TRABAJO FIN DE GRADO

INGENIERÍA EN INFORMÁTICA

OpinionMOOC

La retoalimentación del profesor virtual

Autor

José Antonio Ruiz Millán

Directores

María Victoria Luzón García

Eugenio Martínez Cámara

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada , 27 de agosto de 2019

OpinionMOOC

La retoalimentación del profesor virtual.

Autor
José Antonio Ruiz Millán

Directores
María Victoria Luzón García
Eugenio Martínez Cámara

OpinionMOOC, la retoalimentación del profesor virtual

José Antonio Ruiz Millán

Resumen

Tenemos un problema de clasificación supervisada para análisis de textos, concretamente, queremos clasificar el grado de satisfacción de los usuarios/alumnos que realizan un curso online masivo (MOOC), teniendo que diferenciar entre 6 clases distintas que indican niveles distintos de satisfacción.

El objetivo principal es automatizar el proceso humano de revisar cada uno de los comentarios que los alumnos aportan para poder saber si el curso realizado ha tenido éxito o no dependiendo de cómo se sienten los alumnos con el curso. Claramente éste proceso es lento y con la automatización del mismo podemos saber si un curso está siendo bien o mal recibido por los alumnos, incluso usarlo para mejorar el servicio que se ofrece a los alumnos y así mejorar su aprendizaje.

Para ello se ha realizado un ejercicio de anotación con un conjunto de datos proporcionado por AbiertaUGR. Una vez medido y aceptado el acuerdo del ejercicio de anotación, pasamos a el estudio del problema, entender los datos y entender qué es lo que nos quieren decir. La base de datos proporcionada es algo escasa y no generaliza correctamente, por lo que algunos experimentos no muestran un grado muy elevado de acierto.

Estuvimos probando distintas configuraciones debido a los problemas que los datos nos propusieron, finalmente, decidimos diseñar 5 casos distintos de combinación de los datos, utilizando 3 algoritmos para su clasificación y experimentación. Añadimos un caso base y un caso techo basados en casos simples para las comparaciones directas de estos casos diseñados.

Para poder ejecutar estos casos, se realizó un estudio sobre procesamiento de texto, lo que permite transformar el texto en información útil para los algoritmos de aprendizaje. Una vez transformado el texto, se procesa para extraer la mejor información y se entrenan los distintos algoritmos.

Finalmente ejecutamos los distintos casos de experimentación que diseñamos anteriormente y se analizan los resultados para entenderlos con claridad y facilidad, pudiendo visualizar cómo han funcionado los algoritmos y analizando distintas métricas que nos indican cómo un algoritmo se ha adaptado a los datos y a la resolución del problema.

OpinionMOOC, the feedback of the virtual teacher

José Antonio, Ruiz Millán

Abstract

We have a problem of supervised classification for text analysis, specifically, we want to classify the degree of satisfaction of users/students who perform a massive online course (MOOC), having to differentiate between 6 different classes that indicate different levels of satisfaction.

The main objective is to automate the human process of reviewing each of the comments that the students provide in order to know if the course has been successful or not depending on how the students feel about the course. Clearly this process is slow and with the automation of it we can know if a course is being well or poorly received by the students, even using it to improve the service offered to the students and thus improve their learning.

To do this, an annotation exercise was performed with a set of data provided by AbiertaUGR. Once the agreement of the annotation exercise has been measured and accepted, we turn to the study of the problem, understand the data and understand what they want to tell us. The database provided is somewhat scarce and does not generalize correctly, so some experiments do not show a very high degree of success.

We were testing different configurations due to the problems that the data proposed, finally, we decided to design 5 different cases of data combination, using 3 algorithms for their classification and experimentation. We add a base case and a ceiling case based on simple cases for direct comparisons of these designed cases.

In order to execute these cases, a study on text processing was carried out, which allows to transform the text into useful information for the learning algorithms. Once the text has been transformed, it is processed to extract the best information and the different algorithms are trained with that data.

Finally we execute the different experimentation cases that we designed previously and analyze the results to understand them clearly and easily, being able to visualize how the algorithms have worked and analyzing different metrics that tell us how an algorithm has adapted to the data and the resolution of the problem .

Índice general

Capítulo 1

Introducción

1.1. Motivación

Con la expansión de Internet, los usuarios tienden cada vez más a usar la red mundial para satisfacer sus necesidades de conocimientos y, debido a la naturaleza de la Web 2.0, en muchos de los casos son los mismos usuarios los que responden a las dudas e inquietudes de otros, ya sea mediante las redes sociales, los foros, las páginas de preguntas o publicando un tutorial en Youtube.

Estos medios han proporcionado la aparición de cursos pensados para un gran público: los MOOC. Un fenómeno que algunos autores [1] creen es la mayor revolución en materia de educación y que supondrá el primero de muchos cambios en la calidad de la educación. Al registrarse el proceso de aprendizaje del estudiante el sistema podrá adaptarse a sus necesidades y capacidades y, mediante el análisis de los datos, se podrían mejorar los métodos de enseñanza con el fin de lograr mejores resultados académicos.

Un MOOC (acrónimo en inglés de Massive Open Online Course - curso online masivo abierto), es un curso a distancia en línea dirigido a un amplio número de participantes a través de Internet según el principio de educación abierta y masiva, [5].

AbiertaUGR es el nombre de la plataforma de formación abierta de la Universidad de Granada que ofrece formación de coste gratuito y abierto a cualquier persona interesada. En esta plataforma, los cursos se organizan en torno al seguimiento de los contenidos y actividades propuestos en un rango temporal determinado, con participación activa online y el apoyo de tutores para facilitar el proceso de aprendizaje.

AbiertaUGR ofrece cursos variados:

- La Alhambra: historia, arte y patrimonio. Curso sobre el conjunto monumental e histórico de la Alhambra en el que se analiza su historia, sus procesos constructivos, sus valores estéticos y las razones de su conservación.
- Currículum 2.0. Curso sobre las herramientas de internet para mejorar nuestra inserción laboral.
- Aprendizaje Ubicuo. Curso abierto, gratuito y en línea sobre nuevas formas de aprendizaje basado en TIC y en dispositivos móviles.
- Creative Commons. Curso organizado por el Centro de Enseñanzas Virtuales para conocer y experimentar la difusión de contenidos en abierto.

Desde el punto de vista docente, la plataforma AbiertaUGR proporciona una herramienta muy positiva para los profesores, y se trata de que los estudiantes puedan publicar sus comentarios y opiniones sobre las asignaturas y los propios cursos. Este contenido es de vital importancia para la gestión docente de los cursos, así como para identificar sus fortalezas y debilidades, con la mira siempre puesta en su mejora para ofrecer el mejor contenido docente a los estudiantes y el más adecuado proceso de aprendizaje.

El elevado número de comentarios, y el limitado tiempo del que suelen disponer los docentes y los gestores universitarios, obliga al estudio y desarrollo de herramientas que automaticen el análisis de esos comentarios, y muestre el resultado de dicho análisis de una forma simple y comprensible en el menor tiempo posible. El objetivo de dichas herramientas es el incremento de la productividad de los docentes y los gestores académicos, además de facilitar y reducir la incertidumbre de la toma de decisiones académicas.

1.2. Tema de estudio

El análisis automático de contenido es posible, y el área responsable de ello en el contexto de la Inteligencia Artificial es el Procesamiento del Lenguaje Natural. En concreto, el problema al que nos enfrentamos es la clasificación automática de opiniones y la posterior presentación de los resultados. La tarea que se encarga de ello es el Análisis de Opiniones, que se define como la tarea responsable del tratamiento computacional de la opinión, sentimiento y subjetividad de textos [4]. Aunque la tarea no está resuelta aún, la literatura sobre ella es más que amplia, y muestra de ello son los trabajos [3] [4] y [6].

La interpretación automática del sentido o polaridad de la opinión es una tarea muy compleja, a la que además se le añade su dependencia del dominio

del que se circunscriba la opinión. En nuestro caso de opiniones sobre los cursos y asignaturas de MOOC, el problema de la adaptación al dominio es evidente, dado que cada curso, e incluso cada asignatura de un mismo curso, puede presentar un uso del lenguaje diferente, que obligue adaptar al sistema de identificación de la opinión.

Resumiendo, los problemas que se estudiarán son:

1. Clasificación automática de opiniones.
2. Visualización comprensiva de los resultados.
3. Adaptación al dominio de clasificadores de opinión.

1.3. Objetivos

Los objeticos que se marca este Trabajo Fin de Grado son:

1. Estudiar el estado del arte de clasificación de la opinión, en especial la clasificación de la opinión en español.
2. Estudiar el proceso a seguir para anotar unos datos lingüísticos y cómo hacerlo.
3. Estudiar distintos tipos de métodos para evaluar la calidad de anotación de los datos, haciendo uso de alguno de ellos para evaluar nuestra propia anotación.
4. Desarrollar un clasificador de opiniones sobre un curso específico de la plataforma AbiertaUGR.
5. Visualizar de la forma más comprensible posible el resultado de la clasificación para facilitar la toma de decisiones.
6. Desarrollar técnicas de adaptación al dominio del sistema desarrollado.

1.4. Hipótesis

La hipótesis que da pie al presente Trabajo Fin de Grado es la siguiente: *A partir de un conjunto de comentarios sobre un MOOC es posible el análisis y clasificación automática de la orientación de la opinión que los estudiantes tienen sobre dicho MOOC.*

La planificación del trabajo va a estar dirigida por el método científico, de manera que se le propondrá al estudiante el desarrollo de las siguientes tareas:

1. Estudio de la bibliografía relacionada con el problema.
2. Compilación de datos.
3. Anotación de datos.
4. Propuesta de un clasificador.
5. Evaluación de la propuesta.
6. Análisis de errores.
7. Publicación de los resultados.

Las tareas 4, 5 y 6 constituyen un proceso iterativo, que se detendrá cuando los resultados obtenidos superen el estado del arte en el problema al que se trata de contribuir con una solución.

Capítulo 2

Especificación Requisitos

2.1. Nuestro problema

En estos momentos, con el avance tan rápido de la tecnología, hemos llegado a un nivel en el que el uso de Internet es prácticamente necesario para todas las empresas y personas en el mundo. Esto hace que aparezcan servicios que permiten a través de Internet, aprender y mejorar el estilo de vida sin la necesidad de desplazarse o moverse de tu habitación.

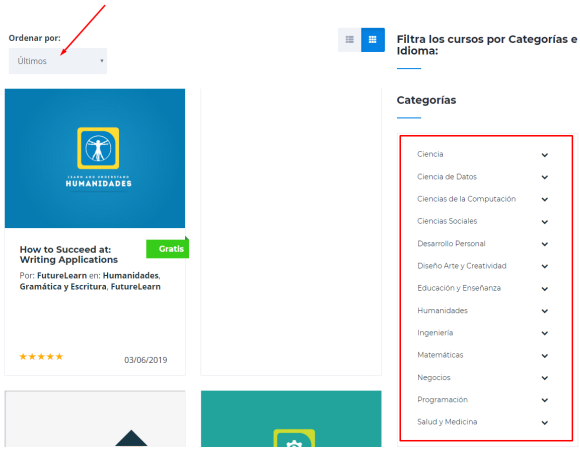
Estos medios han proporcionado la aparición de los MOOC entre otras muchas cosas. Los MOOC, un fenómeno que algunos autores [1] creen es la mayor revolución en materia de educación que supondrá el primero de muchos cambios en la calidad de la educación.

2.1.1. Curso online masivo abierto (MOOC):

Un MOOC (acrónimo en inglés de Massive Open Online Course - curso online masivo abierto), es un curso a distancia en línea dirigido a un amplio número de participantes a través de Internet según el principio de educación abierta y masiva, [5].

Estos cursos almacenan la información de los usuarios que los realizan para así mejorar y poder ofrecerle al usuario la atención adecuada y precisa dependiendo de sus necesidades en cada momento. Con esto, se puede mejorar el rendimiento y la calidad de los resultados obtenidos por el usuario haciendo que la educación mejore en térnminos generales debido a la mejora individual de cada uno de los usuarios.

Podemos acceder por ejemplo a *mooc.es*:



Esta web (entre muchas otras) ofrece cursos que como podemos ver pueden ser filtrados por categorías u ordenados por algún criterio concreto que nos interese. La facilidad con la que podemos buscar y seleccionar un curso para nuestro aprendizaje es tan grande que permite a cualquier usuario independientemente de su experiencia con Internet, poder realizarlos y comenzar a mejorar sus conocimientos tan rápido y fácil como haciendo un simple click.

2.1.2. Historia:

En 1988, Isaac Asimov expresaba la aparición de los MOOC en el vídeo titulado *"Su visión hacia el futuro"*, pero ya anteriormente, en 1922, se produjo un fenómeno similar con los cursos radiofónicos cuyas pretensiones, planteamientos pedagógicos, objetivos económicos y visión de futuro coincidían con los de los MOOC y en particular los xMOOC [9].

Como antecedentes de los cursos MOOC se pueden considerar, entre otras, a las iniciativas OpenCourseWare (OCW) y los recursos educativos abiertos de la UNESCO (en inglés se emplea la sigla OER).

Realmente no podemos otorgar una fecha exacta a la existencia del primer MOOC, pues la discusión es intensa al respecto, y está sujeta a las características que definen un curso como MOOC. Algunos opinan que David Wiley es el autor del primer MOOC en la Universidad Estatal de Utah en agosto de 2007. Se trataba de un curso de educación abierta. Esta iniciativa tuvo continuidad en numerosos proyectos académicos dentro y fuera de los Estados Unidos, pero también existen algunos cursos previos a este que podrían considerarse MOOC, como el curso *"From NAND to Tetris: Building a Modern Computer from First Principles"*, organizado por Noam Nisan y Shimon Schocken en el año 2005 (Shimon Schocken: The self-organizing computer course) [10].

El primer MOOC fue ofrecido en el año 2008 en la universidad de Manitoba, Canadá (Fini, 2009). Stephen Downes y George Siemens abrieron al público un curso en línea llamado *"Connectivism and Connective Knowledge CCK08"* donde se inscribieron 2.300 estudiantes, Dave Cormier y Bryan Alexander sugirieron llamarlo *"Massive Open Online Course"* o MOOC (Siemens, 2012).

