UNIVERSIDAD POLITÉCNICA DE CARTAGENA

TRABAJO FIN DE GRADO

Diseño e Implementación de un Motor de Recomendación de Tweets.

Autor: Juan Benito Pacheco Rubio

Director: Juan José Alcaraz Espín

Autor	Juan Benito Pacheco Rubio
Email del autor	juanbenito.pr@gmail.com
Director	Juan José Alcaraz Espín
Email del director	juan.alcaraz@upct.es
Título del TFG	Diseño e Implementación de un Motor de Recomendación de Tweets.
Descriptores	machine learning, python, Django, y tecnologías web
Resumen	Este trabajo desarrolla un recomendador de tweets capaz de realizar un filtrado de los mismos, mostrando al usuario únicamente los tweets más interesantes para ese usuario en particular. Se emplearán algoritmos de clasificación de machine learning.
Titulación	Grado en Ingeniería Telemática
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Septiembre 2016

ÍNDICE

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 MOTIVACIÓN.

En la actualidad, el procesado y manejo de información está en el orden del día, ya sea ayudando en la toma de decisiones a una empresa, como ayudando al día a día a los usuarios. Para esto están surgiendo numerosas tecnologías que ayudan a darle utilidad a esta información, de aquí es de donde surge el machine learning.

El Machine Learning, es una tecnología que está cogiendo cada vez más fuerza dentro del ámbito de la tecnología en general, y las redes sociales en particular. Y es que esta nueva tecnología se encuentra por todas partes. Desde los productos recomendados por Amazon, hasta las noticias de Facebook, pasando por los resultados de búsquedas en google utilizan el machine learning, para dotar de inteligencia y precisión a sus soluciones.

Hasta el momento son numerosas las herramientas que se han creado usando esta tecnología, sobre todo en el ámbito del e-commerce y las redes sociales. Con esto los desarrolladores intentan ayudar a los usuarios a tomar decisiones fácilmente, sobre todo con la recomendación de los diferentes servicios.

Este proyecto tiene como objetivo el intentar ayudar al usuario en el ámbito del manejo en las redes sociales. En un primer momento las redes sociales permitían compartir fotos, contactos e información de personas con las que tenías relación, pero ahora son usadas cada vez más como un canal de información, y es aquí donde surge el problema. Mientras la información se mantiene con una temática homogénea como los recuerdos, o la información de tus amigos la estructura no supone realmente un problema, ya que una página donde se muestre toda la información es bastante adecuada, pero cuando la información se diversifica más, el mostrar toda la información de manera conjunta ralentiza bastante la labor del usuario, es aquí donde este proyecto crea un punto diferenciador.

La herramienta implementa un algoritmo capaz de estructurar la información de las redes sociales en diferentes categorías creadas previamente por el usuario, esto proporciona una mayor facilidad para que el usuario consiga encontrar toda la

información de una manera mucho más cómoda. Además, esta tecnología se puede aplicar de una forma mucho más horizontal que como se está aplicando actualmente en las herramientas de Twitter o Facebook, ya que la herramienta podría ser extendida para el manejo de varias redes sociales, de esta manera un usuario podría tener todos sus datos clasificados y disponibles en una sola herramienta.

1.1. OBJETIVOS.

El objetivo de este trabajo es la creación de una herramienta gráfica, que consiga clasificar la información proveniente de una red social, para este proyecto la red social escogida ha sido twitter.

Esta herramienta será capaz de estructurar la información de dicha red social, y permitir al usuario poder visualizarla cómodamente. Además, la herramienta deberá contar con algoritmos que nos permitan saber el grado de rendimiento de la misma, con el fin de poder comprobar su funcionamiento, y la posible implementación de mejoras futuras.

1.2 CONTENIDOS DE LA MEMORIA.

En esta memoria vamos a explicar todas las partes del proyecto y como se ha desarrollado. Está estructurada de la siguiente manera:

En el capítulo 1, se introduce todo el proyecto, explicando que queremos conseguir y la motivación que nos ha llevado hasta él.

En el capítulo 2, se explican todas las tecnologías y librerías utilizadas, así como las principales funciones y clases.

En el capítulo 3, se trata toda la arquitectura de la herramienta, desde que es dada de alta en twitter hasta que mide el rendimiento del modelo.

En el capítulo 4, En el capítulo 4, se describe la implementación de la parte de machine learning, los parámetros utilizados, la manera de extraerlos y las hipótesis formuladas.

En el capítulo 5, se explica lo referente a la evaluación de nuestro modelo, los parámetros que hemos utilizado y los datos empleados.

En el capítulo 6, se exponen las conclusiones que hemos sacado del proyecto y las líneas futuras las cuales pueden marcar el desarrollo del mismo.

En el capítulo 7, se encuentra alojada toda la documentación usada, así como los enlaces de referencia a cada una de las páginas.

2 TECNOLOGÍAS USADAS.

En este capítulo se explica con detalle cada una de las diferentes tecnologías que he usado para el desarrollo de esta herramienta.

2.1 PYTHON

Para la implementación del proyecto hemos elegido Python como lenguaje principal, ya que se trata de un lenguaje de propósito general ampliamente usado para el análisis de datos.

Este lenguaje se usa frecuentemente para el procesado de información, creación de plataformas webs y machine learning. Todas estas funcionalidades las aportan el gran número de librerías a su disposición. Python al contar con una gran comunidad tiene un amplio soporte, esto le proporciona un gran número de librerías open source.

Se ha escogido Python por la versatilidad de plataformas que ofrece, ya que ha permitido tanto el desarrollo de la herramienta web, como de la lógica del aprendizaje automático.

Python en el ámbito web se utiliza en el lado del servidor permitiendo crear backend muy robustos, lo más frecuente es utilizarlo junto con un framework como pyramid, flask o Django. Django ha sido el escogido para este proyecto, por ser un framework robusto y muy utilizado, que será explicado más adelante.

2.2 TWEEPY:

Tweepy es la librería empleada para la comunicación con twitter, la comunicación con la red social en cuestión se hace imprescindible, por eso el contar con una buena librería que se encargase de ello es imprescindible.

Tweepy permite poder recuperar tweets, loguearse, buscar según parámetros y ver la información de otros usuarios entre otras funciones.

Esta librería fue la escogida al contar con un estado de desarrollo bastante maduro, además tenía un amplio uso dentro de la comunidad de Python, y su más que bien conseguida documentación.

Tweepy está completamente implementada en Python, lo que permite una comunicación con la API REST de Twitter mucho más cómoda, encapsulando la información devuelta en diferentes objetos Python que permiten una facilidad a la hora de trabajar con los datos.

2.3 GIT:

Git es una tecnología de control de versiones, la cual permite llevar un seguimiento de todos los cambios producidos en un proyecto, volver a versiones anteriores, crear ramas para producciones en paralelo, solución de bugs y poder ver quién es el autor de los cambios en cada momento.

Además, git fue utilizada junto con bitbucket, esto permitió el uso de repositorios remotos con el fin de fomentar el trabajo colaborativo, seguridad ante problemas con los cambios en local, y poder trabajar desde cualquier emplazamiento de una manera muy sencilla y cómoda.

El enlace al repositorio remoto es el siguiente:

https://bitbucket.org/Auros132/tweetrecomend

Accediendo a este enlace se puede ver todo el código del proyecto, como se puede observar el código está abierto para cualquiera que desee verlo.

2.4 CLIENTE TWITTER:

Para el desarrollo de nuestro cliente de twitter hemos utilizado las siguientes tecnologías:

1.1.1. Django:

Django es un framework de Python open source, enfocado a la parte de backend, respetando el patrón de diseño conocido como Modelo–vista–controlador, este framework es caracterizado por su robustez y su amplio uso dentro de la comunidad, es utilizado en múltiples plataformas que tienen que dar soporte a miles de usuarios como Pinterest o Coursera.

Se ha escogido Django por la amplia documentación con la que cuenta, su facilidad de uso ya que es un framework que al estar bien documentado permite un rápido aprendizaje y resolución de problemas. Además, posee componentes especializados que ayudan a la rapidez del desarrollo.

Django tiene un diseño modular, esto le permite interactuar con sus principales componentes por separado, además es posible añadirle módulos nuevos que hayan sido desarrollados por terceros, o quitarle algunos que vengan por defecto.

Los principales módulos usados en Django son:

- Su ORM que te permite el acceso a tu base de datos.
- El sistema de autenticación de Django.
- Template Engine, lo que te permite construir vistas dinámicas con su sistema de sintaxis de plantillas.
- Sistema de administración de la base de datos.
- Implementación de módulos para la ayuda a la seguridad de la comunicación.

El ORM de Django es el que se encarga del acceso a la base de datos de una manera muy cómoda y eficiente, este ORM permite diferentes acciones como la creación de modelos, migraciones y la creación de complejas querys de una manera muy fácil y cómoda, ya que permite acceder a cada uno de los recursos como si se utilizara simplemente un objeto.

El sistema de autenticación de Django permite a los usuarios creados autenticarse mediante la conexión HTTP, además esto lo combina con un sistema de sesiones que permite que el usuario se mantenga autenticado de manera constante lo que evita que el usuario no tenga la necesidad de estar introduciendo su contraseña constantemente. Aunque en nuestro caso la autenticación realmente va a depender de Tweepy que es el que se encarga de la comunicación con Twitter es una característica muy interesante que cabe la pena destacar.

El Template Engine de Django, es el encargado de la creación de los templates en Django de forma dinámica. Ésto lo consigue a través de una sintaxis propia. Además se le une la posibilidad de pasar cualquier tipo de información de Django a la vista.

Django ofrece un framework propio para la interacción con su base de datos como hemos dicho anteriormente, el llamado ORM, pero además un panel de administración web completamente navegable con el que puedes interactuar para hacer cualquier cambio. Algo muy intuitivo que facilita bastante la inserción y el borrado de información.

Por último, los módulos de seguridad de Django, que previenen diversas vulnerabilidades de los lenguajes de backend más comunes, como la inyección de código SQL maligno que pueda afectar a nuestra herramienta, protección CSRF, y CORS, además del cifrado automático de contraseñas y el uso de escape de código HTML.

Además de todo lo mencionado, existen multitud de módulos que permiten mejorar nuestra herramienta, tales como tecnologías de colas de tareas concurrentes como Celery o sistemas de caché como Redis, si nos interesase otro sistema de autenticación o de generador de vistas, por ejemplo, bastaría con instalar otro de los muchos paquetes de terceros con los que cuenta el framework.

2.4.1 Javascript:

Para la implementación de la lógica por parte del cliente se ha utilizado javascript, ya que se trata del lenguaje utilizado por excelencia en el lado del navegador, para un uso más cómodo de este lenguaje se ha empleado la librería jQuery, una librería muy extendida que permite utilizar todas las funciones de javascript, como la modificación de valores y detección de eventos, de forma sencilla, sin tener que recurrir al difícil DOM que nos proporciona la API nativa de javascript.

jQuery ha sido utilizado sobre todo en dos aspectos: el primero ha sido el cambio de clases de los elementos mediante las funciones addClass() y removeClass(), estas funciones permiten cambiar las clases asignadas a un determinado componente html, permitiendo así crear efectos dinámicos en las vistas de nuestra herramienta, y la segunda, para las llamadas a AJAX, para conseguir la clasificación de tweets de una manera que minimizase las molestias al usuario, estas llamadas se trata son llamadas no bloqueantes, es decir, se ejecutan de manera asíncrona que no detiene en ningún momento la ejecución normal de nuestro navegador.

2.4.2 CSS:

CSS es el encargado de darle toda la apariencia a la plataforma. Para conseguir el efecto deseado se han tenido que implementar una gran cantidad de reglas y clases CSS, que dan como resultado la imagen de nuestra plataforma .

Para esto se ha empleado el framework Bootstrap que permite interactuar con la ventana del navegador como si ésta fuese un grid, dividiendo a la ventana en un total de 12 columnas, de manera que se le puede indicar a cada uno de los elementos cuantas columnas tiene que ocupar. Además permite indicar a los componentes como se tienen que situar en la ventana si hubiese alguna variación en el tamaño de la misma, permitiendo crear una aplicación con una apariencia visual atractiva y capaz de adaptarse a diferentes formatos.

De bootstrap se han usado diferentes clases como la barra de navegación, algunos contenedores como el fluid-container, que son los que encapsulan, en función del formato que quieras usar, el contenido de nuestra página web. Por ejemplo, fluid-container ocupa todo el contenido de la página sin márgenes, mientras que container es con márgenes tanto a derecha como a izquierda. Además, para encerrar el contenido de cada tweet hemos utilizado la clase panel.

Para darle una apariencia más personal se han utilizado un conjunto de clases para adaptar los estilos de bootstrap a nuestras necesidades, y esto es lo que ha dado como consecuencia la apariencia de la herramienta.

2.5 LIBRERÍAS DE MACHINE LEARNING:

Para la implementación del algoritmo de aprendizaje automático, se han utilizado librerías especializadas en este propósito, estas librerías son capaces de trabajar con estructuras específicas de datos, y funcionalidades específicas de machine learning.

