Predicción de la clasificación de una película en IMDB

Juan Carlos Barría Rival

ARTICLE INFO

Keywords:
Peliculas
Predicción
IMDb
Netflix
Amazon Prime Video
Hulu
Disney+
Aprendizaje automático
Random Forest

ABSTRACT

La industria del cine no es sólo una industria o un centro de entretenimiento, sino que ahora es un centro de negocios global para las plataformas de streaming. Todo el mundo está entusiasmado con el éxito de una película en la taquilla, su popularidad, etc. Hay una gran cantidad de datos disponibles en línea sobre el éxito o la popularidad de estas películas. Para este proyecto se ha utilizado un dataset con la lista de de películas de las principales plataformas de Streaming (Netflix, Amazon Prime Video, Hulu y Disney+) y su calificación en IMDb, web donde se pueden ver las valoraciones de estas. Se aplicó algoritmo de regresión de aprendizaje automático. Por último, se busca el modelo más eficaz para predecir la calificación de una película en IMDb.

1. Introducción

Hoy en día las plataformas de streaming de películas no es la única fuente de ocio, sino que es una de las principales fuentes de comercio global y el marketing. Las películas crean una nueva moda entre la gente especialmente entre los jóvenes. No sólo los directores de cine y empresas de distribución se preocupan por el éxito de las películas, sino también la gente en general. La gente suele hablar de ellas en las redes sociales. Por lo tanto, el análisis de los datos de las redes sociales sobre las películas es recientemente popular entre los analistas de datos. Aparte de esto otros ámbitos, como el análisis de las historias de éxito de un director o la popularidad de un actor, etc.

El análisis puede ser diferente en los distintos países, naturalmente, las regiones del mundo no reaccionan de forma similar. Ahora las películas están disponibles en Internet para el consumo de cualquier persona. Hay plataformas como IMDb (Internet Movie Database) [1], Rotten Tomatoes [6] , Metacritic [4] etc. donde la gente puede compartir sus críticas sobre las películas. Día a día estas plataformas se están se están volviendo populares, ya que la gente está recibiendo críticas honestas allí. Por lo tanto, hay una gran cantidad de datos disponibles en línea acerca de las críticas y rating de películas. En este trabajo, estos datos sobre las notas de las películas se analizan para predecir estas calificaciones.

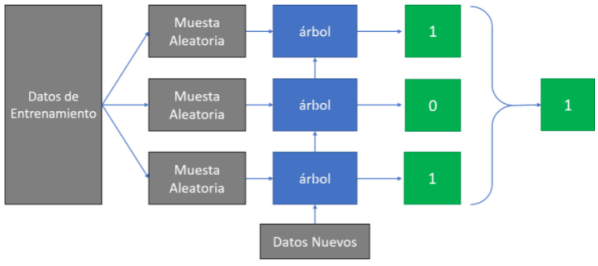
Hay un buen número de estudios para predecir el éxito de las películas de éxito, ya que la gente está muy entusiasmada con las películas. Muy pocos estudios incluyen las características (director, guión actor, actriz, género, etc.) de una película para predecir la tasa de éxito. Por eso, en este trabajo, el objetivo es predecir el índice de éxito basándose en las características propias de la película. Por ejemplo, el director, el guión Actores/actrices, país y género. De nuevo, IMDb es una plataforma que crece día a día. Por lo general, la gente muestra interés si la calificación de IMDb de una película es alta. Por lo tanto, el motivo final es encontrar un modelo que pueda predecir eficientemente la calificación en IMDb de una película.

2. Desarrollo

2.1. Técnica aplicada

Los Bosques Aleatorios (Random Forest) es un algoritmo de aprendizaje supervisado que, como dice su nombre, crea un bosque y lo hace de alguna manera aleatorio. En resumen, el Bosque Aleatorio crea múltiples árboles de decisión y los combina para obtener una predicción más precisa y estable. En general, mientras más árboles en el bosque se vea, más robusto es el bosque. Los Random Forests tienen una capacidad de generalización muy alta para muchos problemas.

En sí, un Random Forest es un conjunto de árboles de decisión combinados con bagging. Al usar bagging, lo que en realidad está pasando, es que distintos árboles ven distintas porciones de los datos. Ningún árbol ve todos los datos de entrenamiento. Esto hace que cada árbol se entrene con distintas muestras de datos para un mismo problema. De esta forma, al combinar sus resultados, unos errores se compensan con otros y tenemos una predicción que generaliza mejor.



