se han seleccionado prácticamente el mismo número de Negativos que Positivos para tener un análisis no desbalanceado y ver la diferencia con un modelo desbalanceado. Para asegurarnos del número de negativos y positivos que tenemos basta con hacer un ‘count’ de la columna “target” y hacer un plot. Para mostrar el gráfico se ha usado factorplot de la librería de seaborn.

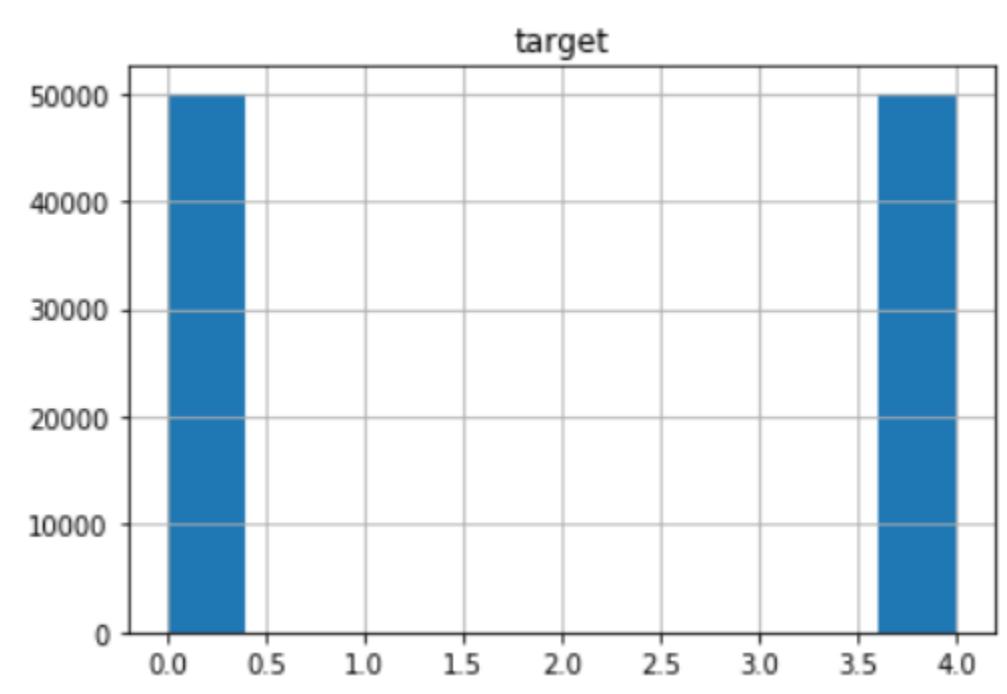


Figura 44: Plot balanced target

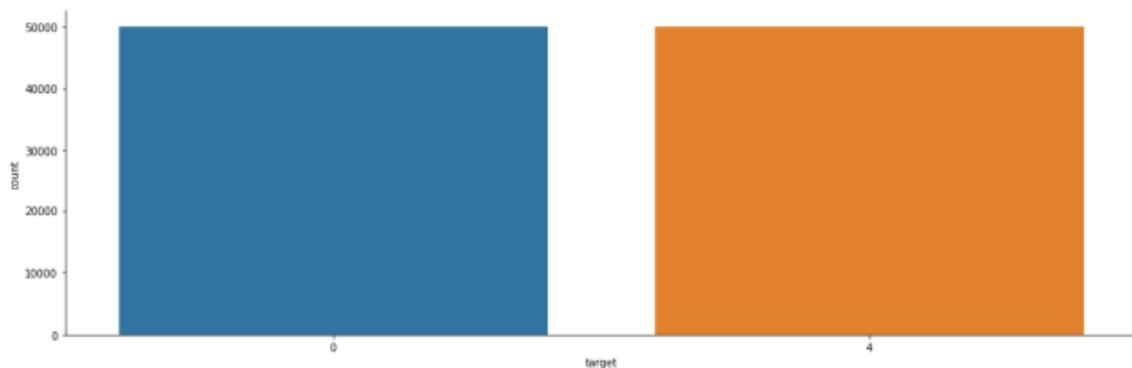


Figura 45: Factorplot balanced target

Vemos en el gráfico que hay la misma cantidad de negativos (color azul) que de positivos (color naranja).

El modelo peor parado es el de K Neighbors (el vecino más cercano). Esto resulta porque el ordenador donde se han realizado las pruebas ha tardado más de 8 horas en ejecutar. Para obtener unos resultados óptimos es necesario saber con exactitud qué valor de  $k$  es el apropiado. Además, este algoritmo no aprende explícitamente el modelo, elige memorizar las instancias de capacitación que se utilizan posteriormente como conocimiento para la fase de predicción. Almacena todos los datos de entrenamiento siendo computacionalmente muy costoso.



Figura 46: KNN y los problemas con nuestro modelo

El siguiente que ha obtenido peores resultados es “Naive Bayes Gaussian”. Veamos la diferencia con “Naive Bayes Multinomial” y el porqué de esa diferencia de resultados.

- Multinomial Naive Bayes: Una distribución multinomial es útil para modelar vectores de características donde cada valor representa, por ejemplo, el número de apariciones de un término o su frecuencia relativa. Su fórmula es:

$$P(X_1 = x_1 \cap X_2 = x_2 \cap \dots \cap X_k = x_k) = \frac{n!}{\prod_i x_i!} \prod_i p_i^{x_i}$$

Figura 47: Multinomial formule

- Gaussian Naive Bayes: Gaussian Naive Bayes es útil cuando se trabaja con valores continuos cuyas probabilidades se pueden modelar utilizando una distribución gaussiana.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Figura 48: Gaussian formule

Por esta razón deducimos que, al hacer un análisis de sentimientos de un modelo supervisado, donde se encuentran nuestros tweets etiquetados con 0 o 4, dependiendo de si es negativo o positivo, es preferible usar el Multinomial debido a que tiene en cuenta la frecuencia con la que aparece un 0 o un 4.

El siguiente en la lista es el algoritmo de Random Forest. Este algoritmo ha devuelto resultados bastante equilibrados. Random Forest es una técnica de aprendizaje automático supervisada basada en árboles de decisión. Su principal ventaja es que obtiene un mejor rendimiento de generalización para un rendimiento durante

entrenamiento similar. Esta mejora en la generalización la consigue compensando los errores de las predicciones de los distintos árboles de decisión. Para asegurarnos que los árboles sean distintos, lo que hacemos es que cada uno se entrena con una muestra aleatoria de los datos de entrenamiento. Esta estrategia se denomina bagging.

Dado que un random forest es un conjunto de árboles de decisión, y los árboles son modelos no-paramétricos, los random forests tienen las mismas ventajas y desventajas de los modelos no-paramétricos:

- ventaja: pueden aprender cualquier correspondencia entre datos de entrada y resultado a predecir
- desventaja: no son buenos extrapolando ... porque no siguen un modelo conocido

En nuestro modelo se le ha asignado a n (n estimators) el valor de 10. Si nos fijamos en el recall de la tabla, para este algoritmo es el que menos positivos pierde, sin embargo, es el segundo que más negativos pierde. El F1 no lo tenemos en cuenta porque este indicador es útil si se tiene una distribución de clases desigual.

El ya mencionado Naive Bayes Multinomial y Logistic Regression son los dos mejores modelos para un análisis de 100000 filas. La matriz de confusión para el Logistic Regression es la siguiente.

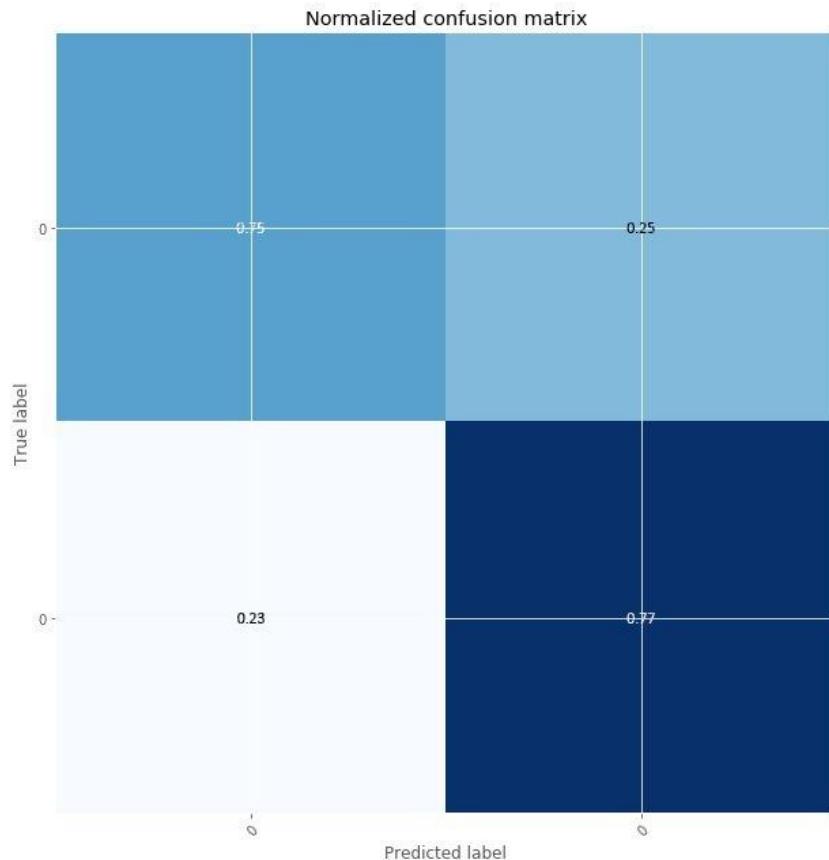


Figura 49: Confusion Matrix n=100000

Son números decentes para un primer análisis, pero no suficientes. Tenemos un 25% de falsos positivos y un 23% de falsos negativos. Por este motivo vamos a estudiar que ocurre si duplicamos el número de filas para el modelo.

