

**DEPARTAMENTO DE INGENIERÍA INDUSTRIAL**

**Profesor**: David Zarruk Valencia

**Estudiantes:**

Integrante 1: David Guillermo Guerrero Calderón Código: 201113487

Integrante 2: Jhon Jairo Melo Jiménez Código: 201818750

Integrante 3: Julián David Moreno Marín Código: 200822194

Integrante 4: Oscar Iván Castro Buitrago Código: 201315998

Integrante 5: Wilson Camilo Sánchez Forero Código: 201820379

# **PROJECT 3**

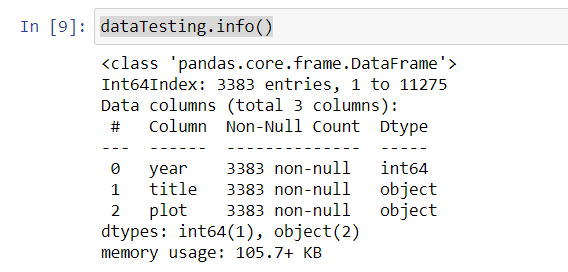
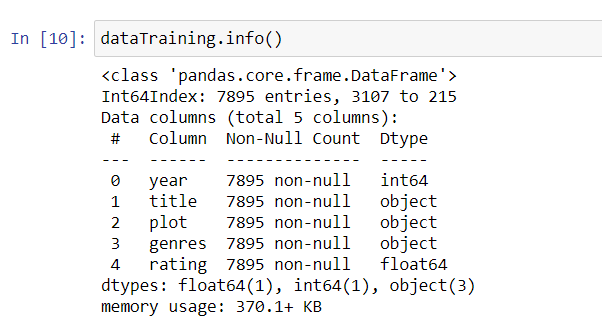
**EXPLORACIÓN DE DATOS**

A la base de datos de Train y Test se le realizaron algunas transformaciones y análisis descriptivo de exploración.

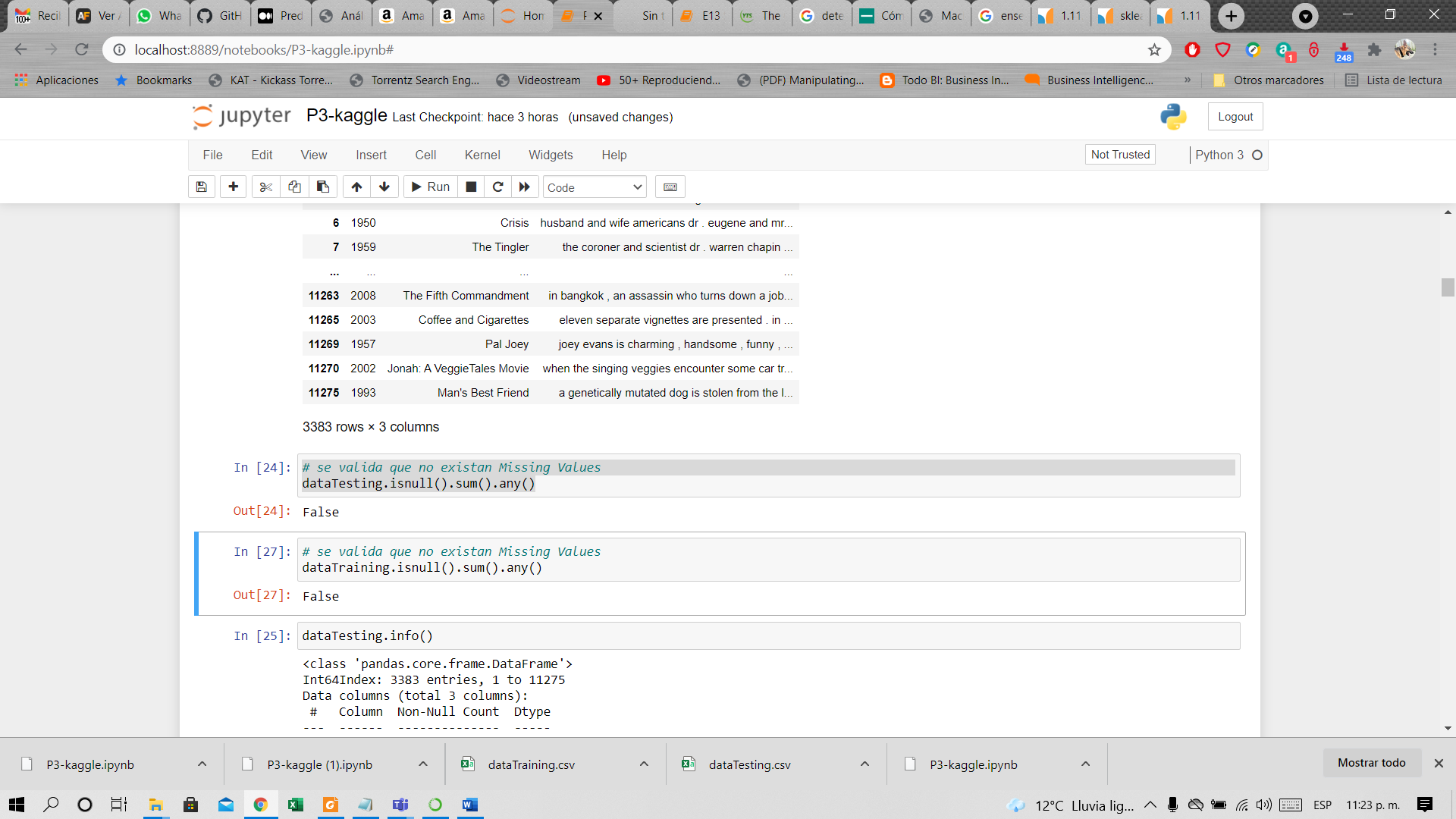
La base de Train contiene 5 Varibales y 7.895 registros y la base de Test contiene 3 variables y 3.383 registros.