2.5.1 Numpy y Scipy:

Numpy es la librería por excelencia para cualquier Data Scientist, esta librería permite el manejo de estructuras de datos de una manera eficiente. Esta librería está implementada en C, gracias a esto consigue optimizar procesos como, los accesos a

memoria y el tiempo de cómputo, evitando así sufrir la lentitud que los lenguajes interpretados como Python incluyen inherentemente.

Numpy ante todo proporciona estructuras para encapsular los datos, como arrays o matrices n-dimensionales implementados de manera propia, para evitar así tener que utilizar los que te da Python mucho más lentos, además de poseer una gran cantidad de funciones para el manejo de dichas estructuras.

Scipy en cambio es una biblioteca de funciones matemáticas de alto nivel, esto permite trabajar con toda clase de datos de una manera muy potente, se llega a comparar con otras herramientas matemáticas como Matlab y Octave. Contiene un gran número de operaciones de optimización, álgebra lineal y procesamiento de señales. El aspecto de Scipy más útil para este proyecto es la gran compatibilidad que tiene con el resto de herramientas, ya que ofrece soporte para varias funciones de Numpy, y para las librerías de machine learning.

2.5.2 Scikit-Learn:

Scikit-Learn es una librería especializada en machine learning, y nuestra tecnología fundamental en el campo del aprendizaje máquina. Esta librería cubre ámpliamente todas las necesidades básicas que se puedan necesitar en una herramienta de aprendizaje automático, tanto algoritmos de aprendizaje, como funciones que proveen diversas utilidades, como, por ejemplo, obtener los mejores parámetros para la creación de un modelo o clases para obtener métricas sobre el rendimiento de tu modelo.

Scikit-Learn hace uso de las librerías Numpy y Scipy explicadas anteriormente, como dijimos la estructura de datos y las funciones matemáticas que te ofrecen Numpy y Scipy son mucho más eficientes y rápidas, y cuando nos encontramos ante un problema de este tipo el tiempo y la eficiencia son dos aspectos muy importantes.

Hemos elegido esta librería por la amplia gama de opciones que te ofrecía, respecto a otras como Theano, TensorFlow, Keras... Ya que normalmente estas se encargan de un tipo de algoritmo de machine learning en particular, y no poseen la caja de herramientas al completo.

Las funciones que hemos empleado de esta librería son:

- **SVC:** Es nuestro estimador, se encarga de crear nuestro modelo, en función de los parámetros que elijamos para ello, se basa en el funcionamiento del algoritmo “máquinas de vector soporte” enfocadas a la clasificación. En nuestra herramienta es el estimador que usamos junto con OneVsRest.
  Tiene como argumentos los parámetros necesarios del algoritmo el parámetro C, tipo de kernel que va a usar para clasificación, tolerancia y gamma si se elige un kernel rbf

- **OneVsRest:** Esta función tiene como argumento un estimador, y se encarga de crear un modelo usando ese estimador para cada clase diferente, se utiliza sobre todo en algoritmos de clasificación donde hay múltiples clases.
- **cross_val_score:** Es la función que nos indica la precisión de nuestro modelo, hay que pasarle como argumentos nuestro estimador, el conjunto de ejemplos de entrenamiento, y el algoritmo que se quiere usar para que indique la precisión. Implementa el algoritmo K-folds con el cual haciendo uso del parámetro cv, se le indica a la función cuantas son las partes en las que tiene que ser dividas nuestro training set, el algoritmo usara todas las partes excepto una que será la que utilice para validarlo.
- **GridSearchCV**: Esta función encuentra los mejores parámetros para un estimador dado, de manera que podemos construir un modelo lo más exacto posible. Este método funciona de la siguiente manera: crea un grid con todas las combinaciones de argumentos posibles y se las pasa al estimador para que cree el modelo, y así evaluar su precisión. Es una función que ahorra varios procesos, y lo sustituye por una función bastante intuitiva.

En definitiva, Scikit-Learn es un kit imprescindible con todo lo necesario para crear, entrenar y validar tu modelo de una forma muy rápida.

1.1.2. NLTK**:**

Natural Language Toolkit (NLTK), es una librería que se encarga de la interpretación del lenguaje de manera escrita o hablada. Esta librería es muy utilizada en el campo del Text Mining o minería de texto con el fin de extraer conclusiones acerca de conjuntos de

texto no estructurado. Esta librería se utiliza habitualmente en combinación con un algoritmo llamado bag of words, que se encarga de la interpretación de las palabras de manera separada intentando no entender las palabras de manera conjunta, sino más bien dándole un significado por separado a cada palabra y extrayendo una conclusión final sobre el texto interpretado.

Una aplicación habitual de NLTK es el análisis de redes sociales, ya que normalmente estas plataformas se caracterizan por la comunicación escrita entre diferentes usuarios, y algoritmos como el de bag of words trabajan muy bien en estos entornos. Existen numerosos proyectos que aplican dicho algoritmo para sacar conclusiones como sentimientos o tendencias.

El aspecto más influyente al emplear NLTK es el idioma de los textos que van a ser analizados. El idioma inglés, al ser el más extendido tiene una mayor cantidad de recursos, así como corpus predefinidos. El corpus es un gran conjunto de palabras en un idioma predeterminado. Es por eso que la posibilidad de encontrar corpus en inglés amplio para determinados proyectos es más sencilla que para otros idiomas.

Para este proyecto ha sido necesario crear nuestro propio corpus, con los tweets que se han clasificado. Con NLTK es especialmente importante decidir primero el idioma de los tweets que vamos a analizar, ya que para poder clasificar satisfactoriamente tweets de diferentes idiomas necesitaríamos un corpus mucho más grande. Para el proyecto se ha elegido el español.

NLTK dispone de varias funciones muy importantes, una de ellas es la función llamada stopwords que proporciona una lista de las palabras que carecen de significado en un idioma determinado, otra sería SwnowBallStemmer que es la encargada de encontrar la raíz de las palabras. Esto resulta muy útil ya que te ahorra gran cantidad de trabajo. Ambas librerías se encuentran disponibles para su uso en multitud de idiomas.

Esta librería, junto con todas sus funciones, puede ser usada se puede ser usado junto con la clase CountVectorice de Scikit-Learn la cual aplica el algoritmo bag of words de una manera casi inmediata, el problema de esta técnica es la necesidad de un corpus muy grande del que no se disponía en el experimento.

3 ARQUITECTURA.

Una de las piezas fundamentales de la herramienta es el cliente gráfico, que nos muestra toda la información relevante acerca de los tweets, la autenticación y la clasificación.

El cliente se encarga de facilitar el trabajo de clasificación, haciéndolo de una manera bastante cómoda e intuitiva.

El cliente se divide en varios componentes.

3.1 LA API DE TWITTER.

Para poder acceder a los datos de twitter, tenemos en primer lugar que dar de alta el proyecto en la página de desarrolladores de twitter.

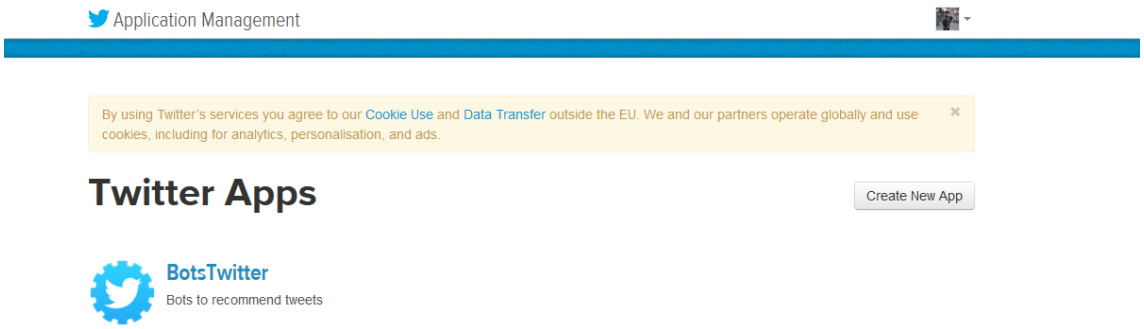

Figura 1 TwitterDev

Pulsamos sobre el botón de crear nueva aplicación, para que nos lleve al formulario donde crearemos nuestra app.

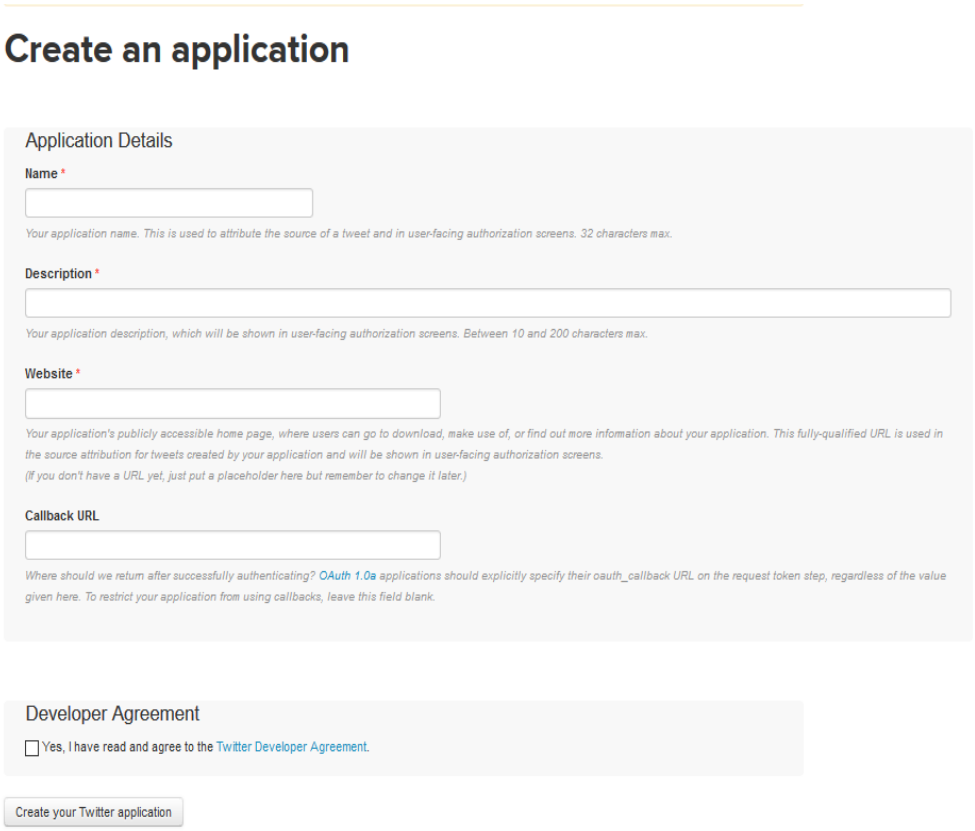

Figura 2 Formulario TwitterDev

Rellenando el formulario, y aceptando las condiciones de uso, finalizamos el proceso de creación de la aplicación. Ahora se pueden ver todos los datos necesarios de nuestra aplicación, así como las keys y token que dan el acceso a la herramienta, los permisos que la plataforma tendrá en twitter o quién la creó.

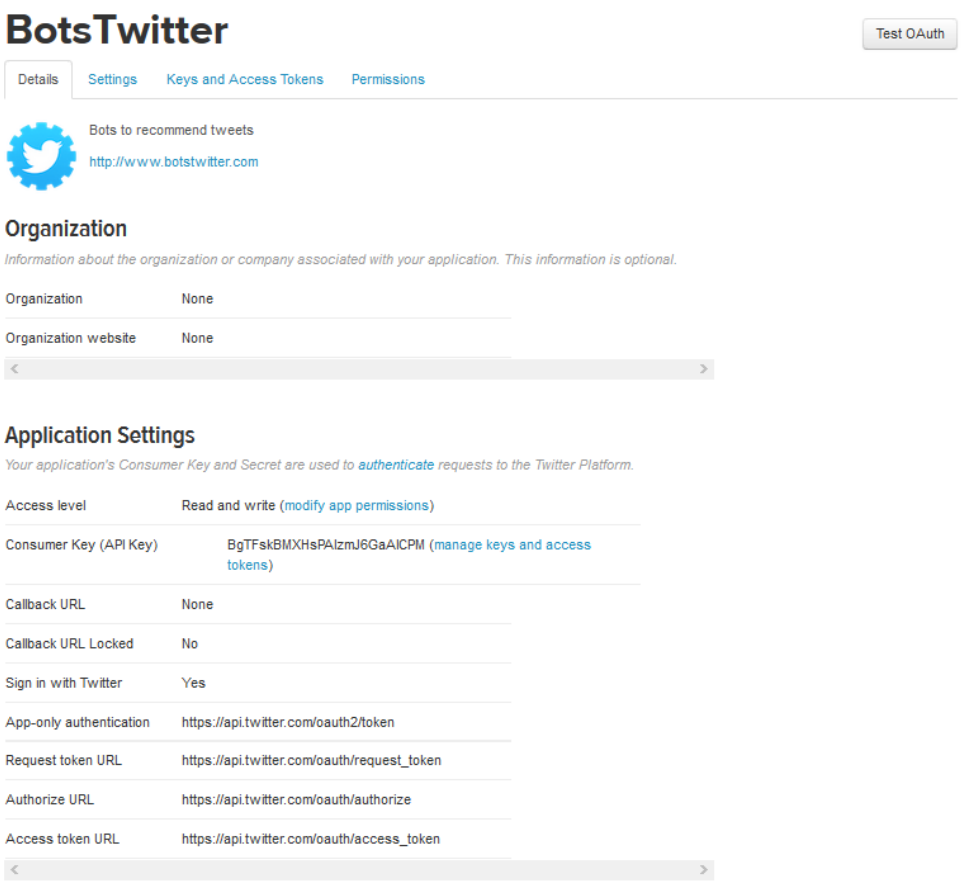

Figura 3 Configuracion TwitterDev

3.2 CONFIGURACIÓN DE DJANGO:

Todo proyecto de Django cuenta con un archivo settings.py, que le indica a Django cuál es su configuración principal, este archivo es principalmente la base de nuestro proyecto e indica cuáles son y dónde se encuentra todo lo relacionado con él. Por ejemplo, donde se encuentran los archivos estáticos, los media files, la base de datos a acceder, las apps instaladas, todos los módulos, la clave secreta con la que serán cifradas las contraseñas o si tiene el modo debug activado. Si quisiésemos añadir otro sistema de autenticación, utilizar otra clase de usuarios por defecto…, bastaría con añadirlo o quitarlo del settings.py.