En 2011, cuando más de 160.000 [11] personas repartidas por todo el mundo se matricularon en un curso de Inteligencia artificial (*"Introduction to Artificial Intelligence"*) ofrecido por Sebastian Thrun y Peter Norvig en la Universidad de Stanford a través de una compañía startup llamada Know Labs. En Francia, Open Classrooms, que va ofreciendo cursos en línea en IT y lenguajes de programación desde 1999, empieza produciendo MOOC en 2012. Hoy en día, ofrece más de 1000 cursos enfocados en tecnología y competencias digitales, principalmente en francés, pero también en inglés y en castellano, producidos internamente o en colaboración con universidades y empresas.

En 2012 se reconoce internacionalmente el auge de los MOOC. El periódico New York Times publicó un artículo titulado *"The Year of the MOOC"*, o *"El año del MOOC"* [12]. En él se declaraba que el 2012 había sido el año de los *Massive Open Online Courses* (MOOC) debido a la gran expectación que había obtenido este nuevo término por parte de los medios de comunicación y la comunidad educativa mundial [12]. Ese mismo año, Stanford y el Instituto Tecnológico de Massachuetts desarrollaron plataformas para impartir sus propios MOOC, Coursera y edX.

España es, desde 2013, y según la Unión Europea, el primer país europeo en la producción de cursos MOOC. Un 27 % de los mismos [13], seguido de Reino Unido convirtiéndose ambos en los países de Europa al frente esta [14]. La cantidad de cursos en línea abiertos masivos que se ofrecen aumentó en el año 2017 a 9.400, frente a los 6.850 de 2016 [15].

2.1.3. Clasificación de los MOOC

Los MOOC pueden dividirse en función del origen, ciclo de vida y finalidad de su material educativo, [7]:

- xMOOC: Estos son los MOOC en los que el material, normalmente vídeos complementados por artículos de texto, se pueden descargar y estudiar de forma «off-line». Estos MOOC suelen tener apoyo de foros para que los usuario puedan interactuar, pero esta interacción no es esencial, ni forma parte de la experiencia del curso.
- cMOOC: En estos cursos, el material proporcionado puede (y debe) ser usado, extendido o propuesto para otro uso distinto al original por

parte de los estudiantes. Estos amplían el contenido del curso mediante entradas a blogs, tweets, podcasts, etc.

En nuestro caso de estudio, tenemos un MOOC que claramente entra dentro de la descripción de xMOOC, pero la tendencia es que los MOOC promuevan cada vez más la comunicación y participación de los estudiantes.

2.1.4. AbiertaUGR

AbiertaUGR es el nombre de la plataforma de formación abierta de la Universidad de Granada que ofrece formación de coste gratuito y abierto a cualquier persona interesada. Ésta será la plataforma que nos brindará los datos y con la que realizaremos el estudio.

Esta plataforma ofrece una diversidad de cursos que podemos realizar cualquiera de nosotros. Éstos cursos nos dan toda la documentación necesaria ya sea a través de ficheros (generalmente PDF) o ya sea través de vídeos. También disponemos de profesores/tutores que nos pueden ayudar a resolver nuestras dudas y cuestiones en los foros de los cursos al igual que cualquier usuario que realice el curso.

Cada curso se divide en distintos módulos para los cuales tendremos que completar cuestionarios y/o actividades relacionadas con dicho módulo para superarlo.

Al finalizar algunos de los cursos, se puede realizar una encuesta de satisfacción, en la cual aparecen algunas preguntas abiertas en las que el usuario puede expresar su nivel de satisfacción con el curso. Éstas preguntas abiertas son nuestros datos, con los que trabajaremos para estudiarlos y poder clasificar el estado de satisfacción de cada uno de los usuarios con el curso realizado.

Algunos ejemplos de cursos ofrecidos en este momento por AbiertaUGR son:

- **La Alhambra: Historia, arte y patrimonio 5ª Edición**

  Este proyecto didáctico tiene como finalidad profundizar en el conocimiento de la Alhambra. Se analizará su historia, los procesos constructivos, sus valores estéticos y las razones de su conservación.

- **MOOC UGR sobre Sierra Nevada**

  Este proyecto didáctico tiene como finalidad profundizar en el conocimiento de Sierra Nevada. Se analizará su historia, las infraestructuras históricas y explotación económica, su medio físico y biológico, sus paisajes, veremos a Sierra Nevada como espacio protegido, así como disfrutar de ella mediante el deporte.

- **Federico García Lorca**

  Este curso pretende ser un recorrido completo por la obra y la trayectoria de Federico García Lorca. Se estudiará su obra poética y teatral, así como las características generales del periodo literario y cultural en el que se desarrolla su producción. Los seis módulos en que se divide, ordenados de manera cronológica, explorarán toda la obra de García Lorca, desarrollándose a lo largo de seis semanas.

- **Emprende: Convierte tu idea en un modelo de negocio**

  El curso analiza los principales elementos que definen a un modelo de negocio, es decir, los que hacen que una empresa pueda crear, proporcionar y captar valor. A lo largo de 6 módulos se desarrolla la herramienta denominada Lienzo del Modelo de Negocio o Business Model Canvas. Guiados por los técnicos de UGR Emprendedora, semana tras semana, el participante podrá ir configurando su idea de negocio en torno a nueve elementos clave: segmento de clientes, proposición de valor, canales con clientes, relaciones con clientes, actividades clave, recursos clave, alianzas estratégicas, costes e ingresos. En cada módulo hay un ejemplo de emprendedores para ilustrar de manera práctica los temas tratados. Durante todo el Mooc se puede acceder a un contenido transversal para motivar e inspirar en la aventura de emprender y no perder el rumbo.

- **MOOC de Software Libre**

  El curso pretende llegar a toda la comunidad, ya sea una persona con un nivel aceptable a la hora de usar las nuevas tecnologías, o no, puesto que la metodología es sencilla y los contenidos no requieren ningún nivel técnico anterior, puesto que se utiliza un lenguaje llano y

accesible en todas sus partes. Lo único necesario para seguir este curso es tener ganas de aprender qué rodea el Software Libre, ya sea como el vehículo que permite hacer ciencia abierta, como la herramienta que se debería utilizar para educar en todos los niveles.

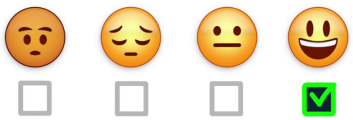
- **Aprendizaje Ubicuo**

  Este es un curso abierto, gratuito y en línea (MOOC) sobre nuevas formas de aprendizaje basado en TIC y en dispositivos móviles / tablets. Este curso de "Aprendizaje Ubicuo" organizado por el Centro de Producción de Recursos para la Universidad Digital se divide en cuatro módulos a lo largo de cuatro semanas.

2.2. Proceso automático de clasificación de opinión

Llegados a este punto, podemos preguntarnos, ¿Para qué tener información sobre el nivel de satisfacción de los alumnos?, ¿De qué nos puede servir?, ¿Cómo lo tratamos? entre otras muchas preguntas.

Como en todos los sectores, la satisfacción del cliente siempre debe ser la mejor esperada, por lo que ésta información es indispensable para los desarrolladores de los cursos en nuestro caso. Gracias a ella nos puede facilitar un orientación sobre los cursos con más éxito o con menos éxito entre otras muchas características. Un ejemplo cualquiera puede ser a la hora de comprar cualquier tipo de producto a través de una web online, si tenemos dos productos prácticamente idénticos en precio y ambos nos ofrecen lo que estamos buscando y lo que necesitamos, siempre vamos a elegir el que mejor valoración tenga por parte del resto de usuarios. Por ello, tener ésta información de una forma automática nos facilitará mucho este tipo de estadísticas.



Y ahora nos preguntamos, si tengo miles o millones de alumnos, ¿Cómo puedo a partir de un comentario abierto escrito por el alumno clasificar su nivel de satisfacción?, ¿Tengo que revisar 1 a 1 las encuestas?, gracias a nuestro trabajo y el estudio que vamos a realizar, este proceso que sería tan pesado y lento de realizar ya que tendríamos que revisar miles o millones de encuestas para poder categorizar todos los comentarios, se hará de forma automática y rápida.

Esto permitirá a los desarrolladores de los cursos poder revisar la satisfacción de todos los alumnos en sus cursos de forma fácil, cómoda y automática sin tener que revisar todas las encuentras de satisfacción realizadas. Este proceso categorizarán todas las opiniones de los alumnos de forma automática siguiendo un criterio de anotación determinado que para nuestro caso, veremos en capítulos posteriores.

2.3. Datos

Como no puede ser de otra forma, para poder resolver este problema, necesitamos datos de los cuales aprender y poder basarnos en ellos para así categorizar los futuros comentarios de los nuevos alumnos. En nuestro caso, el conjunto de datos del que partimos ha sido proporcionado por AbiertaUGR, siendo éste un conjunto de datos compuesto por 2 preguntas distintas, las cuales se corresponden con un comentario abierto.

El problema que tenemos es claramente de clasificación supervisada, es decir, necesitamos que los comentarios que nos proporcionan estén etiquetados siguiendo algún criterio. La estructura de datos de los que disponemos son los siguientes:

1. **Pregunta/entidad Q13**

   Conjunto compuesto de 226 elementos que refleja la respuesta de los alumnos en la que expresan su nivel de satisfacción antes de realizar el curso y así poder ver como se encuentran cada uno de ellos.

2. **Pregunta/entidad Q14**

   Conjunto compuesta de 68 elementos que refleja la respuesta de los alumnos en la que valoran la encuesta realizada, es decir, expresen su nivel de satisfacción respecto a la propia encuesta que acaban de realizar.

Tanto Q13 como Q14 tienen el mismo formato, compuesto por los siguientes atributos:

- **id:** Identificador único del comentario indicando al curso al que pertenece la encuesta.
- **label:** Texto escrito por el alumno en respuesta a la pregunta concreta.
- **y:** Valor de clasificación del comentario después de haber seguido un proceso de anotación que veremos en siguientes secciones.

Este conjunto es el conjunto de datos con el que trabajaremos en este estudio. No obstante, se hará un estudio más intenso de los datos en secciones posteriores ya que ahora únicamente hemos hecho una introducción a los mismos para poder explicar el proceso de anotación que hemos seguido.

2.3.1. Guía de anotación

Partimos de un conjunto de datos que se divide en dos partes (dos entidades o preguntas). Una primera pregunta (Q13) que nos ofrece respuesta sobre el nivel de satisfacción del usuario al comienzo del curso, y por otro lado una segunda pregunta (Q14) que ofrece una respuesta del usuario indicando el nivel de satisfacción con la propia encuesta.

Tenemos 6 grados de satisfacción, enumerados desde el 5 al 0, siendo el 5 el valor que indica el máximo nivel de satisfacción y el 1 el que indica el menor nivel de satisfacción. Hacemos uso de la etiqueta 0 para catalogar los comentarios que no indican un nivel de satisfacción. Mostraremos ejemplos sobre cada uno de los casos, indicando a la entidad a la que pertenecen siendo **E.1** la entidad que se refiere al estado de ánimo del usuario antes del curso y **E.2** la entidad que se refiere a la encuesta:

- **Grado (clase) 5**

  Comentarios totalmente positivos, mostrando un entusiasmo alto respecto a su estado actual.

  - **E.1** - "Muy contenta y con ganas de comenzar el curso"
  - **E.2** - "Muy sencilla y fácil"

- **Grado (clase) 4**

  Comentarios positivos que pueden tener algún aspecto no positivo pero el comentario generalizado es positivo. Comentarios positivos sin mostrar un excesivo entusiasmo.

  - **E.1** - "Animada por aprender cosas nuevas"
  - **E.2** - "Me parece curiosa cuanto menos"

- **Grado (clase) 3**

  Comentarios neutros, tienen parte positiva y negativa a partes iguales. No se puede definir con exactitud si muestran una satisfacción positiva o negativa.

  - **E.1** - "Pues con ilusión aunque creo que va a ser muy duro"
  - **E.2** - "Parece interesante, aunque en algunos aspectos se hace pesada."

- **Grado (clase) 2**

  Comentarios negativos sin excesiva seguridad sobre ellos. En el comentario se muestra negatividad pero con duda. No es una respuesta tajante negativa.

  - **E.1** - "Perdida y agobiada"
  - **E.2** - "No le veo gran utilidad."

- **Grado (clase) 1**

  Comentarios totalmente negativos, todo respecto a su estado de ánimo es negativo y se muestra una actitud totalmente desconforme.

  - **E.1** - "Muy cansado y deprimido"
  - **E.2** - "No sirve para nada"

- **Grado (clase) 0**

  Comentarios que no indican un grado de satisfacción. No aporta nada al problema que estamos estudiando.

  - **E.1** - "Es el primer año que estudio en la Universidad"
  - **E.2** - "Nada que comentar."

2.3.2. Ejercicio de anotación de los datos

Basándonos en la guía anterior, se ha realizado un ejercicio de anotación compuesto por 3 integrantes, los cuales se ha escogido a 2 de ellos para etiquetar cada una de las entradas en el conjunto de datos y así poder comprobar mediante distintas métricas el nivel de concordancia. El tercer anotador se utiliza para decidir la etiqueta final en los casos donde los 2 primeros anotadores no coinciden.

Evaluación de la calidad de la anotación

Para poder evaluar la calidad de la anotación comprobaremos el acuerdo entre los dos anotadores principales comentados anteriormente.

Se define **acuerdo observado** como "*La primera aproximación a la concordancia entre observadores; resulta, por tanto, la más intuitiva. Simplemente expresa el porcentaje de acuerdo entre ellos, es decir, en qué medida hubo coincidencia en la clasificación entre los observadores en relación al total de elementos examinados*".

Si utilizamos únicamente el acuerdo observado estamos obviando la posibilidad de coincidencia por azar entre los anotadores. Es por ello, por lo

que vemos demasiado pobre utilizar únicamente este acuerdo y por lo tanto decidimos utilizar también el acuerdo esperado.

El **acuerdo esperado** es la probabilidad hipotética de acuerdo por azar. Con esto podemos añadir a la evaluación la posibilidad de acierto al azar y hacer esta evaluación de una forma más acertada y que se acerque más a la realidad.

Es por ello, por lo que para esta evaluación hemos decidido utilizar dos métricas[8], *k de Cohen* y π *de Scott*, cada una de ellas se define:

$$k, \pi = \frac{A_0 - A_e}{1 - A_e}$$

Donde:

$$A_0 = \frac{1}{i} \sum_{i \in I} agr_i$$

agr_i =1 si los anotadores coinciden, 0 si no coinciden

$$A_e^{\pi} = A_e^{k} = \sum_{k \in K} P(k|c_1) \cdot P(k|c_2)$$

Con la única diferencia de que en π *de Scott* asumimos la misma distribución de probabilidad para cada anotador $\text{P}(k|c_m) = \text{P}(k|c_n)$, mientras que en *k de Cohen* se asume una distribucion de probabilidad distinta para cada anotador.