La base de Train y Test tienen los siguientes tipos de variables:

**Base Train** **Base Test**



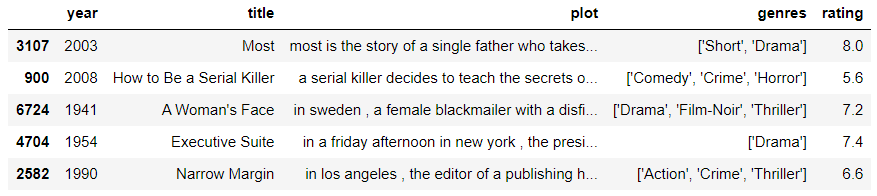
Se valida si las bases registran Missing Values, donde se observa que no tienen.

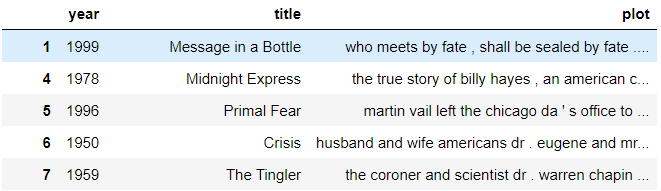


Realizamos un análisis exploratorio de los datos:

Posteriormente se realiza una limpieza de los datos en especifico

# **Movie Genre Classification**[**¶**](https://render.githubusercontent.com/view/ipynb?color_mode=auto&commit=c5b516ad1acf1a6951b8ad90605d365099bc6124&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f64617669647a617272756b2f416476616e6365644d6574686f647344617461416e616c79736973436c6173732f633562353136616431616366316136393531623861643930363035643336353039396263363132342f4578657263697365732f50332d4d6f76696547656e726550726564696374696f6e2e6970796e62&nwo=davidzarruk%2FAdvancedMethodsDataAnalysisClass&path=Exercises%2FP3-MovieGenrePrediction.ipynb&repository_id=383191846&repository_type=Repository#Movie-Genre-Classification)





**Preparación de los datos:**

La información contenida en la variable “plot” es vectorizada para poder realizar el análisis de texto correspondiente:

Escala de tiempo

Descripción generada automáticamente

La variable y corresponde con el genero de cada película, el cual es necesario convertir en matriz para realizar la predicción:

Imagen que contiene Texto

Descripción generada automáticamente

**Modelos:**

Para realizar la clasificación hemos implementado diferentes modelos de predicción, teniendo en cuenta que el resultado debe contemplar una combinación de géneros:

**MPL Classifier**

Perceptron multicapa (MLP) es un algoritmo de aprendizaje supervisado que aprende una función

entrenando en un conjunto de datos, donde es el número de dimensiones para la entrada y es el número de dimensiones para la salida. Dado un conjunto de características y un objetivo, puede aprender un aproximador de función no lineal para clasificación o regresión. Es diferente de la regresión logística, en que entre la capa de entrada y la de salida, puede haber una o más capas no lineales, llamadas capas ocultas. La Figura 1 muestra un MLP de una capa oculta con salida escalar.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

El modelo anterior ha tenido varios ajustes de parámetros, variando la cantidad de redes y de neuronas, sin embargo el roc\_auc\_score no es superior a 0.7



Por lo cual, este modelo ha sido descartado.