3.3 DIRECCIONAMIENTO.

La manera de direccionamiento de nuestro framework es muy sencilla. Cuenta con un archivo llamado urls.py, en el cual se especifican el nombre de los enlaces con el controlador al que tiene que ir la petición. El código de nuestro módulo urls.py es el siguiente:

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', TrainningTweets.as_view()),
    url(r'^login$', LoginView.as_view()),
    url(r'^classifier$',Classifier.as_view(),
    url(r'^metrics$',MetricsView.as_view()),
    url(r'^categorias$',CategoriesView.as_view()),
    url(r'^categoria_classifier/([0-9]+)$',CategoriaClassifierView.as_view()),
    url(r'^metrics_one_categorie$',MetricsOneCategoria.as_view())
]
```

Funciona de la siguiente manera: en el archivo se incluye un array de clases url, en estas clases Django busca el url al que queremos dirigirnos. Cada clase de url se construye a partir de una expresión regular indicando el nombre del enlace, y si posee algún parámetro adicional, por ejemplo **([0-9]+)$** como la que hay en el enlace de categoría classifier, indicamos que en el enlace podría añadirse un número que se repita una o más veces. Esto resulta útil para una petición get cuando necesitemos acceder mediante el identificador de un objeto, en cuyo caso tenemos que indicar en el url cuál es el controlador que contendrá la lógica de esa dirección web.

3.4 BASE DE DATOS.

En la plataforma utilizamos una base de datos sqlite3, que es el tipo de base de datos que viene por defecto al utilizar Django. Es una base de datos bastante ligera con todo lo necesario para conseguir información persistente, y el uso de querys.

3.4.1 Modelos de la Base de Datos.

La creación de modelos (tablas) en la base de datos se hace mediante las clases que Django ha creado para ello models.Model, desde las cuales podemos implementar toda nuestra base de datos con sus correspondientes relaciones.

En primer lugar, hemos creado varias tablas que tienen que guardar todos los datos que necesiten persistencia. Esto es importante, ya que sin la persistencia de los datos cosas como el entrenamiento, que es imprescindible para la creación de un modelo fiable, sería imposible de realizar.

Las tablas utilizadas son las siguientes:

- **Autores**: Esta tabla guarda toda la información relativa a los autores que han escrito los tweets, el lenguaje que utilizan, su nombre, personas que le siguen, identificador de twitter y personas a las que sigue.

- **Categorías**: Guarda las categorías en las que queremos clasificar nuestros tweets, en este momento estamos utilizando 7 categorías: Entretenimiento, Deportes, Noticias, Tecnología, Política, Humor y Ciencia.
  Esta categoría simplemente posee el nombre de dicha categoría.

- **TweetModel:** Esta es una de las tablas más importantes de la base de datos, ya que se encarga de guardar toda la información relativa a los tweets, y los parámetros sobre los cuales vamos a entrenar nuestro modelo.
  Las características que se guardaron aquí han tenido que ser cuidadosamente pensadas. Un error en esta tabla puede tener consecuencias muy importantes a la hora de la construcción de nuestro modelo.
  Finalmente, las características guardadas son: texto que contiene el tweet, número de retweets, número de favoritos, el autor que lo ha escrito, la fecha en la que se escribió el tweet, la fecha en la que se guardó, a que categoría pertenece, si posee alguna foto y el url hacia dicho tweet.

- **TextWord**: Esta es la tabla que nos permite utilizar las técnicas de Text mining, almacenando las palabras más usadas de cada uno de los tweets que fueron clasificados y la que categoría en la que fueron clasificados, además de las repeticiones que ha sufrido dicha palabra.

Realmente lo que se guarda en esta tabla, no es la palabra en sí, sino la raíz de dicha palabra evitando así que palabras como saltar, saltando, saltitos, salto se consideren diferentes, teniendo un significado equivalente.

Cada una de las tablas anteriores, cuentan con un campo adicional llamado id, este campo identifica cada uno de los elementos que compondrán la tabla con un identificador único para ese elemento.

3.4.2 Funciones especiales y de integridad.

La mayoría de las clases hacen uso de una función llamada __unicode__, que le indica al panel de administración de Django como debería mostrar cada una de las filas de la tabla, por ejemplo, la tabla categorías:

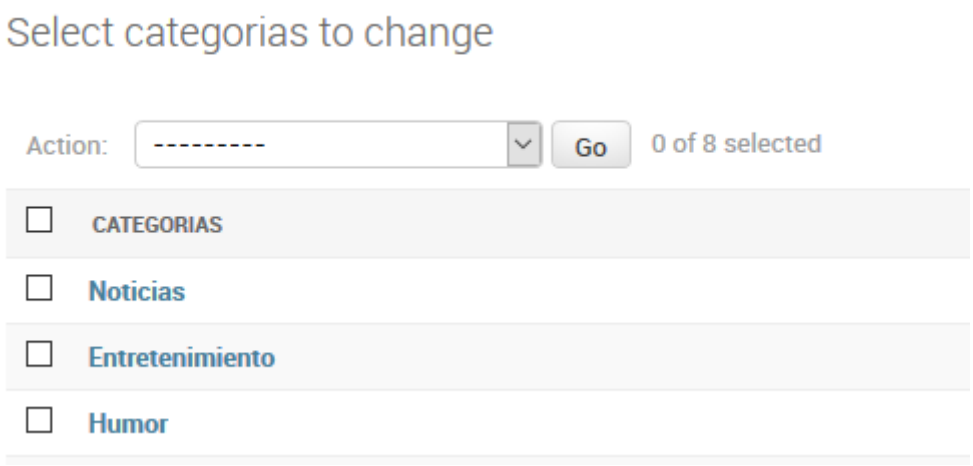

Figura 4 Unicode

Como se puede ver se muestran para cada fila el nombre de cada categoría, como se ve expresado en la propia función __unicode__.

```
class TextWord(Model):
    word = CharField(max_length=30)
    categoria = ForeignKey(Categorias)
    repeticiones = IntegerField(default=1)
    def __unicode__(self):
        return str(self.repeticiones)+' '+self.word
    class Meta:
        unique_together = ("word","categoria")
```

Existen multitud de propiedades adicionales que pueden ser usadas en nuestra base de datos. En la clase TextWord, por ejemplo, hemos hecho uso de la propiedad

unique_together. Esta propiedad indica a Django la restricción de que el conjunto de elementos dentro de la tupla debe ser único en esa tabla, en este caso le hemos indicado que una misma palabra solo puede existir para una categoría diferente. En el caso de que quisiésemos guardar la misma palabra para la misma categoría dos veces distintas, lanzaría un error de integridad en la base de datos.

3.4.3 Administración.

Django proporciona un panel de administración mediante el cual se pueden hacer numerosas operaciones, como especificarle en un archivo de nuestra aplicación, qué modelos queremos que sean visibles en el panel de administración, y qué operaciones podremos realizar en dichos modelos.

El panel de administración de Django es el siguiente:

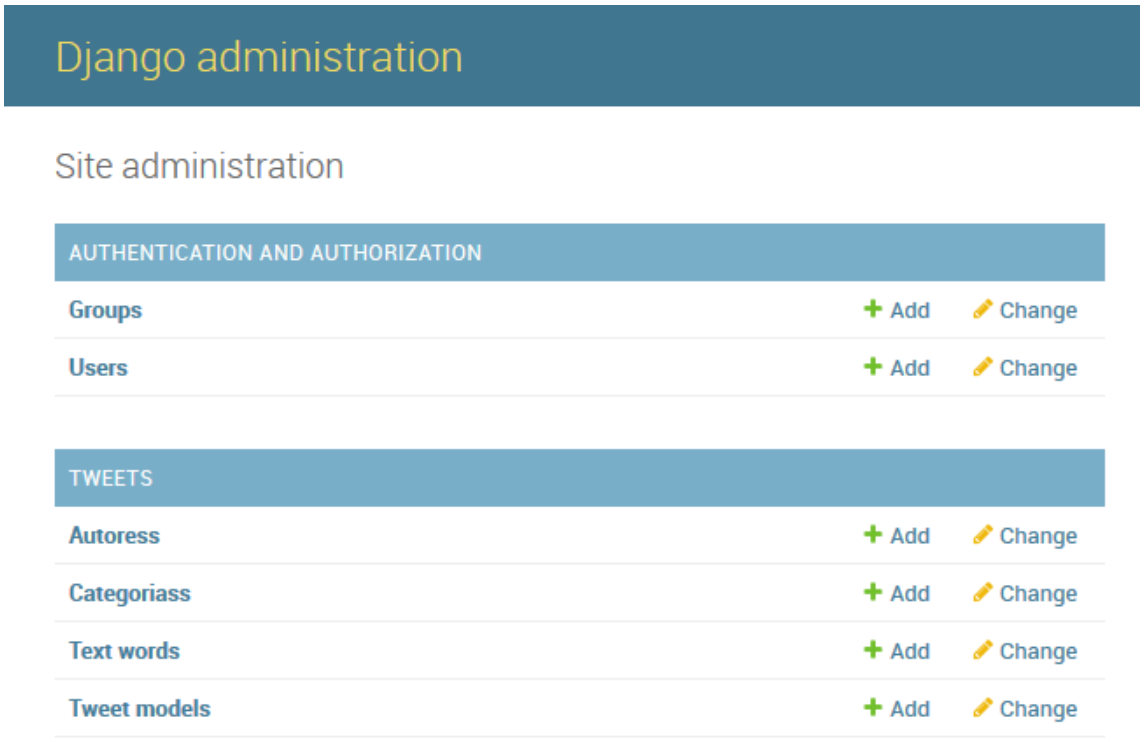

Figura 5 Panel Administración Django

En él se muestran todos los modelos que hemos registrado, además de poder hacer acciones como añadir más registros y modificar los existentes.

El código para registrar nuestros modelos en el panel de administración de Django es el siguiente:

```
class TweetAdmin(ModelAdmin):
    list_display = ('autor','retweeted','retweets')
admin.site.register(TweetModel,TweetAdmin)
admin.site.register(Categorias)
admin.site.register(Autores)
admin.site.register(TextWord)
```

Además de registrarlos, podemos añadir otras opciones como especificar qué campos mostrar en él, y de qué manera. También permite opciones como filtrar por algún campo y hacer búsquedas, pero estas funciones no eran necesarias dentro del proyecto por lo que no han sido desarrolladas.

3.5 VISTAS.

Vistas es como se les llama a los controladores de los templates en django. Django es un framework Modelo-vista-controlador, por tanto, se caracteriza por tener separados los modelos que guardan los datos, la parte gráfica y los agentes que se encargan de coordinar a los modelos con las vistas de una manera totalmente separada. Estos agentes son las Vistas de Django.

Las vistas se encargan de proporcionar la lógica a la parte gráfica de la herramienta. Los controladores extienden la clase Views de Django.

Los controladores de Django cuentan con dos métodos fundamentales: el método get y el método post, que se encargan de manejar las diferentes peticiones que haga el navegador. Cada petición es manejada por el método que lleva su nombre, de manera que cuando el navegador pida la web, se ejecutará el método get, y cuando quiera mandar datos ejecutará el método post.

Además de los controladores, la parte gráfica que es mostrada por el navegador para el usuario, son los templates en Django, pero en este trabajo los llamaremos comúnmente vistas por su semejanza con la parte gráfica, y a las vistas de Django simplemente controladores.

Estos templates se generan utilizando el motor de vistas de Django. Esto permite que en unas pocas líneas de código se puedan generar vistas muy completas.

La herramienta se divide en tres grandes grupos, que se explicarán por separado: Login, Entrenamiento, Clasificación, Métricas.

3.6 LOGIN.

En primer lugar, para poder acceder a toda la información que el usuario tiene disponible en twitter hay que darse de alta en la plataforma. Esto se ha implementado mediante un panel de login, al cual accedemos al entrar por primera vez en la herramienta.

Figura 6 Login Twitter

Pulsando sobre el botón de "logueo con twitter", nos lleva a una ventana la cual nos pide que autoricemos la aplicación, con esto le damos permiso a nuestra herramienta para poder entrar a tus datos personales de twitter.

Figura 7 Autorizar Twitter

Figura 8 Código Twitter

Por último, al pulsar en autorizar nos sale un pin de acceso como este, introduciéndolo en el input que había en el panel de login anterior, daremos a la herramienta acceso a los datos de twitter

Una vez introducimos el código, obtenemos el token necesario para el acceso a los datos de twitter, este token lo guardamos en una cookie que expira en cada sesión, con el objetivo de evitar tener que hacer molestas peticiones de volver a introducir el código constantemente. Ahora la página nos redirige a la ventana de clasificación de tweets, donde podremos comenzar nuestra actividad.

3.7 ENTRENAMIENTO.

Una vez logueados en twitter accedemos de manera automática a la ventana de entrenamiento, en esta ventana se nos mostrará un total de 20 en diferentes paneles, con foto si es que el tweet posee alguna foto y si no la posee solo se verá un espacio en blanco, para poder disfrutar de la experiencia de twitter de manera completa.

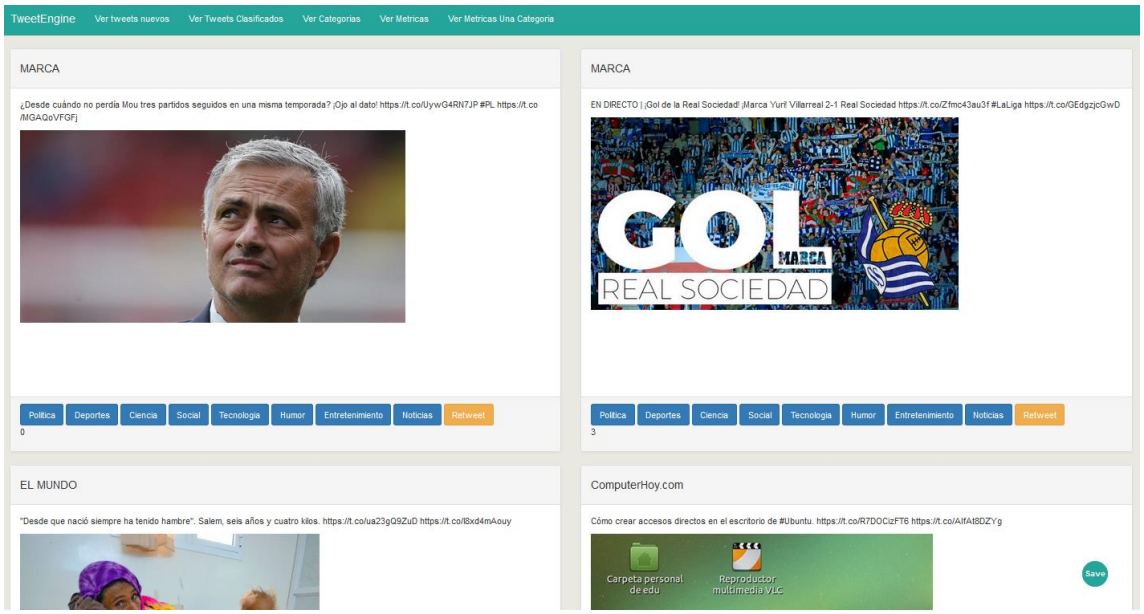

Figura 9 Clasificación Tweets

Como vemos, los tweets poseen las categorías en las que pueden ser clasificados abajo, de manera que con pulsar en cada una el botón de la categoría en la que queremos clasificar el tweet, el botón pasa de azul a verde, esto significa que ya hemos marcado ese tweet como que pertenece a esa categoría. Si quisiésemos indicar que pertenece a dos categorías bastaría con pulsar sobre los dos botones. Cuando hemos acabado de marcar todos los tweets basta con darle al botón save, y este automáticamente guardará los tweets marcados con sus correspondientes categorías en la base de datos.

En esta parte es donde más importancia toma el código en javascript, ya que utiliza jQuery para cambiar los estilos css, y hacer las llamadas a Ajax, para comunicarnos con nuestro servidor. Django hace uso de un método de protección contra csrf (Cross-site request forgery), para poder pasar a través de esa protección tenemos que comprobar

una cookie puesta por django de forma automática en el DOM, y enviar el contenido de esa cookie para que Django corrobore nuestra identidad.

El método get es el quien se ejecuta cuando el navegador pide la página web en primer lugar, cuando se ejecuta recupera los 20 últimos tweets de twitter, los guarda en la base datos y se los pasa al usuario para que los clasifique

Una vez accedemos a la página, encontramos un panel de logueo, que gracias a la librería Tweepy hacemos permitimos el acceso nuestro usuario, mediante un código que nos proporciona twitter.

3.8 CLASIFICACIÓN.

Hemos creado dos vistas adicionales que nos permiten utilizar las diferentes "modalidades" de clasificación desarrolladas para la plataforma, las cuales se pueden acceder una vez que el usuario se ha logeado.

Las vistas se han separado en dos en función de las dos modalidades según si se quieren ver clasificadas en categorías diferenciadas o con todas a la vez.

3.8.1 Clasificación Una Categoría.

Al pulsar sobre ver categorías nos lleva a una pantalla donde podremos seleccionar entre todas las categorías que hemos creado, ver los tweets que corresponden solamente a esa categoría. En esta "modalidad" de clasificación hacemos uso de la clasificación binaria, con la cual nuestro clasificador solo comprueba si pertenece a la clase especificada o no. Así es como se ve la siguiente vista:

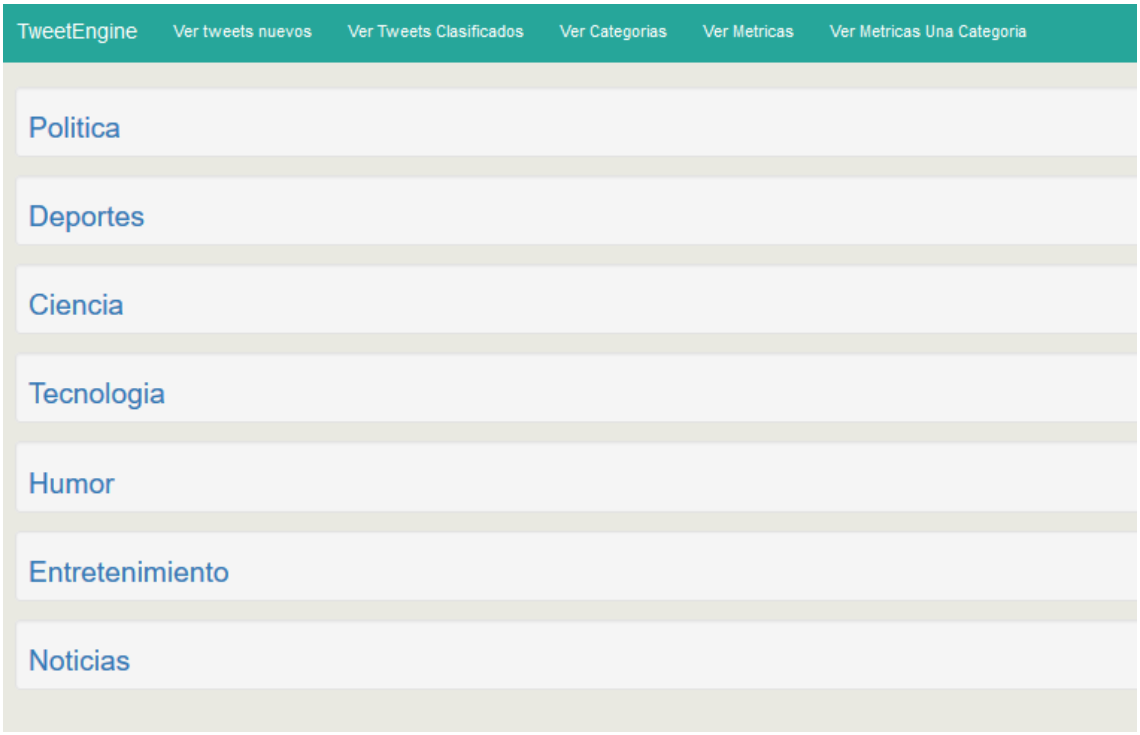

Figura 10 Categorías para ser clasificadas

Al pulsar sobre cualquiera de las categorías avanzamos hasta una vista de los tweets que el algoritmo de aprendizaje clasifica como perteneciente a dicha categoría.

Por ejemplo, si pulsamos sobre la categoría de tecnología nos muestra lo siguiente:

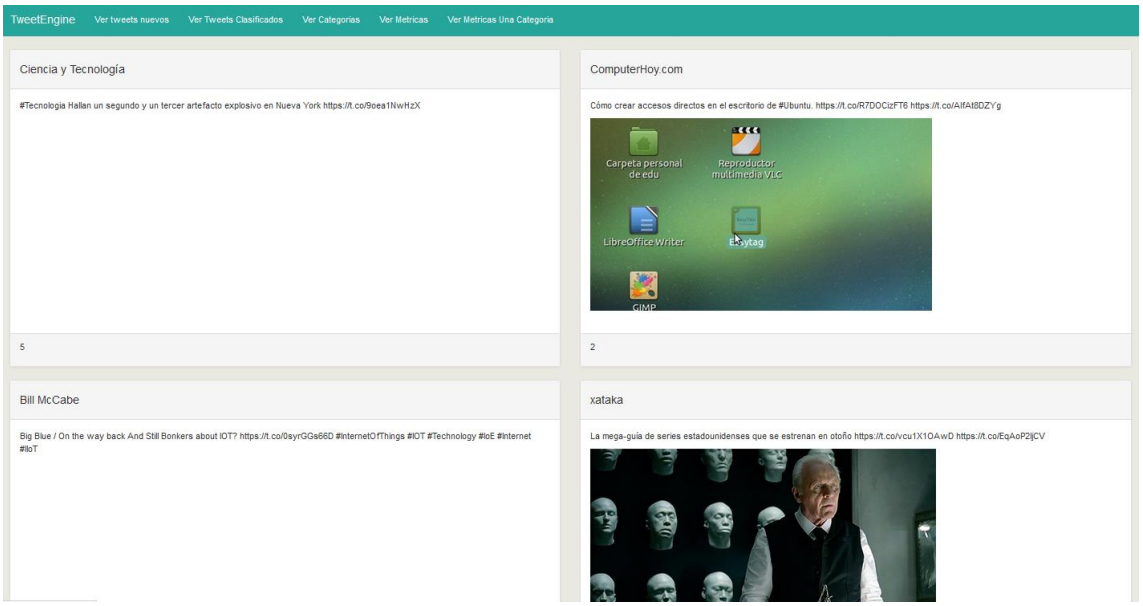

Figura 11 Clasificación una categoría

Nos enseña una ventana donde vemos todos los tweets que han sido catalogados como pertenecientes a la categoría de tecnología, además de toda la información relevante de ellos.

3.8.2 Clasificación Multilabel.

Se han implementado una vista adicional, que consiste en una clasificación de todos los tweets que se piden a twitter de manera multilabel. Esta modalidad, consiste en que ya no solo consideramos que tenemos más de una clase, sino que un elemento puede pertenecer a más de una de las clases. Para esto hacemos uso de un clasificador basado en máquinas vector soporte y en el algoritmo OneVsAll. Finalmente obtenemos una vista como la que se muestra a continuación:

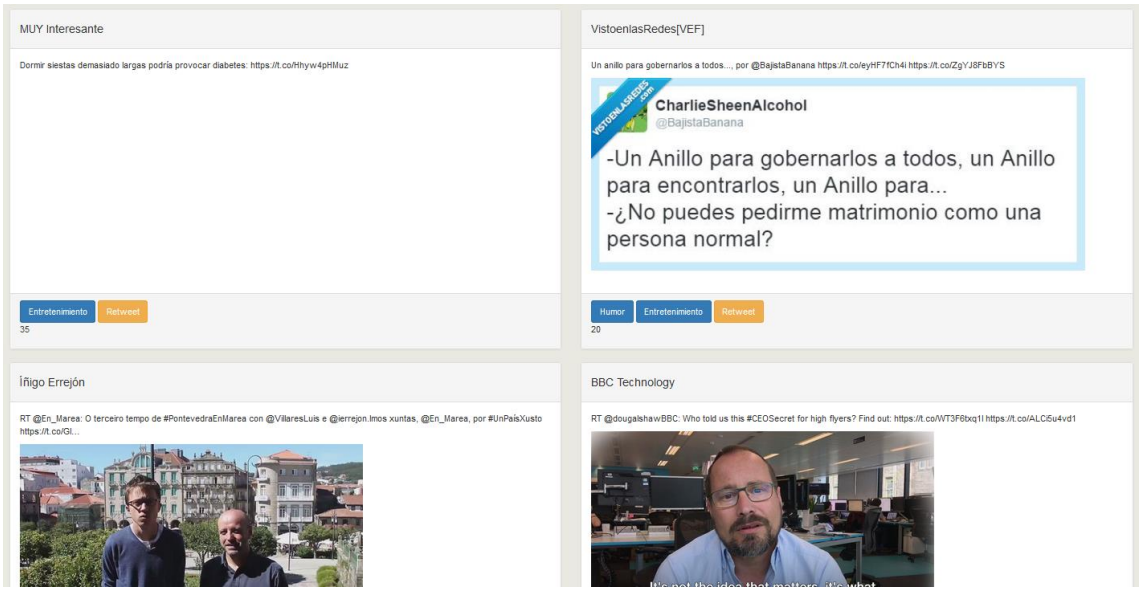

Figura 12 Clasificación multilabel

Como vemos, el clasificador en esta ocasión no solo es capaz de detectar a que clase pertenece, sino si pertenece a más de una al mismo tiempo.

3.9 MÉTRICAS.

Con el fin de obtener una manera de evaluar nuestro modelo de la manera más intuitiva posible, se han creado un par de vistas cuyo rendimiento se evalúa con dos métodos diferentes de estimación de la precisión, además de los mejores parámetros para obtener la mejor precisión en ambos casos. Los métodos de estimación de la precisión

escogidos son “f1_weighted” y “roc_auc” o área bajo la curva. Más adelante se explicará por qué han escogidos estos métodos, y la forma en la que son calculados.

Ambas vistas guardan una gráfica que representa los diferentes datos obtenidos del grid creado para buscar los mejores parámetros, comparando los diferentes métodos, y variando los parámetros mostrando la variación que tiene en la precisión cada uno.

3.9.1 Métricas Una Categoría.

Esta vista muestra la evaluación de los dos métodos de estimación de la precisión en cada una de las diferentes categorías, además de mostrar qué parámetros son los más acertados.

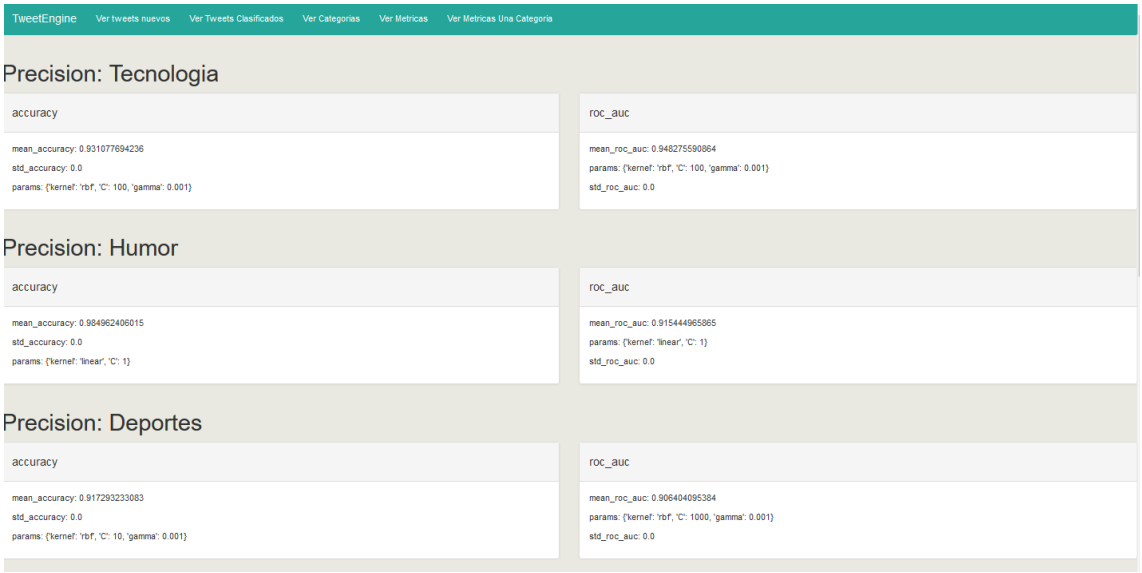

Figura 13 Métricas una categoría

Como vemos se divide cada una de las precisiones en las diferentes categorías que habíamos creado, y se generan diferentes paneles con los datos que nos hacen falta

3.9.2 Métricas Multilabel.

Igual que en la vista anterior, para conseguir ver el rendimiento que se obtenía con nuestro algoritmo multilabel, se ha creado una vista que lo muestra para cada uno de los diferentes algoritmos, y que nos dice los parámetros que tendríamos que usar para obtener la mejor precisión en cada método.

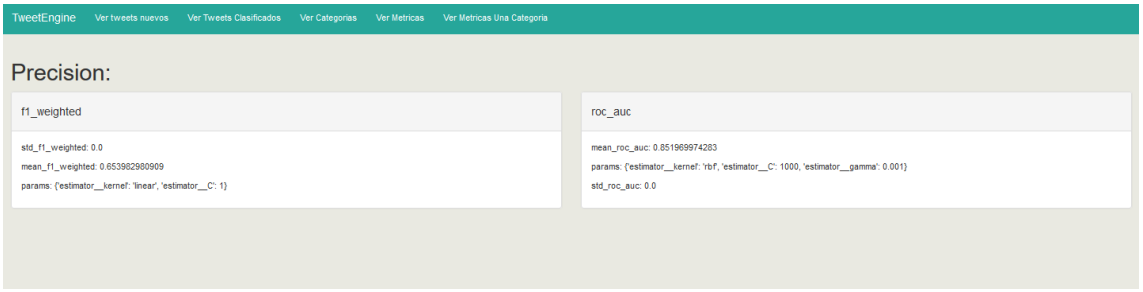

Figura 14 Métricas multilabel

3.10 UTILIDADES

Para la implementación de los objetos y métodos necesarios para todo el desarrollo de la lógica de las vistas de la herramienta, se ha creado un nuevo fichero llamado MethodUtils.py.

3.10.1 TweetObject.

Para poder interactuar con todos los datos relevantes acerca de los tweets he creado el objeto TweetObject, este objeto cuenta con todas las propiedades acerca de que texto guarda, los datos del autor, el número de retweet y el número de favoritos. Es decir, consigo encapsular todos los datos que me da la API de twitter en diferentes clases en un solo objeto. De esta manera solo con la implementación de dos funciones puedo transformar este objeto en un formato representable en mi template o en un elemento de la tabla TweetModel.

3.10.2 MethodUtils.

Esta clase extiende también la clase object lo que permite que sea utilizada como objeto, tiene implementados todos los métodos que van a ser necesarios utilizados por varias clases, además de esta manera se pueden compartir variables de instancia y estado de los métodos.

La lógica de la plataforma se ha dividido en los diferentes métodos que componen este módulo, desde el logueo de la herramienta, hasta la extracción de las features de los tweets y el posterior entrenamiento, y clasificación de nuestro modelo.

3.10.3 TextMining.

Este módulo se implemenó para facilitar el uso de todo lo necesario referente al text mining. Aquí se recogen todas las funciones necesarias para extraer cada una de las

palabras, descartar las palabras que no aportasen información alguna y los signos de puntuación y guardar solo aquellas palabras relevantes, este módulo hace sobre todo uso de la librería NLTK, ya que es la destinada a este tipo de funcionalidad.

4 APRENDIZAJE DE LA HERRAMIENTA.

En este capítulo se va tratar todo lo relativo a las técnicas de machine learning utilizadas en la herramienta, así como los diferentes algoritmos en los que se basa.

4.1 MÁQUINAS DE VECTOR SOPORTE (SVM)

Para este proyecto se decidió hacer uso de un algoritmo basado en Máquinas Vector Soporte, ya que el problema principal al que se enfrenta es a la clasificación, y este tipo de algoritmos consigue una gran precisión sin la necesidad de una cantidad demasiado alta de ejemplos de entrenamiento, además de optimizar la eficiencia trabajando con vectores de características muy largos.

Las Máquinas de Vector Soporte funcionan de la siguiente manera, crean una función particular que siguen un conjunto de puntos pertenecientes a una categoría, y mediante esta crean un hiperplano que consigue acotar dicho conjunto de puntos. Con esto se consigue implementar un modelo capaz de predecir si un futuro punto pertenece a la misma categoría que el resto, o si por el contrario está al otro lado del hiperplano.

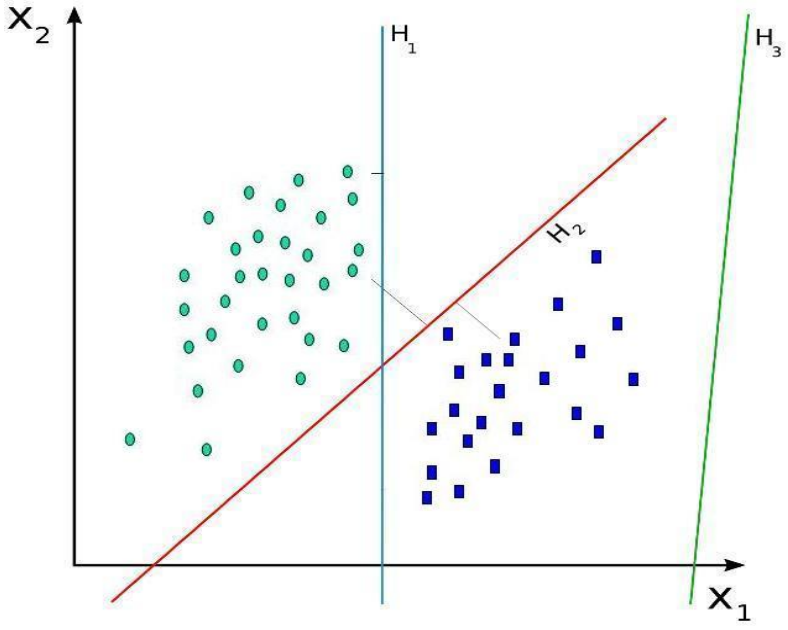

Figura 15 Ejemplo Máquinas Vector Soporte

