MÁSTER EN DEEP LEARNING E INTELIGENCIA ARTIFICIAL

Trabajo de Fin de Máster

IMDB Rating Prediction with Deep Learning.

Rodrigo Díaz Morón

Juan Serrano Vara

2019/20

Listado de Abreviaturas

TFM - Trabajo de Fin de Máster

IA - Inteligencia Artificial

IMDB - Internet Movies DataBase

EEUU - Estados Unidos

MLP - Multilayer Perceptron

CNN -Convolutional Neural Network

EE - Entity Embeddings

Índice

1 Introducción 1

1.1. Estado del Arte 3

2 Descripción y Preprocesamiento de los datos. 5

2.1. Descripción 5

2.2. Preprocesamiento 6

3 Metodología 9

3.1. Scraper 9

3.2. Definición de la variable objetivo 10

3.3. Métricas de la capacidad de los modelos 12

3.4. Baselines 13

3.4.1. Árboles de decisión 13

3.4.2. Random forest 13

3.4.3. Adaboost y gradientboosting 13

3.4.4. SVM 14

3.4.5. Nearest centroid classifier 14

3.4.6. k-nearest neighbors algorithm 14

3.5. Entrenamiento con generador para las imágenes y los datos tabulares. 14

3.6. Clasificador para datos tabulares. 16

3.6.1. Embeddings 16

3.7. Clasificador para imágenes 17

3.8. Clasificador para datos tabulares e imágenes 18

3.9. Clasificador para textos (Título y Descripción) 18

3.9.1. Clasificador por descripción 19

3.9.2. Clasificador por títulos 19

4 Resultados y conclusiones 20

4.1. División del dataset 20

4.2. Modelos Machine Learning (Baselines) 20

4.3. Modelos Deep Learning 21

4.3.1. Clasificador de título 23

4.3.2. Clasificador de descripciones 24

4.3.3. Clasificador de datos tabulares 26

4.3.4. Clasificador de imágenes 28

4.3.5. Clasificador de tabulares + imágenes 28

4.3.6. Comparativa modelos Deep Learning 29

4.4. Comparativa entre los mejores modelos de DL y Baselines 30

4.5. Aplicación web 30

4.5.1. Manual de usuario 30

4.6. Conclusiones y trabajo futuro 32

5 Bibliografía 34

6 Anexos 1

6.1. Anexo A 1

6.2. Anexo B 2

6.3. ANEXO C 1

1 Introducción

El cine es una de las actividades de ocio y entretenimiento más extendidas a lo largo y ancho del planeta, penetrando en prácticamente todas las sociedades y en todos los grupos sociales. ¿Quién no ha visto alguna película que le ha marcado o ha renovado su visión sobre algún tema? Desde nuestros abuelos o abuelas hasta los típicos amigos "raritos" a los que no les gusta nada que goce de popularidad, te podrán responder casi con total seguridad cual es su película favorita: 'Aquella que vi con tu abuelo con 22 años la primera vez que vi una pantalla' o 'esta película de culto que no ha visto nadie más que su director y yo' pasando por la última película de animación de moda que encantará al primer niño que preguntes, serán respuestas probables. Esto demuestra que el cine no discrimina entre edades, rangos sociales o gustos, ya que es una industria lo suficientemente grande como para que cada persona pueda encontrar su mercado.

Así lo demuestran, por otro lado, las cifras de ingresos asociadas a esta industria. Según The Motion Picture Association of America en 2018 la industria del cine generó más 41,1 mil millones de dólares [1]. No obstante, la industria se enfrenta a numerosos retos: las salas de cine se vacían como informan numerosos medios cada año (si bien es cierto que es debido a que mucha parte del público prefiere cada vez más plataformas digitales) lo que está revolucionando la forma de ofrecer este entretenimiento, planteando un cambio de forma más que de contenido. Pero también, en los últimos tiempos, ha sufrido un adelantamiento o acercamiento por parte de otras industrias del entretenimiento, como la de los videojuegos, que desde hace ya varios años factura bastante mas que la industria del cine [2] y es en España la primera opción de ocio audiovisual [3] cuestionando la capacidad del cine para entretener tanto como siempre lo ha hecho. Por otro lado, dentro de la propia industria, también se están produciendo grandes cambios en cuanto al contenido ofrecido: las series le están ganando la batalla a las películas. Esto es debido a factores como la flexibilidad presupuestaria (en una serie, si todo va bien, sigo haciendo capítulos, si no, corto) lo que permite aglutinar enormes cantidades de inversión al tener menos riesgos o que cada vez, en el mundo de la inmediatez, menos gente está dispuesta o puede dedicar 2 horas seguidas de su vida a algo, en cambio 30 o 45 min sí. Por último, esta disminución de los recursos disponibles para repartir entre todo aquel que se lance a hacer una película agudiza el problema de la incapacidad de los pequeños estudios para competir con los grandes de la industria.

Por otro lado, tantos años de industria del cine han generado una gran cantidad de datos recogidos en múltiples sitios, como la web IMDB [4]. En ella, encontramos información histórica acerca de tanto las películas y series en sí, como del impacto que han tenido estas a varios niveles.

Los autores de este trabajo apreciamos el valor de un producto audiovisual meramente expositivo, en el que el autor expone su idea de principio a fin sin interacciones ni incentivos diferentes a que simplemente te cuenten una historia, como puedan ser demostrar tu capacidad de coordinación mano-ojo en un videojuego. Que te cuenten una historia tal y como la imaginó su autor, sin personalizarla a cada uno de los receptores tal y como se hace en la actualidad en muchos ámbitos, dando como resultado el hecho de que ya solo encontramos aquello que a priori queremos encontrar, sin lugar a la sorpresa o a la decepción. También apreciamos las historias cerradas, pensadas desde el comienzo de principio a fin, en las que todo está conectado y tiene un por qué. Las series tienen como objetivo que veas un capítulo más, por lo que a menudo se alargan innecesariamente introduciendo relleno mientras siguen captando atención, para finalizar de forma casi improvisada cuando dejan de hacerlo.

Como trabajo de fin de máster para el Máster en Inteligencia Artificial y Deep Learning proponemos el desarrollo de un sistema basado en Deep Learning, que posea algún grado de capacidad a la hora de anticipar la recepción de una película, a partir de información a priori accesible antes o durante el proceso de producción de esta. El objetivo es desarrollar una aproximación a un posible sistema más completo que fuese capaz de ayudar a los productores a saber qué grado de éxito puede tener un proyecto o qué variables modificar para que este aumente.

Por todo lo comentado anteriormente, la motivación de este proyecto viene dada por:

- El gran tamaño que tiene la industria del cine, tanto a nivel de alcance social como a nivel económico. Por esto, pequeñas mejoras en su producción pueden tener un gran impacto en términos comparativos con otras industrias.
- Los nuevos retos a los que se enfrenta la industria, en concreto, las películas. Las ideas nuevas son más necesarias que nunca en los momentos más complicados.
- La disponibilidad de datos y su heterogeneidad. Una de las ventajas del Deep Learning frente a otras aproximaciones más tradicionales es su capacidad de utilizar datos muy diferentes al mismo tiempo para un mismo objetivo, a cambio de la necesidad de una alta cardinalidad. En cuanto a las películas y en concreto IMDB, se dispone de una enorme cantidad de datos en formas muy diferentes, desde imágenes hasta texto pasando por datos tabulares y numéricos.
- El interés que tenemos por el mundo del cine y, en concreto, por las películas, y la ilusión de ver si es posible aportar algo desde nuestro campo de especialización.

El sistema, en concreto, recibirá información acerca de la duración, el color, el año, el director, la clasificación del contenido, el género, el idioma, las keywords, el país, el escritor, la descripción,

los actores y el cartel de cada película (considerando todo esto información que se puede poseer antes de hacer la película) y hará una predicción clasificando la película en tres categorías posibles relacionadas con la recepción que tendrá dicha película: película mala, película normal, y película de éxito.

1.1. Estado del Arte

Este TFM consiste en la implementación de técnicas de Inteligencia Artificial para predecir el Rating que va a obtener una película en el portal IMDB en función de una serie de variables. En los siguientes párrafos se hará una revisión del estado del arte de los artículos de divulgación científica publicados en el campo de aplicación de técnicas de IA al portal IMDB.

En [5] predicen la cantidad de dinero que va a generar la película en taquilla. Consideran que el éxito de una película varía en función del dinero que se va a generar. La predicción la realizan con técnicas de Machine Learning clásicas como Logistic Regression, SVM Regression y Linear Regression obteniendo en el caso de la regresión lineal el mejor resultado con un 50% de éxito. Los datos que han empleado son datos de 1050 películas hechas en EEUU entre los años 2000 al 2012 con las siguientes variables: actor, director, escritor, género, empresa que ha realizado la producción, presupuesto, imdb rating, número de gente que ha votado, metascorerating y varias variables extraídas de la web de valoración de películas rottentomatoes.com. En [6] predicen el rating que va a obtener una película en función de una serie de variables (ingresos generados por la película, año de lanzamiento, duración de la película y número de votos). Los mejores resultados los obtienen con random forest y alcanzan un 90 % de precisión. Estos son los papers más influyentes que hemos encontrado en el campo de aplicación estudiado en este TFM, podemos encontrar también trabajos en Medium o Github que siguen la misma línea que los dos trabajos mencionados anteriormente.