Interpretación de los resultados

Cuadro 2.1: Interpretación Kappa (k)

Valores de Kappa	**Fuerza del acuerdo**
<0.0	Pobre
0.0-0.20	Leve
0.21-0.40	Justa
0.41-0.60	Moderada
0.61-0.80	Considerable
0.81-1.00	Perfecta

Habiendo seguido esta guía de anotación, tras realizar el ejercicio de anotación obtuvimos un valor de **0.527389** de *k de Cohen* y un valor de **0.52686** de π *de Scott* en **Q13**, mientras que por otra parte, obtuvimos un valor de **0.673273** de *k de Cohen* y un valor de **0.672141** de π *de Scott* en la pregunta **Q14**.

Fijándonos en la tabla de resultados y en los valores obtenidos, podemos concretar que para la pregunta **Q13** hemos obtenido un valor que está considerado de concordancia "*Moderada*" lo que nos permite aceptar el resultado y seguir con el proceso. Por otra parte, en la pregunta **Q14** hemos obtenido

un valor de concordancia "*Considerable*", por lo tanto, aceptamos los resultados obtenidos y concretamos que tenemos una buena concordancia para seguir con el proceso del estudio que estamos realizando.

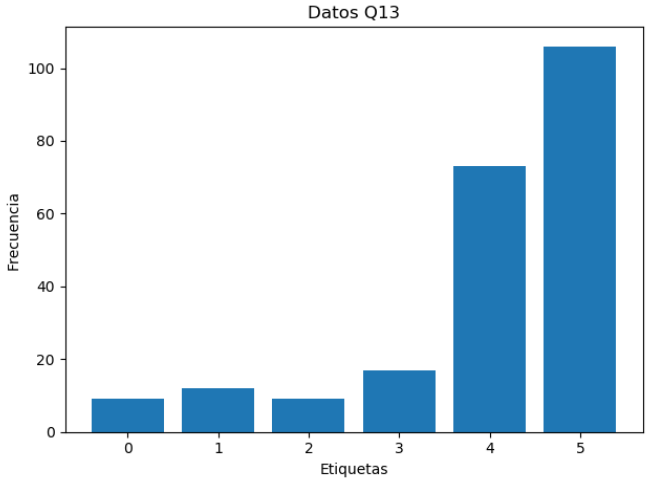
2.4. Conjunto de datos

En este apartado vamos a entrar más a fondo en el conjunto de datos con el que partimos para el problema.

Como hemos comentado, tenemos dos conjuntos distintos:

- **Conjunto Q13**

  Conjunto de datos que contiene información sobre la satisfacción de cada uno de los usuarios con el curso realizado. Este conjunto de datos está compuesto de **226 elementos** clasificados de 0 a 5, teniendo la siguiente distribución.

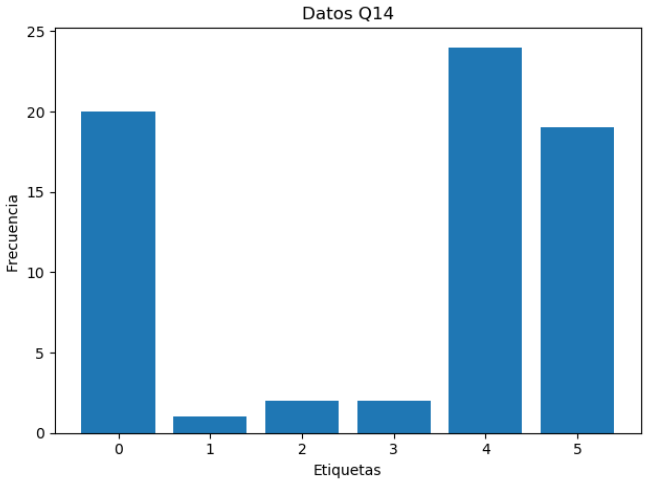


  Como podemos ver, tenemos un conjunto de datos muy desbalanceado, a parte de ser escaso en número de elementos. Esto será un problema a resolver en el futuro, ya que será clave en las decisiones posteriores. Como sabemos, los algorimos de aprendizaje supervisado en nuestro caso, se basan única y exclusivamente en los datos de entrenamiento para poder así predecir otras entradas en el futuro. Es por ello, que si disponemos de tan pocos datos como es nuestro caso, el algoritmo le cuesta más aprender y por ejemplo algoritmos más sofisticados como redes neuronales que necesitan un conjunto de datos muy grande para aprender tienen que ser descartados para este estudio.

Esto quiere decir que debido al número tan bajo de elementos tendremos que utilizar algoritmos que permitan hacer una aproximación lo más certera posible a la realidad con tan pocos datos. Como he comentado, a parte de disponer de tan pocos elementos, el conjunto está desbalanceado haciendo incluso que entradas como por ejemplo los comentarios con etiqueta "2" tengamos alrededor de 10 elementos en total. lo que hace que el aprendizaje se dificulte bastante.

- **Conjunto Q14**

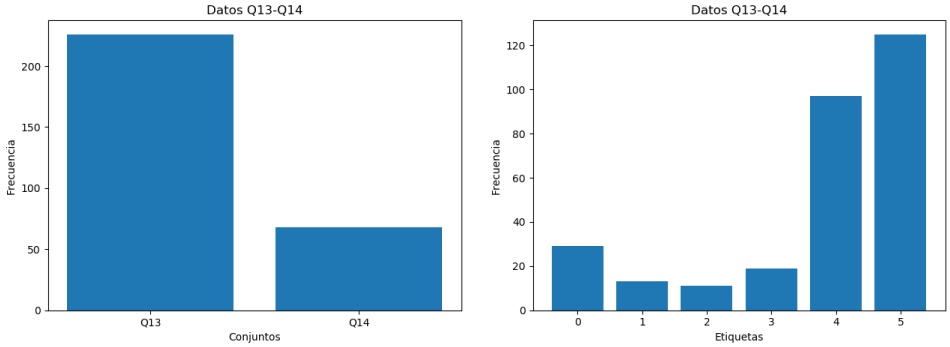
  Conjunto de datos que contiene información sobre la satisfacción de cada uno de los usuarios respecto a la encuesta en sí. Este conjunto de datos está compuesto por **68 elementos**, siendo aún menor que el anterior. Al igual que en Q13, estos datos siguen el mismo etiquetado, por lo que la distribución en este caso es la siguiente.



  Si con el conjunto anterior teníamos pocos datos, pasamos a este conjunto en el que tenemos menos de una tercera parte del anterior. Esto dificulta inmensamente el aprendizaje ya que como hemos comentado, a un algoritmo con tan pocos datos le es muy difícil aprender realmente cómo clasificar los distintos valores de entrada.

  Podemos ver cómo para este conjunto tenemos apenas 2 elementos de la clase "1". Es totalmente comprensible que para cualquier algoritmo usado sea prácticamente imposible aprender las características esenciales de esta clase para poder diferenciarla del resto debido a la escasez de información.

Finalmente, como conjunto de datos completo tenemos **294 elementos** distribuidos de la siguiente forma:



Aún así, seguimos partiendo de un conjunto muy pequeño de datos, que sigue con la escasez de ejemplos para casos determinados como puede ser el caso de las entradas con etiqueta "2" en la que apenas tenemos unos 10 elementos.

Tendremos por lo tanto que utilizar algoritmos que permitan o que funcionen mejor con conjuntos de datos pequeños y desbalanceados aunque como he recalcado, el conjunto de datos en general para este tipo de estudios es pequeño y escaso en ejemplos.

2.5. Requisitos funcionales

Para el desarrollo de nuestra aplicación o modelo tenemos que cumplir los siguientes requisitos funcionales:

1. Capacidad para obtener una cadena de texto y almacenarla para un futuro procesamiento de la misma.
2. Capacidad para transformar las cadenas de texto en valores interpretables para el futuro modelo.
3. Capacidad para crear un modelo de aprendizaje a partir de un conjunto de cadenas de texto anteriormente procesadas.
4. Capacidad para clasificar una nueva cadena de texto nunca vista anteriormente utilizando el modelo creado anteriormente.
5. Capacidad para visualizar los resultados obtenidos de la forma mas comprensible y rápida posible.

2.6. Requisitos no funcionales

Para este apartado, pondré los casos más comunes que cualquier software debe cumplir, por lo tanto:

1. Sistema robusto que responde a fallos.
2. Sistema eficiente, respuesta en tiempo real.
3. Sistema disponible al usuario.
4. Sistema fácil de utilizar para cualquier tipo de usuario.

Capítulo 3

Análisis

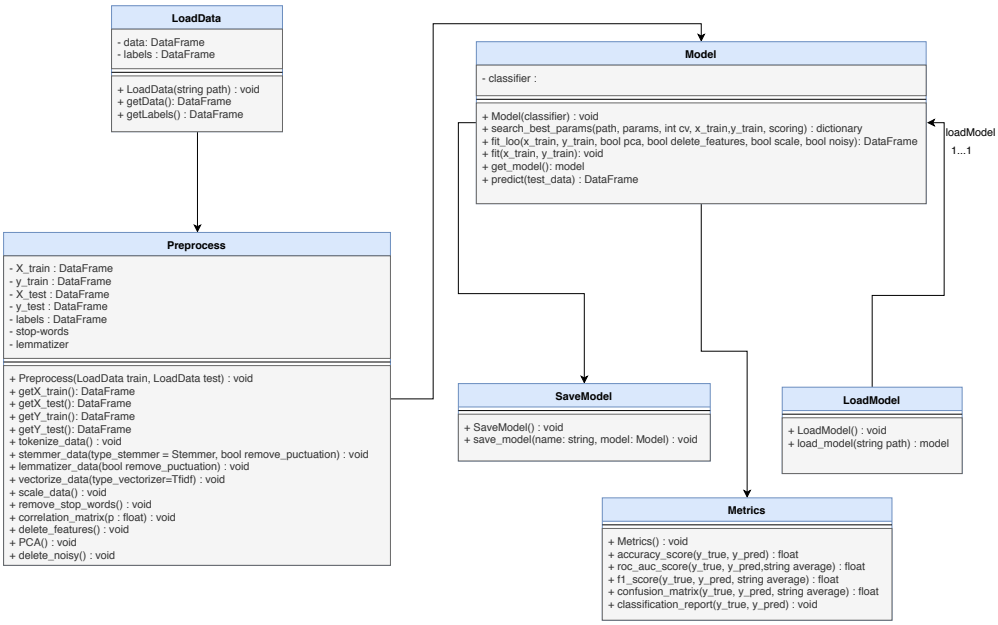
En este capítulo comentaremos el proceso de análisis seguido para éste problema, como hemos indicando en capítulos anteriores, nos centramos en la plataforma *AbiertaUGR*, en concreto en la evaluación de la satisfacción de las diferentes personas que realizan los cursos y las encuestas ofrecidas por la plataforma.

Para ello, desarrollamos un sistema capaz de, dados unos datos de entrada, procesar dichos datos para transformarlos en información útil para el algoritmo de aprendizaje haciendo uso de distintas técnicas que veremos en apartados posteriores. Con esto, conseguiremos desarrollar métodos que nos permitirán la clasificación automática de opiniones comprobando el estado de satisfacción del alumno en el MOOC, facilitando inmensamente el trabajo del desarrollador del curso y poder sacar conclusiones de una forma fácil y cómoda.

Seguidamente, utilizando esos datos procesados y haciendo uso de distintos algoritmos de aprendizaje se consigue crear un modelo que será en encargado posteriormente de la clasificación de nuevos datos de entrada, valorándolos con el etiquetado anteriormente propuesto.

3.1. Diagrama de clases

En este apartado mostraremos el diagrama de clases que forma el sistema creado.



El sistema se ha dividido en diferentes entidades o clases con las que podremos realizar los procesos a seguir para conseguir crear o entrenar un clasificador con el que resolver el problema que estamos estudiando en este proyecto. Se pueden identificar claramente cada una de las tareas que realiza cada una de las entidades, no obstante voy a comentar en general lo que el sistema sería capaz de realizar.

Con este sistema podremos hacer, entre otras cosas:

- Lectura del conjunto de datos etiquetados.
- Tokenización de los datos introducidos.
- Eliminar "stop-words" del conjunto de datos, es decir, las palabras que no aportan información a la frase.
- Aplicar un "Steammer" al conjunto para conseguir unificar las distintas formas de expresar y escribir una misma palabra.
- Eliminación de los signos de puntuación.
- Vectorizar el conjunto de datos para hacer éstos datos factibles para el algoritmo de aprendizaje.

- Escalado de los datos por si necesitamos hacerlo para un algoritmo concreto.
- Entrenamiento del modelo buscando los mejores parámetros posibles (indicando los distintos parámetros a probar).
- Entrenamiento del modelo con unos datos de entrada.
- Predicción de unos datos de entrada, obteniendo la etiqueta predicha de éstos datos.
- Guardar y cargar el modelo para poder acceder a él en cualquier momento sin volver a entrenar.
- Obtener distintos valores de métricas como pueden ser "accuracy score", "f1 score", entre otras.

3.2. Análisis léxico

Hemos decidido para visualizar de una forma más clara y directa la composición de los datos, hacer un estudio léxico de los datos, siguiendo el siguiente procedimiento:

- En primer lugar se hará una agrupación del conjunto de datos de forma que tendremos las entradas con etiqueta "4" y "5" (positivas) en un mismo grupo, las entradas con etiqueta "1" y "2" (negativas) en otro grupo, otro grupo será la etiqueta "0" y el último la etiqueta "3".
- Una vez agrupados, se tokenizarán cada una de las entradas.
- Se eliminarán las *"stop-words"* y los signos de puntuación.
- Por último se pasará un *"Steammer"* sobre el conjunto.

Una vez hecho esto, contaremos la frecuencia con la que aparece cada palabra clave en el conjunto. Con esto tendremos una forma visual de ver qué palabras aparecen más veces y así poder identificar a simple vista las diferencias entre clases.