Este se considera un algoritmo muy útil y fácil de usar ya que los parámetros predeterminados a menudo producen un buen resultado de predicción. De igual forma, el número de parámetros tampoco es tan alto y son fáciles de entender.

Uno de los grandes problemas en Machine Learning es el sobreajuste (overfitting), pero la mayoría de las veces esto no será tan fácil para un algoritmo de Bosques Aleatorios, esto se debe a que, si hay suficientes árboles en el bosque, el algoritmo no se adaptará al modelo.

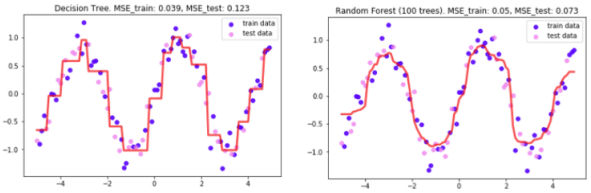
La principal limitación de Bosques es que una gran can-

tidad de árboles puede hacer que el algoritmo sea lento e ineficiente para las predicciones en tiempo real. En general, estos algoritmos son rápidos para entrenar, pero bastante lentos para crear predicciones una vez que están entrenados. Una predicción más precisa requiere más árboles, lo que resulta en un modelo más lento. En la mayoría de las aplicaciones del mundo real, el algoritmo de Bosque Aleatorio es lo suficientemente rápido, pero ciertamente puede haber situaciones en las que el rendimiento en tiempo de ejecución es importante y se prefiera otros enfoques.

Adicionalmente, los Bosques Aleatorios es una herramienta de modelado predictivo y no una herramienta descriptiva. Eso significa que, si estás buscando una descripción de las relaciones en los datos, deberás elegir otro algoritmo.

Entonces podríamos decir, el algoritmo de Bosques Aleatorios es un gran algoritmo para entrenar temprano en el proceso de desarrollo del modelo, para ver cómo se desempeña y es difícil construir un mal modelo con este algoritmo debido a su simplicidad. Es una excelente opción, si se necesita desarrollar un modelo en un corto periodo de tiempo, además de eso, proporciona un buen indicador de la importancia que asigna a sus características.

La diferencia intuitiva entre un arbol de decisión y un random forest (este en particular usa 100 árboles):



Como se puede ver, tanto el árbol de decisión como el bosque aleatorio, tienen un error cuadrático medio (MSE) de entrenamiento pequeño y similar. Sin embargo, el error de generalización del random forest es mucho mejor. Esto se puede apreciar también en que el modelo aprendido es mucho más suave como se puede observar en la figura anterior.

Para problemas de clasificación, se suelen combinar los resultados de los árboles de decisión usando soft-voting (voto suave). En el voto suave, se le da más importancia a los resultados en los que los árboles estén muy seguros.

Para problemas de regresión, la forma más habitual de combinar los resultados de los árboles de decisión, es tomando su media aritmética.

Scikit-learn que es una de las librerías que se usan ofrece dos implementaciones de random forests:

- Para clasificación: RandomForestClassifier
- Para regressión: RandomForestRegressor (que es el que usaremos)

Estos son los hiper-parámetros más útiles:

Propios del Bosque Aleatorio:

- n_estimators: Número de árboles que va a tener el bosque aleatorio. Normalmente cuantos más mejor, pero a partir de cierto punto deja de mejorar y sólo hace que vaya más lento. Un buen valor por defecto puede ser el uso de 100 árboles.
- n_jobs: Número de cores que se pueden usar para entrenar los árboles. Cada árbol es independiente del resto, así que entrenar un bosque aleatorio es una tarea muy paralelizable. Por defecto sólo utiliza 1 core de la CPU. Para mejorar el rendimiento se pueden usar tantos cores como se estime necesario. Si se usa n_jobs = -1, estás indicando que quieres usar tantos cores como tenga la máquina.
- max_features: Una forma de garantizar que los árboles son diferentes, es que todos se entrenan con una muestra aleatoria de los datos. Si queremos que todavía sean más diferentes, podemos hacer que distintos árboles usen distintos atributos. Esto puede ser útil especialmente cuando algunos atributos están relacionados entre sí. Hay varias estrategias para elegir el número máximo de atributos que se pueden usar.