Nuestra manera de afrontar el problema es diferente a la mencionada en el primero de papers anteriores ya que partimos de la premisa que una película será considerada como buena o mala en función de la nota media que le den los usuarios del portal IMDB, por lo cual la variable a predecir en nuestro TFM es el Rating. Respecto al segundo paper discrepamos en la medida que utilizan como variable de entrada a la red el número de votos y el ingreso que genera la película, nosotros consideramos que son variables que obtienes una vez lanzada la película por lo cual estaríamos introduciendo información que contiene información de la salida de la red en los datos que se usan para predecir. Es decir, nuestro caso de uso sería el siguiente: Una productora quiere saber qué nivel de aceptación por el público tendrá su película y para ello deberá ingresar a la red todas las variables que están en su mano de modificar, y ni el número de votos ni los ingresos en

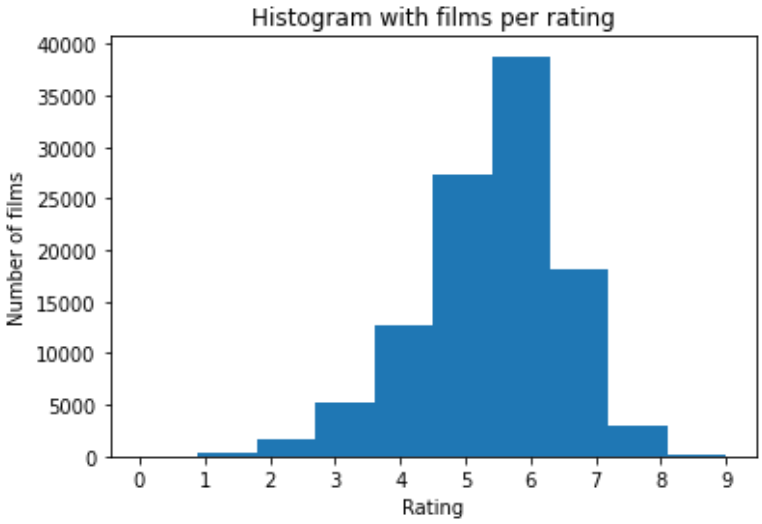
taquilla los puede seleccionar. Respecto a todos los trabajos publicados hasta la fecha nosotros utilizamos un dataset mucho más grande (aproximadamente 107000 registros) explicado en el apartado número 2 y hemos implementado tanto técnicas de Machine Learning como de Deep Learning (embeddings, MLP, CNN, RNN...). Por otro lado, también hemos estudiado la viabilidad de utilizar para predecir el rating la imágen promocional de la película, su descripción y su título.

2 Descripción y Preprocesamiento de los datos.

2.1. Descripción

Los datos utilizados en este TFM se han obtenido a través de un scraper (explicado detalladamente en el punto 3) de la página web de valoración de películas y series IMDB.
Inicialmente se "scrapearon" 542077 películas con la siguiente información:

- **Code**: Código identificativo de la película (Disponible en la web de imdb).
- **Imagen**: Imagen con la que se promociono la película. Las imágenes se han descargado de tal forma que el archivo que la contiene se llama igual que el código de la película.
- **Duración**: Duración de la película.
- **Título**: Título de la película.
- **Descripción**: Breve resumen de la película.
- **Color**: Información acerca del color de la película (blanco y negro/color).
- **Año de publicación:** Año en que se publicó la película.
- **Director**: Nombre del director de la película.
- **Número de votos:** Número de gente que ha puntuado la película.
- **Géneros**: Lista con los géneros de la película.
- **Lenguajes**: Lengua original de la película.
- **País**: País de origen de la película.
- **Keywords**: Palabras claves que identifican la película.
- **Escritores**: Nombre de los guionistas de la película.
- **Actores**: Lista de los nombres de los actores principales de la película.
- **Rating** (variable a predecir): Puntuación media entre 0-10 que le han dado los usuarios de la plataforma.

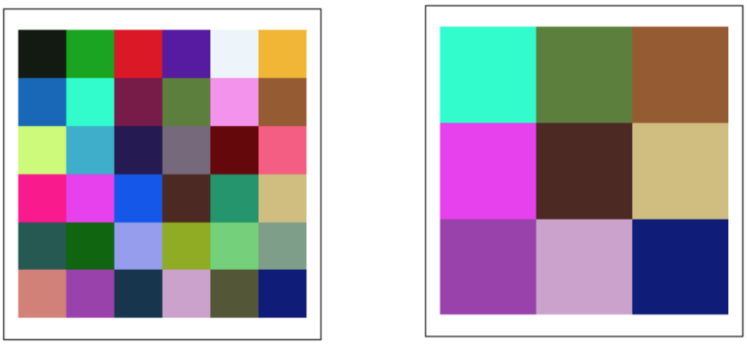


- **Content Rating:** Evaluación que le da cada país a una película para saber si es apropiada para cierto público en función de su edad.
- **Metascore Rating**: Puntuación media entre 0-10 que le han dado los usuarios de la plataforma Metascore.
- **Budget**: Presupuesto utilizado para hacer la película.
- **CWG (Cumulative Worldwide Gross):** Cantidad de dinero ganada a nivel mundial.

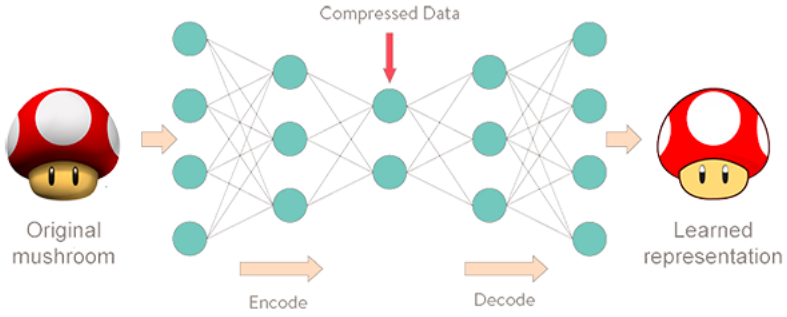
2.2. Preprocesamiento

- **Imagen**: La dimensión original de las imágenes es de 268,182,3. A través del algoritmo de escalado de imágenes Nearest Neighbor se ha reducido su dimensionalidad a 32,32,3. La elección de este tamaño ha sido debida a diversas pruebas consensuando coste computacional y rendimiento de las modelos que utilizan las imágenes.

  Ejemplo de hacer downsampling sobre una imagen a través de Nearest Neighbor:



También se probó a reducir la dimensionalidad de las imágenes a través de autoencoders, obteniendo la representación de la imagen reducida que salía del encoder.



Tras realizar varias pruebas y tras una evaluación manual con un pequeño set de datos prueba llegamos a la conclusión de que debido al coste computacional no merecía la pena reducir la dimensión de las imágenes con autoencoders.

- **Duración**: La duración se obtuvo en horas y minutos y se convirtió todo a minutos.
- **Título**: Se eliminaron todos los caracteres no relevantes.
- **Descripción**: Para el preprocesamiento de las descripciones se tomó la decisión de transformar cada texto en una palabra clave que lo identificara. Es decir, a través de Yake [7] se realizó un proceso de extracción de keywords para cada una de las descripciones de las películas. Como resultado se le añadió al DataSet una nueva categoría llamada Keyword_Description que contiene la palabra clave. Yake es una metodología no supervisada que utiliza un enfoque heurístico para extraer las Keywords en un periodo corto de tiempo a través de características estadísticas extraídas del texto.
- **Año de publicación:** Se quitaron los paréntesis que acompañaban a cada año.
- **Géneros, Idiomas y Keywords:** Al obtener una lista para cada variable seleccionamos el número de entradas que queremos para cada una. 3 géneros 1 Idioma 3 Keywords. Como no todas las películas tendrán ese mismo número de registros cambiamos los valores NaN por el primero de los valores que obtenemos, es decir, si una película no tiene los 3 géneros y solo tiene dos ponemos el primer género en la columna GENRES_3. Como resultado obtendremos tres columnas con un género cada una, una con un idioma y otras 3 con una keyword cada una.
- **Director, Escritores y Actores:** Nos vemos obligados a reducir la dimensionalidad de estas columnas debido a que al calcular el número de directores, escritores y actores que hay es un número muy elevado. Como solución solo nos quedamos con los directores, escritores y actores que han hecho más de x películas y al resto de películas sustituimos su valor por uno en concreto, por ejemplo, en el caso de los directores por JohnDoe. Es decir, todas las películas que tengan directores que hayan hecho menos de x películas su director será “JohnDoe”.

- **Content Rating:** Cada país tiene una manera diferente de identificar la categoría de cada película se buscaron manualmente la conversión a la edad, una vez hecho esto se establecieron 3 clases.
    - Categoría 1: Lo puede ver todo el mundo.
    - Categoría 2: Solo mayores de 13 años.
    - Categoría 3: Lo pueden ver solo mayores de 18 años.
- **Budget y CWG:** Las cantidades vienen indicadas en la moneda local donde se ha desarrollado la película, se han convertido todas a euros buscando de manera manual la conversión de la moneda a euros.

Una vez procesada la información se realizó un conteo de qué categorías teníamos disponibles por película. Obtuvimos que solo teníamos 54.000 con información del Budget, 15.000 con CWG y 14.000 con Metascore Rating. Por lo cual decidimos quedarnos con las películas que tuvieran la información completa de las siguientes categorías: título, descripción, duración, color, año, director, rating, actor votos, géneros, lenguaje, país de origen, keywords y escritores. Como resultado obtuvimos 113000 películas de las que descartamos las que no tenían imagen y nos quedamos con un dataset total de 107181 películas. Por último, filtramos el dataset por las películas que tenían más de 125 votos ya que consideramos que con menos de 125 votos los resultados podrían no ser fiables o estar "manipulados", quedándonos finalmente con **59.625 películas**. El valor exacto de 125 votos ha sido obtenido realizando pruebas sobre los modelos (tratado como un hiper parámetro más), determinando que en 125 votos nos quedamos con suficientes películas como para que los todos los modelos entrenen, y desechamos suficientes películas como para que la distorsión que genera meter ratings con pocos votos sea mínima.

3 Metodología

3.1. Scraper

Este proyecto requiere la mayor cantidad de películas posible, toda la información disponible de cada película, la disposición de todas las películas en un formato estructurado y accesible por Python y la posibilidad de llevar a cabo actualizaciones frecuentes de los datos usados.

Por todo esto, ha sido necesario el desarrollo de un scraper flexible, es decir, capaz de adaptarse a los posibles cambios que se produzcan en la web de modo que permita tener el historial de películas lo más actualizado posible y completo, esto es, que sea capaz de descargar toda la información disponible. Además, es preciso que guarde toda esta información de un modo estructurado y accesible para el cometido de este proyecto: hemos elegido guardar todos los datos estructurados en forma de csv.

La librería utilizada para desarrollar dicho scraper ha sido Selenium [8] debido a los siguientes factores:

- Utiliza Python como lenguaje de programación al igual que el resto del proyecto, de modo que no ha sido necesario cambiar de entorno de programación.
- Es bastante potente tanto en la descarga del contenido de las webs como en la navegación que permite a través de dichas webs.
- Existe bastante información y ejemplos en la red acerca de cómo usar esta herramienta para este cometido.
- Contamos con experiencia previa en el uso de esta herramienta.

El funcionamiento del scraper está basado en una característica concreta de IMDB: cada película tiene asociada un código (CODE) a través del cual es posible acceder directamente a la información de dicha película modificando una pequeña parte de una url. Por esta razón comenzamos consiguiendo la lista de códigos de todas las películas y después utilizamos dicha lista para descargar toda la información de cada una de ellas.