3.2.1. Clase positiva (4 y 5)

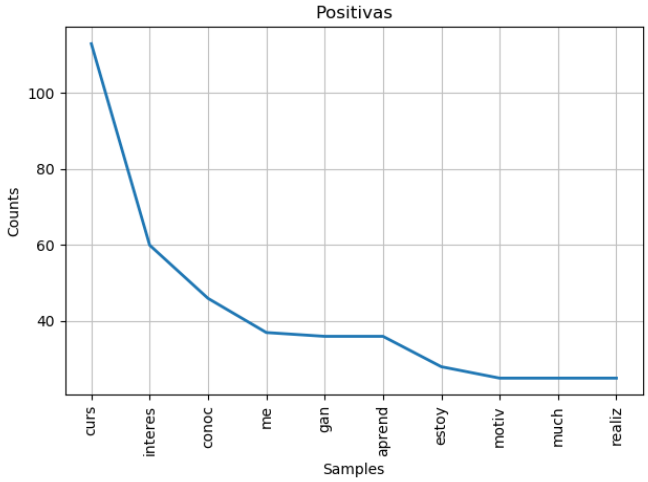
Los resultados obtenidos para este experimento en la clase positiva son los siguientes:

Palabra	Frecuencia	Frecuencia/Total
curs	113	0.17
interes	60	0.092
conoc	46	0.07
me	37	0.057
gan	36	0.055
aprend	36	0.055
estoy	28	0.043
motiv	25	0.038
much	25	0.038
realiz	25	0.038

Cuadro 3.1: Tabla frecuencias grupo positivo.

Podemos ver en la tabla las palabras destacadas (ya procesadas) cuando agrupamos la clase positiva. Esta tabla nos muestra la palabra en sí, la frecuencia en número de veces que aparece y la frecuencia respecto al número de palabras totales en el grupo positivo.

Podemos también visualizar el resultado mediante un gráfico como el siguiente:



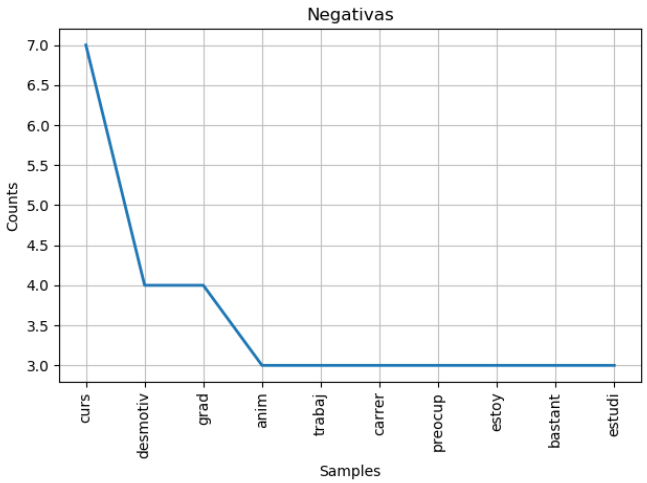
Viendo los resultados obtenidos, se pueden sacar o al menos discutir que tienen sentido las palabras clave para este grupo positivo, no obstante, dejaremos las conclusiones para el final cuando tengamos los resultados de cada grupo creado.

3.2.2. Clase negativa (1 y 2)

Al igual que en el caso anterior, mostraremos los resultados en forma de tabla y los resultados en forma de gráfico:

Palabra	Frecuencia	Frecuencia/Total
curs	7	0.068
desmotiv	4	0.039
grad	4	0.039
anim	4	0.039
trabaj	3	0.029
carrer	3	0.029
preocup	3	0.029
estoy	3	0.029
bastant	3	0.029
estudi	3	0.029

Cuadro 3.2: Tabla frecuencias grupo negativo.



Esta vez, podemos comprobar que el gráfico es mucho más fijo que el anterior, simplemente se debe a lo ya comentado anteriormente, la falta de datos que tenemos no permite realmente poder visualizar el experimento con claridad ya que como podemos ver aunque algunas de las palabras puedan coincidir con la expresión de negatividad que queremos aprender, se puede ver como la palabra más común sólo tiene 7 apariciones, y eso que estamos tomando el caso de todas las negativas, juntándolas en un mismo grupo.

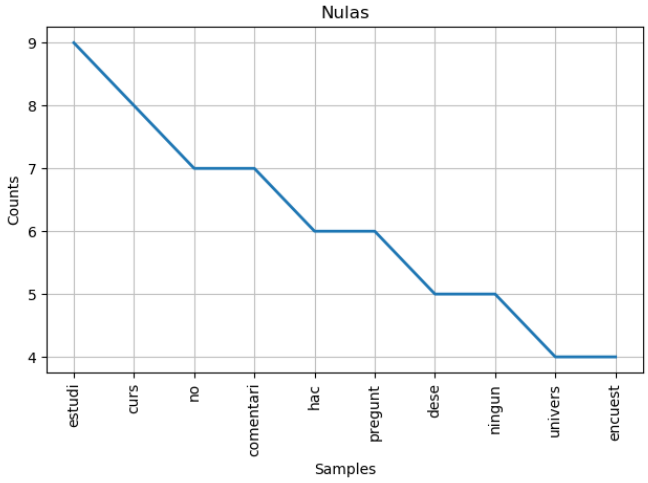
También hay que destacar comprobando la frecuencia total, que incluso la palabra que más aparece, aparece muy pocas veces respecto al conjunto de palabras del grupo negativo.

3.2.3. Clase Nula (0)

Mostramos de nuevo la tabla de frecuencias y el gráfico:

Palabra	Frecuencia	Frecuencia/Total
estudi	9	0.05
curs	8	0.044
no	7	0.039
comentari	7	0.039
hac	6	0.033
pregunt	6	0.033
dese	5	0.028
ningun	5	0.028
univers	4	0.022
encuest	4	0.022

Cuadro 3.3: Tabla frecuencias grupo nulo.



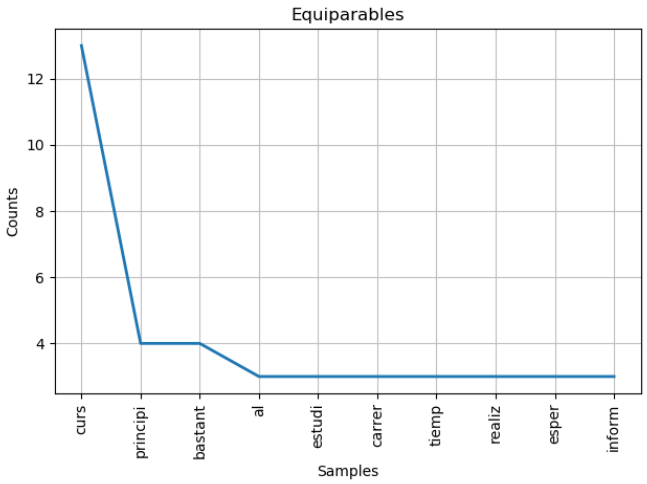
Se puede identificar que las palabras para este grupo son palabras sin significado en el sentido de expresar algún sentimiento relacionado con el nivel de satisfacción de una persona. No obstante, como en los caso anteriores, seguimos teniendo un número muy bajo de apariciones de las palabras más comunes. Si nos fijamos en la frecuencia total, vemos que las palabras siguen siendo muy poco significativas respecto al conjunto de palabras clave del grupo nulo completo.

3.2.4. Clase Neutra (3)

Veremos al igual que en los apartados anteriores, la tabla de frecuencias y su correspondiente gráfica:

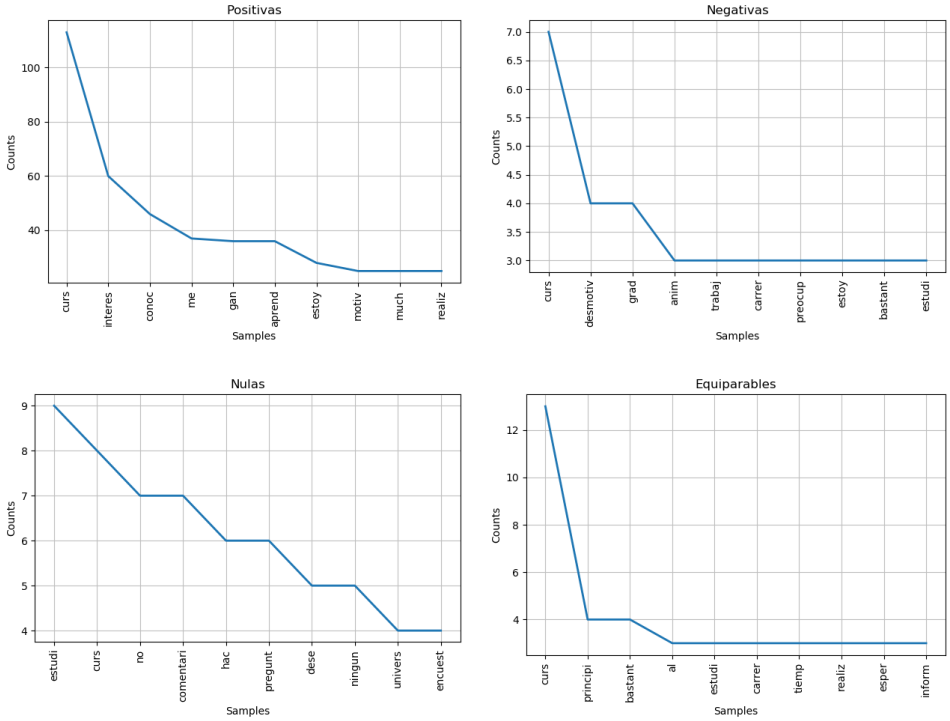
Palabra	Frecuencia	Frecuencia/Total
curs	13	0.082
principi	4	0.025
bastant	4	0.025
al	3	0.019
estudi	3	0.019
carrer	3	0.019
tiemp	3	0.019
realiz	3	0.019
esper	3	0.019
inform	3	0.019

Cuadro 3.4: Tabla frecuencias grupo equiparable.



Como podemos ver para este caso, prácticamente ninguna palabra nos puede indicar nada a nivel de frecuencia ya que tiende muy rápido a bajar a un numero de apariciones por palabra prácticamente constante y bajo, al igual que seguimos con los mismos problemas anteriores de los demás casos, éste nos aumenta más aún la dificultad no sólo por la poca aparición de palabras si no por que prácticamente todas las palabras tienen la misma distribución respecto a lo que a la frecuencia se refiere.

3.2.5. Conclusiones



Viendo las gráficas y las tablas anteriores me doy cuenta de que efectivamente hay palabras que sabemos que expresan una satisfacción positiva o negativa y se encuentran en su correspondiente grupo, lo que nos dice que hemos sido capaces de separar correctamente el nivel de satisfacción.

Por otra parte, tenemos los conjuntos de lo que consideremos nulo y el conjunto que consideramos neutro. Para el conjunto que indica que un comentario es nulo podemos percibir que las palabras más significativas son palabras que no expresan un grado de satisfacción y por lo tanto podemos dar por correcto esta forma de separar los conjuntos, sin embargo para el caso de la clase neutra, ésto no ocurre ya que no tenemos un conjunto de palabras que realmente expresen el sentimiento indeciso de que puede ser positivo o negativo y ésto nos va avisando de antemano que será complicado para el algoritmo.

Lo que todas tienen en común es la falta de información como ya hemos hecho hincapié en otros apartados. Aunque el caso de la clase positiva tenemos más frecuencia en las palabras debido a que tenemos un conjunto más grande, vemos que no hay ninguna palabra que realmente exprese un positividad y sea significativa respecto al conjunto de datos completo ya que su aparición total es muy baja.

En resumen, aunque las palabras en algunos casos sí que pueden indicar-

los que la clasificación y separación de clases podrá hacerse correctamente, con éste experimento vemos que ése mismo conjunto de palabras son muy poco significantes respecto al respectivo conjunto completo del grupo, lo que hace que el valor de las mismas se vea algo suprimido por el resto de palabras.

Capítulo 4

Diseño

En este apartado expondremos los diferentes casos que se han diseñado para las siguientes experimentaciones explicando cada uno de los elementos utilizados.

Hemos decido realizar en primer lugar un **caso base** simple del que partiremos y compararemos con el resto. Al igual que el caso base, hemos definido un **caso techo** en el que nos apoyaremos cuando obtengamos los resultados de las diferentes experimentaciones.

4.1. Casos iniciales

4.1.1. Caso base

Este caso base está basado en el algoritmo *OneR*, lo que hace es clasificar todas las salidas con el valor de la clase que más apariciones tiene, por lo que en nuestro caso, como vimos en el apartado anterior dónde mostrábamos el conjunto de datos que teníamos, este caso base lo que hará será crear una salida donde todos los elementos a clasificar llevarán el valor de etiqueta "5".

Veremos el resultado y los valores de este caso cuando tengamos que usarlo en las diferentes experimentaciones siguiente.

4.1.2. Caso techo

Este caso se define como el acuerdo observado entre las 2 personas que han realizado el ejercicio de anotación comentado en apartados anteriores. Éste caso nos dice el acuerdo observado que dos personas humanas han conseguido obtener al clasificar los datos uno a uno sin ninguna influencia

externa. Basándonos en esto, podemos partir de un valor inicial el cuál nos muestra cómo dos humanos han sido capaces de estar de acuerdo.

Aunque haremos referencia a este caso en las experimentaciones, voy a comentar que en el ejercicio de anotación realizado, una vez realizadas todas las métricas, obtuve un valor de **0.681** en **Q13** y un valor de **0.764** en **Q14** de acuerdo observado.

Ésto nos dice que los dos anotadores han conseguido coincidir en un 68 % de los datos de la pregunta Q13 y un 76 % en la pregunta Q14. Si dos personas han conseguido estos valores, debemos de considerar que una máquina tendrá dificultad en superar estos valores.

4.2. Casos diseñados

Una vez definidos estos dos casos base y techo, definimos distintos casos para estudiar posteriormente donde iremos probando diferentes combinaciones de los conjuntos de datos de los que partimos para poder seleccionar el mejor caso posible entre todos.

Antes de nada he de comentar que debido a la escasez de los datos, decidimos hacer uso en los entrenamientos de *"LeaveOneOut"* para el entrenamiento y así tener evaluados todos los elementos del conjunto y de esa clasificación sacar las distintas métricas. Explicaremos qué significa y entraremos más a fondo en apartados posteriores ya que para éste momento no es necesario entender el funcionamiento del mismo.

Para todos los casos que hemos diseñado se sigue un proceso de transformación del texto, limpieza del mismo y pre-procesamiento de los datos del cuál haremos mas énfasis en los apartados siguientes.

Una vez realizados lo comentado, se pasa ha hacer los diferentes casos:

4.2.1. Caso 1: Pregunta Q13

Para este caso lo que haremos será entrenar un clasificador usando *"LeaveOneOut"* con todos los elementos de la pregunta Q13. En el entrenamiento se va guardando la salida de la predicción en cada caso, para que al final tengamos un conjunto de predicción igual al tamaño del conjunto completo.