LSTM

LSTM es un tipo de red neuronal recurrente, pero es mejor que las redes neuronales recurrentes tradicionales en términos de memoria. Tener un buen control sobre la memorización de ciertos patrones, los LSTM teóricamente deberían funcionar bastante mejor. Este modelo lo desarrollamos para poner a prueba el rendimiento de un modelo de Deep learning que requiere pocos recursos de maquina ya que dentro de nuestro grupo no contamos con gran capacidad de procesamiento.

A pesar de las expectativas iniciales, el modelo de LSTM arrojo los siguientes resultado:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Al correr el modelo obtenemos un roc\_auc\_score inferior a 0.6 por lo cual descartamos el modelo para esta aplicación.

**Random Forest**

Nuestro tercer modelo consistio en un Random Forest que se compone de una gran cantidad de pequeños árboles de decisión que son nuestros estimadores. Cada estimador produce sus propias predicciones. El modelo de Random Forest combina las predicciones de los estimadores para producir una predicción más precisa.

Nuestro tercer modelo conto con un proceso de tuning en donde se establecieron los siguientes hiperparametros: 20.000 estimadores, una profundidad máxima de 24 niveles en las ramificaciones de los arboles, un radom state de 47 y “entropy” como criterio para la medición de impureza en la partición.

Este modelo nos arrojo el mejor resultado en comparación al resto de modelos desarrollados en este competencia con un roc\_auc\_score de 0.8314.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

### **AdaBoostClassifier**

Un clasificador AdaBoost es un metaestimador que comienza ajustando un clasificador en el conjunto de datos original y luego ajusta copias adicionales del clasificador en el mismo conjunto de datos, pero donde los pesos de instancias clasificadas incorrectamente se ajustan de modo que los clasificadores posteriores se enfocan más en casos difíciles.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

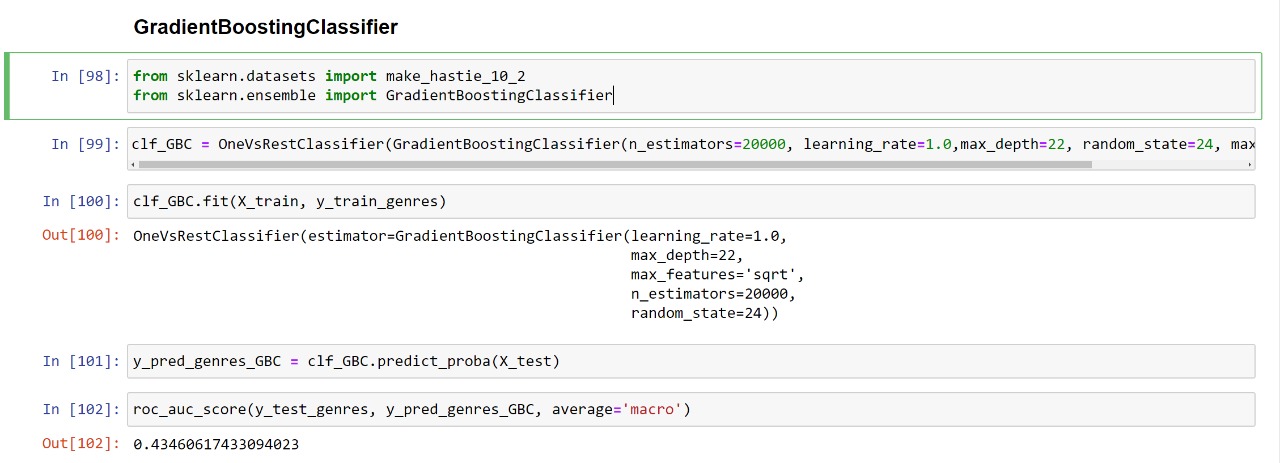
Descripción generada automáticamente

Luego del procesamiento del modelo con 20000 estimadores el score obtenido es de 0.76, por lo cual se descarta el modelo.