En cuanto al funcionamiento del scraper, este cuenta con tres partes muy simples:

- El código principal, que cuenta con dos funciones:
    - Recorrer la lista de todos los códigos de las películas para acceder a la información de estas una a una.

    - Abrir un csv en el que será escrita toda la información acerca de cada película que devuelva la segunda parte del código.
- Una función que se encarga de extraer y guardar en variables la información de la película que está siendo descargada. Esta función retorna toda la información a la primera. Las principales características de esta función son:
    - Se accede a cada dato que queremos descargar mediante su xpath en el código de la página.
    - Se utilizan excepciones para lidiar con la falta de información requerida, sustituyendo dicha información por la palabra 'unknown' en caso de no encontrarse.
- Una función que define el cabecero del archivo.

El resultado de este script es un dataset en formato csv que contiene absolutamente todas las películas presentes en la web en el momento de descarga de la lista de códigos de las películas con toda la información considerada relevante acerca de ellas.

3.2. Definición de la variable objetivo

Una de las decisiones más importantes que se han tomado ha sido la forma en la que se ha definido la variable objetivo o salida del modelo. El dato que se ha de predecir (el rating de la película) se encuentra en la página de donde es escrapeado en forma de número, con posibles decimales y en el rango de 1 a 10. La primera opción que barajamos fue la de afrontar el problema como un problema de regresión, en el que tratamos de predecir el rating de la película. Sin embargo, tras algunas pruebas y sondeos desechamos esta idea debido a las siguientes razones:

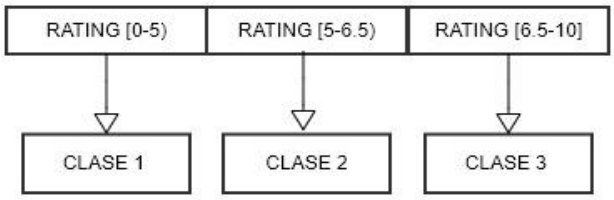
- Pensando en las características e intereses del usuario final, consideramos que no toda predicción del sistema tiene igual valor, como se comentará en el apartado de *Métrica de la capacidad de los modelos*. Esta variación en la importancia de la predicción en función de la naturaleza de esta es mucho más fácil de modelar y ajustar en un problema de clasificación que en uno de regresión.
- Debido a la naturaleza del problema, la precisión de la predicción dista de ser exacta. Por esta razón, al final acabamos por medir la capacidad del modelo viendo si la predicción se encontraba en un espacio razonable en el que también se encuentra el valor real, creándose así de forma natural varias clases.
- Pensando en el uso final de la aplicación, no es razonable pensar que alguien espere recibir el rating exacto que tendrá su película, dado que este depende de muchas variables que no son posible medir o incluso de naturaleza aleatoria. Una predicción de un rating exacto podría incluso restar credibilidad al sistema a ojos de un usuario, dado que sería

inverosímil. Sin embargo, si puede ser de utilidad de cara al usuario una valoración general acerca de si tu película tiene altas posibilidades de no gustar nada, de gustar a una parte de la población, o de tener bastante éxito gustando a la mayoría.

Una vez decidido tratar el problema como un problema de clasificación, se procedió a determinar los rangos que compondrían cada clase. Estos, finalmente, han sido [0-5) (clase 1), [5-6.5) (clase2), [6.5-10](clase3). Esta elección ha sido motivada por lo siguiente:

- En los ratings bajos es donde menos observaciones encontramos. Con el fin de obtener una distribución no excesivamente desbalanceada, el rango de la primera clase debe ser el más extenso. Entre 5 y 6.5 es donde la concentración de películas por punto de rating es mayor, por lo que el rango que abarque estas películas debe ser el menos extenso si queremos mantener dicho balanceo.
- Pensando en el usuario final (aunque siendo conscientes de un posible sesgo cultural), una película con un rating por debajo de 5 sobre 10 puede considerarse de forma consensuada como un fracaso o suspenso, una película con un rating de entre un 5 y un 6.5 como una película aceptable y una película con un rating por encima de un 6.5 como una película exitosa. Por este motivo, creemos que esta división de clases puede aportar una buena visión general intuitiva al usuario acerca de qué esperar de su película.

Por otro lado, somos conscientes del principal problema que acarrea afrontar el problema como un problema de clasificación: al tratarse de un rating, las clases resultantes no son categorías nominales sino ordinales, ya que poseen un orden. Además, el error cometido al decidir que es de clase 3 (6.5-10] una película cuyo rating real es 6.5, no es el mismo que si lo dijéramos de una película cuyo rating real es 5.1. Esto sin duda es un problema a tratar como trabajo futuro. Sin embargo, las ventajas comentadas anteriormente y los medios a nuestra disposición nos llevaron a finalmente afrontar el problema como un problema de clasificación.



3.3. Métricas de la capacidad de los modelos

Una decisión central en este proyecto ha sido la definición de la métrica utilizada para medir la capacidad de todos los algoritmos probados. Como se ha comentado con anterioridad, la importancia de las predicciones acertadas o erradas no es homogénea. Esta heterogeneidad se explica del siguiente modo:

- Desde la perspectiva del usuario final, creemos que el sistema debe minimizar los errores por exceso en la clase 1, dado que el impacto de que una película que pensemos que es buena o muy buena sea finalmente un fracaso es muy grande. Esto es debido a que este tipo de error puede ocasionar pérdidas de inversión, mientras que otro tipo de errores pueden ocasionar como mucho que dejes de ganar dinero.

- Relacionado con lo anterior, creemos que el sistema tiene que maximizar la confianza a la hora de decir que una película va a ser un éxito (clase 3), para que el usuario pueda seguir adelante con la inversión con ciertas garantías de que al menos puede recuperar su inversión.

Debido a estas razones, se ha decidido elaborar una métrica customizada que se amolde a los intereses del usuario. Describimos esta métrica del siguiente modo:

- Por la importancia de no cometer errores relacionados con predecir que la película es buena o muy buena cuando realmente es un fracaso, medimos el ***recall*** **de la clase 0,** dado que penaliza este tipo de errores.

$$R\_0 = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- Por la importancia de no cometer errores relacionados con predecir que la película es muy buena (clase 3) cuando realmente es buena o un fracaso, medimos la ***precisión*** **de la clase 3,** dado que penaliza este tipo de errores.

$$P\_2 = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- La clase 2, debe estar equilibrada por lo que medimos el f1-score **de la clase 2.**

$$F1\_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Una vez hechas estas medidas calculamos la media de las tres, definiendo así nuestra métrica.

$$Custom\_Metric = 2 * \frac{R\_0 + P\_2 + F1\_1}{3}$$

El objetivo, por tanto, será buscar modelos capaces de tener una buena solución de consenso entre nuestra métrica y el accuracy total del modelo. En general, aceptaremos un accuracy mínimo y a partir de este buscaremos la solución que maximice nuestra métrica.

3.4. Baselines

Los resultados de los modelos descritos en esta sección se encuentran en el apartado resultados. Los algoritmos utilizados como baselines son los siguientes:

3.4.1. Árboles de decisión

Los árboles de decisión son modelos cuyo objetivo es predecir la variable objetivo mediante el este modo, dado un conjunto de datos se fabrican diagramas de construcciones lógicas que sirven para representar la categorización de los datos bajo una serie de condiciones aplicadas de forma sucesiva.

3.4.2. Random forest

Random forest es un método basado en el aprendizaje mediante árboles de decisión. En concreto, Random forest es un estimador que se ajusta a una serie de árboles de decisión en varias submuestras del conjunto de datos de entrada y utiliza el resultado de combinar todos estos árboles para intentar mejorar la precisión de las predicciones o clasificaciones que cada uno de ellos haría por separado. Cada árbol de decisión depende de una serie de valores establecidos aleatoriamente.

3.4.3. Adaboost y gradientboosting

Boosted trees o árboles potenciados son un conjunto de técnicas basadas también en los árboles de decisión. Estas técnicas se basan en lo que se conoce como gradient boosting o potenciación de gradiente, haciendo uso de varios árboles de decisión para mejorar la precisión de las predicciones que se llevan a cabo. El término potenciación de gradiente se refiere al uso de diferentes técnicas que permiten optimizar y mejorar el descenso de gradiente sobre la función de coste o error, problema al cual se enfrentan todos los algoritmos de aprendizaje automático supervisado. De entre estas técnicas, las utilizadas en este estudio son la básica conocida como Gradient Boosted Classification Trees y una modificación conocida como AdaBoost.

3.4.4. SVM

Las máquinas de soporte vectorial o support vector machine (SVM) tratar de llevar a cabo la tarea de clasificación encontrando el mejor hiperplano que sea capaz de separar los puntos que representan cada uno de los datos de entrada en el espacio n-dimensional de representación, de modo que dicha separación coincida con la separación indicada mediante los datos de entrenamiento. De este modo, el modelo será capaz de decir a qué clase pertenece un nuevo punto (y llevar a cabo la tarea de clasificación) identificando a qué lado del hiperplano se encuentra. Hasta este momento, no estaríamos hablando más que de un clasificador lineal ordinario, pero SVM añade algunas características. De todos los hiperplanos que son capaces de separar dos puntos de diferentes clases, SVM busca aquel que maximice su distancia a los dos puntos más cercanos de cada lado, lo que constituiría un clasificador de margen máximo.

3.4.5. Nearest centroid classifier

aprendizaje de reglas de decisión simples inferidas a partir de las características de los datos. De El clasificador de centroide más cercano es un algoritmo de clasificación que asigna a cada una de las observaciones la etiqueta de clase del ejemplo de entrenamiento cuyo centroide se encuentre más cerca de dicha observación.

3.4.6. k-nearest neighbors algorithm

k-NN classification (la versión del algoritmo utilizada para problemas de clasificación) es un algoritmo que clasifica cada una de las observaciones mediante la información ponderada de sus *k* vecinos más cercanos, asignando a cada observación la clase más común entre estos vecinos teniendo en cuenta su proximidad.