4.2.2. Caso 2: Pregunta 14

Al igual que en el anterior, entrenaremos un clasificador usando *"LeaveOneOut"* con todos los elementos de la pregunta Q14. Al terminar el

entrenamiento, tendremos un conjunto de etiquetas predichas del tamaño del conjunto de datos completo.

4.2.3. Caso 3: Pregunta 13 + Pregunta 14

Para este caso, juntaremos los dos conjuntos en uno, seguiremos usando *"LeaveOneOut"* ya que aunque estén juntos sigue siendo un conjunto pequeño de datos. Al terminar el entrenamiento valoraremos los resultados ya que tendremos un conjunto de etiquetas predichas del tamaño del conjunto completo.

4.2.4. Caso 4: Entrenamiento Pregunta 13, Test Pregunta 14

Para este caso, entrenaremos con la pregunta 13 completa y usamos la pregunta 14 para predecir todos los elementos de la misma y así poder valorar el caso de que con una pregunta que valora la satisfacción, aunque la pregunta no sea la misma, ser capaces de etiquetar otra pregunta distinta que al final expresa el mismo grado de satisfacción.

4.2.5. Caso 5: Entrenamiento Pregunta 14, Test Pregunta 13

Al igual que el caso anterior pero cambiando las preguntas, ésta vez entrenaremos con la pregunta 14 para después evaluar si somos capaces de detectar el grado de satisfacción evaluando los elementos de la pregunta 13.

Para todos los resultados utilizaremos el caso base y caso techo para las comparaciones.

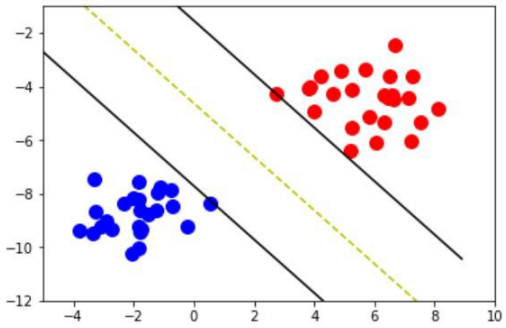
4.3. Algoritmos de aprendizaje

Para este experimento hemos decidido probar 3 algoritmos de aprendizaje para cada uno de los casos diseñados y comentados anteriormente. Los algoritmos de aprendizaje nos permitirán poder distinguir entre las distintas clases que tenemos y poder etiquetar y predecir nuevos datos que nos vengan en el futuro.

4.3.1. SVM

El algoritmo SVM[16] es un algoritmo de aprendizaje supervisado que puede ser usado tanto para regresión como para clasificación como para detección de *outliers*.

Este algoritmo intenta separar dos o más clases de forma lineal haciendo que la distancia entre la función lineal en el plano y los puntos de cada una de las clases estén a una distancia máxima posible entre ellas.



Como podemos ver la intención del algoritmo es dejar la máxima distancia entre los puntos de cada clase y la propia función lineal, con esto intentamos asegurarnos de que puntos futuros conflictivos que puedan estar cerca de la frontera estén en un margen considerable para ser seleccionados como una clase u otra.

Algunas de las ventajas de éste algoritmo son:

- Es un algoritmo efectivo en grandes dimensiones.
- Sigue siendo efectivo en casos donde el número de dimensiones es mayor que el número de muestras.
- Utiliza un subconjunto de puntos los cuales se encuentran en la frontera y ésto hace que el algoritmo sea eficiente en memoria.
- Es versátil, de tal forma que le pueden ser especificados diferentes funciones de kernel e incluso kernel personalizados.

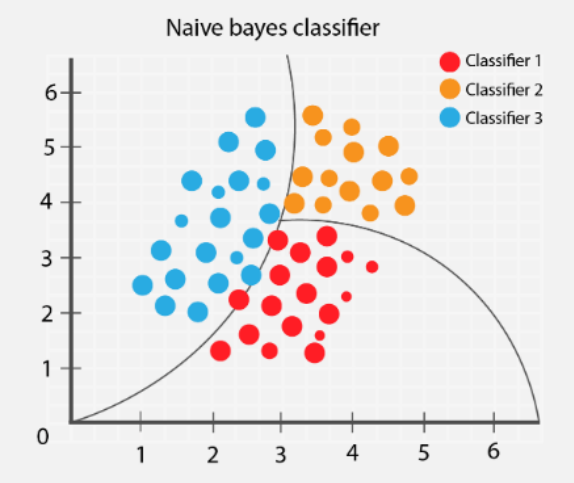
Por otra parte, algunas de las desventajas de este algoritmo son:

- Si el número de características es mucho más grande que el número de muestras, es muy probable que se produzca un sobreajuste de la función sobre los datos.
- SVM no proporciona directamente estimaciones de probabilidad si no que éstas se calculan haciendo validación cruzada, lo que puede llegar a ser costoso.

Como podemos ver por las diferentes ventajas y por las desventajas, es un algoritmos que encaja perfectamente en nuestro problema.

4.3.2. Naive Bayes

El algoritmo Naive Bayes[17] es un algoritmo de aprendizaje supervisado basado en la aplicación del teorema de Bayes, es uno de los clasificadores probabilísticos más simples. A menudo funciona sorprendentemente bien en muchas aplicaciones del mundo real, sobre todo en problemas de documentos y el filtrado de correo no deseado. También éste algoritmo supone que todas las características son condicionalmente independientes respecto a la clase.



Algunas de las ventajas de este algoritmo son:

- Necesita pocos datos para poder estimar la predicción medianamente bien.
- Es extremadamente rápido en comparación con otros métodos sofisticados.
- Al suponer las características independientes, pueden calcularse como una distribución unidimensional y así eliminar el problema de el número de dimensiones elevado.

Como principal desventaja tenemos que:

- Se conoce que este algoritmo no es muy buen clasificador o estimador a lo que a resultados probabilísticos se refiere, por lo que las salidas predecidas mediante probabilidad no se suelen tener muy en cuenta.

Principalmente éste algoritmo fué elegido por su rapidez y por la necesidad de pocos datos para estimar.

Naive Bayes Multinomial

Uno de los tipos de Naive Bates a utilizar será el algorimo conocido como Naive Bayes Multinomial, claramente sigue toda la estructura del Naive Bayes que acabamos de ver, con la diferencia de que éste algoritmo está preparado para trabajar con variables numéricas enteras, no obstante se ha comprobado en la práctica que éste algoritmo trabaja correctamente con variables reales como por ejemplo los valores que nos devuelve una trasformación de un conjunto de texto a números con Tfidf, por lo que nos será útil en nuestro problema.

Naive Bayes Complement

Otro tipo del algoritmo Naive Bayes, definido y explicado en el apartado anterior, con la diferencia de que ahora utilizamos una modificación del mismo.

Ésta modificación es una modificaón sobre el algoritmo Naive Bayes Multinomial, con la diferencia de que en éste caso no sólo se tiene en cuenta la distribucion multinomial de los datos (normalmente utilizada en clasificación de texto), si no que también se centra en conjuntos de datos donde la distribución de los mismos está desbalanceada.

Se conoce que éste método suele mejorar al clásico Naive Bayes multinomial por lo que hemos decidido utilizar junto al multinomial para compararlos.

Capítulo 5

Implementación

En este capítulo vamos a explicar cómo se ha programado el sistema, indicando el lenguaje de programación como los módulos utilizados para la resolución del problema.

5.1. Especificación inicial:

El sistema ha sido desarrollado con el lenguaje de programación Python 3.6 ya que para mí es un lenguaje muy fácil de entender, facil de programar, eficaz y muy bien preparado para este tipo de tareas con muchísimos paquetes y módulos implementados para nuestra ayuda. He utilizado un Sistema Operativo Linux, en concreto Linux Mint y Pycharm como editor de texto.

El sistema ha sido diseñado siguiendo el diagrama de clases descrito en secciones anteriores. Por lo que el proceso que se ha seguido ha sido crear una clase para cada entidad representada en el diagrama. Cada una ha sido almacenada en un fichero distinto para una mejor separabilidad y modularización. Por lo que finalmente hemos diseñado un total de 6 clases que nos ayudarán y nos facilitarán la resolución del problema incluso pudiendo hacer distintas combinaciones a lo que a la resolución del problema se refiere.

5.2. Definición de las clases

En esta sección explicaremos qué hace cada una de las clases y qué módulos se permiten utilizar, explicando los distintos módulos externos que utilice.

5.2.1. Clase LoadData

Esta clase es la encargada de cargar los datos de un fichero para poder procesarlos desde el sistema. Su funcionamiento es muy simple ya que su único cometido es leer los datos indicados por la ruta de los mismos y guardarlos en una variable.

Paquetes utilizados

Para esta clase se han utilizado el paquete **numpy** para almacenar los datos de entrada, tando los comentarios en sí como las etiquetas, y **pandas** para la lectura de el fichero de entrada (formato xlsx).

Variables internas

- **- data:** Variable donde tendremos almacenados los comentarios de los usuarios.
- **- labels:** Variable donde tendremos las etiquetas correspondientes a los comentarios.

Métodos

- **+ getData():** Devuelve los datos (comentarios).
- **+ getLabels():** Devuelve las etiquetas.

5.2.2. Clase Preprocess

Esta clase es una de las clases principales del sistema, ya que es la encargada de el procesamiento de texto y del pre-procesamiento de los datos para dejar toda la información disponible para los algoritmos que vamos a usar.

Este paso es imprescindible e importante ya que por muy buen algoritmo que utilicemos para resolver el problema, si los datos no están bien preparados y transformados, el algoritmo será incapaz de sacar información y aprender algo de los datos.

Paquetes utilizados

Principalmente para esta clase necesitamos utilizar los siguientes paquetes:

- **nltk:** NLTK es la plataforma líder para la creacion de programas Python para trabajar con datos en lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 recursos corporales y léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico. Este paquete será el que nos permita trabajar con el texto para que el algoritmo pueda así entender el significado de las frases.

  Permite trabajar con lenguaje en español, lo que facilita el uso para nuestro caso.

- **sklearn** Paquete principal e importantísimo para resolver problemas de este estilo en Python. Nos permitirá hacer preprocesado de datos, utilizar algoritmos de clasificación, utilizar métricas para medir los resultados, métodos de ajuste de hyperparámetros, validación cruzada, etc... .

  Es la base del sistema y en la que se centran la mayoría de módulos, sobre todo en la clase Models que veremos a continuación.

- **spacy:** Lo utilizamos principalmente para poder importar un Lematizador al sistema por si se necesita de su uso.

Variables internas

- **- X_train:** Variable donde se almacenan los datos (comentarios) de los cuales utilizaremos para entrenar los modelos.

- **- y_train:** Variable donde se almacenan las etiquetas correspondientes a los datos de entrenamiento.

- **- X_test:** Variable donde se almacenan los datos (comentarios) con los cuales evaluaremos el modelo.

- **- words:** Variable donde se almacena el modelo de *"stop-words"* para poder utilizarlo y eliminar las propias *"stop-words"*.

- **- nlp:** Variable que nos permite realizar una lematización sobre el conjunto de datos.

Métodos

- **+ getX_train():** Devuelve *X_train*.

- **+ getX_test():** Devuelve *X_test*.

- + **getY_train():** Devuelve *y_train*.
- + **getY_test():** Devuelve *y_test*.
- + **tokenize_data():** Tokeniza cada una de las frases del conjunto para dejar las frases separadas por tokens.
- + **remove_stopwords():** Elimina las palabras que no aportan significado a las frases como los determinantes.
- + **stemmer_data(remove_punctuation,stemmer):** Pasa el Stemmer *SnowBallStemmer* al conjunto de datos para reducir el número de palabras que aparecen en el conjunto de datos y relacionar palabras con la misma raíz. Se le puede indicar si se quiere eliminar los signos de puntuación o indicar un Stemmer diferente.

  SnowBallStemmer es utilizado principalmente porque nos permite aplicarlo sobre lenguaje español directamente. Su cometido consiste en acortar palabras similares para dejarla como una única palabra y así facilitar el conjunto y resumirlo. Por ejemplo, si tenemos la palabra "comería" y "comiendo", después de aplicar ésta técnica obtendremos la palabra "com".
- + **lemmatizer_data(remove_punctuation):** Lematiza el conjunto de datos haciendo coincidir el lema de las palabras que lo comparten para reducir el conjunto y relacionar las palabras. Se le puede indicar si se quiere eliminar signos de puntuación o no.

  Para este caso utilizamos el Lematizador que tiene el paquete spacy, que nos permite diferenciar las palabras entre sustantivos, verbos, etc... dentro de una frase. A su vez, transforma estas palabras para que palabras similares queden iguales, por ejemplo, en la frase "procesamiento de textos", indica que procesamiento es un sustantivo, y que su lema es procesamiento, de indica que es un preposición y que su lema es de, y por último indica que textos es sustantivo y su lema es texto.
- + **vectorizer_data(vectorizer):** Vectoriza el conjunto de palabras que tenemos utilizado *TfidfVectorizer* para conseguir transformar las palabras en información útil para los algoritmos. Se puede indicar algún otro tipo de vectorizador.

  TfidfVectorizer nos permite representar de forma matricial el conjunto de datos (texto) y transformarlo en números para que un algoritmo pueda entenderlo.

  Tf nos viene a decir la frecuencia de un término, mientras que tf-idf nos dice la frecuencia de un término multiplicado por la frecuencia del término en el documento. Es un esquema de ponderación muy usado y con buenos resultados en problemas de clasificación de documentos.

El principal cometido es restar importancia a los token que aparecen en un texto muchas veces ya que nos dan menos información que los token que aparecen menos veces en el documento. Por lo tanto, aumenta la importancia de los token que aparecen menos veces en el documento para que no pasen desapercibidos respecto a los token que aparecen más veces en el documento.

$$\text{tfidf}(t,d) = \text{tf}(t,d) * \text{idf}(t)$$
$$\text{idf}(t) = \log(n/\text{df}(t)) + 1$$

Donde n es el numero total de documentos en el conjunto de documentos y $\text{df}(t)$ es la frecuencia de t en el documento. La frecuencia de documento es el número de documentos en el conjunto de documentos que contienen a t

- **+ scale_data():** Estandariza el conjunto mediante la eliminación de la media y la escala a la variación de la unidad. Seguidamente escala los datos entre 0 y 1.
- **+ correlation_matrix(p):** Elimina las características que cumplan que tienen una correlación superior a p respecto a otra característica, reduciendo la dimensionalidad del problema.
- **+ delete_features():** Utiliza un algorimo de aprendizaje para seleccionar las características menos importantes a la hora de realizar la clasificación. Una vez se tienen cuáles son, se eliminan estas características del conjunto de datos, reduciendo la dimensionalidad del problema. En mi caso, utilizo *Random Forest* para este proceso.
- **+ PCA():** Permite eliminar características reduciendo la dimensionalidad mientras se consiga explicar un porcentaje determinado de información del conjunto.
- **+ delete_noisy():** Elimina instancias que se consideran ruido. Éste proceso se hace mediante un proceso de *cross-validation* con *LeaveOneOut* y clasificando el elemento de test con 3 algoritmos de clasificación distintos simples. Si ninguno de los algoritmos es capaz de acertar, éste elemento es eliminado del conjunto y considerado como ruido.