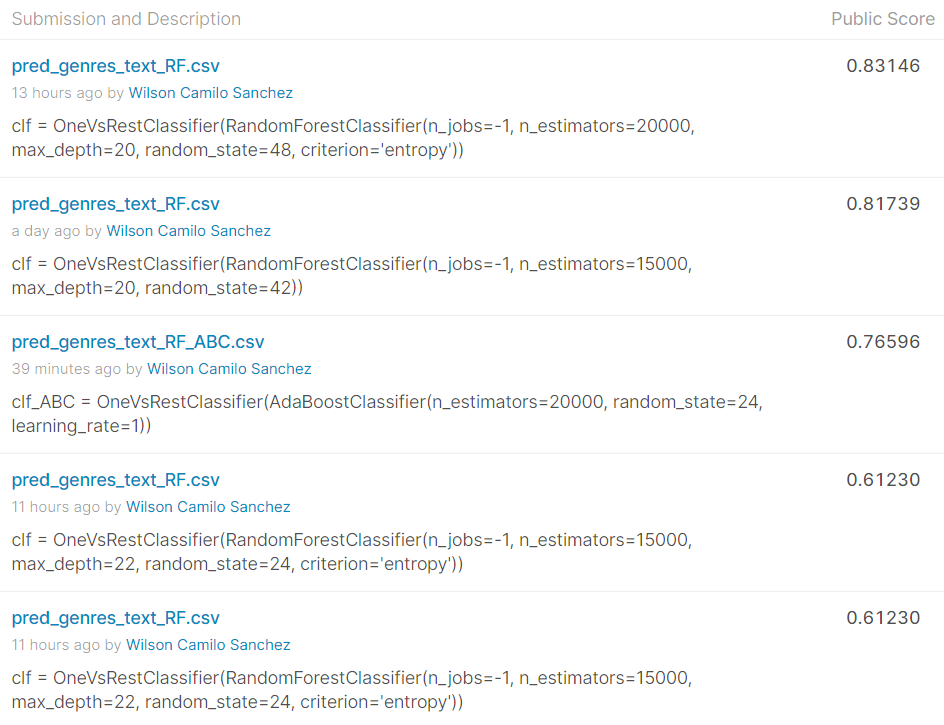
**GradientBoostingClassifier**

El modelo de Gradient Boosting permite la optimización de funciones de pérdida diferenciables arbitrarias. En cada etapa, n\_classes\_ los árboles de regresión se ajustan al gradiente negativo de la función de pérdida de desviación binomial o multinomial. La clasificación binaria es un caso especial en el que solo se induce un único árbol de regresión.

En nuestro caso esperamos un desempeño mas alto que el alcanzado en los arboles de decisión, situación que no se dio. El roc\_auc\_score fue uno de los mas bajos con un resultado de 0.43, siendo uno de los peores modelos dentro de nuestros planteamientos. Este resultado puede deberse a que debido a los limitados recursos de maquina con los que contamos, se tomo la decisión de configurar el hiperparametro “max\_feature = sqrt” buscando limitar el numero de feature utilizada en cada iteración a la raíz cuadrada del total de features en el modelo, situación que claramente limita la capacidad de ajuste del modelo.



De todos los modelos de predicción de clasificación implementados el que mejor tubo resultado en la competencia de kaggle de acuerdo a los datos de test fue el de random forest, con un score de 0.83:



**Conclusiones**

En la búsqueda por generar la mejor predicción multi-clase para la categorización de los géneros de las películas se corrieron modelos de deep learning como redes neuronales y modelos de machine learning como arboles de decisión y gradientes como AdaBoost y GradientBoosting. Estos modelos variaron ampliamente en sus resultados, mostrando niveles de accuracy muy diferentes según la métrica roc\_auc\_score.

Dentro de la implementación de diferentes modelos y generando procesos de tuning en los hiperparametros de los mismo, se llegó a la conclusión que el modelo con el mejor desempeño fue el random Forest con un roc\_auc\_score, incluso sobre modelos como las redes neuronales que inicialmente se consideraban los mejores candidatos para una tarea de clasificación con relativamente pocos “features”, situación que en generar debería mejorar su eficiencia.

Este ejercicio resulto muy interesante y enriquecedor ya que requirió el desarrollo de nuevo conocimiento por parte del equipo de trabajo en el proceso de Tuning y sobre todo en el planteamiento de un ejercicio de predicción multi-clase que no se había abordado con anterioridad en clase.

Por último, se recomienda revisar la literatura para el posterior desarrollo de modelos de ensamble con para predicciones de clasificación multi-clase. Esta metodología podría resultar más eficiente y arrojar una roc\_auc\_score más alto. Para este ejercicio se revisaron varias fuentes de información en internet, pero no se obtuvo literatura clara sobre como implementar este tipo de metodología dentro de los lenguajes de programación como Python o R.