Las SVM poseen parámetros que le ayudan a encontrar el mejor modelo posible, estos son:

- kernel que puede ser linear, gaussiano(rbf) o polinomial. El kernel es como se llama al método seguido para la creación de las fronteras de clasificación.
- parámetro C, controla la compensación entre los errores de entrenamiento. Este parámetro permite que el modelo tenga unos "márgenes blandos" los cuales permiten algunos errores en la clasificación de los ejemplos de entrenamiento, con el fin de ser un poco más flexibles en ciertas muestras.
- Parámetro gamma, es un parámetro del kernel gaussiano para controlar las clasificaciones no lineales, ayuda a controlar si hay overfitting o underfitting sobre los ejemplos de entrenamiento.

Estos parámetros tienen un papel fundamental en el correcto ajuste del modelo, es por eso tan importante encontrar los parámetros más adecuados.

4.2 CARACTERÍSTICAS.

Para la implementación del algoritmo de aprendizaje máquina hemos combinado el uso de dos técnicas de extracción de features y Text Mining, con el fin de aumentar la precisión a la hora de clasificar cuentas que publiquen tweets de diversas categorías.

4.2.1 Features.

Para el correcto aprendizaje de nuestro algoritmo, se tienen que extraer una serie de parámetros los cuales conforman nuestro vector de entrenamiento, esos parámetros son las features.

Al hablar de features en el proyecto, nos referimos a las características principales de los tweets, en función de las cuales hemos ajustado el modelo, tales como el autor, lenguaje, número de retweets…

En primer lugar, para entrenar nuestro modelo tenemos que extraer una serie de parámetros, su número de retweets, el autor que lo ha escrito, si posee foto o no, y el texto que contiene. Todas estas cosas son guardadas en nuestra base de datos para usarlas en el entrenamiento posteriormente.

Cuando accedemos a la vista de TweetRecomendados es cuando se ejecuta el algoritmo. Este crea un vector para cada tweet, donde agrupa las características principales, y la clase a la que pertenece en caso de ser uno de los tweets del conjunto de

entrenamiento. Si es un tweet nuevo que tiene que ser clasificado únicamente se crea el vector de características.

4.2.2 TextMining.

La segunda técnica de extracción de características se llama TextMining. Esta técnica consiste en la extracción de las palabras usadas en los tweets para la creación de nuestro vector de características.

Este algoritmo se compone de los siguientes pasos:

1. Se extraen todas las palabras de cada uno de los tweets.
2. Se eliminan las palabras que son repetidas frecuentemente, pero carecen de significado, a estas les llamamos stopword, y los signos de puntuación
3. Se utiliza un objeto SnowBall el cual con su método stem, transforma la palabra en su raíz eliminando cualquier prefijo o sufijo, evitando así diferencias innecesarias.

Una vez hecho esto guardamos cada una de las palabras por separado en la tabla TextWord esta tabla es la encargada de guardar la palabra que corresponde a cada categoría y el número de veces que ha sido repetida.

Así finalmente cuando necesitemos recuperarla para nuestro algoritmo con una simple query podremos obtener todas las palabras referentes a una categoría, en función de las veces que han sido repetidas.

4.2.3 Vector de características.

El vector de características es un vector que agrupa todos los parámetros reseñables de los elementos queremos clasificar, para que el modelo pueda ajustarse a ellos. Para la clasificación de los tweets en nuestro caso, está constituido de la siguiente manera.

En primer lugar, construimos un array de ceros con todos los autores de los que tenemos tweets, hasta ahora. Este array será el primer elemento del vector de características, solo que pondremos 1 en el índice del autor que corrresponda con el autor que ha escrito el tweet. Agregamos a este vector inicial un vector de 3 posiciones en función del número de retweets que tenga el tweet en cuestión, por ejemplo, en este momento la frontera está en 25 retweets y en 50. Añadimos un vector de una posición cuyo valor

puede ser 1 o 0 según si el tweet posee una foto o no, y por último añadimos un vector de 30 posiciones por el número de categorías que posee la herramienta, un total de 210 posiciones, las cuales pueden ser 0 o 1 dependiendo de que palabras posea el tweet y en relación a que categoría pertenezcan dichas palabras.

Nuestro vector final sería el siguiente.

[autores, retweets, foto, palabras de interés]

Por ejemplo, pongamos que tenemos un total de 5 autores, y 3 categorías posibles para clasificar, y que sacamos 5 palabras de interés en cada categoría. Los vectores resultantes para un tweet que posea una foto, haya sido publicado por el autor 3, cuente con 30 retweets y posea 3 palabras de interes de la categoría 1 y 2 de la categoría 2 será el siguiente.

- Autores → [0,0,1,0,0]
- Retweets → [0,1,0]
- Foto→ [1]
- Palabras de interés→ [1,1,0,1,0,1,0,0,1,0,0,0,0,0,0]

Concatenándolos todos da nuestro vector final:

[0,0,1,0,0, 0,1,0,1,1,1,0,1,0,1,0,0,1,0,0,0,0,0,0]

Estos vectores son los que finalmente le pasamos a nuestro clasificador, tanto para ajustar nuestro modelo como para clasificar los nuevos tweets que nos lleguen a la herramienta.

4.3 MÉTODO DE CLASIFICACIÓN.

Como se mencionó anteriormente, esta herramienta implementa diferentes "modalidades" de clasificación, en función de si queremos clasificar los tweets en una sola categoría o en varias, ambas siguen un funcionamiento parecido, cambiando sobre todo la clase de estimador que hemos usado.

4.3.1 Binario.

Esta “modalidad” es la utilizada para conseguir clasificar los tweets en una sola categoría diferenciada, es básicamente la que se usa cuando entramos en la vista de clasificación a una categoría y clicamos sobre cualquiera de ella.

Para esta clasificación lo que se hace básicamente es separar los tweets en dos grandes grupos: los pertenecientes a la categoría que tenemos que clasificar y lo que no lo son. Ésto disminuye mucho la complejidad y la evaluación de nuestro modelo, ya que corresponde a una clasificación binaria: o pertenece a la categoría en cuestión o no pertenece.

Nuestra misión, es crear un estimador que sea capaz de decidir si un tweet pertenece o no a una categoría en cuestión. Para esto en primer lugar, extraemos de todos los tweets guardados en nuestra base de dados el vector de características del que hemos hablado antes. Una vez lo hemos extraído y les hemos puesto el label en función del grupo al que pertenecen utilizamos el método fit de nuestro estimador para ajustar el modelo a los datos de entrenamiento.

Cuando ya hemos conseguido ajustar el modelo, ya tenemos nuestro clasificador y podemos empezar a clasificar tweets. Ésto se hace extrayendo los vectores de características de los nuevos tweets, y con el método predict del clasificador se comprueba el resultado.

Finalmente, lo que obtendremos serán unos tweets etiquetados tal y como lo habríamos hecho nosotros en función de en qué grupo nuestro modelo predice que estarán.

4.3.2 Multilabel.

Esta otra “modalidad” es algo más compleja que la anterior, ya que aumentamos el número de grupos posibles en los que son clasificados cada tweet, y además de esto le añadimos que un tweet va a poder pertenecer a varias clases simultáneamente, esto confiere una mayor complejidad a la hora de clasificar y a la hora de evaluar nuestro modelo ya que evaluar la precisión de un modelo multilabel es algo más complejo que un modelo binario.

Con el fin de mejorar la escalabilidad de nuestro sistema, y alejarnos lo mínimo posible de la técnica de clasificación binaria que hemos hecho antes, hemos usado la técnica llamada, OneVsAll. Esta técnica consiste en crear un modelo diferente para cada una de nuestras categorías, lo que produce básicamente modelos binarios como de los que hemos hablado antes, que únicamente son capaces de clasificar en dos grupos, si un tweet pertenece a su categoría en cuestión o no. Esta técnica consigue una mayor eficiencia y disminuir la complejidad del sistema. También permite predecir si un tweet pertenece a más de una categoría. Por ejemplo, un tweet puede ser tanto una noticia, como ser una noticia sobre tecnología, o estar hablando con un invento que ayudará a los deportes. Con esto se consigue una mayor precisión en nuestra herramienta de cara al usuario, ya que una herramienta que solo sea capaz de clasificar en una categoría, se vuelve en algunos casos insuficiente.

5 EVALUACIÓN DE RENDIMIENTO:

5.1 DATOS USADOS

Para el entrenamiento de nuestro modelo hemos usado un dataset como el siguiente:

- 708 tweets
- 7 categorías
- 6521 palabras extraídas
- Idioma: español.
- 58 Autores diferentes.

También se han desglosado los tweets en función de la categoría a la que pertenecen.

- Política 91
- Deportes 160
- Ciencia 80
- Tecnología 178
- Humor 33
- Entretenimiento 194
- Noticias 220

Este dataset resulta un poco escaso para conseguir una herramienta que obtuviese resultados realmente significativos, sobre todo por la parte de la cantidad de tweets usados y el número de palabras extraídas, pero resulta suficiente para mostrar el funcionamiento de la herramienta.

5.2 GRÁFICAS

En este capítulo veremos varias gráficas que tratan de mostrar el rendimiento obtenido de una forma más clara, para la generación de estas gráficas se hace uso del paquete matplotlib de Python. Éste paquete es capaz de representar cualquier conjunto de datos con un amplio abanico de formato y opciones de configuración. Muchas de estas opciones han sido usadas para este proyecto, como el cambio de estilo en las líneas, el crear un grid o el uso de los ejes en escala logarítmica.

Las gráficas que se verán más adelante en el capítulo se muestran en función de varios parámetros, los cuales han sido diferenciados en las gráficas por colores y estilos de línea.

Kernel.	**Binario. Métodos**	**Multilabel.Métodos**
Rbf → Azul	Roc_auc → sólido	F1_weighted → sólido
Linear →Rojo	Accuracy → dashed	Roc_auc → dashed

5.3 RENDIMIENTO

Una de los aspectos más importantes de una herramienta de machine learning es la eficacia del modelo que se construye, ya que no nos serviría de nada una herramienta muy buena que obtuviese un mal rendimiento.

Para comprobar la eficacia del modelo existe un método muy utilizado en estos casos, consiste en separar en varios lotes nuestros ejemplos de entrenamiento, y usar solo parte de ellos para entrenar nuestro modelo y otra parte para validarlo. La parte que se encarga de entrenar nuestro modelo se llama traning_set y la parte que se encarga de validar el modelo es el cross_validation. Con esto tratamos de evitar el problema del overfiting ya que tendremos que ajustar los parámetros en función del rendimiento que obtenemos en el cross_validation, aunque sin utilizar estos para entrenar el modelo. Por suerte, Scikit Learn ya tiene implementada varias funciones y clases que nos ayudan en la misión de evaluar nuestro modelo, para este propósito se ha utilizado la clase cross_validation.cross_val_score, y la grid_search.GridSearchCV de scikit_learn. Con estas dos clases se ha conseguido hallar la precisión de nuestro modelo de varias formas diferentes. La clase GridSearch utiliza diferentes parámetros con el fin de hallar el más adecuado para nuestro experimento.

Scikit-Learn nos ayuda de muchas maneras, una de ellas es utilizar un parámetro de las clases que se encargan de evaluar nuestro modelo, para especificar que método utilizar para calcular la precisión, y es que existen muchas, y cada una toma realmente significado en determinadas circunstancias.

Estas clases las hemos usado de la siguiente manera.