3.5. Entrenamiento con generador para las imágenes y los datos tabulares.

En total en el dataset tenemos información de 107181 películas cargar todas esas imágenes en memoria y preprocesarlas es intratable con una memoria ram de 8G, si a esto le sumamos los datos tabulares que pertenecen a cada película para después entrenar con todos estos datos sería imposible. Ante la imposibilidad de cargar todas las imágenes en memoria para introducirlas al modelo nos hemos visto obilgados a usar generadores. Los generadores nos permiten cargar las imágenes por batches, preprocesarlas, concatenarlas con los datos tabulares que corresponden a esa película y después entrenar el modelo, una vez terminado ese batch el espacio de memoria se

libera y continua con el siguente set de imágenes. Del tal manera que solo cargamos en memoria el tamaño del batch. Como consecuencia de esto el pipeline de entrenamiento básico de tensorflow (model.fit (x_train,y_train,validation_split...)) no lo podemos usar. Nuestro entrenamiento es el siguiente:

1. Seleccionamos un tamaño del batch (intentamos que sea el máximo que cabe en memoria).
2. Establecemos el número de pasos totales para hacer un iteración de entrenamiento sobre el dataset total. Steps = Longuitud_DataSet/Batch.
3. Por cada Step hacemos un model.fit con los datos cargados de ese batch.
4. Una vez terminada la iteración se realiza el mismo proceso pero con los datos de test.

Como resultado de proceso por cada iteración obtenemos las métricas que deseamos tanto para los datos de entrenamiento como los de test

Ejemplo:

```
longuitud_dataset_train = 1000
longuitud_dataset_test = 100
batch = 10
steps_train = longuitud_dataset_train/batch
steps_test = longuitud_dataset_test/batch
iteraciones = 30

for iter in range (0,len(iteraciones)):

    for step in range (0,len(steps_train)):

        model.fit(datos_train...)

    for step_test in range (0,len(steps_test)):

        model.evaluate(datos_test...)
```

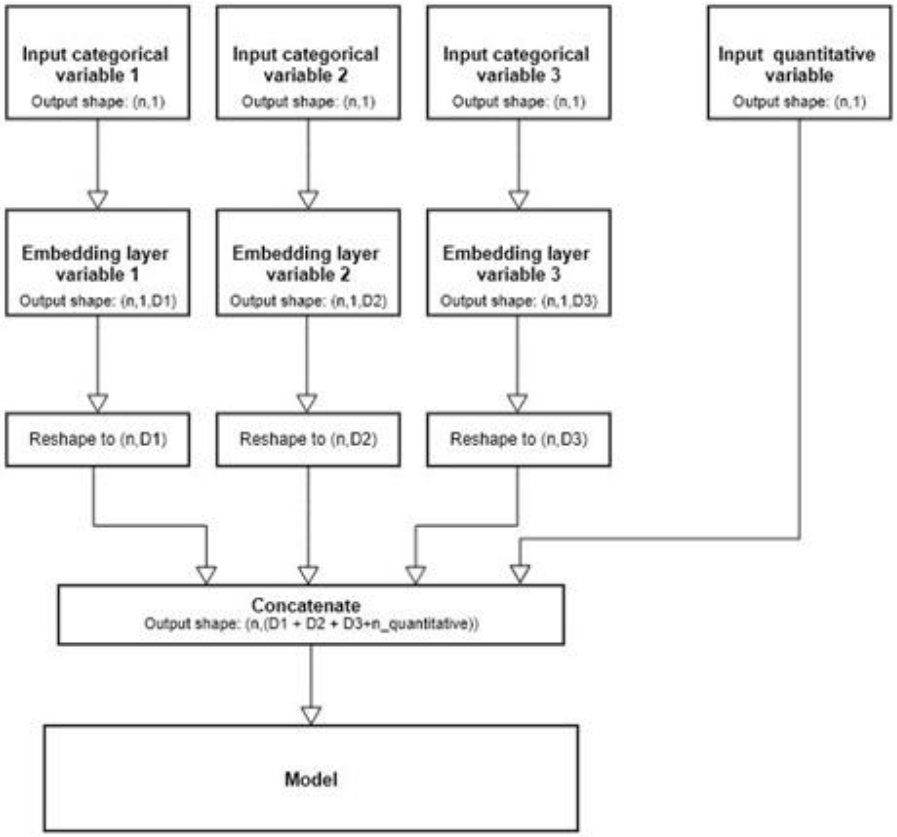
Notas:

- No hay que confundir el batch que seleccionamos para determinar la cantidad de datos en memoria que cargamos con el batch_size del model.fit() de tensorflow. Cada vez que hagamos un model.fit() hara tantos pasos como datos que hayamos cargado en memoria entre el batch_size seleccionado.
- En los ejemplos de clasificación por Rating (en los casos de uso solo con imágenes) hemos utilizado el generador de tensorflow *ImageDataGenerator- flow_from_dataframe,* por lo cual el proceso indicado con anterioridad **solo se utiliza cuando entrenamos las imágenes con los datos tabulares,** aunque la idea entre el generador de tensorflow y la que hemos implementado es la misma.
- Para el caso de clasificación solo con datos tabulares no ha sido necesario implementar ningún generador ya que todos los datos caben en memoria.

3.6. Clasificador para datos tabulares.

3.6.1. Embeddings

La codificación mediante EE se lleva a cabo mediante capas de embedding. Cada variable categórica que vaya a ser introducida al modelo pasará antes por una capa de embedding diferente, de modo que las características intrínsecas de cada variable categórica serán obtenidas de forma aislada. La dimensión de salida de cada una de estas capas es un hiper parámetro que deberá ser modificado con el objetivo de obtener los mejores resultados. Después, todas las variables codificadas serán concatenadas resultando un vector que representa la codificación completa de cada película que ha sido introducida, siendo estos vectores la entrada para el modelo que suceda a la codificación.



El funcionamiento de una capa de embedding, conceptualmente, consiste en la elaboración de una tabla en la que cada entrada diferente(ya sea un valor numérico o un string) tiene asociado un índice. El número de índices, por tanto, que posee la tabla de una capa de embedding que tiene como entrada una variable categórica es igual al número de categorías diferentes que posee dicha variable. Cada índice, a su vez, está asociado a un vector de embedding de tamaño D, que será la

representación que haga la capa de embedding de la categoría que está asociada a dicho índice. Esta es la razón por la que las capas de embedding son utilizadas para codificar variables categóricas nominales: al tener cada una de las categorías de cada variable categórica asociada una clave única, la red las acaba tratando como entidades separadas y no relacionadas (al contrario que una dense normal, que trataría cada uno de los valores que recibiera del mismo modo).

Cada vector de embedding se inicializa de forma aleatoria, y va siendo modificado conforme al error que comete el modelo en sus predicciones, de manera muy parecida a la que lo hacen las capas normales de las redes neuronales. Esta codificación se ha basado en el artículo de *Entity Embeddings of Categorical Variables* [9]

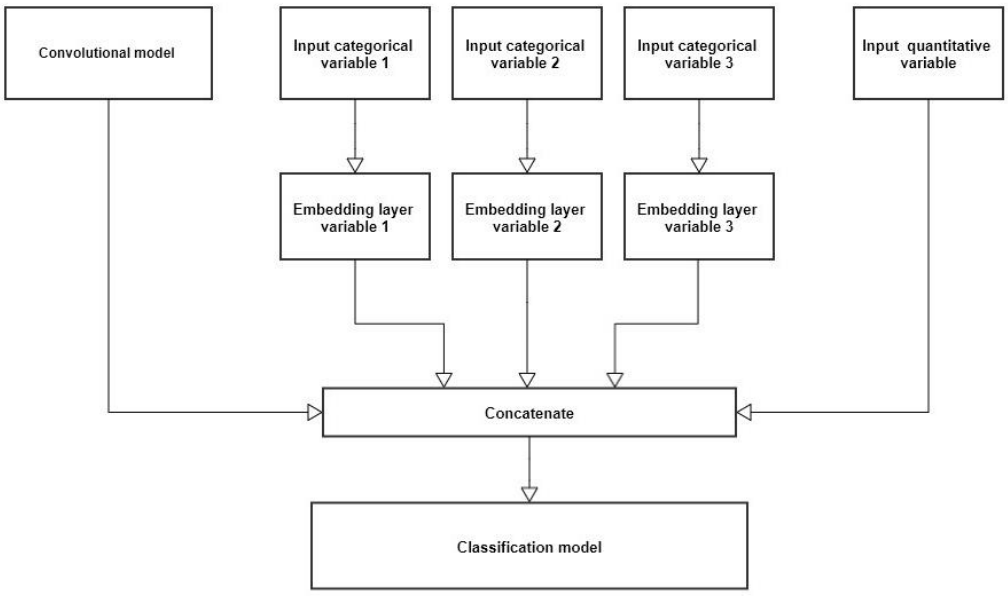
3.7. Clasificador para imágenes

Con el fin de intentar buscar patrones en los datos obtenidos de las películas se ha desarrollado un clasificador basado en Redes Neuronales Convolucionales para intentar clasificar las películas en función de su Rating teniendo en cuenta la imagen promocional de la película. Para la realización de la CNN se han tomado dos opciones: Transfer Learning e Implementarla de cero.

Transfer Learning: Se han aprovechado los pesos pre-entrenados (en el set de datos de ImageNet) de las redes VGG16 e XCeption de tal manera que ajustamos las capas superficiales de su red e implementamos un clasificador para adaptar el aprendizaje adquirido de esas redes a nuestro dominio. Hay que tener en cuenta que el dominio de nuestro problema (clasificar la imagen promocional de una película) es peculiar y resulta complicado encontrar redes con pesos pre entrenados en alguna tarea que traten imágenes parecidas. Las imágenes con las que trabajamos son imágenes con muchísima información en la que se pueden visualizar muchos elementos superpuestos lo que complica el aprendizaje de la red (visualizar imagen inferior). Debido a este motivo se decidió desarrollar un modelo neuronal con Redes Convolucionales de cero y comparar los resultados obtenidos.

3.8. Clasificador para datos tabulares e imágenes

Considerando los puntos 3.6 y 3.7 se decidió realizar con los mejores modelos obtenidos un clasificador de películas teniendo en cuenta la imagen y los datos tabulares de cada película. De esta manera creemos que la precisión del clasificador mejorará al combinar tanto los patrones obtenidos de los datos tabulares como de los patrones obtenidos de las imágenes.



3.9. Clasificador para textos (Título y Descripción)

Una vez realizada la clasificación por los datos tabulares teniendo en cuenta las keywords extraídas de las descripción optamos por implementar varios modelos neuronales para clasificar tanto el texto y la descripción, tratando de responder a la pregunta: ¿Hay alguna información en

la descripción de la película o en el texto que nos permita identificar si pertenece a una clase u otra?