Regularización (también disponibles para Decision Trees):

- max_depth: La profundidad máxima del árbol.
- min_samples_split: Número mínimo de muestras necesarias antes de dividir este nodo. También se puede expresar en porcentaje.
- min_samples_leaf: Número mínimo de muestras que debe haber en un nodo final (hoja). También se puede expresar en porcentaje.
- max_leaf_nodes: Número máximo de nodos finales

Algunas de las ventajas de los **Bosques Aleatorios** son:

- Funciona bien aún sin ajuste de hiperparámetros
- Funciona bien para problemas de clasificación y también de regresión (como en este caso).
- Al utilizar múltiples árboles se reduce considerablemente el riesgo de overfiting
- Se mantiene estable con nuevas muestras puesto que al utilizar cientos de árboles sigue prevaleciendo el promedio de sus votaciones.

Y sus desventajas:

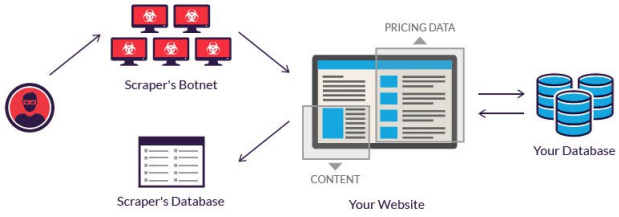
- En algunos datos de entrada "particulares" random forest también puede caer en overfitting
- Es mucho más "costo" de crear y ejecutar que "un sólo árbol" de decisión.
- Puede requerir muchísimo tiempo de entrenamiento
- Random Forest no funciona bien con datasets pequeños.
- Es muy difícil poder interpretar los ¿cientos? de árboles creados en el bosque, si quisiéramos comprender y explicar a un cliente su comportamiento.

2.2. Tipo de proyecto

Este proyecto se hizo con el objetivo de obtener las predicciones de las clasificaciones en el sitio web IMDb (Internet Movie Database) que es, como dice su nombre, una base de datos en línea que almacena información relacionada con películas, personal de equipo de producción (incluyendo directores y productores), actores, series de televisión, programas de televisión, videojuegos, actores de doblaje y, más recientemente, personajes ficticios que aparecen en los medios de entretenimiento visual. Esta Recibe más de 100 millones de usuarios únicos al mes y es de las más populares web de este tipo junto a Rotten Tomatoes.

Se centra en la prueba de diferentes modelos para predecir la calificación de una película en este sitio web, utilizando la información de la película obtenida del dataset como el director, el guión, el actor, la actriz, el género y el país como características de entrada. En el conjunto de datos, el título de la película y el año su estreno tambien se indican. Los modelos se construyen utilizando la técnica de de aprendizaje automático mencionada. Se entrena y se prueba con el conjunto de datos preparado obtenido. Por lo tanto, los 5 modelos construidos aquí se entrenan y se prueban mediante la técnica de aprendizaje supervisado Random Forest. El conjunto de datos utilizado aquí contiene la clase atributo "IMDb" que contiene el rating de esta, puede ser fracaso, inferior a la media, media y éxito, esto depende de la calificación de 1 a 10 del sitio web IMDb.

2.3. Detalle de conjunto de datos usado



El conjunto de datos [8] (dataset) de 15 columnas, tiene más de 16.000 títulos fue obtenido desde la página web Kaggle, que es una comunidad en línea de Data Scientists y profesionales del aprendizaje automático, este permite a los usuarios encontrar y publicar conjuntos de datos, explorar y crear modelos en un entorno de ciencia de datos basado en la web, trabajar con otros científicos de datos, etc. El dataset llamado "Movies on Netflix, Prime Video, Hulu and Disney+" contiene información que fue extraída a través de web scraping, que se trata de un proceso de usar bots para extraer contenido y datos de un sitio web, en este caso, con todas las películas disponibles en las siguientes plataformas de streaming:

- Netflix
- Amazon Prime Video
- Hulu
- Disney+

que incluye:

- id: valor identificador de la película
- title: Título de la película
- year: Año en que fue producida
- age: edad de público a la que se dirige
- imdb: clasificación (nota) en la página IMDb
- rottem tomatoes: porcentaje de aprobación en el sitio Rotten Tomatoes
- netflix: si la película se encuentra en la plataforma Netflix (1=si; 0=no)
- hulu: si la película se encuentra en la plataforma Hulu (1=si; 0=no)
- prime video: si la película se encuentra en la plataforma Amazon Prime Video (1=si; 0=no)
- disney+: si la película se encuentra en la plataforma Disney+ (1=si; 0=no)
- type: película o serie de tv (en este caso, todas son películas)
- directors: directores de la película
- genres: generos de la película
- country: país donde fue realizada
- language: idioma/s que está disponible
- runtime: duración (en minutos) de la película

2.4. Explicación de la implementación

El código [7] que se encontraba en la plataforma Kaggle y en Github [2], se pasó a un archivo de jupyter notebook en la plataforma Google Colab [5] la cual te permite ejecutar y programar en Python en tu navegador, da acceso gratuito a GPUs y permite compartir contenido fácilmente. También, fue subido finalmente a la plataforma GitHub [3].

Lo primero que se hace es importar las librerías, para este caso se utilizaron las siguientes:

- numpy: Librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.
- pandas: Paquete de Python que proporciona estructuras de datos similares a los dataframes de R. Esta depende de Numpy, la librería que añade un potente tipo matricial a Python. Esta nos permite por ejemplo leer y escribir datos en archivos CSV como se utilizó en este proyecto.
- matplotlib: Biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy.
- missingno: Biblioteca de python compatible con pandas que ofrece una forma muy agradable de visualizar la distribución de los valores NaN (nulos).
- seaborn: Librería de visualización de datos para Python desarrollada sobre matplotlib . Nos ofrece una interfaz de alto nivel para la creación de atractivas gráficas.
- scikit-learn: La librería más útil para Machine Learning, nos proporciona una gama de algoritmos de aprendizaje supervisados y no supervisados en Python.

Luego, viene la carga del dataset, el cual viene un archivo con extension .csv, y mostramos las primeras n filas en función de la posición, gracias al metodo .head() de pandas.

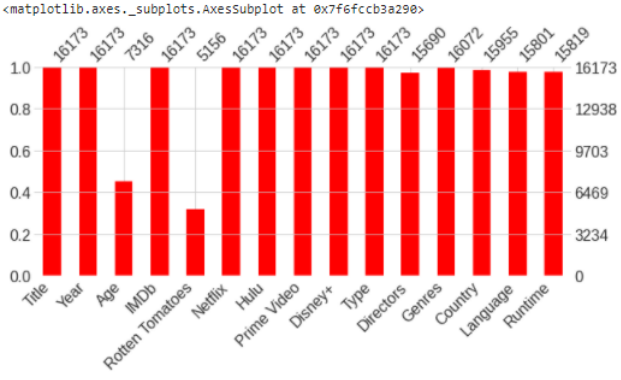
Eliminamos de la lista los "Unnamed", osea, película sin nombre. Ahora queremos ver una descripción general de los datos de nuestro DataFrame que tenemos, para eso ocupamos el metodo .info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16744 entries, 1 to 16744
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Title            16744 non-null  object 
 1   Year             16744 non-null  int64  
 2   Age              7354 non-null   object 
 3   IMDb             16173 non-null  float64
 4   Rotten Tomatoes  5158 non-null   object 
 5   Netflix          16744 non-null  int64  
 6   Hulu             16744 non-null  int64  
 7   Prime Video      16744 non-null  int64  
 8   Disney+          16744 non-null  int64  
 9   Type             16744 non-null  int64  
 10  Directors        16018 non-null  object 
 11  Genres           16469 non-null  object 
 12  Country          16309 non-null  object 
 13  Language         16145 non-null  object 
 14  Runtime          16152 non-null  float64
dtypes: float64(2), int64(6), object(7)
memory usage: 2.0+ MB
```

Comprobamos el número de total de valores nulos en cada columna:

```
Title                0
Year                 0
Age               9390
IMDb               571
Rotten Tomatoes  11586
Netflix              0
Hulu                 0
Prime Video          0
Disney+              0
Type                 0
Directors          726
Genres             275
Country            435
Language           599
Runtime            592
dtype: int64
```

Ya quedándonos con los no nulos, visualizamos la cantidad de datos que faltan usando la librería missingno, a través del siguiente gráfico ocupando matplotlib:



Se elimina la columna de Rotten Tomatoes, ya que no la tomaremos en cuenta para esto, también las columnas de "Title" y "Type" que también son irrelevantes para esta predicción.