(nrows=200000)								
	Classifier	Accuracy	Precision		Recall		F1	
			P	N	P	N	P	N
1	Logistic Regression	0.82965	0.69954338	0.86232927	0.56068672	0.91952771	0.62246519	0.89001044
2	Random Forest	0.807833333	0.65251559	0.84548485	0.50586054	0.90939665	0.56990451	0.87627693
3	KNeighbors	0.537645634	0.5256389	0.352839	0.51329487	0.37129884	0.55578	0.5449837
4	Naive Bayes Multinomial	0.82025	0.81511372	0.82090815	0.3683687	0.97195013	0.50742179	0.89006677
5	Naive Bayes Gaussian	0.5008	0.32211688	0.91355437	0.89591498	0.36845033	0.47386172	0.52511416

Figura 50: Resultados para 200000 datos de entrada

Ahora nuestro número de filas son 200000, el doble de las que teníamos anteriormente. A continuación, se muestra gráfica la desproporción entre los positivos y los negativos.

```
print(df.groupby('target').size())
target
0    50001
4   149999
dtype: int64
```

```
df.hist()
plt.show()
```

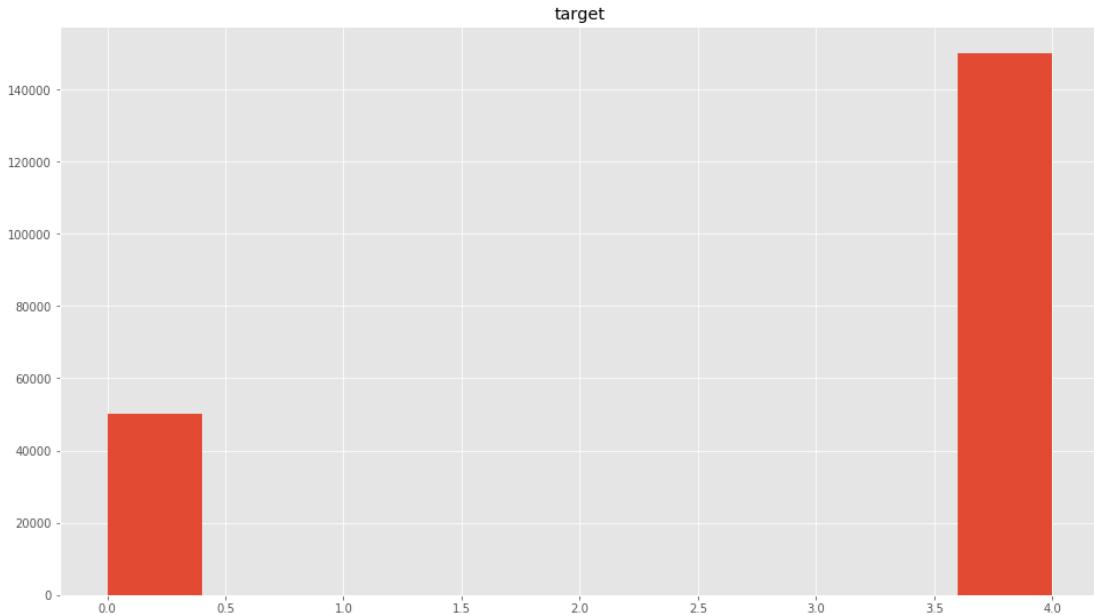


Figura 51: Plot Unbalanced target

Tenemos un total de 50001 filas de tweets negativos, y 149999 de tweets positivos.

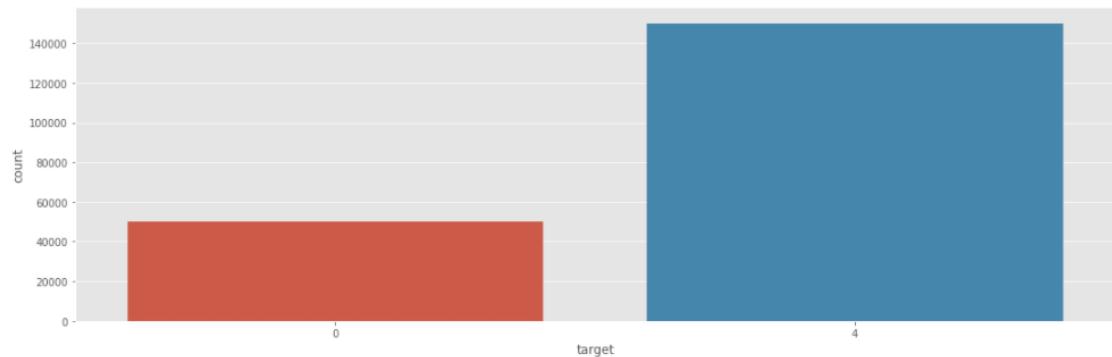


Figura 52: Factorplot unbalanced target

En el gráfico se aprecia claramente lo desbalanceado que está el modelo, y analizaremos en la tabla si se mantienen resultados parecidos que con un modelo balanceado.

A primera vista se van a descartar dos modelos que siguen el mismo patrón. Cuanto mayor número de filas y desbalanceo tenga entre positivos y negativos, peor será el modelo. Esto ocurre para Kneighbors y Naive Bayes Gaussian, que ha perdido un 20% en el accuracy respecto al anterior análisis.

En los otros tres modelos ocurre algo similar entre ellos. La precisión y el recall son bajos para los positivos, esto no significa que no logra clasificar el modelo correctamente. Sin embargo para los negativos ocurre el caso contrario, la precision y el recall son muy altos. Quiere decir que maneja perfectamente los casos negativos. Para confirmar este desbalanceo nos fijaremos en la matriz de confusión:

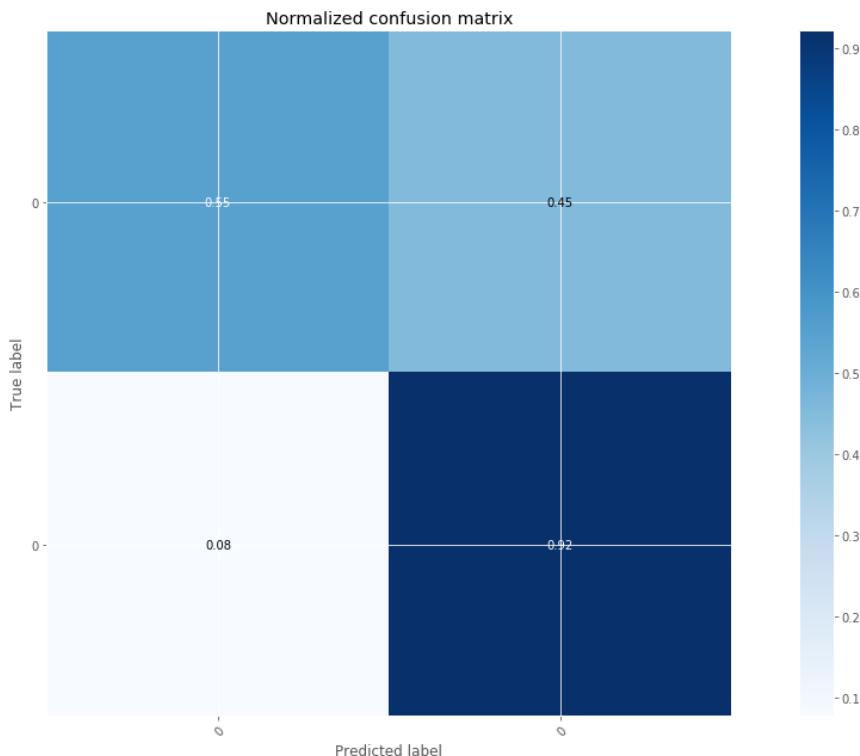


Figura 53: Confusion Matrix  $n=200000$

La matriz de confusión se asemeja con los resultados de la tabla de la columna de F1. En la diagonal principal nos encontramos que detecta un 55% de verdaderos positivos y un 92% de verdaderos negativos.

Con estos resultados podríamos deducir que manteniendo el mismo número de filas para casos negativos y aumentando exponencialmente el número de filas para casos positivos, el porcentaje de verdaderos negativos irá aumentando a la vez que el porcentaje de verdaderos positivos disminuirá considerablemente.

Para confirmar nuestra hipótesis vamos a añadir 100000 filas más a nuestro modelo.