3.9.1. Clasificador por descripción

Para la clasificación de las películas teniendo en cuenta su descripción (todas estaban en inglés) se utilizó un modelo pre-entrenado de Tensorflow Hub. Este modelo ha sido entrenado con 130GB de Noticias de Google en Inglés [10]. El modelo tiene un vocabulario de un tamaño de 20.000 tokens más un token extra para los caracteres no reconocidos. No es necesario realizar un preprocesamiento porque el propio modelo importado se encarga de ello (realiza una separación de caracteres por espacio).

3.9.2. Clasificador por títulos

Una característica de los títulos es que están en diferentes idiomas, es decir, el título Scrapeado es el original que puso el director de la película en el idioma que él consideró conveniente, por otro lado hay títulos con palabras inventadas y nombres propios. Teniendo en cuenta estos factores se tomó la decisión de crear un embedding de palabras propio sin utilizar uno pre-entrenado como en el caso de las descripciones.

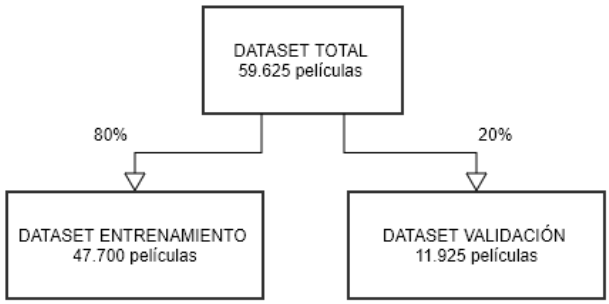
- En primer lugar se tokenizaron todos los títulos para realizar el proceso de conversión de palabras a números, esta tokenización se llevó a cabo con la librería de preprocesamiento de keras (keras.preprocessing import text).
- Una vez creados los token se convirtieron a números cada token tenía asociado un ID.

Una vez realizada la conversión de tokens a números ya estábamos en condiciones de pasar los datos a un embedding que acompañado de un clasificador iba a realizar el entrenamiento pertinente.

4 Resultados y conclusiones

4.1. División del dataset

El dataset final utilizado contiene 59.625 películas como ha sido comentado en el apartado 2. Para medir la capacidad de los modelos, se ha dividido el dataset en 80% para entrenamiento y 20% para validación, resultando como se muestra a continuación.



Todos los resultados mostrados en esta sección corresponden a los resultados medidos para **validación.**

4.2. Modelos Machine Learning (Baselines)

Debido a la naturaleza de los algoritmos de clasificación no neuronales, el único tipo de dato que pueden recibir son los datos tabulares (los títulos y descripciones son texto, mientras que los posters de las películas son imágenes). Por esta razón los resultados mostrados a continuación son los resultados de clasificación mediante datos tabulares.

Además, cabe destacar que ha sido probado la eliminación de cada una de las variables por separado con el objetivo de ver si el rendimiento de los modelos con mejor rendimiento se mantenía igual, tratándose entonces de variables superfluas. El resultado no ha sido concluyente, observándose siempre en alguna de las ejecuciones una disminución (aunque en algunos casos mínima) del rendimiento de los modelos, por lo que se ha decidido dejar todas.

Estos modelos reciben como **entrada** información de las siguientes variables categóricas:
['COLOR','DIRECTOR','GENRES_0','GENRES_1','GENRES_2','LANGUAGE_0','COUNTRY_0','KEYWORDS_0','KEYWORDS_1','KEYWORDS_2','WRITERS_0','CONTENT_RATING','KEYWORDS_DESCRIPTION','ACTOR_0','ACTOR_1','ACTOR_2']

Todos los valores mostrados a continuación se encuentran en tanto por uno.

					weighted avg			
Algoritmo ML	Recall (Clase 0)	F1 (Clase1)	Precision (Clase2)	Recall	F1	Precision	ACC	Custom metric
AdaBoosting	0.29	0.60	0.63	0.58	0.57	0.59	0.58	0.506
DecisionTree	0.44	0.50	0.56	0.51	0.51	0.51	0.51	0.5
GradientBoosting	**0.41**	**0.63**	**0.68**	**0.62**	**0.62**	**0.63**	**0.62**	**0,57**
Kneighbors	0.25	0.49	0.40	0.39	0.38	0.39	0.39	0.38
NearestCentroid	0.29	0.42	0.37	0.36	0.37	0.37	0.36	0.36
SVM	0.27	0.53	0.38	0.38	0.35	0.38	0.38	0,393
Random Forest	**0.40**	**0.62**	**0.67**	**0.62**	**0.61**	**0.63**	**0.62**	**0.56**

El único algoritmo que permite ajustar la penalización por clases del error cometido con la librearía utilizada es Random Forest. Lo mostramos a continuación.

					weighted avg			
Algoritmo ML	Recall (Clase 0)	F1 (Clase1)	Precision (Clase2)	Recall	F1	Precisión	ACC	Custom metric
Random Forest	0.36	0.63	0.69	0.61	0.60	0.63	0.61	0.563

4.3. Modelos Deep Learning

Para seleccionar la arquitectura más apropiada en cada uno de los modelos se ha seguido la técnica de ***grid search,*** es decir, se han ido probando todas las combinaciones posibles de los hiper parámetros que hemos pre seleccionado como "importantes" a partir de nuestros conocimientos. El *grid search* no ha sido exhaustivo, es decir, hemos probado primero pocas combinaciones con saltos grandes entre los posibles valores y posteriormente hemos probado algunas combinaciones

más alrededor del valor que mejores resultados ha dado. La selección de solo algunos hiper parámetros considerados como "importantes", y la realización de un *grid search* poco exhaustivo ha sido debida a las limitaciones de tiempo y sobre todo al límite de uso que impone Google Colab, que tras algunas horas de ejecución te restringe el uso de GPU e incluso el mero uso de la plataforma.

Los hiper parámetros seleccionados como "importantes" y por tanto probados en las arquitecturas de Deep Learning son los siguientes:

- Neuronas/filtros/tamaños de pooling: En todas las arquitecturas se han probado diferentes números de neuronas, y de filtros en caso de las redes convolucionales, así como el tamaño de los filtros.
- Dropout/Batch normalization: En todas las arquitecturas se ha probado la inclusión de capas de Dopout y Batch normalization, así como los valores de Dropout más apropiados.
- Optimizador: Se han probado diferentes optimizadores, así como sus valores de learning rate.
- Penalización del error por clase: uno de los hiper parámetros que más importancia ha tenido a la hora de mejorar nuestros modelos frente a la métrica customizada ha sido la modificación del peso que damos a los errores cometidos en cada clase. En general, el aumento del peso dado al error cometido en la clase 0 frente al resto de errores nos ha permitido mejorar el recall de la clase 0 sin apenas disminuir la la precisión general de nuestro modelo. También se ha penalizado de forma ligeramente superior el error cometido en la clase 2 frente a la clase 1, de modo que ha aumentado la precisión de la clase 2 de forma notable.
- Batch size: En todas las arquitecturas se ha probado varios valores de batch size.
- En el caso de las capas de embedding, el tamaño del vector de embedding se ha determinado a partir de la siguiente fórmula recomendada por Jeremy Howard en su curso de fast.ai y considerada como una regla empírica por la comunidad.

  $$\text{embedding\_size} = \min(50, m/2)$$

  Siendo *m* el número de categorías de la variable categórica que pasa por esa capa de embedding.

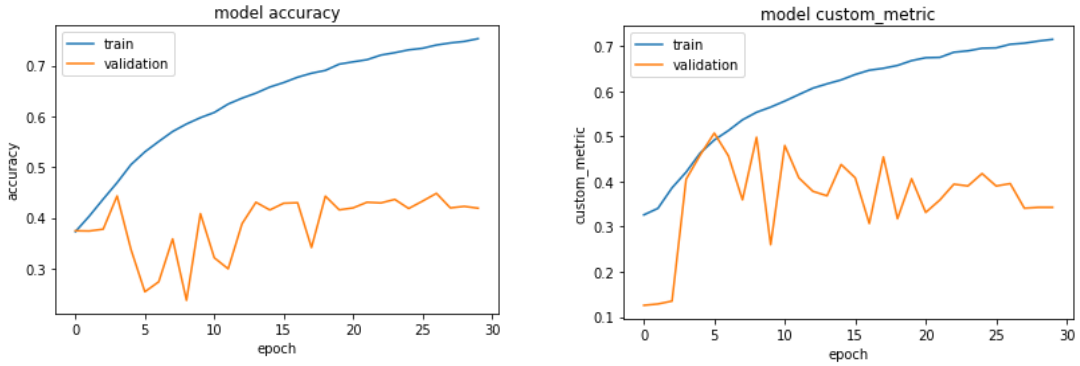
El resto de hiper parámetros han conservado su valor por defecto. En el caso de las funciones de activación, se ha optado por utilizar siempre "relu" en caso de necesitar una función de activación no lineal y "softmax" para las capas de salida por tratarse de un problema de clasificación.

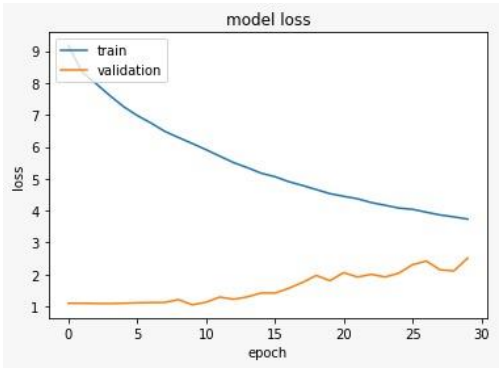
Cabe destacar que todas las pruebas han contado **check points** de modo que, aunque en las gráficas se observe sobre entrenamiento, los modelos obtenidos a partir de estos entrenamientos suelen pertenecer a las epochs anteriores al sobre entrenamiento.

4.3.1. Clasificador de título

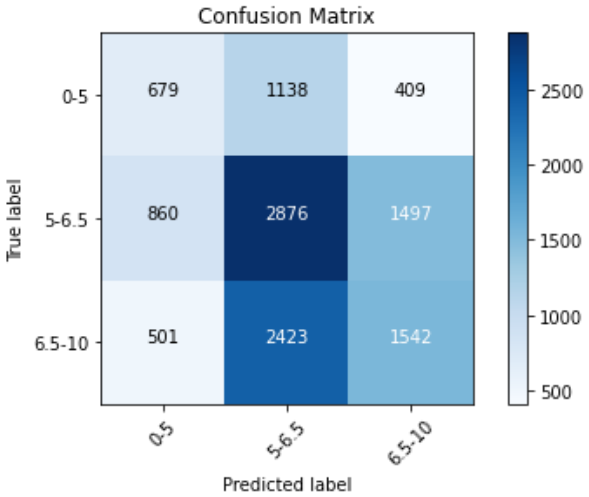
Este modelo recibe como **entrada** un texto (el título de la película).