```
Year            0
Age          8857
IMDb            0
Netflix         0
Hulu            0
Prime Video     0
Disney+         0
Directors     483
Genres        101
Country       218
Language      372
Runtime       354
dtype: int64
```

Vemos los valores únicos de la columna "Age":

```
array(['13+', '18+', '7+', nan, 'all', '16+'], dtype=object)
```

Imputamos los valores nulos en "age" con "all", SimpleImputer es la que nos permite sustituir valores nulos por otros valores, ahora nos queda así:

```
Year            0
Age             0
IMDb            0
Netflix         0
Hulu            0
Prime Video     0
Disney+         0
Directors     483
Genres        101
Country       218
Language      372
Runtime       354
dtype: int64
```

Ahora simplemente eliminamos las entradas en las celdas que no hay datos:

```
Year            0
Age             0
IMDb            0
Netflix         0
Hulu            0
Prime Video     0
Disney+         0
Directors       0
Genres          0
Country         0
Language        0
Runtime         0
dtype: int64
```

Visualizamos el gráfico nuevamente para ver si faltan más datos o no:

Vemos ya que no hay datos nulos, verificamos con el método .info() de pandas:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15233 entries, 1 to 16741
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Year          15233 non-null  int64
 1   Age           15233 non-null  object
 2   IMDb          15233 non-null  float64
 3   Netflix       15233 non-null  int64
 4   Hulu          15233 non-null  int64
 5   Prime Video   15233 non-null  int64
 6   Disney+       15233 non-null  int64
 7   Directors     15233 non-null  object
 8   Genres        15233 non-null  object
 9   Country       15233 non-null  object
 10  Language      15233 non-null  object
 11  Runtime       15233 non-null  float64
dtypes: float64(2), int64(5), object(5)
memory usage: 1.5+ MB
```

Ahora ya se pasa a elegir y dividir nuestros datos de entrenamiento y prueba en las variables X e y.

- Las columnas categóricas que se tomarán en cuenta como características son ["Age", "Directors, "Genres", "Country", "Language"]
- Las columnas numéricas que se tomarán en cuenta como características son ['Year', 'Runtime']

A continuación, se visualizan los **datos de entrenamiento**:

ID	Year	Runtime	Age	Directors	Genres	Country	Language
13576	1972	88.0	18+	Vernon Zimmerman	Action,Comedy,Drama,Sport	United States	English
9829	2011	93.0	13+	Nayan Padrai	Comedy,Romance	United States	English
1256	2020	104.0	all	William Wyler	Drama,Romance	United States	English,French
9800	2017	149.0	all	Boyapati Srinu	Action,Romance	India	Telugu
11212	1941	66.0	all	Howard Bretherton	Action,Adventure,Western	United States	English

Ahora los **datos de prueba:**

ID	Year	Runtime	Age	Directors	Genres	Country	Language
8874	2016	90.0	all	Ross Kohn	Thriller	United States	English
12421	2016	87.0	all	Melissa Finell	Comedy	United States	English
8191	2018	79.0	all	Svend Ploug Johansen	Sci-Fi	United States	English,Danish
12796	2017	120.0	18+	Christopher Compton	Action,Adventure,Drama	United States	English
4721	2018	136.0	all	Zhangke Jia	Crime,Drama,Romance	China,France,Japan	Chinese,Mandarin

Copiamos estos datos de entrenamiento y testeo para evitar cambios del dataset original y aplicamos el LabelEncoder que codifica etiquetas de una característica categórica en valores numéricos entre 0 y el número de clases menos 1. Una vez instanciado, el método fit lo entrena (creando el mapeado entre las etiquetas y los números) y el método transform transforma las etiquetas que se incluyan como argumento en los números correspondientes. El método fit_transform realiza ambas acciones simultáneamente.

2.5. Modelos

Pasamos a definir los modelos que en este caso serán 5, ocupando el algoritmo **RandomForestRegressor**:

- El primer modelo (model_1), lo definimos con 50 árboles de estimación y random_state=1 que su función es controlar tanto la aleatoriedad de las muestras utilizadas al construir los árboles, como es 1 el muestreo de las características a considerar al buscar la mejor división en cada nodo será (si max_features < n_features)
- El segundo modelo (model_2), lo mismo solo que con la diferencia que esta vez con 100 árboles de estimación.
- El tercer modelo (model_3), usamos 100 árboles también pero esta vez le añadimos la función ***criterion***, que es una función para medir la calidad de una división (split). Los criterios admitidos son "mse" para el error medio al cuadrado, que es igual a la reducción de la varianza como criterio de selección de características, y "mae" para el error medio absoluto.
- El cuarto modelo (model_4), para este se definen 200 árboles y el número mínimo de muestras necesarias para dividir un nodo interno serán 20.
- Finalmente, el quinto modelo (model_5), se definen 100 árboles y la profundidad máxima del árbol con 20.