```
C_parameter = [1, 10, 100, 1000]
gamma_parameter = [1e-3]
parameters = [{'kernel': ['rbf'], 'gamma': gamma_parameter,
                     'C': C_parameter},
                    {'kernel': ['linear'], 'C': C_parameter}]
scores = ['roc_auc','accuracy']
for sc in scores:
    clasif = GridSearchCV(SVC(C=1), parameters, cv=5, scoring=sc)
```

Como se aprecia en el código se llama iterativamente a la función GridSearchCV, cambiando el método de cómputo, esta función es quien prueba cada uno de los parameters y nos indica el mejor valor.

Dada la gran diferencia de funcionamiento de nuestros estimadores, se han tenido que usar parámetros diferentes para computar la precisión según si estamos hablando del estimador binario o multilabel.

5.3.1 Clasificador Binario:

Para evaluar el rendimiento del clasificador binario se han empleado los métodos "área bajo la curva", y "accuracy". Además, para ver gráficamente el resultado, se han implementado gráficas donde representar los datos obtenidos.

5.3.1.1 Métodos

- Área Bajo la curva. En el caso del área bajo la curva, es un método bastante popular de computar el rendimiento y efectividad de un clasificador binario y multilabel, consiste en computar una curva que te ayuda a encontrar el clasificador óptimo para tu conjunto de datos, esta técnica es especialmente buena para los clasificadores binarios y multilabel.
- Accuracy. Este método consiste básicamente en comprobar si el resultado es exactamente el predicho, este método es relativamente bueno para clasificación binaria, pero terriblemente malo para una clasificación multilabel, ya que los clasificadores multilabel pueden clasificar en varias categorías de manera simultánea, entonces un tweet puede pertenecer a noticias y tecnología por ejemplo. Si nuestro clasificador lo clasifica como tecnología solo, este método detecta que ha tenido un error mientras que lo ha clasificado casi bien, es por