(nrows=300000)									
	Classifier	Accuracy	Precision		Recall		F1		
			P	N	P	N	P	N	
Logistic Regression	0.871822222	0.68163567	0.89417155	0.43081341	0.95984059	0.52794828	0.92584307		
Random Forest	0.856088888	0.61944751	0.88254478	0.37091438	0.9540101	0.4639603	0.91688698		
KNeighbors	0.55809234	0.57983298	0.4673899	0.5143798	0.45329804	0.65938	0.61387924		
Naive Bayes Multinomial	0.862111112	0.83133733	0.86354545	0.22116023	0.99097873	0.34937611	0.92288381		
Naive Bayes Gaussian	0.436366666	0.21365227	0.94474231	0.89822652	0.34483585	0.34519614	0.50525207		

Figura 54: Resultados para 300000 datos de entrada

Vamos a analizar nuestros modelos con 300000 filas y un desbalance aun mayor entre tweets positivos y negativos.

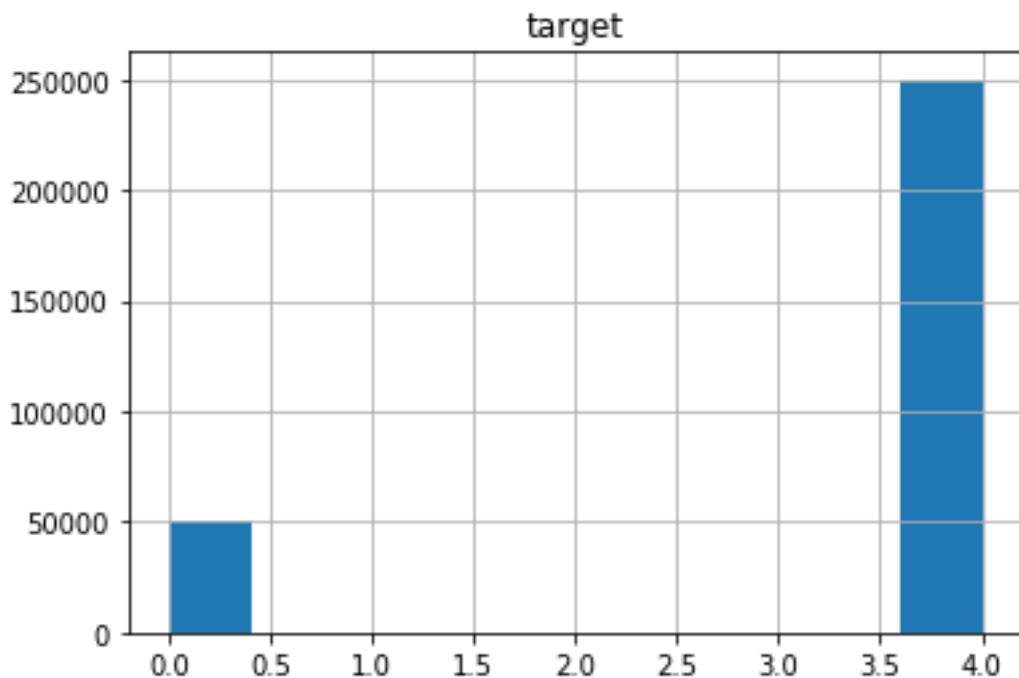


Figura 55: Plot very unbalanced target

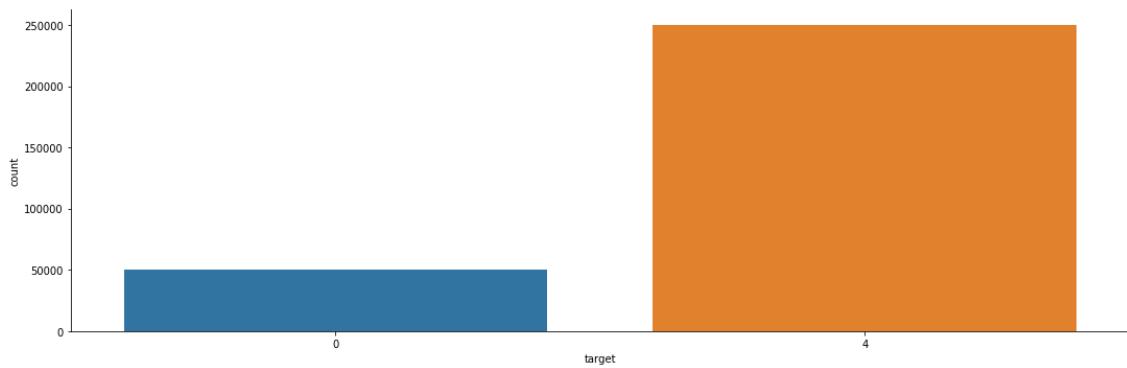


Figura 56: Factorplot very unbalanced target

Ahora es todavía más desproporcionada la diferencia entre tweets positivos y negativos.

Como en el anterior caso el accuracy ha seguido disminuyendo para el algoritmo de Naive Bayes Gaussian. Sin embargo, es el que mejor precisión tiene para los casos negativos. Si prestamos atención al recall de los casos negativos, es muy bajo. Podemos descartar este modelo.

Kneighbors no lo tendremos en cuenta debido a que sigue teniendo números muy mediocres.

Para los casos negativos los otros 3 modelos siguen siendo muy buenos en cuanto a precisión y recall. Nos encontramos con un 99% de recall para el modelo de Naive Bayes Multinomial. Por el contrario, el modelo que se ha mantenido más estable a lo largo de todas las pruebas, y que en estas tiene mejor accuracy y F1-Score es el algoritmo de Regresión logística. Su matriz de confusión es la siguiente:

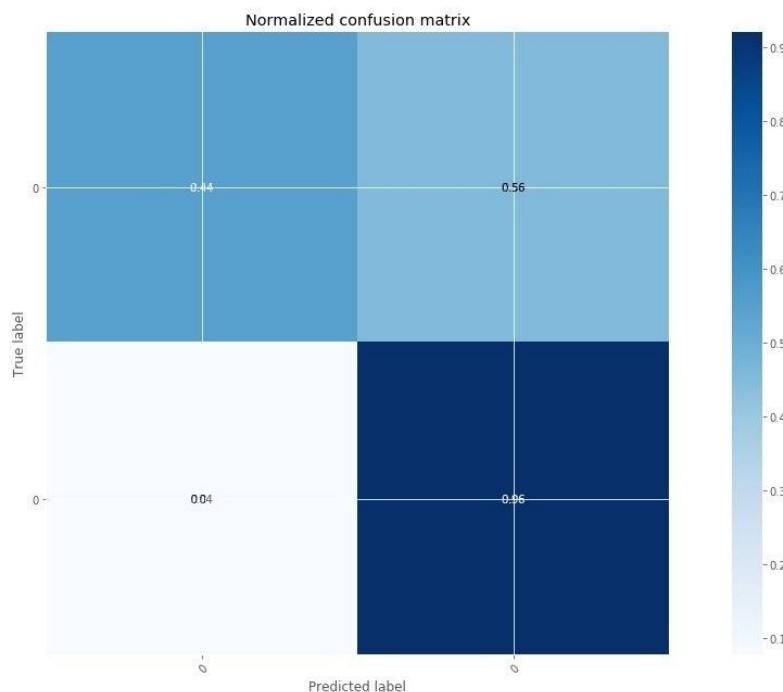


Figura 57: Confusion Matrix  $n=300000$

Como se pronosticaba, para los casos negativos el porcentaje es sumamente alto, un 96%, sin embargo, para los casos positivos sigue siendo muy poco fiable.

### 8.3. Resultados Adicionales

A continuación, se detallan los resultados de las variables de los modelos de forma adicional, para entender cada una de ellas.

Índice	target	text	cleaned_text
0	0	no, it was rescheduled for Thursday	wa reshedul thursday
1	0	@Opotopo small slip on Tryfan few weeks bac...	opotopo small slip tryfan week back felt si...
2	0	@Idristtwilight You can post HAN when you wa...	idristtwilight post han want great still work
3	0	@rose_7 Ohh poor jan please tell her that if she cans, send us an email!!	ohh poor jan pleas tell send us email
4	0	Finally home from work...It was a looong day!! And it's only Monday	final home work wa looong day onli monday
5	0	im very sad 4 chantelle and tom	im veri sad chantel tom
6	0	I chatted with someone on the online Apple ...	chat someon onlin appl store said would better buy new one wast
7	0	Back to office to empty aircon water tank ...	back offic empti aircon water tank empti offic give much time reflect
8	0	@ToxicMelvin Too late However it works now. Am really happy!	toxicmelvin late howev work realli happi
9	0	@exljbris it can't connect	exljbri ca connect
10	0	Missing my 20yr old baby-moved to WA.	miss old wa
11	0	@SaulaSmurf How old's ur bro?? mine was 15 ...	saulasmurf old ur bro mine wa happen look horribl accid
12	0	I miss caitlin already	miss caitlin alreadi
13	0	I'm bored at the dr's office for my mommy. And I miss my jeremy. He works far away now.	bore dr offic mommi miss jeremi work far away
14	0	nothing on fucking tv to watch. i hate not ...	noth fuck tv watch hate fuck ipod imac god damn fuck phone im fall fuck apart
15	0	I'm gonna have to give away my dog. &lt;3	gon na give away dog lt
16	0	I forgot to eat cookies at the barn today Travesty!	forgot eat cooki barn today travesti
17	0	can't walk anymore!	ca walk anymor
18	0	@vegancheze can i just say.... hot!? haha. ...	veganchez say hot haha wish wa sunni rain day
19	0	@itakepeektures at least for a week.i'm ver...	itakepeektur least veri sick block follow annoy much xx
20	0	At Foothills Brewing Co. Left my I.D in th...	foothill brew left hotel room oh well drink beer anyway
21	0	@jmroskell has she said some sensible thing...	jmroskel ha said sensibl thing like onli could understand word
22	0	I'd prefer a whole host of other symptoms b...	prefer whole host symptom besid constant nausea last week
23	0	omg just finished writing all of the names out My back is absolutely killing me	omg finish write name back absolut kill
24	0	had all my wage stole	wage stole
25	0	tyler rock but idk wat i should do	tyler rock idk wat
26	0	Just saw that video of Neda... Lord! Going to bed now with a dull feeling...	saw video neda lord go bed dull feel
27	0	its very warm tonight	veri warm tonight
28	0	@JeanGrae That's fucked up... I was just messing around. COLDBLOODED!	jeangra fuck wa mess around coldblood
29	0	@devinalexander it makes my tummy sad	devinalexand make tummi sad
30	0	@sims then dont look on twitter, i was spoiled 2...	sim dont look twitter wa spoil
31	0	@elyseholladay Since that's totally better	elyseholladay sinc total better layoff sorri