Gráficos de entrenamiento

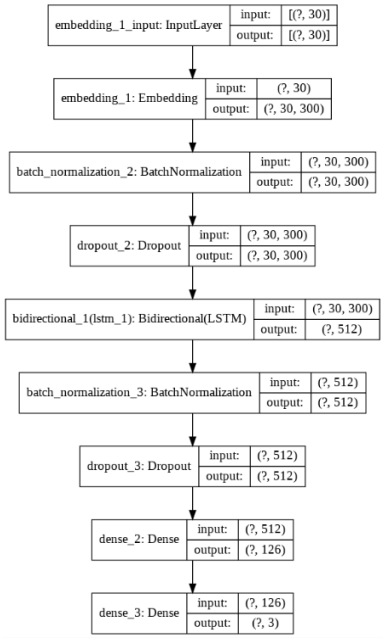




Matriz de confusión



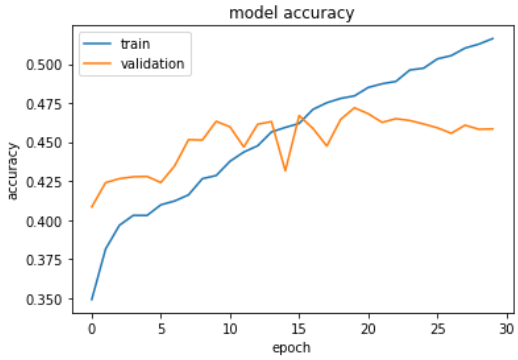
Arquitectura del modelo

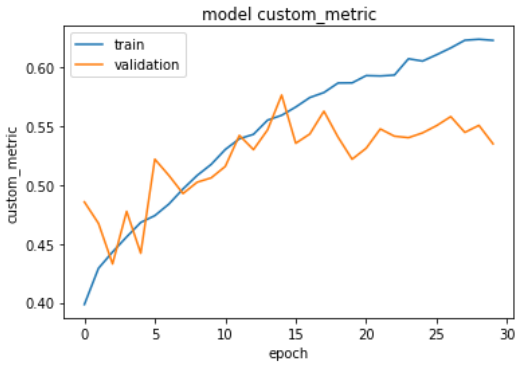


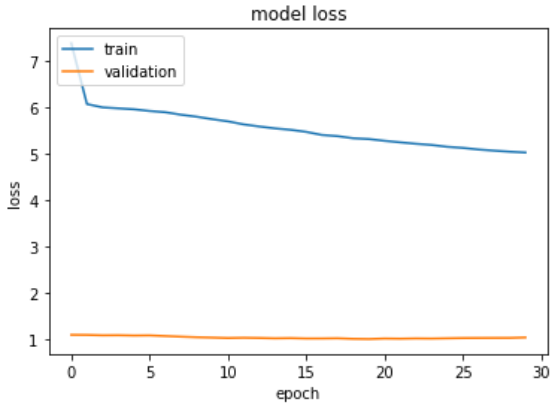
4.3.2. Clasificador de descripciones

Este modelo recibe como **entrada** un texto (la descripción de la película).

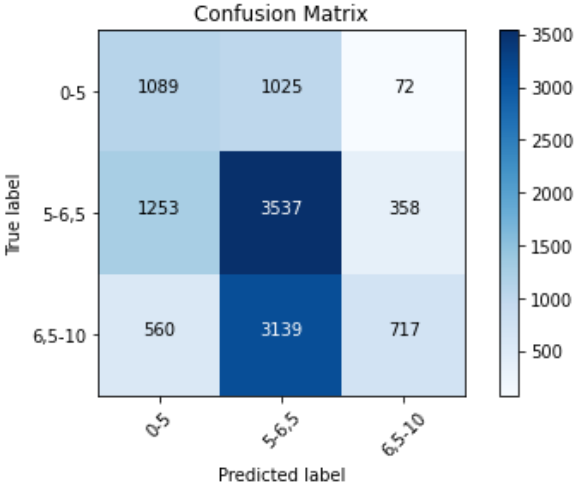
Gráficos de entrenamiento



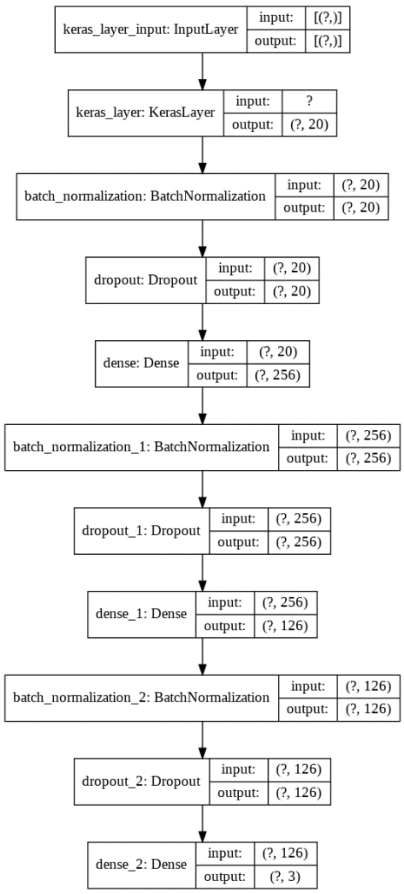




Matriz de confusión



Arquitectura del modelo



4.3.3. Clasificador de datos tabulares

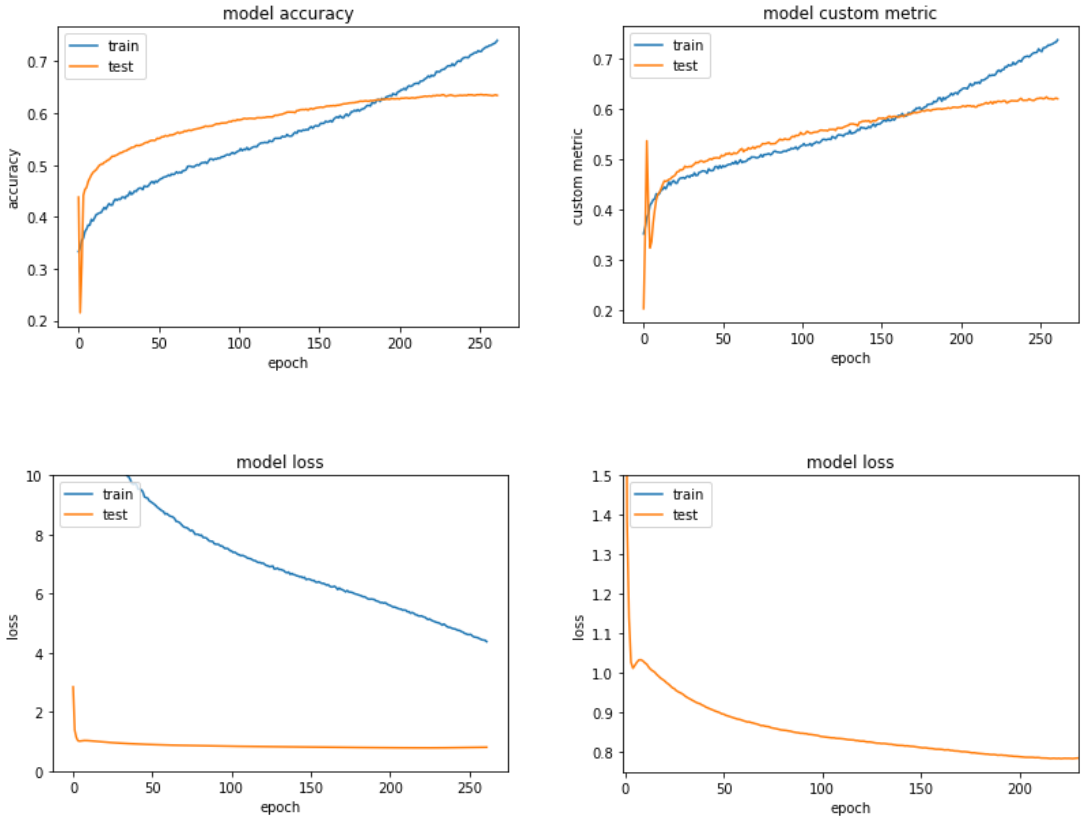
Este modelo recibe como **entrada**

- La información de las siguientes variables categóricas:

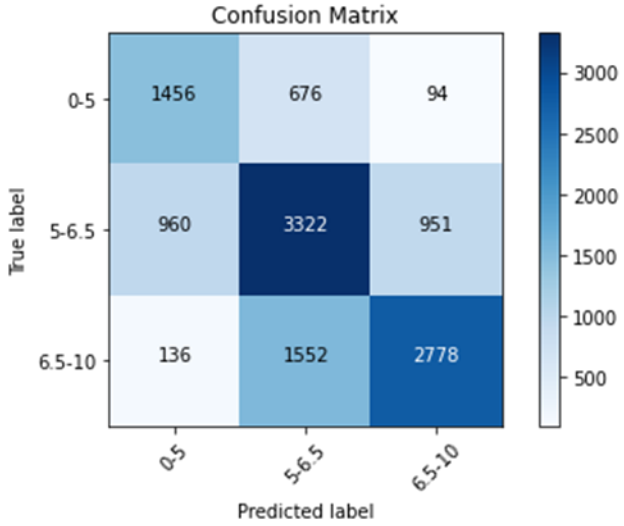
  ['COLOR','DIRECTOR','GENRES_0','GENRES_1','GENRES_2','LANGUAGE_0','COUNTRY_0','KEYWORDS_0','KEYWORDS_1','KEYWORDS_2','WRITERS_0','CONTENT_RATING', 'KEYWORDS_DESCRIPTION','ACTOR_0','ACTOR_1','ACTOR_2']

- La información de las siguientes variables numéricas: ['DURATION','YEAR']

Gráficos de entrenamiento



Matriz de confusión

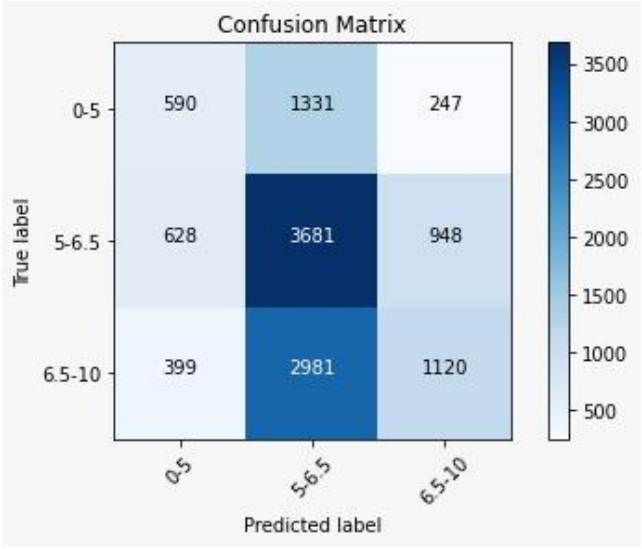


La imagen del modelo se encuentra en el **Anexo A** de la presente memoria por motivos de tamaño.