5.2.3. Clase Model

Esta es la clase que se encarga de gestionar los modelos de aprendizaje que necesitemos crear, con la cual podremos entrenar el modelo, buscar los mejores hyperparámetros y predecir respecto al modelo.

5.2.4. Paquetes utilizados

Para ésta clase se utiliza claramente el paquete **sklearn** explicado anteriormente y también hemos utilizado **numpy** para el almacenamiento de algunas de las variables y el procesado de las mismas.

Variables internas

- **- classifier:** Modelo a utilizar, será el que utilizaremos para entrenar con los datos para posteriormente predecir.
- **- loadModel:** Variable de la clase *LoadModel* que veremos posteriormente. Ésto nos permitirá cargar un clasificador directamente si ya lo tenemos almacenado en lugar de tener que volver a ajustar los hiperparámetros.

Métodos

- **+ search_params(path,params,cv,x_tran,y_train,scoring):** Método que nos va a permitir realizar un ajuste de hyperparámetros del modelo para poder ajustarnos más y mejorar el resultado obtenido.

  Para ello, hemos utilizado **GridSerachCV** del paquete *sklearn* que hace una búsqueda exhaustiva sobre valores de parámetros especificados para un estimador. Lo que hace es ejecutar el modelo con todas las combinaciones posibles de los parámetros que le indiquemos y quedarse con el mejor modelo obtenido siguiendo un criterio de *scoring* preestablecido.

- **+ fit_loo(x_train,y_train,pca,delete_features,scale):** Permite hacer un ajuste del modelo utilizando validación cruzada con *LeaveOneOut*. Se le pueden indicar mediante booleanos si queremos que realizar un preprocesado de los datos o no.

  Lo que hace principalmente es que en el proceso de aprendizaje, como tenemos todos los elementos como train menos 1 que será el test, para hacer el proceso algo más cercano a la realidad, se ajusta el preprocesado sobre el conjunto de train para aplicarlo sobre el test. Con esto conseguimos no sólo que el algoritmo no conozca a ese elemento de train, si no que las transformaciones de los datos también se realizan sin que éste elemento sea visto.

  Devuelve el conjunto de datos con la etiqueta predicha para cada uno de los elementos.

- **+ fit(x_train,y_train):** Si no vamos a realizar *LeaveOneOut*, podemos entrenar el modelo con el conjunto de datos completo.

- **+ getMOdel():** Devuelve *classifier*, que contiene el modelo.
- **+ predict(x_test):** Predice un conjunto de datos indicado por parámetros y devuelve el etiquetado predicho por el modelo.

5.2.5. Clase LoadModel

Clase que nos permite cargar un modelo guardado anteriormente para no tener que entrenarlo de nuevo ni tener que ajustar los hyperparámetros, facilitando y agilizando el proceso en caso de tener clasificadores complejos en tiempo.

Paquetes utilizados

Para esta clase utilizaremos un paquete llamado **joblib**, que un paquete que nos permitirá guardar y cargar documentos en binario. Lo utilizaremos para guarda y cargar el modelo. Aunque éste paquete tiene más usos, en mi caso únicamente lo utilizo para ésto, por lo que no explicaré los diferentes usos del mismo.

Métodos

- **+ load_model(path):** Carga el modelo que se encuetre almacenado en *path*.

5.2.6. Clase SaveModel

Clase que nos permitirá guardar un modelo entrenado previamente para poder hacer uso del mismo en momentos posteriores sin tener que realizar de nuevo todo el proceso de aprendizaje.

Paquetes utilizados

Al igual que en el caso anterior, únicamente se ha utilizado el paquete **joblib**.

Métodos

- **+ save_model(path,model):** Guarda el modelo *model* en *path* de forma eficiente en espacio y velocidad para sus posibles futuros usos.

5.2.7. Clase Metric

Clase que nos permitirá ejecutar un grupo de métricas respecto a los resultados obtenidos en la predicción del modelo creado.

Paquetes utilizados

Utilizaremos únicamente el paquete **sklearn** ya que como hemos comentado anteriormente dispone de un grande repertorio de métricas para evaluar el resultado de una clasificación.

Métodos

- **+ accuracy_score(y_true,y_pred):** Devuelve el *accuracy* (acuerdo observado) entre un etiquetado y el otro. Ésto no es mas que un valor $[0-1]$ que indica el porcentaje de concordancia entre los dos.
- **+ f1_score(y_true,y_pred,average):** F1 score puede ser interpretada como el promedio ponderado de la precisión y el acuerdo entre los dos etiquetados para no tener únicamente en cuenta qué datos hemos acertado. El valor de F1 está acotado a $[0, 1]$ siendo 1 el mejor valor posible y 0 el peor. La definición del F1 se realiza asumiendo un etiquetado binario.

  $$F1 = 2 * (\text{precision} * \text{acuerdo}) / (\text{precision} + \text{acuerdo})$$

  Definimos la precisión como el valor de aciertos positivos (TP) respecto al subconjunto formado por los aciertos positivos mas los fallos positivos (TP+FP).

  $$\text{precision} = \frac{TP}{TP+FP}$$

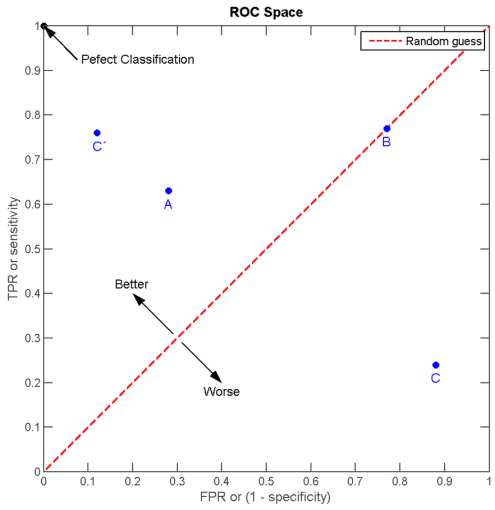
  Definimos acuerdo como el valor de los aciertos positivos (TP) respecto al subconjunto formado por los aciertos positivos mas los falsos negativos (TP+FN)

  $$\text{acuerdo} = \frac{TP}{TP+FN}$$

- **+ confusion_matrix(y_true,y_pred):** Calcula la matriz de confusión entre las dos listas de etiquetas. Ésta matriz nos permite visualizar cuántos elementos han sido correctamente etiquetados y cuántos han sido etiquetados de forma incorrecta, pudiendo ver a qué clase se le ha etiqueda el dato incorrecto, con lo que no sólo nos permite ver el fallo, si no que también podemos ver cómo de cerca o lejos está etiquetando ese valor respecto al que debería ser su valor real.

- **+ classification_report(y_true,y_pred):** Nos hace un informe con distintos *average* donde podremos ver el valor de F1, precision y de acuerdo (recall) para cada una de las clases que tengamos.
- **+ roc_auc_score(y_true,y_pred,average):** Calcula el valor ROC. Hay que tener en cuenta que este valor se calcula únicamente para etiquetados binarios, por lo que se ha tenido que hacer una función que permite traspasar el valor de la funciona ROC a un problema con más de 1 clase.

  Para definir esta métrica, partiremos de la siguiente imagen:



Como podemos ver, es una función en la cuál influye en valor de *precision* en el eje Y y de FPR en el eje X.

$$\text{FPR}= \frac{FP}{FP+TN}$$

Por lo que buscamos un valor elevado de *precision* que nos diría el porcentaje de acierto de la clase positiva, y un valor muy bajo de *FPR* que nos dice el porcentaje de fallo en la clase positiva.

Con esto conseguimos afinar el acierto y reducir el fallo.

Capítulo 6

Pruebas

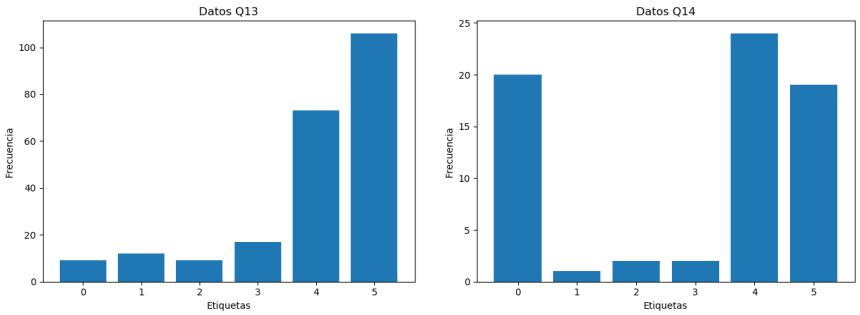
En este capítulo expondremos los resultados de los distintos diseños que hemos montado que se pueden recordar en apartados anteriores.

Haciendo un breve resumen, tenemos 5 diseños distintos los cuales compararemos con el caso base y caso techo explicados anteriormente. Comentaremos los resultados obtenidos y el proceso seguido en cada uno de los casos intentado mostrar de la forma más clara posible los resultados para su fácil entendimiento.

6.1. Introducción

Antes de empezar con los casos de diseño, vamos a hacer un pequeño recordatorio de los conjuntos de datos que tenemos, para así no tener que estar comentándolo en cada uno de los casos.

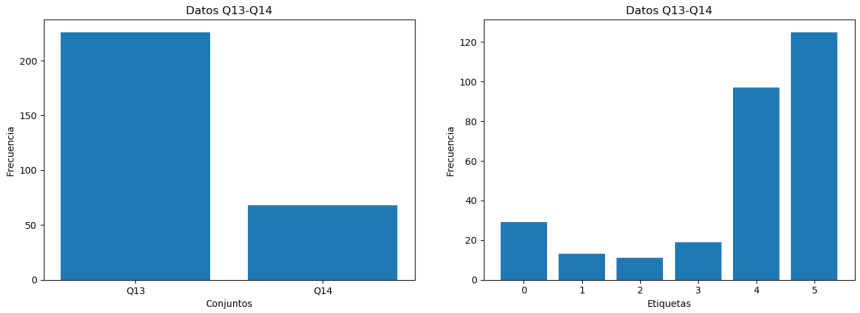
Partimos de dos conjuntos de datos, **Q13** y **Q14** compuestos por **226 elementos** y por **68 elementos** respectivamente.



Los datos están etiquetados de “0” a “5” donde “1” indica el nivel de satisfacción mas bajo, “5” el nivel de satisfacción más alto, “0” respuestas

que no expresan nivel de satisfacción y "3" que expresa una satisfacción media.

Como conjunto de datos completo, tenemos un conjunto compuesto por **294 elementos**.



Como vemos, tenemos un conjunto que tiene una escasez considerablemente alta sobre todo el valores con etiqueta "1" y "2" que indican la clase negativa.

6.2. Caso de diseño 1: Pregunta 13

Como explicamos en el desarrollo del diseño, tenemos 3 algoritmos distintos para ejecutar en este conjunto de datos (SVM, Naive Bayes Multinomial, Naive Bayes Complement).

Debido a la escasez de los datos, decidimos hacer *LeaveOneOut* sobre el conjunto de datos completo y al finalizar obtener un conjunto de etiquetas predichas de igual tamaño al conjunto original. Con esta salida calcularemos las distintas métricas para mostrarlas y visualizarlas.

En primer lugar, se sigue un proceso de procesado de textos como ya se ha indicado anteriormente, en el cual tokenizamos las frases, eliminamos las *stop-words*, le pasamos un Stemmer y vectorizamos los datos.

Para este caso en concreto, no vamos a realizar preprocesado de datos debido a que al realizar *LeaveOneOut*, se hace muy pesado y lento realizar el preprocesado de datos para los datos de entrenamiento en cada una de las particiones, ya que tenemos un número de particiones igual al número de datos.

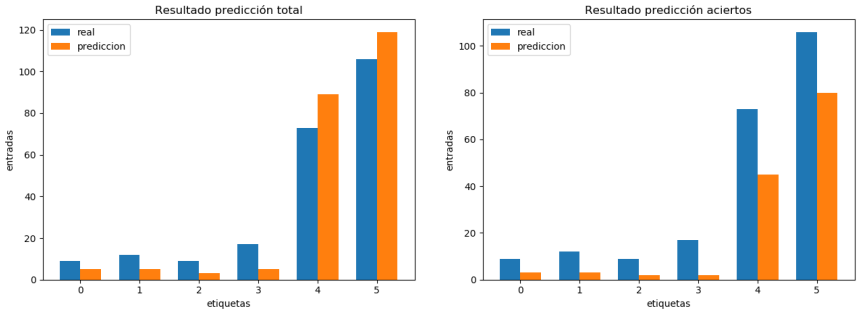
Cuadro 6.1: Resultados experimentos Q13

Q13	Accuracy	ROC	F1	Precision	Recall	Tiempo entrenamiento (s)
SVM	0.597	0.675	0.576	0.59	0.6	13.42
NB Multinomial	0.575	0.679	0.567	0.57	0.58	0.21
NB Complement	0.527	0.647	0.521	0.53	0.53	0.21
Caso base	0.469	0.5	0.299	0.22	0.47	0

Podemos ver que obviando Naive Bayes Complement y el caso base, el resto de algoritmos tienen unos valores muy parecidos, no obstante, como SVM tiene algo mejores los valores en general, decidimos usar SVM como modelo si el problema sólo constase de este proceso. Mostraremos la matriz de confusión para comentarla un poco.

Obviamente todos los algoritmos mejoran al caso base, sobre todo en lo que hemos hecho más esfuerzo en el entrenamiento es en la mejora de la métrica F1 ya que al tener tan pocos datos y estar desbalanceados nos va a ofrecer un mejor algoritmo si mejoramos esa métrica en vez de el *accuracy* por ejemplo.