Figura 58: cleaned\_text

Este es el resultado de la variable cleaned\_text. En esta variable se añade una tercera columna a la derecha a las que ya teníamos. Esta columna contiene los mismos caracteres que la columna de su izquierda, "text", pero una vez ya pasado por el preprocesamiento. Se puede apreciar que desaparecen las minúsculas, los signos de puntuación, etc.

Parte de ese preprocesamiento son los stops, que como indicamos anteriormente son palabras vacías, que no tienen significado por sí solas, como lo pueden ser los artículos, pronombres, preposiciones...

Dentro de bag\_dictionary se almacena el número de veces que aparece una palabra de los tweets analizados. Esto es gracias al proceso de Bag of Words.

En la clase RandomForestEntreno.py podemos ejecutar el bag of words para que nos muestre todas las palabras que existen en el dataframe y el número de veces que aparecen.

bag_dictionary - DataFrame			
Índice	ngram	count	fifd_feature
9758	work thi	1	0.355161
9689	work	1	0.172795
6847	regular	1	0.371162
3816	hope	1	0.199676
6042	one time	1	0.400575
802	blue	1	0.318448
5386	monday	1	0.290985
6006	one	1	0.191885
6278	perfect	1	0.310486
7375	show	1	0.238246
8276	thi	1	0.168611
9447	weekend	1	0.257746
8513	time	1	0.187039

stops - Conjunto (179 elementos)

Tipo	Tamaño	
str	1	below
str	1	into
str	1	was
str	1	shan
str	1	that'll
str	1	re
str	1	each
str	1	then
str	1	between
str	1	he
str	1	yourself
str	1	yourselves
str	1	won't
str	1	they
str	1	hers
str	1	under
str	1	now
str	1	haven
str	1	during
str	1	mustn't
str	1	while
str	1	other
str	1	it's
str	1	shouldn
str	1	or
str	1	theirs
str	1	up
str	1	myself
str	1	all
str	1	does
str	1	off

Figura 59: stops

Índice	Tipo	Tamaño
0	str	1
1	str	1
2	str	1
3	str	1
4	str	1
5	str	1
6	str	1
7	str	1
8	str	1
9	str	1
10	str	1
11	str	1
12	str	1
13	str	1
14	str	1
15	str	1
16	str	1
17	str	1
18	str	1
19	str	1
20	str	1
21	str	1
22	str	1
23	str	1
24	str	1
25	str	1
26	str	1
27	str	1
28	str	1
29	str	1
30	str	1

Figura 60: X\_train

Índice	Tipo	Tamaño
0	str	1
1	str	1
2	str	1
3	str	1
4	str	1
5	str	1
6	str	1
7	str	1
8	str	1
9	str	1
10	str	1
11	str	1
12	str	1
13	str	1
14	str	1
15	str	1
16	str	1
17	str	1
18	str	1
19	str	1
20	str	1
21	str	1
22	str	1
23	str	1
24	str	1
25	str	1
26	str	1
27	str	1
28	str	1
29	str	1
30	str	1

Figura 61: X\_test

Figura 62: y\_test

Índice	Tipo	Tamaño	
0	int	1	4
1	int	1	4
2	int	1	0
3	int	1	0
4	int	1	0
5	int	1	0
6	int	1	0
7	int	1	0
8	int	1	0
9	int	1	4
10	int	1	0
11	int	1	4
12	int	1	4
13	int	1	4
14	int	1	0
15	int	1	4

Para nrows=100000 podemos ver que para y\_train se le pasan 70000 elementos, y para y\_test 30000 elementos. Esto se debe a que a la hora de entrenar el entrenamiento indicamos que el 70% fuera para el entreno y el 30% para el test.

Figura 63: y\_train

Índice	Tipo	Tamaño	
0	int	1	0
1	int	1	4
2	int	1	0
3	int	1	4
4	int	1	4
5	int	1	0
6	int	1	0
7	int	1	0
8	int	1	0
9	int	1	4
10	int	1	4
11	int	1	0
12	int	1	0
13	int	1	4
14	int	1	0
15	int	1	4

Figura 63: y\_train

	0
0	0
1	4
2	4
3	4
4	0
5	0
6	0
7	0
8	0
9	0
10	4
11	0
12	0
13	4
14	0
15	4
16	0
17	0
18	4
19	4
20	0
21	0
22	4
23	0
24	0
25	4
26	0
27	0
28	0
29	4

Nuestros modelos tienen dos tipos de sentimiento, negativo (se indica con un 0) y positivo (se indica con un 4). Cuando aplicamos el algoritmo de lenguaje supervisado a nuestro modelo, tomará como predicción los valores de la figura de la izquierda.

La figura de la derecha analiza si ha devuelto falsos positivos y negativos.

	0
0	True
1	True
2	False
3	True
4	False
5	True
6	True
7	True
8	True
9	False
10	True
11	True
12	True
13	True
14	True
15	True
16	True
17	False
18	True
19	False
20	True
21	True
22	False
23	True
24	False
25	True
26	True
27	True
28	False
29	True

Para terminar, la última imagen muestra el resultado final en la terminal de Spyder de un modelo, en este caso de NaiveBayesMultinomial.py.

```
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Yeray/Desktop/TFG/DataSets/sentiment140/NaiveBayesMultinomialModel.py', wdir='C:/Users/Yeray/Desktop/TFG/DataSets/sentiment140')
Creating bag of words...
    ngram count  tfidf_features
9758  work thi      1      0.355161
9689   work       1      0.172795
6847  regular      1      0.371162
3816    hope       1      0.199676
6042 one time      1      0.400575
802     blue       1      0.318448
5386  monday      1      0.290985
6006     one       1      0.191885
6278  perfect      1      0.310486
7375    show       1      0.238246
Training the Naive Bayes model...
Finished the Naive Bayes model
Accuracy = 77%
Accuracy: 0.7653666666666666
Precision: [0.77371913 0.757428 ]
Recall: [0.75196118 0.77884744]
F1: [0.76268501 0.7679884 ]
```

Figura 64: Resultados NaiveBayesMultinomial

La consola muestra el print del Bag of Words, el resultado de las 10 palabras que aparecen mas veces, los print cuando empieza y acaba de entrenar el modelo, y el resultado de las métricas. Para el caso de Precision, Recall y F1-Score son tuplas indicando el resultado para los verdaderos positivos y los verdaderos negativos, en el lado derecho e izquierdo respectivamente.

## 9. CONCLUSIONES

Como hemos podido comprobar, cuando hay el mismo número o similar de sentimientos positivos y negativos, los resultados de accuracy, precision, recall y f1-score permanecen bastante equilibrados entre ellos.

No obstante, a medida que aumenta la desproporción entre el número de sentimientos negativos y positivos, los resultados entre accuracy, precision, recall y f1-score empiezan a desbalancearse, donde el sentimiento minoritario, en este caso el negativo, cada vez posee un porcentaje más alto en dichas métricas, mientras que los sentimientos positivos descienden su porcentaje quedando muy lejos de los porcentajes de su sentimiento contrario.

En el gráfico se muestra el cambio de porcentaje del accuracy en los 5 modelos entre 100000 filas y 300000 filas de datos de entrada, siendo evidente el descenso para el modelo Gaussiano de Naive Bayes, un incremento prácticamente nulo para Kneighbors y un incremento entre el 9 y 11 % para los otros 3 modelos.