4.3.4. Clasificador de imágenes

Este modelo recibe como **entrada** una imagen (el cartel de la película)

Matriz de confusión



La imagen del modelo se encuentra en el **Anexo C** de la presente memoria por motivos de tamaño.

La información de las gráficas de entrenamiento no puede ser mostrada debido al método de entrenamiento y nuestras limitaciones técnicas.

4.3.5. Clasificador de tabulares + imágenes

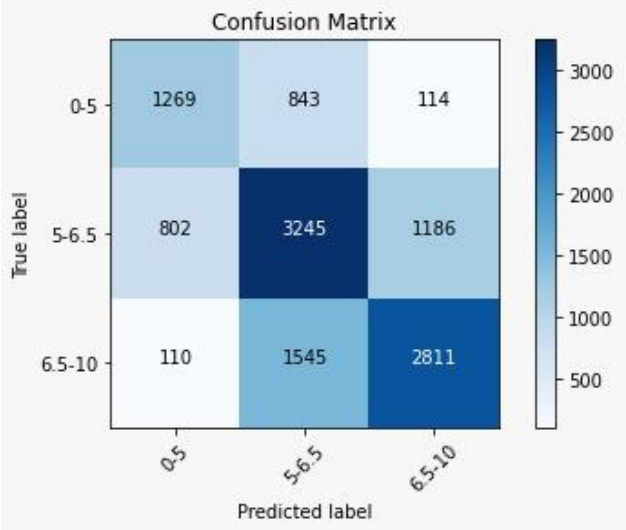
Este modelo recibe como **entrada:**

- Una imagen (el cartel de la película)
- La información de las siguientes variables categóricas:

  ['COLOR','DIRECTOR','GENRES_0','GENRES_1','GENRES_2','LANGUAGE_0','COUNTRY_0','KEYWORDS_0','KEYWORDS_1','KEYWORDS_2','WRITERS_0','CONTENT_RATING','KEYWORDS_DESCRIPTION','ACTOR_0','ACTOR_1','ACTOR_2']

- La información de las siguientes variables numéricas: ['DURATION','YEAR']

Matriz de confusión



La imagen del modelo se encuentra en el **Anexo B** de la presente memoria por motivos de tamaño.

La información de las gráficas de entrenamiento no puede ser mostrada debido al método de entrenamiento y nuestras limitaciones técnicas.

4.3.6. Comparativa modelos Deep Learning

Todos los valores mostrados a continuación se encuentran en tanto por ciento.

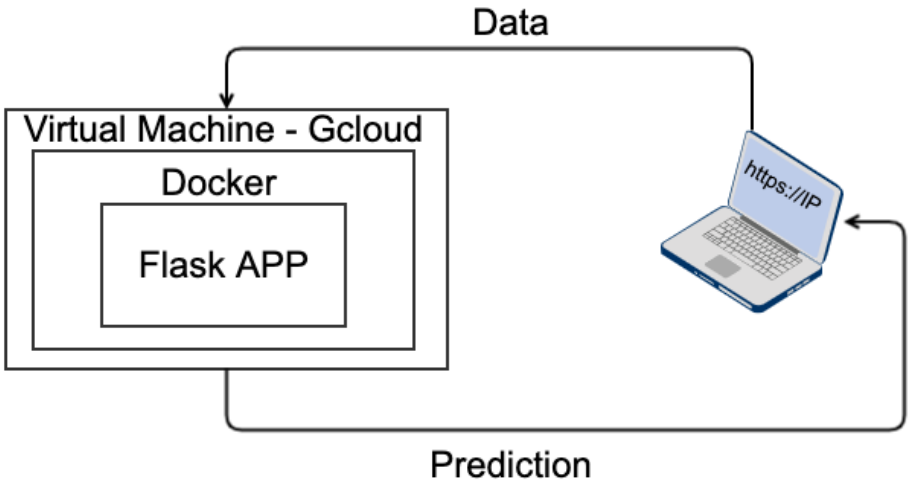
Tipo dato	Recall (Clase 0)	F1_Score (Clase 1)	Precision (Clase 2)	ACC	Custom Metric
Tabular	**65**	**62**	**73**	**63**	**66**
Img	27	56	48	45	44
Img+Tabular	**57**	**60**	**68**	**62**	**62**
Descripción	50	55	63	45	56
Título	31	49	45	42	42

4.4. Comparativa entre los mejores modelos de DL y Baselines

Tipo dato	Recall (Clase 0)	F1_Score (Clase 1)	Precision (Clase 2)	ACC	Custom Metric
DL Tabular	**65**	**62**	**73**	**63**	**66**
DL Img+Tabular	**57**	**60**	**68**	**62**	**62**
Gradient Boosting	41	63	68	62	57
Random Forest	40	62	67	62	56

4.5. Aplicación web

Para la presentación de los resultados obtenidos se ha desarrollado una aplicación web con el framework para python "Flask". Esta aplicación se encuentra alojada en una máquina virtual de Google Cloud Platform.



4.5.1. Manual de usuario

Instalación:

Pre-Requisitos: Tener instalado Docker y tener descargada la carpeta flask_app del reposiotio en el ordenador, dentro de esa carpeta descargarse la siguiente carpeta y meterla en dentro de flask

_app.https://drive.google.com/file/d/1R7FlSWD1lKl91_9lmHRhf_k90ruYoh_k/view?usp=sharing.

Opción A:

Dentro de la carpeta:

1. Ejecutar el fichero .bash "docker_buildImg.sh" (bash docker_buildImg.sh). este fichero creará la imagen de docker.
2. Ejecutar el fichero .bash "docker_buildContainer.sh". Dentro del fichero tenemos que cambiar el ID de la imágen por el ID que le ha asignado docker en nuestra máquina. (Ejecutar docker image ls)
3. En nuestro navegador introducir en el campo de la URL: http://localhost:5000.

Opción B:

Otra opción es descargarse la imagen de DockerHub mediante el siguiente comando (de esta manera después de ejecutar ese comando pasaríamos directamente al paso 3) de la opción A:

docker run -d -p 5000:5000 juan97serrano/imdb_app:1

Pasos para realizar una predicción:

1. Elegir una opción, posibles opciones:
    a. Opción 1: Clasificar por la imagen: Seleccionar una imagen en formato JPEG.
    b. Opción 2: Clasificar por los datos tabulares (incluida descripción): Para rellenar los campos de texto para los datos tabulares deberemos descargar el fichero .zip que se encuentra haciendo click sobre el texto que pone "Link con los datos disponibles para introducir". En este fichero .zip encontramos varios ficheros (uno por cada variable categórica) que contienen todos los datos disponibles para introducir: nombres de actores, directores, keywords… En caso de introducir un valor que no aparezca se le dará un valor por defecto. En el caso de la variable año y duración no se tendrán ficheros, se deberá introducir la duración en minutos y el año con números (ejemplo:2020). Una vez rellenados las variables tabulares deberemos introducir la descripción (la aplicación extraerá automáticamente las keywords).
    c. Opción 3: Clasificar por la imagen + datos categóricos (incluida descripción): Cumplir los pasos 1 y 2.
    d. Opción 4: Clasificar por el título: Introducir un título.

   e. Opción 5: Clasificar por la descripción: Introducir una descripción.
2. Dar al botón verde “Predict”
3. Visualizar los resultados.

4.6. Conclusiones y trabajo futuro

En cuanto a las conclusiones, comenzamos por hacer una aclaración. La capacidad que hemos tenido de probar el modelo multi input ha sido muy inferior a la que hemos tenido con el resto de modelos, debido al coste computacional requerido para entrenarlo. Sabemos que, con una mayor capacidad de cómputo y suficiente tiempo, los resultados del modelo multi input serían como mínimo iguales a los del modelo tabular, si no superiores. La sensación que hemos tenido al entrenar el modelo multi input es de que no hemos llegado a exprimirlo, dado que la mayor parte de las veces la ejecución ha sido interrumpida por cortes en la sesión de Google Colab y no por los Early Stopping, como en el resto de modelos.

Dicho esto, las principales conclusiones son las siguientes:

- Los títulos por sí solos no aportan información, al menos que seamos capaces de aprovechar, dado que el rendimiento conseguido es igual al de un modelo que siempre diga que la película pertenece a la clase más común.
- Las descripciones e imágenes aportan algo de información (aunque no demasiada) dado que los clasificadores aislados consiguen entrenar un poco y hacer una clasificación mejor que la que haría un modelo que siempre diga que la clase a la que pertenece la entrada es la clase más común.
- Los mejores modelos baselines consiguen precisiones bastante altas pero la distribución de los errores que cometen no es la ideal para el problema que abordamos, por lo que obtienen custom metrics no muy altas.
- El modelo multi input, a pesar de la falta de optimización ya descrita al principio de este apartado, consigue una precision tan buena como la de los modelos baseline y una custom metric superior.
- El modelo de datos tabulares, consigue una precisión algo superior a la de los modelos baseline y una custom metric muy superior.

Es evidente, por tanto, que la capacidad de los modelos de DL para este problema tal y como ha sido enfocado es superior a la de los modelos baseline, sobre todo en cuanto al rendimiento

relativo a la custom metric. Esto último probablemente sea debido a la gran capacidad de adaptación al problema que permiten los modelos de DL, como por ejemplo el ajuste de la penalización por errar en cada una de las clases que hemos llevado a cabo en los modelos neuronales, tal y como se describe en el apartado 4.3.