Respecto al caso techo, sabemos que obtuvimos un valor de 0.681 en acuerdo observado (accuracy), como podemos ver, no hemos sido capaces de mejorar el acuerdo entre dos personas, aunque hemos estado muy cerca, no obstante, es una marca que sabemos que es difícil de superar debido a que el acuerdo entre humanos debe ser superior en estos experimentos al valor que devolverá el algoritmo.



Matriz de confusión de SVM:

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 4 & 2 \\ 0 & 3 & 1 & 2 & 3 & 3 \\ 0 & 1 & 2 & 1 & 2 & 3 \\ 1 & 0 & 0 & 2 & 10 & 4 \\ 1 & 0 & 0 & 0 & 45 & 27 \\ 0 & 1 & 0 & 0 & 25 & 80 \end{bmatrix}$$

Podemos apreciar varias cosas, una de ellas es que el algoritmo parece

haber aprendido bastante claro cuando un comentario positivo es realmente positivo, es decir un comentario que es "4" o "5" no suele fallar y ponerlo como comentario negativo. Sin embargo podemos ver como sí que le cuesta diferenciar dentro de los positivos, cuál sería "4" y cuál sería "5".

Por lo general, vemos que sobre todo le cuesta predecir la clase negativa, esto es claramente porque tenemos un conjunto tan minúsculo de éstos datos, que el algoritmo le cuesta bastante aprender ésto y opta por predecir más siempre a las clases más comunes para así fallar menos.

Vamos a ver un par de ejemplos de clasificación errónea para comprobar los posibles fallos que está teniendo.

- Etiqueta real = 5; Etiqueta predicha = 1: "me gusta mucho, es intuitivo".

  Aunque claramente este comentario es positivo, el algoritmo no ha sido capaz de clasificarlo. No obstante, es el único fallo respecto a positivo negativo de todas las etiquetas "5" por lo que no podemos tomarlo como un fallo crítico.

- Etiqueta real = 1; Etiqueta predicha = 5: "no me gusta esta asignatura", "de bajón", "pfffffff esto es muy difícil".

  Como podemos comprobar, son comentarios cortos de apenas 3 palabras clave que aunque podrían indicar negatividad, debido a la falta de datos de este tipo y por lo tanto a la aparición de estas palabras, pueden verse influenciadas por otras palabras que aparezcan en los elementos positivos que a su vez son más y por lo tanto se realice la confusión.

6.3. Caso de diseño 2: Pregunta 14

Al igual que el caso anterior, seguimos ahora replicando el caso pero para la pregunta 14.

Debido a la escasez de los datos, decidimos hacer *LeaveOneOut* sobre el conjunto de datos completo y al finalizar obtener un conjunto de etiquetas predichas de igual tamaño al conjunto original. Con esta salida calcularemos las distintas métricas para mostrarlas y visualizarlas.

En primer lugar, se sigue un proceso de procesado de textos como ya se ha indicado anteriormente, en el cuál tokenizamos las frases, eliminamos las *stop-words*, le pasamos un Stemmer y vectorizamos los datos.

Para este caso en concreto, no vamos a realizar preprocesado de datos debido a que al realizar *LeaveOneOut*, se hace muy pesado y lento realizar

el preprocesado de datos para los datos de entrenamiento en cada una de las particiones, ya que tenemos un número de particiones igual al número de datos.

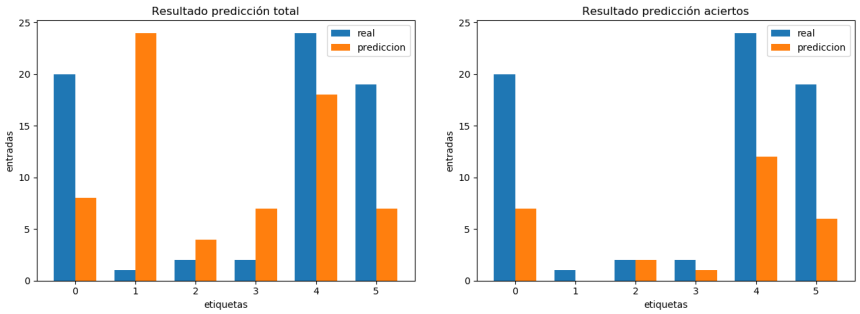
Cuadro 6.2: Resultados experimentos Q14

Q14	Accuracy	ROC	F1	Precision	Recall	Tiempo entrenamiento (s)
SVM	0.559	0.668	0.531	0.59	0.56	0.18
NB Multinomial	0.441	0.669	0.509	0.67	0.44	0.03
NB Complement	0.412	0.6715	0.504	0.75	0.41	0.04
Caso base	0.279	0.5	0.122	0.08	0.28	0

En este caso tenemos más diferencia entre SVM y los Naive Bayes, ésta vez, nos hemos decantado por Naive Bayes Complement debido a que aunque tenga un *accuraccy* menor, consigue una curva ROC mejor, que nos permite poder clasificar elementos de la clase negativa, cosa que SVM no consigue clasificar correctamente ni 1 elemento de la clase negativa. También tenemos que tener en cuenta que este conjunto es mucho menor incluso que el primero, no tenemos apenas datos negativos en el conjunto, lo que hace que acertarlos ya sea un logro.

Obviamente todos los algoritmos mejoran al caso base, sobre todo en lo que hemos hecho más esfuerzo en el entrenamiento es en la mejora de la métrica F1 y ROC ya que al tener tan pocos datos y estar desbalanceados nos va a ofrecer un mejor algoritmo si mejoramos esa métrica en vez de el *accuracy* por ejemplo.

Respecto al caso techo, sabemos que obtuvimos un valor de 0.764 en acuerdo observado (accuracy), como podemos ver, no hemos sido capaces de mejorar el acuerdo entre dos personas, y ésta vez si nos hemos quedado bastante más lejos que en el caso anterior. Aunque el acuerdo entre las personas es más alto, hay un número de datos prácticamente nulo en algunos casos como en la etiqueta "1" por lo que es entendible que una máquina baje este porcentaje en algunos niveles.



Matriz de confusión Naive Bayes Complement:

$$\begin{bmatrix} 7 & 11 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 6 & 1 & 4 & 12 & 1 \\ 0 & 7 & 0 & 1 & 5 & 6 \end{bmatrix}$$

Ahora tenemos el caso un poco distinto al anterior, se ve cómo ahora le ha dado más importancia y prefiere clasificar negativo en casos de duda, ya que si vemos la matriz, se equivoca más poniendo casos positivos como negativos que a la inversa. Otro problema que tiene es al diferenciar comentarios que no aportan información, ya que mayoritariamente los clasifica como negativos.

Hemos de tener en cuenta que con un número de datos tan bajo es complicado que el algoritmo llegue a aprender correctamente, si nos fijamos en este caso concreto, únicamente tenemos un comentario con la etiqueta de valor "1", por lo que en casos futuros, es lo más probable que no sea capaz de entender qué significa que un comentario tenga la etiqueta "1".

Está claro que al no conocer qué es "1" se apoya en ello para etiquetar algunos de los comentarios, ya que no sabe qué es "1" y por lo tanto no sabe como clasificar ese tipo de elementos. Para poder clasificar correctamente no sirve únicamente en saber como clasificar un elemento, si no también en saber cómo no clasificarlo. Como podemos ver, el número de fallos que comete el algoritmo se centran en la etiqueta "1".

Vamos a ver un par de ejemplos de clasificación errónea para comprobar los posibles fallos que está teniendo.

- Etiqueta real = 5; Etiqueta predicha = 1: "Entretenida y rápida un placer haberla hecho", "Mu buena".

  Podemos ver como el algoritmo claramente esta orientado a etiquetar como negativo debido a los pocos datos que tenemos y así poder igualar fuerzas con los positivos. Éstos comentarios se ven que son claramente positivos y el algoritmo no es capaz de reconocerlos, una de las razones puede ser por el peso que tengan puestas las palabras que aparecen en estos textos, que ya vimos en su momento que era prácticamente nulo, por lo que puede ser que apoye mucho a las palabras negativas y muy poco a las positivas haciendo que un comentario positivo como éstos pueda ser tomado como negativo.

- Etiqueta real = 0; Etiqueta predicha = 1: "Es gracioso responder a esto antes de empezar el curso", "¿Para qué se realiza dicha encuesta?", "Estudios que estoy realizando Educación Infantil".

Aunque alguno de estos comentarios puede ser medianamente interpretable por negativo como algunas personas, éstos comentarios principalmente sí es cierto que no expresan un nivel de satisfacción, lo que su etiqueta debe de ser "0", mientras que el algoritmo lo está clasificando como negativa.

6.4. Caso de diseño 3: Pregunta 13 + Pregunta 14

Al igual que el caso anterior, seguimos ahora replicando el caso pero para este caso concreto, agruparemos los dos conjuntos de datos que tenemos en uno sólo para que se complementen.

Debido a la escasez de los datos, decidimos hacer *LeaveOneOut* sobre el conjunto de datos completo y al finalizar obtener un conjunto de etiquetas predichas de igual tamaño al conjunto original. Con esta salida calcularemos las distintas métricas para mostrarlas y visualizarlas.

En primer lugar, se sigue un proceso de procesado de textos como ya se ha indicado anteriormente, en el cuál tokenizamos las frases, eliminamos las *stop-words*, le pasamos un Stemmer y vectorizamos los datos.

Para este caso en concreto, no vamos a realizar preprocesado de datos debido a que al realizar *LeaveOneOut*, se hace muy pesado y lento realizar el preprocesado de datos para los datos de entrenamiento en cada una de las particiones, ya que tenemos un número de particiones igual al número de datos.

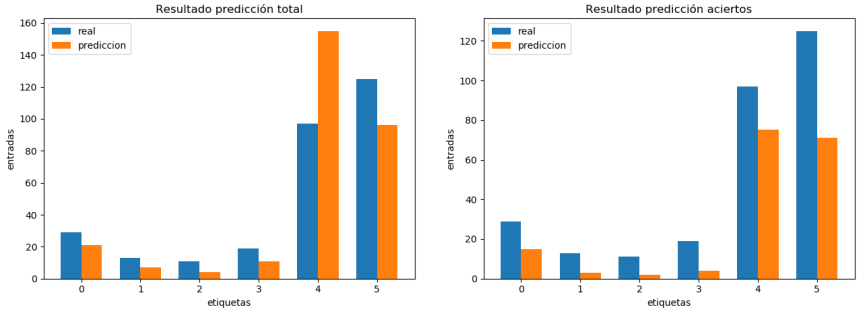
Cuadro 6.3: Resultados experimentos Q13+Q14

Q13Q14	Accuracy	ROC	F1	Precision	Recall	Tiempo entrenamiento (s)
SVM	0.578	0.688	0.569	0.61	0.58	30.42
NB Multinomial	0.551	0.688	0.559	0.57	0.55	0.42
NB Complement	0.544	0.6672	0.54	0.54	0.54	0.5
Caso base	0.425	0.5	0.254	0.18	0.43	0

Tenemos ahora un "claro" ganador ya que SVM supera prácticamente en todas las métricas al resto de algoritmos, por ello, decidimos seleccionar este algoritmo para mostrar los resultados obtenidos con el mismo.

Obviamente todos los algoritmos mejoran al caso base, sobre todo en lo que hemos hecho más esfuerzo en el entrenamiento es en la mejora de la métrica F1 y ROC ya que al tener tan pocos datos y estar desbalanceados nos va a ofrecer un mejor algoritmo si mejoramos esa métrica en vez de el *accuracy* por ejemplo. Como en este caso, tenemos un número elevado de elementos de la clase mayoritaria, tenemos incluso un *accuracy* algo alto para ser el algoritmo que es, por eso tenemos en cuenta las demás metricas que nos permiten ajustar más el algoritmo a lo que realmente necesitamos.

Respecto al caso techo, hemos calculado el valor, obteniendo 0.70 en acuerdo observado (accuracy), como podemos ver, no hemos sido capaces de mejorar el acuerdo entre dos personas, ni mucho menos, ya que nos encontramos a una distancia bastante notable del porcentaje de acierto respecto a los datos. Sabemos que es difícil superar la concordancia humana y por ello es el caso techo. No obstante, nuestro objetivo marcado no es mejorar el *accuracy* ya que le damos preferencia a intentar predecir correctamente elementos de todas las clases teniendo en cuenta la distribución de las mismas.



Matriz de confusión SVM:

$$\begin{bmatrix} 15 & 0 & 0 & 0 & 11 & 3 \\ 0 & 3 & 1 & 3 & 4 & 2 \\ 1 & 1 & 2 & 1 & 6 & 0 \\ 1 & 0 & 0 & 4 & 12 & 2 \\ 2 & 1 & 1 & 0 & 75 & 18 \\ 2 & 2 & 0 & 3 & 47 & 71 \end{bmatrix}$$

Podemos ver como en este caso, tenemos una mezcla de los dos casos anteriores, por una parte tenemos que por lo general el algoritmo tiende a etiquetar valores positivos que negativos, ya que seguimos teniendo un conjunto de datos donde prevalecen éstos valores. No obstante, como también tenemos los valores de Q14, se equilibra y en algunos casos tiende a clasificar los elementos positivos como negativos pero en comparación con lo anterior, mucho menos común.

Al igual que en los casos anteriores suele clasificar bastante más elementos de los que luego realmente son para cada una de las clases, es normal que por ejemplo valores con etiqueta “2” sea complejo su aprendizaje y predicción ya que de un conjunto de 294 elementos, que ya es por sí son escasos, sólo tenemos 11 casos con etiqueta “2”, lo que hace muy complicado generalizar al algoritmo para que sea capaz de valorar estos casos tan inapreciables.

Vamos a ver un par de ejemplos de clasificación errónea para comprobar los posibles fallos que está teniendo.

- Etiqueta real = 0; Etiqueta predicha = 5: "Estudio Grado en Derecho y en las respuestas 09 y 10 he dudado donde encuadrarme y lo he hecho en ciencias sociales o del comportamiento.", "Es gracioso responder a esto antes de empezar el curso."

  Como vemos, realmente son comentarios que no indican un nivel de satisfacción, no obstante el algoritmo determina que son un comentario muy bueno. Este error es debido al desbalanceamiento que hace que el algoritmo se decante más y de más importancia a los elementos que componen las clases positivas y esto hace que cualquier comentario parta de que es probable que sea positivo.

- Etiqueta real = 1; Etiqueta predicha = 5: "no me gusta esta asignatura" , "no le veo gran utilidad".