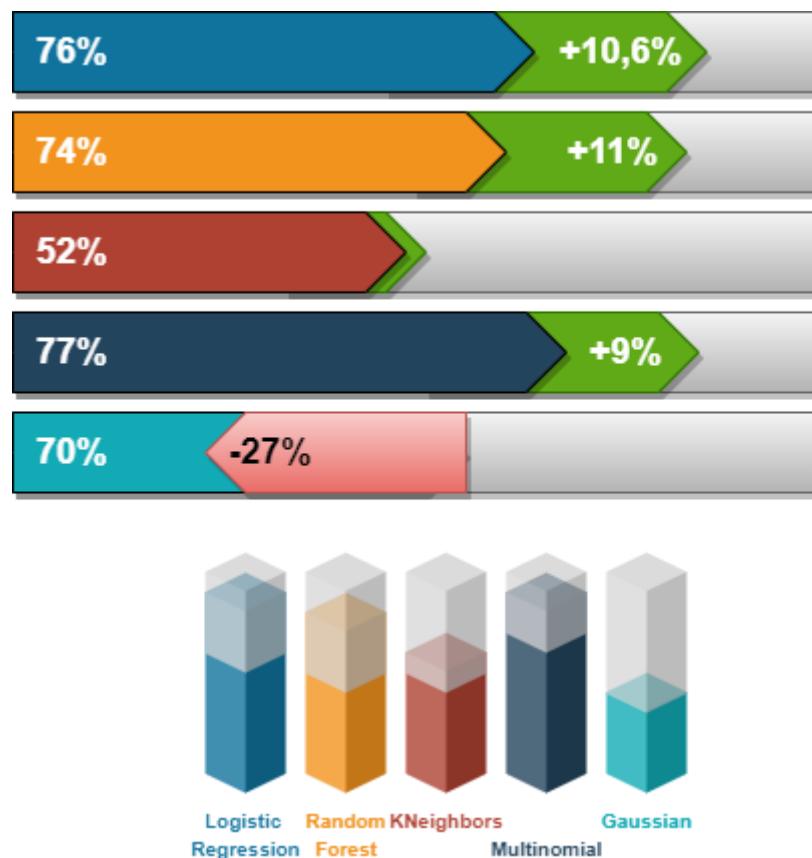


Figura 65: Porcentaje de evaluación triplicando el número de datos de entrada

Durante el desarrollo del proyecto solamente se tuvo en cuenta una métrica, el accuracy. A medida que he ido indagando más en varios libros y documentación en internet, descubrí que existen un montón de métricas, las relaciones entre ellas, y que el accuracy no es siempre una métrica fiable, pudiendo falsear los resultados sobre todo cuando los datos etiquetados con un sentimiento tienen un número desbalanceado entre ellos. Es por esto por lo que se recomienda tener en cuenta todas las posibilidades, distintos casos de prueba y analizar con varias métricas, siendo F1-Score y la matriz de confusión las que nos ayudarán más a entender los resultados.

He realizado dos tipos más de modelos que realizan un preprocesamiento distinto:

- RandomForestEntrenamiento.py: este modelo sigue un preprocesamiento parecido al desarrollado a nuestros 5 modelos anteriormente comentados. La diferencia está a la hora de hacer el test de entrenamiento. Para 100000 filas de datos de entrada tiene un accuracy del 97%, mientras que su hermano con los mismos valores nos da un 74%. Esto se debe a que en los datos de entrenamiento le estamos introduciendo todos los datos de entrada que ya conocía. Esto es un error porque es hacer el test sobre el propio entrenamiento no dejando hueco a la aleatoriedad, siendo un modelo inservible porque se basa en datos ya conocidos.
- Alternativa3Modelos.py: este modelo no sigue un preprocesamiento lógico. No se ha estudiado el CSV, ni analizado de qué forma hacer la limpieza de nuestros datos de entrada. Tampoco se ha usado la técnica de Bag of Words. La parte más importante de un buen modelo supervisado es el procesamiento, y en este modelo se está obviando.

Para llegar al final de mis resultados el proyecto ha experimentado las siguientes fases:

1. Periodo de investigación: esta fase ha sido la más larga y que ha permanecido hasta el final. Han sido semanas de estudio diario e investigación sobre Machine Learning, Python, Skicit-learn y algoritmos de aprendizaje supervisado, así como refinamiento del código y de los resultados.
2. Estructurar el proyecto: esta parte ha sido complicada porque es difícil partir de una base nula y saber cómo va a ser el proyecto y cómo queremos hacerlo. Según avanzaba en conocimientos teóricos, más claro podía ver como quería estructurar el proyecto
3. Análisis de resultados: cuando pensaba que había terminado el código no sabía explicar la última métrica que tenía por entonces, el accuracy. Gracias a las otras métricas y a la matriz de confusión he podido hacer los análisis mostrados en el proyecto.

Como comentario final, quisiera apuntar que ha sido un proyecto muy bonito, por todo el periodo de estudio y conocimiento que he experimentado, y he descubierto que el Machine Learning tiene unas capacidades infinitas y muy interesantes.

## 10. FUTURO DEL PROYECTO

Si quisieramos mejorar los resultados de los datos desbalanceados habría que aplicar técnicas para su manejo:

- Ajuste de parámetros de modelo: nuestra clase minoritaria, los sentimientos negativos deberían equilibrarse con los positivos para tener resultados más convincentes.
- Modificar el DataSet: podríamos eliminar muestras de la clase mayoritaria para reducirla y equilibrar la situación. O podríamos agregar nuevas filas con los mismos valores de las clases minoritarias. Modificar el DS es peligroso, porque podría suponer que el modelo cayera en overfitting (el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado).
- Balanced Ensemble Methods: Utiliza las ventajas de hacer ensamble de métodos, es decir, entrenar diversos modelos y entre todos obtener el resultado final, pero se asegura de tomar muestras de entrenamiento equilibradas.

Además de la mejora y tratamiento de nuestros datos, se podría incorporar una extensión de la librería pandas de Python, geopandas, que vaya seleccionando la ubicación de cada uno de los tweets y haga un mapa mundial de calor según el índice de tweets que se escriba por región.

Por último, que este proyecto sirva como base de estudio a cualquier persona interesada en el Machine Learning.

## 11. BIBLIOGRAFÍA

- [1] [https://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)
- [2] <https://www.anaconda.com/>
- [3] <https://www.twilio.com/blog/2017/12/sentiment-analysis-scikit-learn.html>
- [4] <https://medium.com/focus-on-vanilla-javascript/sentiment-analysis-using-scikit-learn-3ba5a63121dc>
- [5] <https://www.geeknetic.es/Noticia/11610/Como-abrir-correctamente-un-archivo-CSV-en-Excel.html>
- [6] <http://www.pitt.edu/~naraehan/presentation/Movie%20Reviews%20sentiment%20analysis%20with%20Scikit-Learn.html>
- [7] [https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)
- [8] <https://pypi.org/project/langdetect/>
- [9] [https://scikit-learn.org/stable/modules/feature\\_extraction.html#the-bag-of-words-representation](https://scikit-learn.org/stable/modules/feature_extraction.html#the-bag-of-words-representation)
- [10] <https://pythonhealthcare.org/2018/12/15/104-using-free-text-for-classification-bag-of-words/>
- [11] <https://www.linkedin.com/pulse/text-classification-using-bag-words-approach-nltk-scikit-rajendran/>
- [12] [https://scikit-learn.org/stable/modules/model\\_evaluation.html](https://scikit-learn.org/stable/modules/model_evaluation.html)
- [13] [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.GridSearchCV.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)
- [14] <https://stackoverflow.com/questions/29334463/how-can-i-partially-read-a-huge-csv-file>
- [15] <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>
- [16] <https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/>
- [17] <https://www.iartificial.net/clustering-agrupamiento-kmeans-ejemplos-en-python/>
- [18] <https://jarroba.com/k-means-python-scikit-learn-ejemplos/>
- [19] <https://towardsdatascience.com/clustering-with-k-means-1e07a8bfb7ca>

- [20] <https://www.pluralsight.com/guides/building-a-twitter-sentiment-analysis-in-python>
- [21] <https://rpubs.com/lstd/169935>
- [22] <https://www.linkedin.com/pulse/recomendador-de-an%C3%A9mico-utilizando-knn-supervisado-y-en/?originalSubdomain=es>
- [23] <https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-emplo-en-python/>
- [24] <https://eprints.ucm.es/49774/1/TFM%20Veronica%20Chamorro%20Alvarado.pdf>
- [25] <https://www.kaggle.com/prakharrathi25/weather-data-clustering-using-k-means>
- [26] <https://bookdown.org/dparedesi/data-science-con-r/aprendizaje-no-supervisado.html>
- [27] <https://www.dezyre.com/student-project/toly-novik-text-mining-and-clustering-of-tweets-based-on-context/2>
- [28] <https://www.kaggle.com/thebrownviking20/k-means-clustering-of-1-million-headlines>
- [29]  
[https://github.com/bigsnarfdude/word2vec\\_experiments\\_kaggle\\_popcorn/blob/master/BagOfWords.py](https://github.com/bigsnarfdude/word2vec_experiments_kaggle_popcorn/blob/master/BagOfWords.py)
- [30] <https://www.kaggle.com/rochachan/part-1-for-beginners-bag-of-words>
- [31] <https://www.datacamp.com/community/tutorials/random-forests-classifier-python>
- [32] <https://www.kaggle.com/kazanova/sentiment140>
- [33] <https://www.brandwatch.com/es/blog/analisis-de-sentimiento/>
- [34] <https://manueldelgado.com/que-es-el-analisis-del-sentimiento/>
- [35] <https://medium.com/@juanzambrano/aprendizaje-supervisado-o-no-supervisado-39ccf1fd6e7b>
- [36] [https://es.wikipedia.org/wiki/Aprendizaje\\_supervisado](https://es.wikipedia.org/wiki/Aprendizaje_supervisado)  
<https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/>