eso que otros métodos como el f1 que tiene en cuenta el número de falsos negativos y falsos positivos es más óptimo para esta tarea.

5.3.1.2 Gráficas

Se han generado varias gráficas para cada clasificador con el fin de mostrar más claramente los resultados obtenidos, cada una de las gráficas muestra el resultado para diferentes parámetros y diferente kernel, en el kernel rbf hemos mantenido gamma constante en 0.001 ya que se ha observado empíricamente que la herramienta funcionaba en todos los casos mejor con ese valor, para el clasificador binario hemos obtenido un total de 7 gráficas, una para cada categoría.

Ciencia

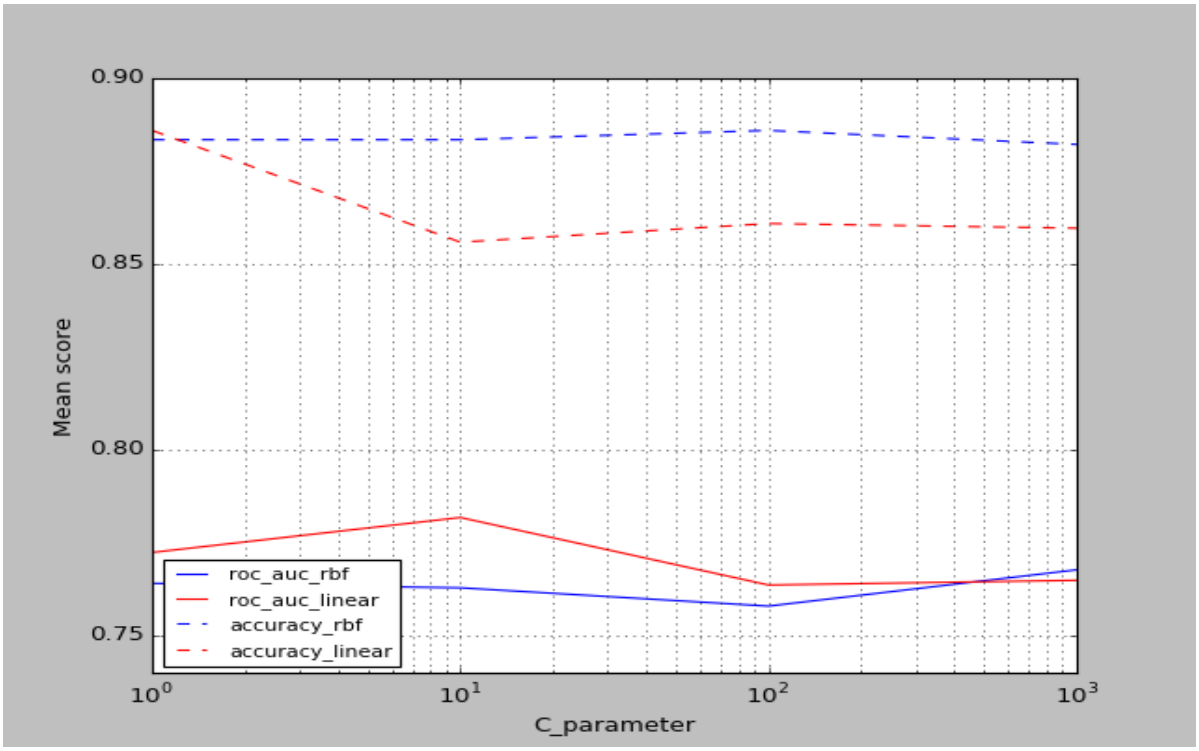

Figura 16 Gráfica Rendimiento Ciencia

Deportes

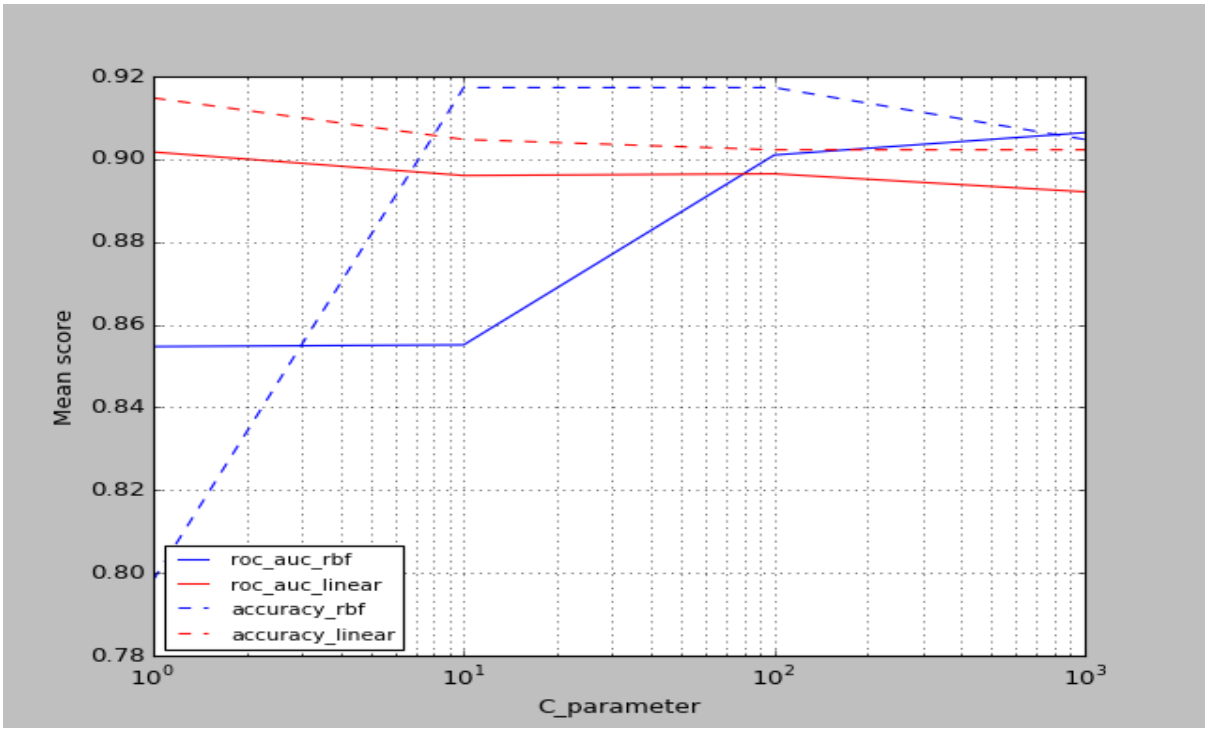

Figura 17 Gráfica Rendimiento Deportes

Entretenimiento

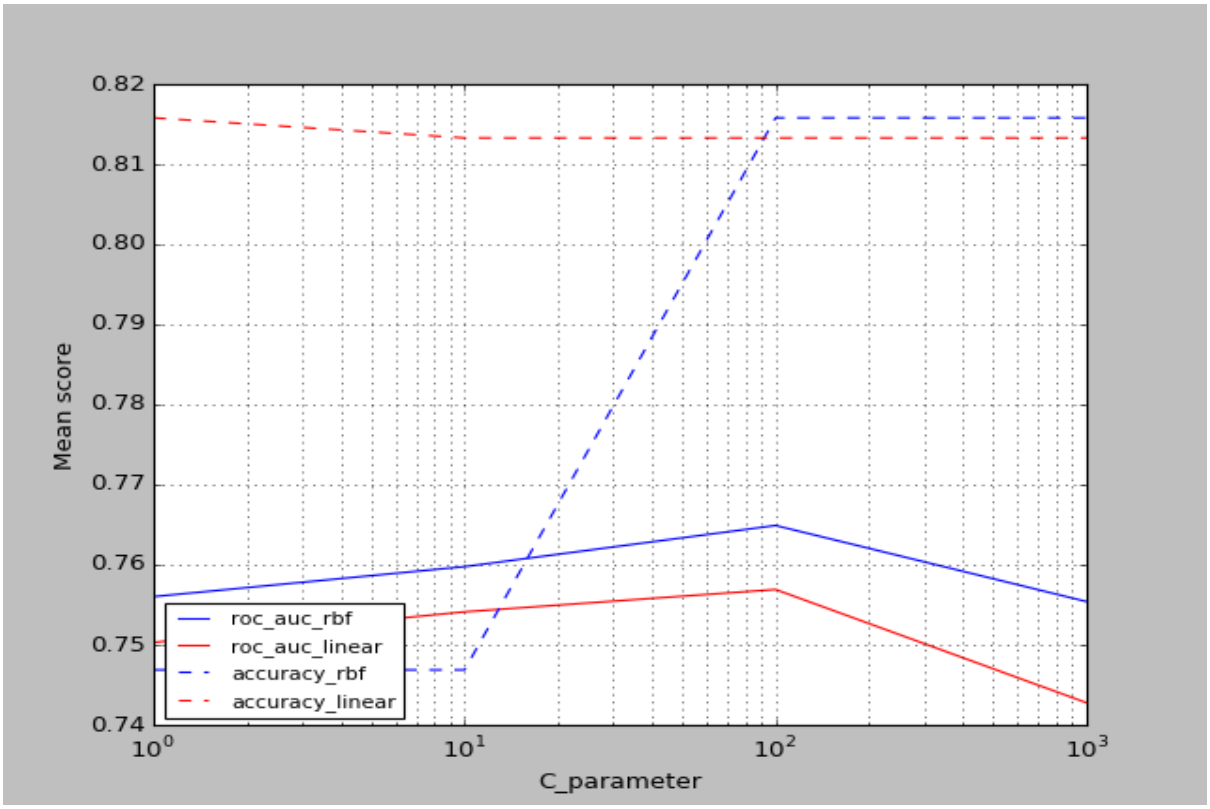

Figura 18 Gráfica Rendimiento Entretenimiento

Humor

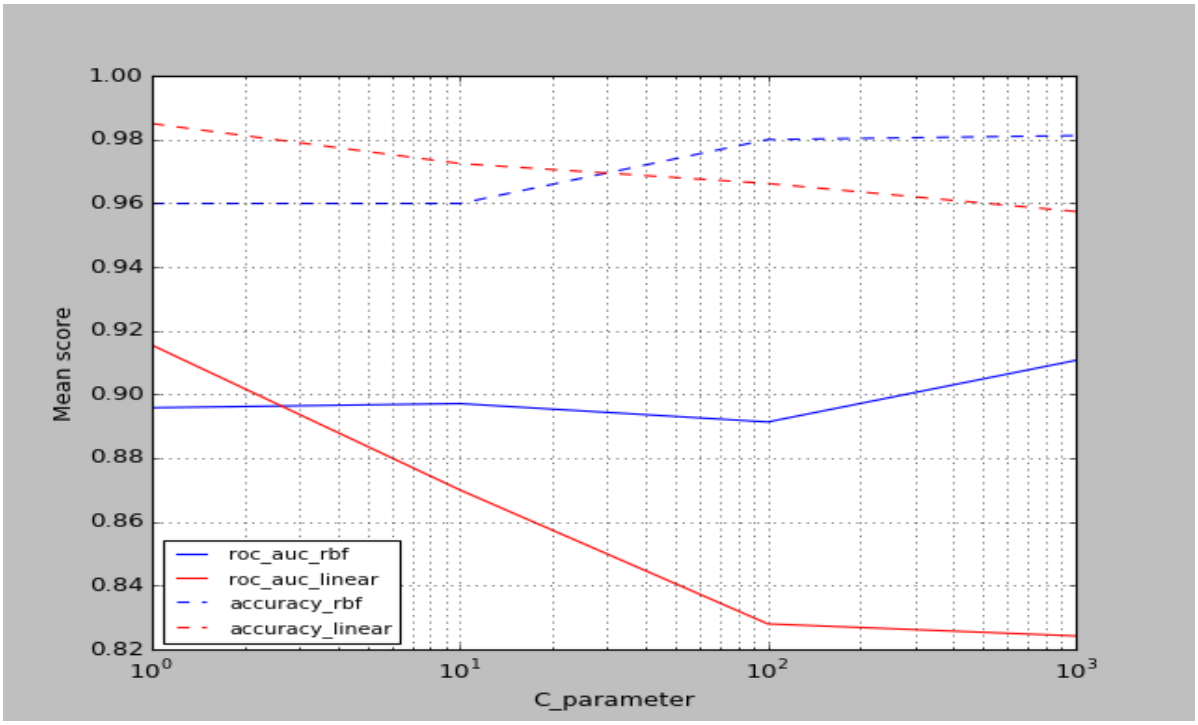

Figura 19 Gráfica Rendimiento Humor

Noticias.

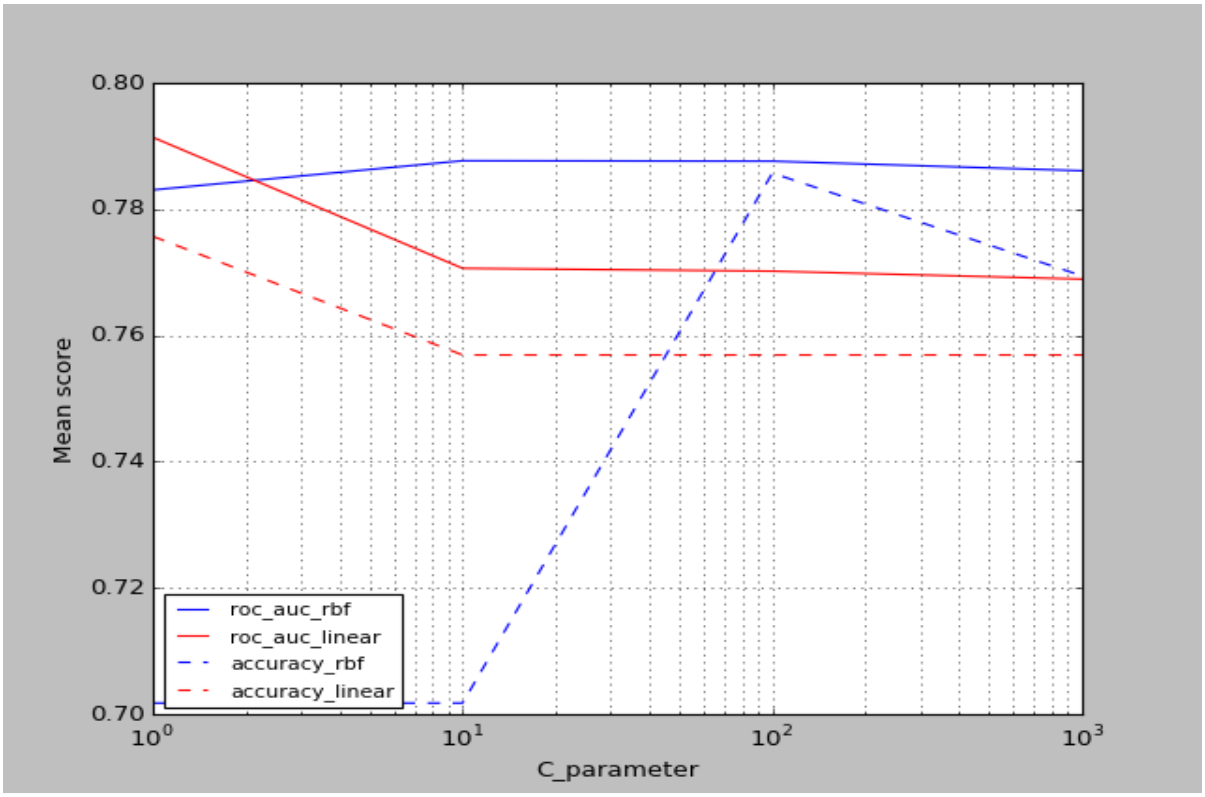

Figura 20 Gráfica Rendimiento Noticias

Política

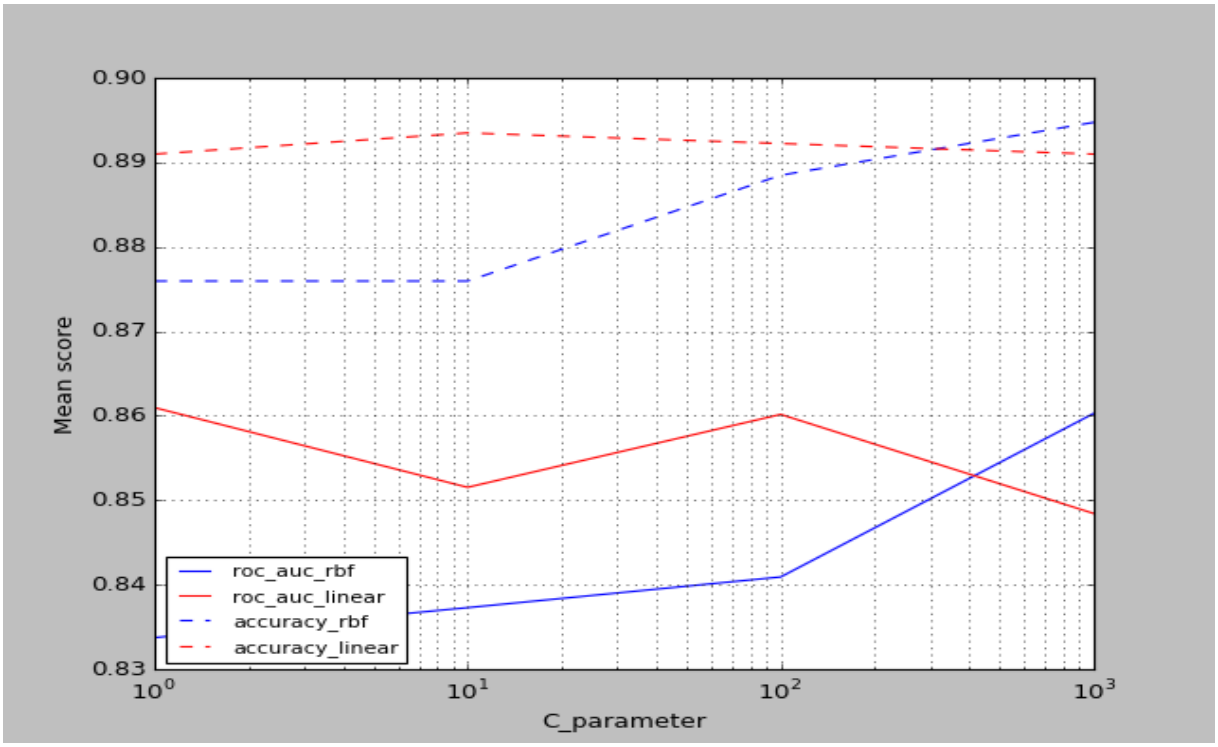

Figura 21 Gráfica Rendimiento Política

Tecnología

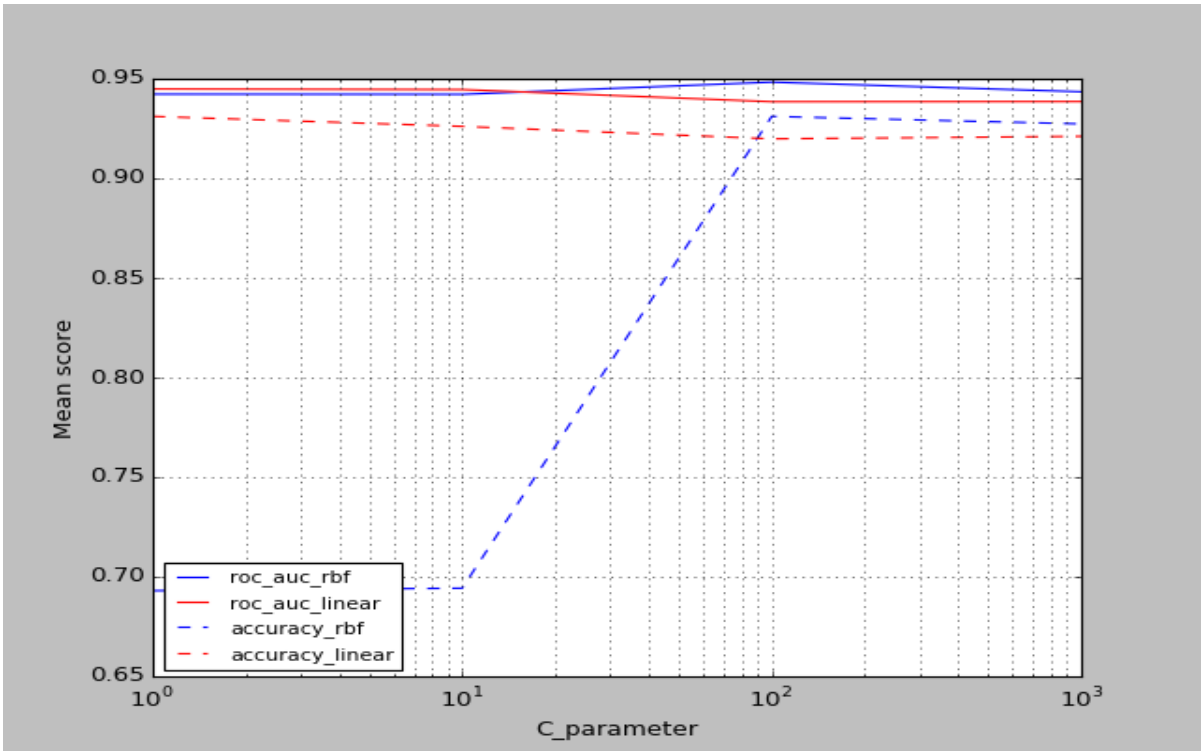

Figura 22 Gráfica Rendimiento Tecnología

5.3.2 Clasificador Multilabel:

Para el clasificador multilabel se tienen que emplear métodos diferentes para evaluar su rendimiento, estos son los métodos de área bajo la curva y el f1. Además de la gráfica correspondiente para evaluarlo.

5.3.2.1 Métodos.

- El método f1 es bastante utilizado para obtener la eficacia de un clasificador multi-clase y multilabel se sirve de la siguiente fórmula para obtener la precisión:

  ```
  F1 = 2 * (precision * recall) / (precision + recall)
  ```

  Precision = verdaderos positivos / (verdaderos positivos + falsos positivos)

  Recall = verdaderos positivos / (verdaderos positivos + falsos negativos)

  Nosotros estamos usándolo en su variante weighted esto le indica a scikit que tiene que calcular esa medida para cada una de las clases por separado y finalmente hacer la media.

- El método roc_auc también ha sido probado para este clasificador ya que también obtiene resultados bastante fiables para él, y poder contrastar los resultados de un mismo método en dos clasificadores diferentes.

5.3.2.2 Gráficas

El estimador multilabel al computar todas las clases de manera conjunta, solamente posee una gráfica que indique su rendimiento, para la clasificación el kernel gaussiano tomamos el mismo gamma que para el clasificador binario, 1e-3.

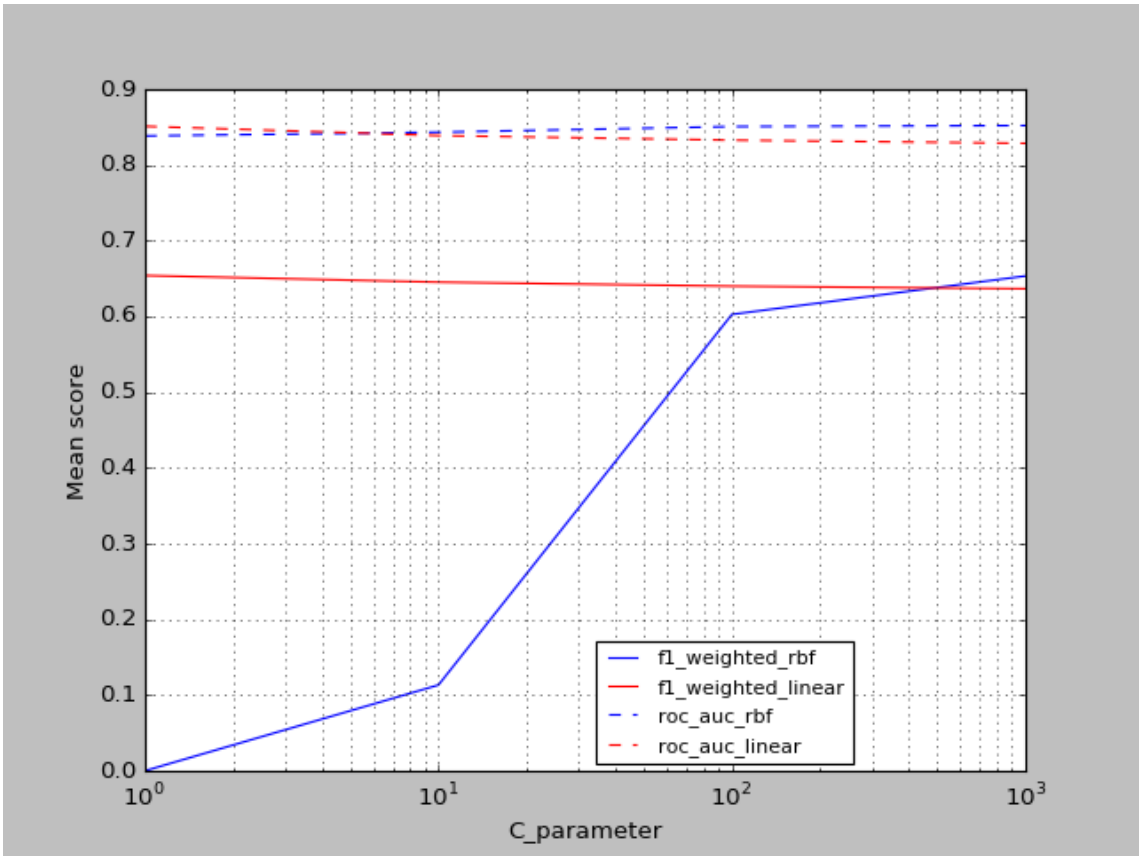

Figura 23 Gráfica Rendimiento Multilabel

5.4 CONCLUSIONES

Se ha podido apreciar en todas las gráficas, que el modo de obtener el rendimiento varía mucho para determinar si un modelo es bueno o no, es por eso que elegir un buen método es imprescindible.

En el clasificador binario, la gran mayoría de pruebas son bastante satisfactorias la mayoría con un rendimiento superior del 80 % de acierto y algunas rondando el 94%, aunque hay significativas diferencias entre una categoría y otra, esto se puede deber a las diferencias que existen entre el número de tweets de cada uno de los ejemplos de entrenamiento, ya que una baja cantidad de tweets y de palabras interesantes produce un bajo rendimiento de nuestro modelo, pero para casos como el de la categoría de noticias, la cual posee un número bastante amplio de tweet y sigue teniendo un bajo rendimiento, se puede deber a la gran subjetividad de la categoría, ya que es muy difícil decir que puede ser una noticia o algo que no nos parece tan relevante.

En el clasificador multilabel se puede apreciar un rendimiento mucho más bajo que en el clasificador binario, esto se debe principalmente a la diferencia de complejidad entre ambos, además de resultar más difícil estimar una buena precisión del clasificador multilabel. A esto se le une la escasa cantidad de tweets que teníamos a nuestra disposición para ejecutar el entrenamiento, ya que para conseguir ejecutar un algoritmo de este tipo es recomendable utilizar un número mucho más amplio de tweets, y un corpus en el que aplicar las técnicas de text mining mucho más grande.

6 CONCLUSIÓN Y LÍNEAS FUTURAS

Finalmente, se han conseguido abordar y finalizar satisfactoriamente todos los objetivos puestos al principio del proyecto, creando una herramienta gráfica la cual es capaz de comunicarse con una red social, en este caso twitter, y clasificar su contenido en función de las necesidades del usuario.

Esta herramienta muestra como el machine learning nos ayuda a construir soluciones a medida para las necesidades del usuario.

El proyecto inicia una base sobre la cual se pueden crear numerosas líneas de trabajo.

- Aumentar el rendimiento de la herramienta en la clasificación multilabel
- Utilizar diversos algoritmos de machine learning, combinando por ejemplo la clasificación y la recomendación, con el fin de dar una experiencia más personalizada.
- Un mejor soporte multiusuario por parte de la herramienta.
- Soporte multi-red social, donde cada usuario podría tener clasificada la información de más de una red social diferente.
- Traspaso de la herramienta a una posible versión móvil.

7 BIBLIOGRAFÍA

http://scikit-learn.org/

https://docs.djangoproject.com/en/1.9/

http://tweepy.readthedocs.io/en/v3.5.0/

https://dev.twitter.com/overview/documentation

http://www.nltk.org/

http://stackoverflow.com/

http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/

https://marcobonzanini.com/2015/05/17/mining-twitter-data-with-python-part-6-sentiment-analysis-basics/

http://pybonacci.org/2015/11/24/como-hacer-analisis-de-sentimiento-en-espanol-2/

http://pybonacci.org/2015/04/06/introduccion-a-machine-learning-con-python-parte-2/

http://pybonacci.org/2015/11/16/dibujando-100k-tweets-de-mi-ciudad/

https://pythonprogramming.net/sklearn-scikit-learn-nltk-tutorial/

https://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine

https://en.wikipedia.org/wiki/Support_vector_machine

Building Machine Learning Systems with Python - Luis Pedro Coelho, Willie Ritchie