  Como vemos, tenemos dos casos el cual cada uno de ellos pertenece a una pregunta distinta, aunque los dos expresen un nivel de satisfacción similar, el vocabulario utilizado para indicarlo es algo distinto, esto puede confundir al algoritmo ya que tenemos dos tipos de formas de expresar la negatividad o la positividad y esto puede dificultar el aprendizaje.

- Etiqueta real = 5; Etiqueta predicha = 1: "Me veo con muchos ánimos", "me gusta mucho, es intuitivo".

  A la vista está que éstos comentarios son positivos, una de las posibles de la mal clasificación puede ser la inclusión de la pregunta 14, que hace que el algoritmo en algunos de los casos le de más puntos a ser negativo que positivo como pudimos ver en el ejemplo anterior.

6.5. Caso de diseño 4: Pregunta 13 Train, Pregunta 14 Test

Ahora tenemos un caso distinto a los anteriores, vamos a utilizar una de las preguntas para entrenar el modelo y posteriormente evaluaremos sobre la otra pregunta.

Lo que realizaremos en este caso será un entrenamiento con todos los datos del conjunto de la pregunta 13 para posteriormente predecir las etiquetas de la pregunta 14.

En primer lugar, se sigue un proceso de procesado de textos como ya se ha indicado anteriormente, en el cuál tokenizamos las frases, eliminamos las *stop-words*, le pasamos un Stemmer y vectorizamos los datos.

Por otra parte, haremos también un preprocesado de datos, primero haciendo una eliminación de características utilizando PCA y Random Forest. Seguidamente, haremos otro experimento añadiendo a lo anterior una eliminación de instancias eliminando el ruido.

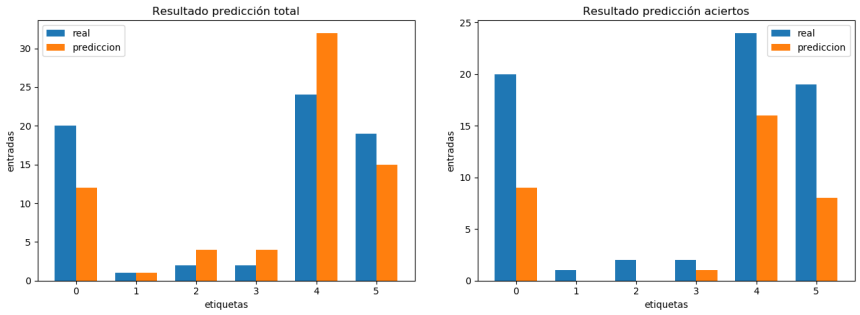
Cuadro 6.4: Resultados experimentos Q13 Train Q14 Test

Q13Q14	Accuracy	ROC	F1	Precision	Recall	Instancias	Características	Tiempo entrenamiento (s)
SVM	0.426	0.589	0.419	0.54	0.43	226	687	0.075
NB Multinomial	0.5	0.655	0.508	0.55	0.50	226	687	0.002
NB Complement	0.485	0.642	0.495	0.53	0.49	226	687	0.001
SVM	0.471	0.61	0.443	0.57	0.47	226	92	0.008
NB Multinomial	0.456	0.615	0.452	0.54	0.46	226	92	0.001
NB Complement	0.471	0.617	0.453	0.53	0.47	226	92	0.001
SVM	0.426	0.593	0406	0.56	0.43	174	75	0.004
NB Multinomial	0.485	0.627	0.454	0.54	0.49	174	75	0.0006
NB Complement	0.471	0.618	0.435	0.58	0.47	174	75	0.001
Caso base	0.279	0.5	0.122	0.08	0.28	226	687	0

Aunque tenemos ahora un conjunto más grande de posibles combinaciones a elegir, aunque reduzcamos las dimensiones o las instancias, debido a los pocos datos que tenemos, no mejoramos prácticamente nada en tiempo y como en métricas tampoco mejoramos nada, tenemos finalmente que el algoritmo que se pone en cabeza para este experimento es Naive Bayes Multinomial, utilizando todos los datos al completo.

En este caso tenemos una mejora bastante significativa sobre el caso base, ya que tenemos un caso base muy pobre en este problema, lo que nos dice que almenos comparando con el caso base hemos podido dar un salto de mejora y poder predecir los datos.

Como se puede comparar, no obtenemos unos datos muy buenos si lo comparamos al apartado donde sólo utilizábamos la pregunta 14 para evaluarse a sí misma. Esto nos dice que no tenemos una relación directa entre una pregunta y la otra a la hora de expresar el nivel de satisfacción, aunque las dos preguntas expresen lo mismo, se utiliza lenguaje diferente para expresarlo.



Matriz de confusión Naive Bayes Multinomial:

$$\begin{bmatrix} 9 & 0 & 2 & 2 & 4 & 3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 2 & 0 & 16 & 3 \\ 0 & 0 & 0 & 1 & 10 & 8 \end{bmatrix}$$

Como vemos, le cuesta bastante predecir elementos, ya que incluso algunas de las clases no es capaz de predecir ni 1 valor (aunque en alguna clase únicamente exista 1, lo que dificulta ésto inmensamente). Ésta vez tiene problema para clasificar incluso las etiquetas de valor "5" que son las etiquetas que normalmente suele estar bien clasificadas. Viendo los resultados concluiría que aunque las preguntas expresen lo mismo, no se expresa de la misma forma en cada una de ellas, lo que hace que el vocabulario sea distinto y cueste clasificar viendo sólo una de ellas para entrenar y clasificando valores de otras preguntas.

Vamos a ver un par de ejemplos de clasificación errónea para comprobar los posibles fallos que está teniendo.

- Etiqueta real = 2; Etiqueta predicha = 4: "debería profundizar un poco más", "La encuesta debería incluir mas preguntas acerca de los contenidos del curso".

  Como vemos son comentarios en los que no aparecen palabras clave negativas, esto puede hacer que cómo tenemos muy pocos datos en los que fijarnos, no pueda ayudar al algoritmo a aprender ésto y como las palabras son muy poco significativas respecto al texto como vimos en apartados anteriores, puede dificultar el aprendizaje.

6.6. Caso de diseño 5: Pregunta 14 Train, Pregunta 13 Test

Como en el caso anterior, vamos a entrenar utilizando una pregunta de las que disponemos y evaluaremos con la otra. Haremos el mismo caso anterior, pero cambiando las preguntas.

Lo que realizaremos en este caso será un entrenamiento con todos los datos del conjunto de la pregunta 14 para posteriormente predecir las etiquetas de la pregunta 13.

En primer lugar, se sigue un proceso de procesado de textos como ya se ha indicado anteriormente, en el cuál tokenizamos las frases, eliminamos las

stop-words, le pasamos un Stemmer y vectorizamos los datos.

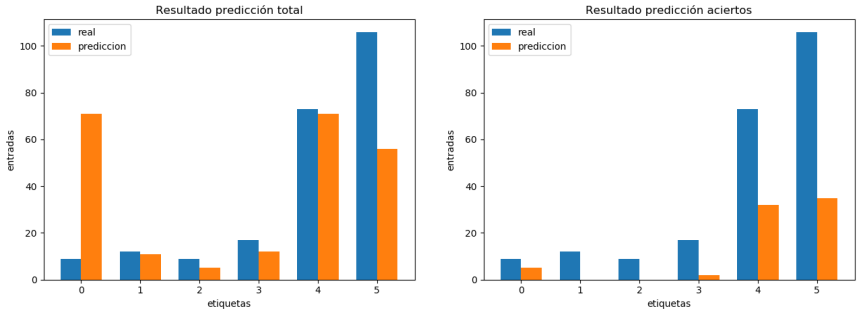
Por otra parte, haremos también un preprocesado de datos, primero haciendo una eliminación de características utilizando PCA y Random Forest. Seguidamente, haremos otro experimento añadiendo a lo anterior una eliminación de instancias eliminando el ruido.

Cuadro 6.5: Resultados experimentos Q14 Train Q13 Test

Q13Q14	Accuracy	ROC	F1	Precision	Recall	Instancias	Características	Tiempo entrenamiento (s)
SVM	0.30	0.544	0.297	0.42	0.30	226	687	0.003
NB Multinomial	0.314	0.554	0.33	0.42	0.31	226	687	0.001
NB Complement	0.327	0.572	0.362	0.45	0.33	226	687	0.0006
SVM	0.367	0.543	0.295	0.40	0.037	226	92	0.0009
NB Multinomial	0.389	0.549	0.329	0.42	0.39	226	92	0.0005
NB Complement	0.372	0.556	0.344	0.45	0.37	226	92	0.001
SVM	0.394	0.559	0.277	0.56	0.39	174	75	0.0007
NB Multinomial	0.380	0.55	0.285	0.44	0.38	174	75	0.0009
NB Complement	0.381	0.553	0.268	0.55	0.38	174	75	0.0005
Caso base	0.469	0.5	0.299	0.22	0.47	226	687	0

Con estos datos, podemos concluir que este modelo no es factible por ningún motivo. Cualquier algoritmo escogido de la experimentación muestra un nivel de desacierto tan elevado que se acerca incluso al caso base escogido en el proyecto.

No obstante, vamos a mostrar las gráficas y la matriz de confusión de Naive Bayes Complement con todos los datos sin preprocesamiento.



Matriz de confusión Naive Bayes Complement:

$$\begin{bmatrix} 5 & 0 & 0 & 0 & 3 & 1 \\ 6 & 0 & 0 & 0 & 2 & 4 \\ 6 & 0 & 0 & 0 & 3 & 0 \\ 8 & 0 & 1 & 2 & 3 & 3 \\ 17 & 4 & 2 & 5 & 32 & 13 \\ 29 & 7 & 2 & 5 & 28 & 35 \end{bmatrix}$$

Como podemos comprobar al entrenar con la pregunta 14, donde tenemos un número grande de 0, los algoritmos se han sobreajustado al conjunto de entrenamiento y al predecir la otra pregunta abundan el número de "0"

que predice cuando apenas tenemos valores con etiqueta “0”. Como he comentado anteriormente, este experimento en concreto da unos resultados que no se pueden dar como aceptables.

Capítulo 7

Conclusiones

El problema a resolver es un problema que hoy en día está muy cotizado, por ello, creo que la importancia de resolverlo es muy relevante para el futuro de la tecnología y de las empresas en general. Partía de poca base de conocimiento relacionada con este tema y gracias a este proyecto he conseguido aprender y profundizar en este tema que me parece bastante interesante y útil para las personas.

Partíamos de una base de datos escasa y con poca información como se ha podido comprobar durante el desarrollo del proyecto, no obstante para los distintos casos que hemos realizado y teniendo en cuenta los casos base y techo, tenemos algunos resultados que pueden ser interesantes a nivel experimentación ya que al partir de este conjunto, los resultados pueden no ser los esperados en resultado global.

Si seguimos este procedimiento para un conjunto que generalice más y que sea más extenso, podemos llegar a realizar esta clasificación de una forma bastante satisfactoria, ya que como hemos podido comprobar los algoritmos llegan a entender lo que queremos, sólo que en nuestro caso no terminan de generalizar debido a los pocos ejemplos de los que dispone.

El proceso para poder solucionar estos problemas tiene varios escalones críticos de los cuales depende el problema, tenemos que procesar y transformar el texto, procesar los datos y prepararlos para facilitar el trabajo a los algoritmos y luego entrenar. Cada uno de estos pasos se pueden realizar con distintas combinaciones y debemos de tener cuidado ya con cualquier fallo en alguno de los pasos puede hacer que el algoritmo de aprendizaje no sea capaz de interpretar los resultados después de estos procesos.

Respecto al problema, hemos conseguido crear algún modelo que puede defenderse respecto a nuevas entradas si se pusiese en funcionamiento, no obstante, la mejor opción sería seguir recogiendo información y añadírsela al modelo para que pueda seguir aprendiendo y mejorar así su ajuste.

Como conclusión final y generalizada sobre el problema creo que seguir investigando sobre éstos tipos de problemas y conseguir que los algoritmos sean capaces de entender cómo nos expresamos o lo que queremos expresar puede ser un avance muy grande de cara a facilitar los comentarios masivos en todo Internet, ya sea en cursos como nuestro caso, como en redes sociales, etc... . Pudiendo ayudar y mejorar la experiencia de usuario de cada uno de los servicios de los que nosotros los usuarios partícipes.

Bibliografía

[1] A. Regalado, "The most important education technology in 200 years", MIT Technology Review, 2012.

[2] M. V. Mäntylä, D. Graziotin y M. Kuutila, "The evolution of sentiment analysis - a review of research topics, venues, and top cited papers", CoRR, vol. abs/1612.01556, 2016.

[3] B. Liu, "Sentiment analysis and opinion mining", 1, vol. 5, Morgan & Claypool Publishers, 2012, págs. 1-167.

[4] B. Pang y L. Lee, "Opinion mining and sentiment analysis", Found. Trends Inf. Retr., vol. 2, n.o 1-2, págs. 1-135. 2008.

[5] A. M. Kaplan y M. Haenlein, "Higher education and the digital revolution: About moocs, spocs, social media, and the cookie monster", Business Horizons, vol. 59, n.o 4, págs. 441-450, 2016.

[6] Cambria, E., & Hussain, A. (2012). Sentic computing: Techniques, tools, and applications (Vol. 2). Springer Science & Business Media.

[7] C. M. Vizoso, Los MOOCs un estilo de educación 3.0. SCOPEO INFORME No2. MOOC: Estado de la situación actual, posibilidades, retos y futuro, 239-261, 2013.

[8] R. Artstein & M. Poesio, "Inter-Coder Agreement for Computacional Linguistics", págs. 07-14, 2007.

[9] A. R. Bartolomé & K. Steffens, ¿Son los MOOC una alternativa de aprendizaje?, 2015.

[10] Sergio Mora, Preguntas y respuestas - ¿Qué son los MOOCs?, 2012.

[11] Richard Pérez Peña, Top Universities Test the Online Appeal of Free, 2012.

[12] P. Pernías & S. Luján, Los MOOC: orígenes, historia y tipos, 2014.

[13] 20Minutos, España es el primer país europeo en producción de MOOC, los cursos 'online' abiertos, 2017.

[14] A. Lequerica, MOOCWatch Feb 2016: More Students, More Price Points, More Models, 2016.

[15] D. Lederman, MOOCs: Fewer New Students, but More Are Paying, 2018.

[16] D. Fradkin & I. Muchnik : Support Vector Machines for Classification

[17] S. Taheri & M. Mammadov : Learning the Naive Bayes classifier with optimization models,2013