- [37] <http://openaccess.uoc.edu/webapps/o2/bitstream/10609/81435/6/jsobrinosTFM0618memoria.pdf>
- [38] <https://www.meaningcloud.com/es/productos/analisis-de-sentimiento>
- [39] <https://www.questionpro.com/blog/es/herramienta-de-analisis-de-sentimientos/>
- [40] <https://blog.karmapulse.com/analisis-de-sentimiento-en-redes-sociales/>
- [41] [https://mott.marketing/origen-historia-e-informacion-completa-sobre-la-red-social-twitter/#:~:text=Los%20creadores%20del%20microblogging%20se%C3%B1alan,%E2%80%9D%20\(ajustando%20mi%20twtrr\).](https://mott.marketing/origen-historia-e-informacion-completa-sobre-la-red-social-twitter/#:~:text=Los%20creadores%20del%20microblogging%20se%C3%B1alan,%E2%80%9D%20(ajustando%20mi%20twtrr).)
- [42] <https://cleverdata.io/que-es-machine-learning-big-data/>
- [43] <https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/>
- [44] <https://www.apd.es/que-es-machine-learning/>
- [45] <https://www.senpaiacademy.com/blog/noticias/detalle/machine-learning-analizar-textos>
- [46] <https://sitiobigdata.com/2019/12/24/clasificacion-de-aprendizaje-automatico-supervisado/#>
- [47] <https://aprendeia.com/libreria-scikit-learn-de-python/>
- [48] <https://ubunlog.com/spyder-entorno-desarrollo-python/>
- [49] <https://www.iartificial.net/fases-del-proceso-de-machine-learning/>
- [50] <https://medium.com/escueladeinteligenciaartificial/extracci%C3%B3n-de-features-con-bag-of-words-bow-y-tf-idf-nlp-f89d678abc0e>
- [51] <http://exponentis.es/como-dividir-un-conjunto-de-entrenamiento-en-dos-partes-train-test-split>
- [52] <https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/>
- [53] <https://www.iartificial.net/error-cuadratico-medio-para-regresion/>
- [54] <https://towardsdatascience.com/complete-guide-to-pythons-cross-validation-with-examples-a9676b5cac12>
- [55] <https://empresas.blogthinkbig.com/ml-a-tu-alcance-matriz-confusion/>

[56] <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>

[57] <https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/>

[58] <https://www.hackdeploy.com/python-logistic-regression-with-scikit-learn/>

[59] <https://www.kaggle.com/grfiv4/plot-a-confusion-matrix>

[60] <https://www.machinecurve.com/index.php/2020/05/05/how-to-create-a-confusion-matrix-with-scikit-learn/>

[61] <https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos>

[62] <https://www.iartificial.net/random-forest-bosque-aleatorio/>

[63] McKinney, W. (2012). Python for Data Analysis. California: O'Reilly

[64] Carlos Moreno Merchant, Diseño y Configuración de una Red de Sensores en el Contexto de Smart Cities, 2015

[65] Pablo Arias Oñate, Planificación, despliegue y supervisión de las redes de sensores de monitorización ambiental y de análisis de flujos de personas para la plataforma Smart City del CEI de Moncloa, 2016

[66] Carlos Alonso Aguilar, Design and development of a personality prediction system based on mobile-phone based metrics, 2017

[67] Marta Martín Sánchez, Especificación y desarrollo de un cuadro de mandos para la visualización de datos y monitorización de la plataforma SmartCity del CEI Moncloa, 2015

[68] Documentación de la asignatura SIBA (Sistemas Inteligentes Basados en Aprendizaje Automático)

[69] Domingos, P. (2015). The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Londres: Penguin Books

[70] H. Witten, I., Frank (2005). Data Mining: Practical Machine Learning Tools and Techniques. California: Elsevier

## 12. ANEXO

Se ha decidido incorporar capturas del código para poder hacer referencia en el anexo y servir como guía para la explicación de los modelos y como ayuda a futuros alumnos.

### LogisticRegressionModel.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed May 27 17:33:55 2020
4
5  @author: Yeray
6  """
7
8  import nltk
9  import pandas as pd
10 import numpy as np
11 from nltk.stem import PorterStemmer
12 from nltk.corpus import stopwords
13 from sklearn.model_selection import train_test_split
14 from sklearn.feature_extraction.text import CountVectorizer
15 from sklearn.feature_extraction.text import TfidfTransformer
16
17 def load_dataset(filename, cols):
18     header_names=['target', 'id', 'date', 'flag', 'user', 'text']
19     df = pd.read_csv(filename, encoding='latin-1', sep=',', engine='python', skiprows=750000, nrows=100000, names=header_names)
20     df.columns = cols
21     return df
22
23 def remove_unwanted_cols(df, cols):
24     for col in cols:
25         del df[col]
26     return df
27
28 df = load_dataset('training.1600000.processed.noemoticon.csv', ['target', 'id', 'date', 'flag', 'user', 'text'])
29 df = remove_unwanted_cols(df, ['id', 'date', 'flag', 'user'])
30
31 stemming = PorterStemmer()
32 stops = set(stopwords.words("english"))
33
34 def apply_cleaning_function_to_list(X):
35     cleaned_X = []
36     for element in X:
37         cleaned_X.append(clean_text(element))
38     return cleaned_X
39
40
41 def clean_text(raw_text):
42     """Esta función funciona en una cadena de texto sin formato y:
43         1) cambios en minúsculas
44         2) tokeniza (se descomponen en palabras)
45         3) elimina la puntuación y el texto que no es de la palabra
46         4) encuentra tallos de palabras
47         5) elimina las palabras de detención
48         6) se une a las palabras significativas del tallo"""
49
50
51     # Transforma a minusculas
52     text = raw_text.lower()
53
54     # Tokenizamos
55     tokens = nltk.word_tokenize(text)
56
57     # Mantiene solo palabras (elimina signos de puntuación y números)
58     # Usar .isalnum para mantener los números
59     token_words = [w for w in tokens if w.isalpha()]
60
61     # Stemming
62     stemmed_words = [stemming.stem(w) for w in token_words]
```

```

63     # Eliminamos palabras vacías
64     meaningful_words = [w for w in stemmed_words if not w in stops]
65
66     # Volvemos a unir las palabras en una cadena separada por espacio
67     joined_words = (" ".join(meaningful_words))
68
69     # Devuelve el resultado
70     return joined_words
71
72
73     # Devuelve el texto Limpio
74     text_to_clean = list(df['text'])
75
76     # Clean text
77     cleaned_text = apply_cleaning_function_to_list(text_to_clean)
78
79     # Añade una columna al DF con el texto Limpio
80     df['cleaned_text'] = cleaned_text
81
82     del cleaned_text
83
84     X = list(df['cleaned_text'])
85     y = list(df['target'])
86     X_train, X_test, y_train, y_test = train_test_split(
87         X, y, test_size = 0.3)
88
89     def create_bag_of_words():
90
91         print ('Creating bag of words...')
92         # Initialize the "CountVectorizer" object, which is scikit-Learn's
93         # bag of words tool.
94
95         # In this example features may be single words or two consecutive words
96         # (as shown by ngram_range = 1,2)
97         vectorizer = CountVectorizer(analyzer = "word", \
98                                     tokenizer = None, \
99                                     preprocessor = None, \
100                                    stop_words = None, \
101                                    ngram_range = (1,2), \
102                                    max_features = 10000
103                                    )
104
105        # fit_transform() does two functions: First, it fits the model
106        # and learns the vocabulary; second, it transforms our training data
107        # into feature vectors. The input to fit_transform should be a list of
108        # strings. The output is a sparse array
109        train_data_features = vectorizer.fit_transform(X)
110
111        # Convert to a NumPy array for easy of handling
112        train_data_features = train_data_features.toarray()
113
114        # tfidf transform
115        tfidf = TfidfTransformer()
116        tfidf_features = tfidf.fit_transform(train_data_features).toarray()
117
118        # Get words in the vocabulary
119        vocab = vectorizer.get_feature_names()
120
121        return vectorizer, vocab, train_data_features, tfidf_features, tfidf
122
123    vectorizer, vocab, train_data_features, tfidf_features, tfidf = \
124    create_bag_of_words(X_train)

```

```

125
126    bag_dictionary = pd.DataFrame()
127    bag_dictionary['ngram'] = vocab
128    bag_dictionary['count'] = train_data_features[0]
129    bag_dictionary['tfidf_features'] = tfidf_features[0]
130
131    # Ordenar por count
132    bag_dictionary.sort_values(by=['count'], ascending=False, inplace=True)
133    # Mostrar top 10
134    print(bag_dictionary.head(10))
135
136    def train_logistic_regression(features, label):
137        print ("Training the logistic regression model...")
138        from sklearn.linear_model import LogisticRegression
139        ml_model = LogisticRegression(C = 100,random_state = 0)
140        ml_model.fit(features, label)
141        print ('Finished the logistic regression model')
142        return ml_model
143
144    ml_model = train_logistic_regression(tfidf_features, y_train)
145
146    test_data_features = vectorizer.transform(X_test)
147    # Convertimos a numpy array
148    test_data_features = test_data_features.toarray()
149
150    test_data_tfidf_features = tfidf.fit_transform(test_data_features)
151    # Convertimos a numpy array
152    test_data_tfidf_features = test_data_tfidf_features.toarray()
153
154    predicted_y = ml_model.predict(test_data_tfidf_features)
155    correctly_identified_y = predicted_y == y_test
156    accuracy = np.mean(correctly_identified_y) * 100
157    #Para visualizar la exactitud redondeada con dos decimales
158    print ('Accuracy = %.0f%%' %accuracy)
159
160    #Obtenemos los valores exactos de Accuracy, Precision, Recall y F1
161    from sklearn import metrics
162    print("Accuracy:",metrics.accuracy_score(y_test, predicted_y))
163    print("Precision:",metrics.precision_score(y_test, predicted_y, average=None))
164    print("Recall:",metrics.recall_score(y_test, predicted_y, average=None))
165    print("F1:",metrics.f1_score(y_test, predicted_y, average=None))
166
167

```

## NaiveBayesGaussianModel.py