Todos estos modelos se guardan en un array llamado ***models***

2.6. Experimentos y resultados

Definimos la función para comparar los modelos, corremos el código y obtenemos los siguientes resultados:

```
Modelo 1 MAE(Error medio absoluto): 0.834292
Modelo 2 MAE(Error medio absoluto): 0.831473
Modelo 3 MAE(Error medio absoluto): 0.827311
Modelo 4 MAE(Error medio absoluto): 0.812408
Modelo 5 MAE(Error medio absoluto): 0.823902
```

El mejor puntaje vemos que corresponde al modelo 4, el cuál es: **0.8124076242731043**

La ejecución de esta función e imprimiendo los resultados duró 213.078 segundos, con una pequeña diferencia respecto al proyecto original que duró 211.1 segundos.

A modo de prueba se le agregó un nuevo modelo (model_6), el cual se definieron 50 árboles y la profundidad máxima del árbol con 7, para ver si salía un mejor resultado:

```
Modelo 1 MAE(Error medio absoluto): 0.834292
Modelo 2 MAE(Error medio absoluto): 0.831473
Modelo 3 MAE(Error medio absoluto): 0.827311
Modelo 4 MAE(Error medio absoluto): 0.812408
Modelo 5 MAE(Error medio absoluto): 0.823902
Modelo 6 MAE(Error medio absoluto): 0.824177
```

Este demoró 188.767 segundos, el resultado como se puede ver del modelo de prueba fué: **0.824177**, por lo que no es mejor que el mejor encontrado anteriormente.

3. Conclusión

Como el mercado de las industrias cinematográficas es cada vez más grande, la competencia también es cada vez más compleja. Por lo tanto, la predicción de la calificación de una película también es cada vez más compleja. Este testeo de estos modelos se ha desarrollado a partir de un conjunto de datos obtenidos en Kaggle del mundo real. El modelo más optimizado también podría utilizarse para predecir otras calificaciones como las de Rotten Tomatoes, etc. Además de las películas, el modelo más óptimo encontrado podría predecir programas de televisión, música, etc. utilizando las características que se propusieron. Habrá que definir algunas características que tienen más influencia en el éxito de las películas y otras que tienen menos o ninguna influencia. Quizás, el presupuesto tiene una pequeña influencia positiva, pero el reparto o el actor/actriz también puede tener influencia en la industria cinematográfica. Quizás se podría trabajar junto con el conjunto de datos de películas de Hollywood, también se puede utilizar el conjunto de datos de películas de Bollywood u otros para que el modelo sea más eficaz. El dataset también puede enriquecerse incluyendo las películas de otras plataformas de streaming (HBO, Paramount, etc.).

References

[1] , . IMDb internet movie database. https://www.imdb.com/.

[2] , . IMDb-Rating-Prediction-. https://github.com/diptaraj23/IMDb-Rating-Prediction-.

[3] , . IMDb-Rating-Prediction-. https://github.com/thejuanka/machine-learning.

[4] , . Metacritic metacritic - movie reviews, tv reviews, game reviews, and ... https://www.metacritic.com/.

[5] , . Predicción IMDb - Proyecto ML.ipynb. https://colab.research.google.com/drive/1Kc4H7G_CgA7M2rTSojcH-IilGXvjB24c?usp=sharing.

[6] , . Rotten Tomatoes rotten tomatoes: Movies | tv shows | movie trailers. https://www.rottentomatoes.com/.

[7] diptaraj23, 2021. IMDb Rating Prediction from a data set of Movies. URL: https://www.kaggle.com/diptaraj23/imdb-rating-prediction-from-a-data-set-of-movies.

[8] ruchi798, 2020. Movies on Netflix, Prime Video, Hulu and Disney+ a collection of movies found on these streaming platforms. URL: https://www.kaggle.com/ruchi798/movies-on-netflix-prime-video-hulu-and-disney.