Otra conclusión que hemos sacado es la importancia que tiene alcanzar un equilibrio adaptado a tus posibilidades entre el coste material (tanto en tiempo como en coste computacional) que tiene entrenar tus modelos y la capacidad de expresión de estos. Si tienes un modelo enorme, capaz de exprimir al máximo los datos, pero no eres capaz de llevarlo a su máxima capacidad obtendrás un rendimiento inferior al deseado. No obstante, si tienes un modelo demasiado simple probablemente quede mucha información oculta en los datos que no serás capaz de exprimir, y también obtengas un rendimiento inferior al deseado. La solución óptima será la que maximice ambas variables en tu situación concreta.

En cuanto al trabajo futuro, las principales opciones que se nos ocurren son:

- Introducir en la salida de los modelos, de alguna manera, el numero de votos. Por ejemplo, otras tres categorías obtenidas por tres rangos del numero de votos, convirtiéndose así el problema en un problema de clasificación multi label. De este modo, harías predicciones tanto del rating de la película como de la popularidad de esta.
- Los mismo del punto anterior, pero con los ingresos generados.
- Aumentar la capacidad de cómputo y hacer uso de imágenes más detalladas, en lugar de reducirlas tanto.
- Cruzar el dataset screapeado con información de otros sitios para hacer uso de un número mayor de películas.

5 Bibliografía

[1] [En línea]. Available: https://www.motionpictures.org/wp-content/uploads/2019/03/MPAA-THEME-Report-2018.pdf)..

[2] [En línea]. Available: https://www.eladelantado.com/segovia/publicatessen_revela_que_los_videojuegos_facturan_ya_mas_que_el_cine_o_la_musica/) .

[3] [En línea]. Available: https://www.rtve.es/noticias/20190507/industria-videojuegos-factura-espana-doble-musica-cine-juntos/1932840.shtml), .

[4] [En línea]. Available: https://www.imdb.com.

[5] N. VR, «Predicting Movie Success Based on IMDB Data,» *International Journal of Data Mining Techniques and Applications*, p. 4, 2014.

[6] M. C. Bayar, «Movie Rating Prediction with Machine Learning Algorithms on IMDB Data Set,» *International Conference on Advanced Technologies, Computer Engineering and Science*, p. r, 2018.

[7] R. Campos, «YAKE! Collection-Independent Automatic Keyword Extractor,» 2018.

[8] [En línea]. Available: https://selenium-python.readthedocs.io/.).

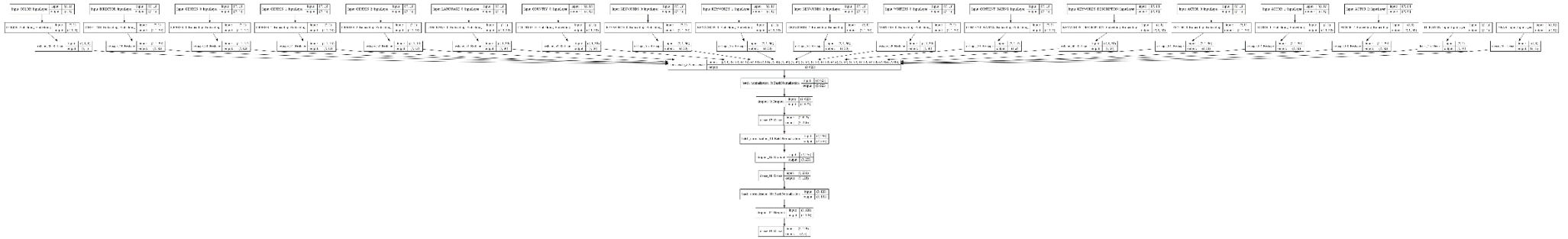
[9] F. B. Cheng Guo, «arxiv,» 25 4 2016. [En línea]. Available: https://arxiv.org/pdf/1604.06737v1.pdf. [Último acceso: 1 2020].

[10] [En línea]. Available: referencia https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1.

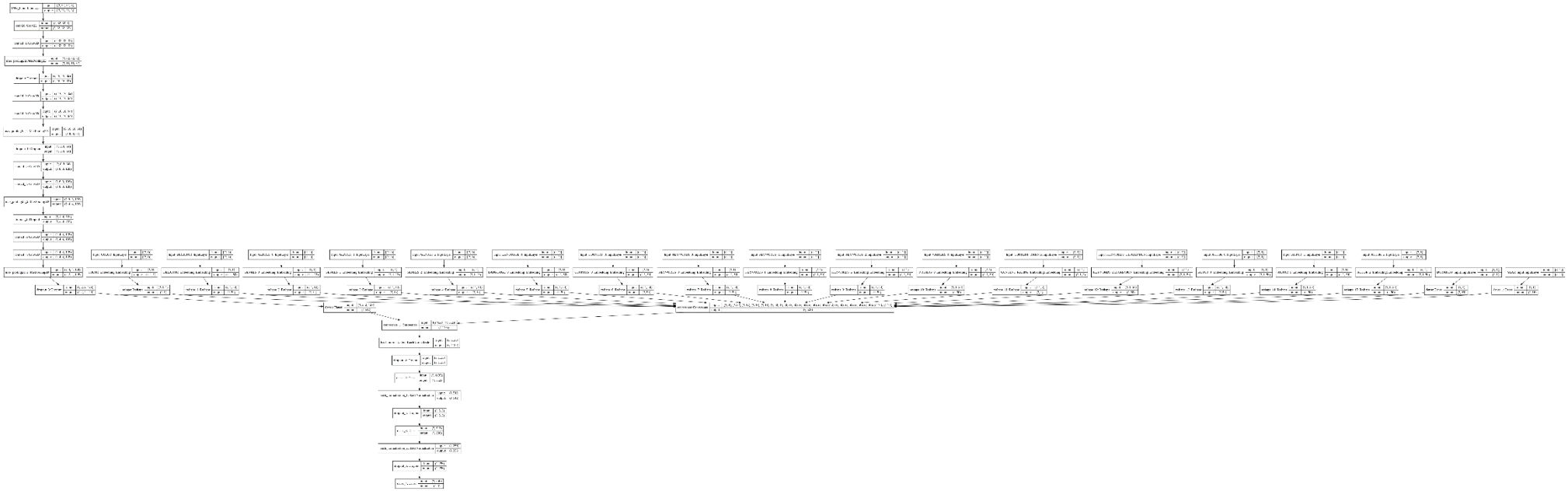
[11] 2. Ricardo Campos1, «YAKE! Collection-Independent Automatic Keyword Extractor».

6 Anexos

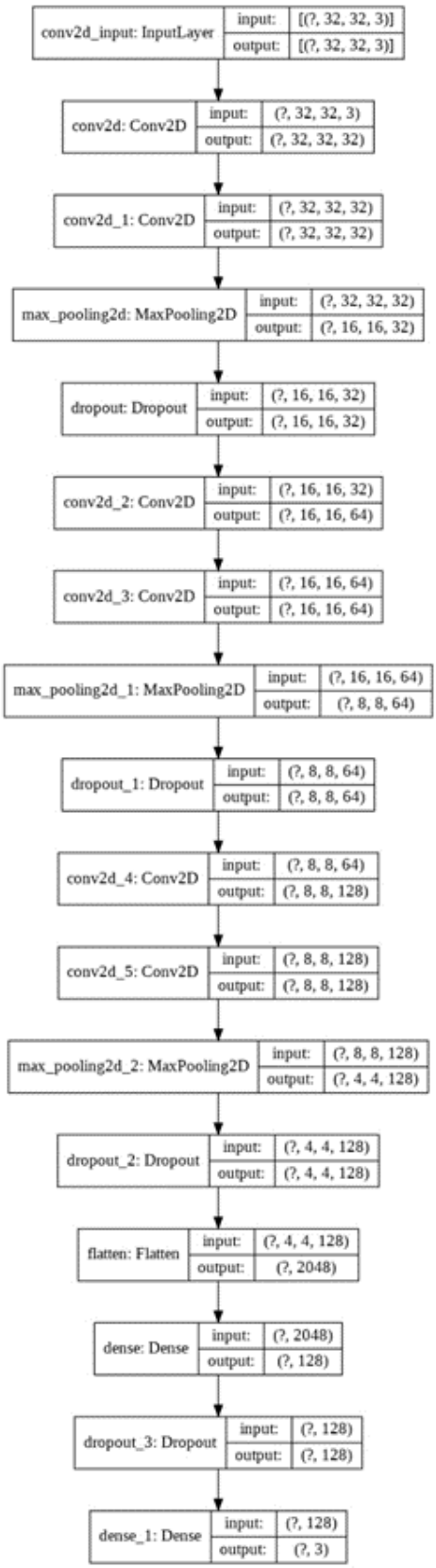
6.1. Anexo A



6.2. Anexo B



6.3. ANEXO C


conv2d_input: InputLayer
input: [(?, 32, 32, 3)]
output: [(?, 32, 32, 3)]
conv2d: Conv2D
input: (?, 32, 32, 3)
output: (?, 32, 32, 32)
conv2d_1: Conv2D
input: (?, 32, 32, 32)
output: (?, 32, 32, 32)
max_pooling2d: MaxPooling2D
input: (?, 32, 32, 32)
output: (?, 16, 16, 32)
dropout: Dropout
input: (?, 16, 16, 32)
output: (?, 16, 16, 32)
conv2d_2: Conv2D
input: (?, 16, 16, 32)
output: (?, 16, 16, 64)
conv2d_3: Conv2D
input: (?, 16, 16, 64)
output: (?, 16, 16, 64)
max_pooling2d_1: MaxPooling2D
input: (?, 16, 16, 64)
output: (?, 8, 8, 64)
dropout_1: Dropout
input: (?, 8, 8, 64)
output: (?, 8, 8, 64)
conv2d_4: Conv2D
input: (?, 8, 8, 64)
output: (?, 8, 8, 128)
conv2d_5: Conv2D
input: (?, 8, 8, 128)
output: (?, 8, 8, 128)
max_pooling2d_2: MaxPooling2D
input: (?, 8, 8, 128)
output: (?, 4, 4, 128)
dropout_2: Dropout
input: (?, 4, 4, 128)
output: (?, 4, 4, 128)
flatten: Flatten
input: (?, 4, 4, 128)
output: (?, 2048)
dense: Dense
input: (?, 2048)
output: (?, 128)
dropout_3: Dropout
input: (?, 128)
output: (?, 128)
dense_1: Dense
input: (?, 128)
output: (?, 3)

Trabajo de Fin de Máster