```
136 def train_naive_bayes(features, label):
137     print ("Training the Naive Bayes model...")
138     from sklearn.naive_bayes import GaussianNB
139     GNB_model = GaussianNB()
140     GNB_model.fit(features, label)
141     print ('Finished the Naive Bayes model')
142     return GNB_model
143
144 GNB_model = train_naive_bayes(tfidf_features, y_train)
145
146 test_data_features = vectorizer.transform(X_test)
# Convertimos a numpy array
147 test_data_features = test_data_features.toarray()
148
149 test_data_tfidf_features = tfidf.fit_transform(test_data_features)
# Convertimos a numpy array
150 test_data_tfidf_features = test_data_tfidf_features.toarray()
151
152 predicted_y = GNB_model.predict(test_data_tfidf_features)
correctly_identified_y = predicted_y == y_test
153 accuracy = np.mean(correctly_identified_y) * 100
print ('Accuracy = %.0f%%' %accuracy)
154
155 from sklearn import metrics
156 print("Accuracy:",metrics.accuracy_score(y_test, predicted_y))
157 print("Precision:",metrics.precision_score(y_test, predicted_y, average=None))
158 print("Recall:",metrics.recall_score(y_test, predicted_y, average=None))
159 print("F1:",metrics.f1_score(y_test, predicted_y, average=None))
160
161
162
163
164
```

## NaiveBayesMultinomialModel.py

```
136 def train_naive_bayes(features, label):
137     print ("Training the Naive Bayes model...")
138     from sklearn.naive_bayes import MultinomialNB
139     NB_model = MultinomialNB()
140     NB_model.fit(features, label)
141     print ('Finished the Naive Bayes model')
142     return NB_model
143
144 NB_model = train_naive_bayes(tfidf_features, y_train)
145
146 test_data_features = vectorizer.transform(X_test)
# Convertimos a numpy array
147 test_data_features = test_data_features.toarray()
148
149 test_data_tfidf_features = tfidf.fit_transform(test_data_features)
# Convertimos a numpy array
150 test_data_tfidf_features = test_data_tfidf_features.toarray()
151
152 predicted_y = NB_model.predict(test_data_tfidf_features)
correctly_identified_y = predicted_y == y_test
153 accuracy = np.mean(correctly_identified_y) * 100
print ('Accuracy = %.0f%%' %accuracy)
154
155 from sklearn import metrics
156 print("Accuracy:",metrics.accuracy_score(y_test, predicted_y))
157 print("Precision:",metrics.precision_score(y_test, predicted_y, average=None))
158 print("Recall:",metrics.recall_score(y_test, predicted_y, average=None))
159 print("F1:",metrics.f1_score(y_test, predicted_y, average=None))
160
161
162
163
164
```

## KNeighborsModel.py

```
136 def train_k_neighbors_classifier(features, label):
137     print ("Training the kneighbors classifier model...")
138     from sklearn.neighbors import KNeighborsClassifier
139     knn = KNeighborsClassifier(n_neighbors = 3)
140     knn.fit(features, label)
141     print ('Finished k neighbors classifier model')
142     return knn
143
144 knn = train_k_neighbors_classifier(tfidf_features, y_train)
test_data_features = vectorizer.transform(X_test)
# Convertimos a numpy array
145 test_data_features = test_data_features.toarray()
146
147 test_data_tfidf_features = tfidf.fit_transform(test_data_features)
# Convertimos a numpy array
148 test_data_tfidf_features = test_data_tfidf_features.toarray()
149
150 predicted_y = knn.predict(test_data_tfidf_features)
correctly_identified_y = predicted_y == y_test
151 accuracy = np.mean(correctly_identified_y) * 100
print ('Accuracy = %.0f%%' %accuracy)
152
153 from sklearn import metrics
154 print("Accuracy:",metrics.accuracy_score(y_test, predicted_y))
155 print("Precision:",metrics.precision_score(y_test, predicted_y, average=None))
156 print("Recall:",metrics.recall_score(y_test, predicted_y, average=None))
157 print("F1:",metrics.f1_score(y_test, predicted_y, average=None))
158
159
160
161
162
163
```

## RandomForestModel.py

```
136     def train_random_forest_classifier(features, label):
137         print ("Training the random forest classifier model...")
138         from sklearn.ensemble import RandomForestClassifier
139         forest = RandomForestClassifier(n_estimators = 10)
140         forest.fit(features, label)
141         print ('Finished random forest classifier model')
142         return forest
143
144
145     forest = train_random_forest_classifier(tfidf_features, y_train)
146     test_data_features = vectorizer.transform(X_test)
147     # Convertimos a numpy array
148     test_data_features = test_data_features.toarray()
149
150     test_data_tfidf_features = tfidf.fit_transform(test_data_features)
151     # Convertimos a numpy array
152     test_data_tfidf_features = test_data_tfidf_features.toarray()
153
154     predicted_y = forest.predict(test_data_tfidf_features)
155     correctly_identified_y = predicted_y == y_test
156     accuracy = np.mean(correctly_identified_y) * 100
157     print ('Accuracy = %.0f%%' %accuracy)
158
159     from sklearn import metrics
160     print("Accuracy:",metrics.accuracy_score(y_test, predicted_y))
161     print("Precision:",metrics.precision_score(y_test, predicted_y, average=None))
162     print("Recall:",metrics.recall_score(y_test, predicted_y, average=None))
163     print("F1:",metrics.f1_score(y_test, predicted_y, average=None))
164
165
166
```

## PlotsParaLosModelos.py

```
1  #-*- coding: utf-8 -*-
2  """
3  Created on Sun Nov 15 22:48:32 2020
4
5  @author: Veray
6  """
7
8  from sklearn.metrics import confusion_matrix
9  from sklearn.metrics import classification_report
10 from sklearn.metrics import plot_confusion_matrix
11 from sklearn.datasets import make_blobs
12 import matplotlib.pyplot as plt
13 import seaborn as sb
14
15 print(df.groupby('target').size())
16
17 df.hist()
18 plt.show()
19
20 sb.factorplot('target',data=df,kind="count", aspect=3)
21
22 print(confusion_matrix(y_test, predicted_y))
23 print(classification_report(y_test, predicted_y))
24
25 plot_confusion_matrix(y_test, predicted_y, df.target, normalize=True)
26
27 # Generate scatter plot for training data
28 plt.scatter(X_train[:,0], X_train[:,1])
29 plt.title('Linearly separable data')
30 plt.xlabel('X1')
31 plt.ylabel('X2')
32 plt.show()
```

## Alternativa3Modelos.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Sep 30 18:49:05 2020
4
5 @author: Yeray
6 """
7
8
9 import pandas as pd
10 import numpy as np
11 import re
12 import string
13 from nltk.corpus import stopwords
14 from nltk.tokenize import word_tokenize
15 from sklearn.feature_extraction.text import TfidfVectorizer
16 from sklearn.model_selection import train_test_split
17 from sklearn.ensemble import RandomForestClassifier
18 from nltk.stem import PorterStemmer
19 from nltk.stem import WordNetLemmatizer
20 # ML Libraries
21 from sklearn.metrics import accuracy_score
22 from sklearn.naive_bayes import MultinomialNB
23 from sklearn.linear_model import LogisticRegression
24 from sklearn.svm import SVC
25
26 # Global Parameters
27 stop_words = set(stopwords.words('english'))
28
29 def load_dataset(filename, cols):
30     header_names=['target', 'id', 'date', 'flag', 'user', 'text']
31     dataset = pd.read_csv(filename, encoding='latin-1', sep=',', engine='python', skiprows=750000, nrows=100000, names=header_names)
32     dataset.columns = cols
33     return dataset
34
35 def remove_unwanted_cols(dataset, cols):
36     for col in cols:
37         del dataset[col]
38     return dataset
39
40 def preprocess_tweet_text(tweet):
41     tweet.lower()
42     # Remove urls
43     tweet = re.sub(r"http\S+|www\S+|https\S+", '', tweet, flags=re.MULTILINE)
44     # Remove user @ references and '#' from tweet
45     tweet = re.sub(r'@\w+|\#', '', tweet)
46     # Remove punctuations
47     tweet = tweet.translate(str.maketrans('', '', string.punctuation))
48     # Remove stopwords
49     tweet_tokens = word_tokenize(tweet)
50     filtered_words = [w for w in tweet_tokens if not w in stop_words]
51
52     #ps = PorterStemmer()
53     #stemmed_words = [ps.stem(w) for w in filtered_words]
54     #lemmatizer = WordNetLemmatizer()
55     #lemma_words = [lemmatizer.lemmatize(w, pos='a') for w in stemmed_words]
56
57     return " ".join(filtered_words)
58
```

```

59  def get_feature_vector(train_fit):
60      vector = TfidfVectorizer(sublinear_tf=True)
61      vector.fit(train_fit)
62      return vector
63
64  def int_to_string(sentiment):
65      if sentiment == 0:
66          return "Negative"
67      elif sentiment == 2:
68          return "Neutral"
69      else:
70          return "Positive"
71
72  # Load dataset
73 dataset = load_dataset('training.1600000.processed.noemoticon.csv', ['target', 'id', 'date', 'flag', 'user', 'text'])
74 # Remove unwanted columns from dataset
75 dataset = remove_unwanted_cols(dataset, ['id', 'date', 'flag', 'user'])
76 #Preprocess data
77 dataset.text = dataset['text'].apply(preprocess_tweet_text)
78 # Split dataset into Train, Test
79
80 # Same tf vector will be used for Testing sentiments on unseen trending data
81 tf_vector = get_feature_vector(np.array(dataset.iloc[:, 1]).ravel())
82 X = tf_vector.transform(np.array(dataset.iloc[:, 1]).ravel())
83 y = np.array(dataset.iloc[:, 0]).ravel()
84 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=30)
85
86 # Training Naive Bayes model
87 NB_model = MultinomialNB()
88 NB_model.fit(X_train, y_train)
89 y_predict_nb = NB_model.predict(X_test)
90 print(accuracy_score(y_test, y_predict_nb))
91
92 # Training Logistics Regression model
93 LR_model = LogisticRegression(solver='lbfgs')
94 LR_model.fit(X_train, y_train)
95 y_predict_lr = LR_model.predict(X_test)
96 print(accuracy_score(y_test, y_predict_lr))
97
98 #Training Random Forest Model
99 RF_model = RandomForestClassifier(n_estimators = 15)
100 RF_model.fit(X_train, y_train)
101 y_predict_rf = RF_model.predict(X_test)
102 print(accuracy_score(y_test, y_predict_rf))
103
104 import matplotlib.pyplot as plt
105
106 plt.scatter(X[y == 0, 0], X[y == 0, 1],
107             c='blue', s=40, label='0')
108 plt.scatter(X[y == 1, 0], X[y == 1, 1],
109             c='red', s=40, label='1', marker='s')
110
111 plt.xlabel('primera característica')
112 plt.ylabel('segunda característica')
113 plt.legend(loc='upper right');

```

## RandomForestEntreno.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 10 18:34:20 2020
4
5 @author: Yeray
6 """
7
8 import pandas as pd
9 import numpy as np
10 from bs4 import BeautifulSoup
11 from nltk.corpus import stopwords
12 from sklearn.model_selection import train_test_split
13 from sklearn.feature_extraction.text import CountVectorizer
14 from sklearn.ensemble import RandomForestClassifier
15 import re
16
17 # Procedemos a Leer nuestro csv, que carece de una primera fila indicando el nombre de cada columna. Para ello utilizaremos "header"
18 # El csv contiene 1600000 filas. Al ser de un tamaño tan grande un ordenador de uso cotidiano no podría procesar toda esta información
19 # Iniciaremos el análisis desde la fila 750000 y leeremos las 100000 siguientes
20 header_names=['target', 'id', 'date', 'flag', 'user', 'text']
21 df = pd.read_csv('training.1600000.processed.noemoticon.csv',sep=',',engine='python', skiprows=750000, nrows=100000, names=header_na
22
23 # comprobamos que seguimos teniendo 1600000 filas y se han borrado las columnas, quedándonos solo 2
24 print("The shape of our data:",df.shape,"\n")
25
26 # Pintamos el nombre de las columnas
27 print("Our column names are:",df.columns.values)
28
29 # Inicializamos el objeto BeautifulSoup en el primer tuit del dataframe
30 example1 = BeautifulSoup(df["text"][0])
31
32 # Print the raw review and then the output of get_text(), for comparison
33 print (df["text"][0])
34 print (example1.get_text())
35
36 # Usamos expresiones regulares para encontrar y reemplazar
37 letters_only = re.sub("[^a-zA-Z]", " ", "# El patrón para buscar
38 " ", "# El patrón para reemplazarlo
39 example1.get_text() ) # El texto a buscar
40 # print (letters_only)
41
42 lower_case = letters_only.lower()      # Convertir a minúsculas
43 words = lower_case.split()           # Dividido en palabras
44
45 # Eliminar palabras vacías de "palabras"
46 words = [w for w in words if not w in stopwords.words("english")]
47 print (words)
48
49 def text_to_words( raw_text ):
50     # Function to convert a raw review to a string of words
51     # The input is a single string (a raw movie review), and
52     # the output is a single string (a preprocessed movie review)
53     # Función para convertir un tweet sin procesar en una cadena de palabras
54     # La entrada es una sola cadena (un tweet sin procesar), y
55     # la salida es una sola cadena (un tweet procesado)
56     #
57     # 1. Eliminamos HTML
58     #raw_text = BeautifulSoup(raw_text).get_text()
```

```

58     text_text = BeautifulSoup(raw_text).get_text()
59     #
60     # 2. Eliminar caracteres que no sean Letras
61     letters_only = re.sub("[^a-zA-Z]", " ", text_text)
62     #
63     # 3. Convertir a minúsculas, dividir en palabras individuales
64     words = letters_only.lower().split()
65     #
66     # 4. En Python, buscar un conjunto es mucho más rápido que buscar
67     # una lista, así que convertimos las palabras vacías en un conjunto
68     stops = set(stopwords.words("english"))
69     #
70     # 5. Eliminamos palabras vacías
71     meaningful_words = [w for w in words if not w in stops]
72     #
73     # 6. Vuelvemos a unir las palabras en una cadena separada por espacio,
74     # y devuelve el resultado.
75     return( " ".join( meaningful_words ) )
76
77 clean_text = text_to_words( df[ "text" ][ 0 ] )
78 # print (clean_text)
79
80 # Devuelve el numero de tuits basado en la columna text del dataframe
81 num_texts = df[ "text" ].size
82
83 # Inicializa una lista vacía de tuits
84 clean_df_texts = []
85
86 # Recorre cada tuit; crea un índice i que va de 0 a la longitud
87 # de la lista
88 for i in range( 0, num_texts ):
89     # Call our function for each one, and add the result to the list of
90     # clean reviews
91     # Llama a nuestra función para cada uno y agrega el resultado a la lista de
92     # tuits vacío
93     clean_df_texts.append( text_to_words( df[ "text" ][ i ] ) )
94
95 print ("Cleaning and parsing the training set tuits...\n")
96 clean_df_texts = []
97 for i in range( 0, num_texts ):
98     # Escribirá un mensaje por cada tuit Leido
99     if( (i+1)*1 == 0 ):
100         #print ("Text %d of %d\n" % ( i+1, num_texts ))
101         clean_df_texts.append( text_to_words( df[ "text" ][ i ] ) )
102
103 print ("Creating the bag of words...\n")
104
105 # Inicializa el objeto "CountVectorizer" ,que es una herramienta de
106 # scikit-learn's bag of words.
107 vectorizer = CountVectorizer(analyzer = "word", \
108                             tokenizer = None, \
109                             preprocessor = None, \
110                             stop_words = None, \
111                             max_features = 5000)
112
113 # fit_transform () realiza dos funciones: Primero, se ajusta al modelo
114 # y aprende el vocabulario; segundo, transforma nuestros datos de entrenamiento
115
116 # en vectores de características. La entrada para fit_transform debe ser una e
117 # lista de cadenas.
118 df_data_features = vectorizer.fit_transform(clean_df_texts)
119
120 # Los Numpy arrays son fáciles de trabajar con ellos, convertimos el resultado
121 # en array
122 df_data_features = df_data_features.toarray()
123
124 # Vemos los datos limpios
125 # print (train_data_features.shape)
126
127 # Echamos un vistazo a nuestro vocabulario
128 vocab = vectorizer.get_feature_names()
129 # print (vocab)
130
131 dist = np.sum(df_data_features, axis=0)
132
133 # Para cada uno, escribe la palabra de vocabulario y el número de veces que
134 # aparece en el conjunto de entrenamiento
135 for tag, count in zip(vocab, dist):
136     print (count, tag)
137
138 print( "Training the random forest..." )
139
140 # Inicializamos el Random Forest classifier con 10 árboles
141 X_train, X_test, y_train, y_test = train_test_split(df_data_features, df[ "target" ], test_size=0.3)
142 forest = RandomForestClassifier(n_estimators = 10)
143
144 # Podemos adaptar el bosque al conjunto de entrenamiento, usando bag of words
145 # como características y las etiquetas de sentimiento como la variable de respuesta
146
147 # Esto puede tardar unos minutos en ejecutarse
148 forest = forest.fit( df_data_features, df[ "target" ] )
149 y_pred=forest.predict(X_test)
150 #Import scikit-learn metrics module for accuracy calculation
151 from sklearn import metrics
152 # Model Accuracy, how often is the classifier correct?
153 print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
154 print("Precision:",metrics.precision_score(y_test, y_pred, average=None))
155 print("Recall:",metrics.recall_score(y_test, y_pred, average=None))
156 print("F1:",metrics.f1_score(y_test, y_pred, average=None))
157
158 import matplotlib.pyplot as plt
159 plt.scatter(X_train[:,0], X_train[:,1])

```

