TP2: Críticas Cinematográficas - Grupo 3

Introducción

- Breve descripción del problema a resolver

Se va a tener que construir una serie de modelos donde se va a predecir si una crítica cinematográfica es una crítica positiva o negativa. Y seleccionar aquellos 2 modelos que se consideren los mejores para la competencia en kaggle.

- Breve descripción del dataset: cantidad de registros y columnas, particularidades, etc.

Se cuenta con 2 datasets, un dataset de entrenamiento que consta de 50000 registros y 3 columnas, donde se tiene la ID, la review (el texto de la crítica cinematográfica) y el sentimiento de la crítica (positiva y negativa). Y se tiene un dataset de este donde se tiene 8599 registros y 2 columnas donde cada registro tendrá su ID y su review (el texto de la crítica cinematográfica). Donde la función del dataset de entrenamiento se usará para entrenar a los modelos y el dataset de test serán las observaciones a predecir.

- Breve comentario de **todas** las técnicas exploradas, pruebas realizadas y modificaciones realizadas sobre el dataset. Cualquier implementación realizada por el equipo se debe detallar en esta sección.

Al dataset de entrenamiento se detectar el idioma de la review del filme y asegurándose de que esté en español creando una nueva columna en el dataset indicando el idioma de la crítica.

- Detallar las técnicas de preprocesamiento de datos utilizadas.

Se eliminaron aquellos registros que tienen una review en inglés.

Se hizo una limpieza del texto, colocando todos los caracteres en minúsculas, eliminando acentos, eliminando todos los caracteres que no sean letras (de la A a la Z, sin importar mayúsculas o minúsculas) o espacios, eliminando cualquier carácter no alfabético reemplazandolo por un espacio y reemplaza múltiples espacio consecutivos por un solo espacio.

Se insertó aquella lista de las palabras en español del stop_words que son las palabras más usuales en textos, por ende cualquier palabra que pertenezca a dicha lista será eliminada del texto.

También se lematizo el texto extrayendo el lema de cada palabra del texto y sustituyéndolo en el mismo.

- Listar hipótesis o supuestos que tomaron:

1. Al conjunto de datos para el train (train.csv) proporcionado no se le dividió en un porcentaje para train y test ya que se consideró tomar el mayor conjunto de datos para entrenar a los modelos.

Para calcular las metricas de los modelos se va a utilizar un dataset que consta de 4800 criticas cinematograficas, dicho dataset fue bajado en el siguiente link:
(https://raw.githubusercontent.com/captain500/elmundodelosdatos/main/5_sentiment_analysis/data/criticas.csv)

Que es un data set que proporciona la página *elmundodelosdatos* en el siguiente link:

(https://elmundodelosdatos.com/sentiment-analysis-criticas-peliculas-regresion-logistica/)

Que toma críticas cinematográficas de la página filmaffinity (www.filmaffinity.com). Las críticas están clasificadas con una nota de 1 a 10, se tomó la hipótesis que aquellas películas con un nota mayor o igual a 7 es una crítica positiva, de lo contrario es una crítica negativa.

2. Seleccionamos aquellos modelos con el cual nos de un mejor resultado al momento de calcular las métricas. ya que queremos alcanzar un buen balance entre las métricas.

Cuadro de Resultados

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test	Kaggle
Bayes Naive	0.78698	0.85304	0.73041	0.80229	0.73822
Random Forest	0.74481	0.84505	0.66583	0.77187	0.74297
XG Boost	0.77819	0.82002	0.74041	0.78895	0.70478
Red Neuronal	0.80867	0.87222	0.75375	0.82166	0.75014
Ensamble	0.79853	0.88250	0.72916	0.81604	0.75906

Descripción de Modelos

- Indicar brevemente en qué consiste cada modelo de la tabla: hiperparámetros, modelos ensamblados, técnicas de preprocesamiento de datos, entrenamiento, etc,

    - **Bayes Naive:**

El modelo Bayes Naive es un modelo clasificador probabilístico que utiliza el Teorema de Bayes para calcular probabilidades de cada clase de un conjunto de clases (en nuestro caso las clases son Positiva y Negativa) dado una observación d. Para clasificar una observación nueva, se elige la clase c con la probabilidad más alta del conjunto. Es uno de los modelos más usados para la clasificación de documentos y la clasificación de texto.

Para la búsqueda de hiperparametros se utilizó Randomized Search.
Se mejoraron los siguientes hiperparametros:

1. alpha: Es el parámetro que permite suavizar el Laplace.
2. fit_prior: Es un booleano que indica si se deben aprender las probabilidades a priori (True) o no (False). Si es False, se asumirá que las clases tienen la misma probabilidad a priori.
3. class_prior: permite proporcionar manualmente las probabilidades a priori de las clases.

Por otra parte, se usó K-fold Cross Validation con 10 folds

Los mejores hiperparametros encontrados fueron:

```
{'force_alpha': True, 'fit_prior': False, 'class_prior': [0.2, 0.8],
'alpha': 1.0714285714285714}
```

    - **Random Forest:**

Es un algoritmo de aprendizaje automático que se basa en la técnica de ensamblaje de modelos, especialmente en el ensamblaje de árboles de decisión, que se construyen en forma paralela. Cuando se realiza una predicción, cada árbol emite una predicción, entre todas las predicciones elegidas, se elige el mejor clasificador.
Para la búsqueda de hiperparametros se utilizó Randomized Search y se mejoraron los siguientes hiperparametros.

1. Criterion : función para medir la calidad de la división.
2. Min_samples_leaf: el número mínimo de muestras requeridas en un nodo hoja.

3. Min_samples_split: el número mínimo de muestras requeridas para dividir un nodo
4. N_estimators: el número de árboles en el ensamble

Por otra parte, se usó K-fold Cross Validation con 10 folds

Los mejores hiperparametros encontrados fueron:

```
{'n_estimators': 57, 'min_samples_split': 14, 'min_samples_leaf': 5,
'criterion': 'entropy'}
```

- **XGBoost:**

Está basado en boosting y construye un conjunto de árboles de forma secuencial, y cada uno se enfoca en corregir los errores de los anteriores. Incluye términos de regularización en la función de pérdida para controlar el sobreajuste y mejorar la generalización del modelo.
Para la búsqueda de hiperparametros se utilizó Randomized Search y se mejoraron los siguientes hiperparametros.

1. Learning_rate: Es la tasa de aprendizaje que controla la contribución (los pesos)de cada árbol al modelo final.
2. N_estimators: el número de árboles en el ensamble.
3. Max_depth: Es la profundidad máxima de un árbol.
4. Subsample: La proporción de muestras utilizadas en cada árbol.
5. Lambda: Término de regularización L2 en ponderaciones.
6. Gamma: Es la reducción de pérdida mínima necesaria para realizar una partición adicional en un nodo hoja del árbol.
7. Eta: conocido también como la tasa de aprendizaje. Se puede ajustar para modificar el impacto de cada árbol individual en el modelo final.

Por otra parte, se usó K-fold Cross Validation con 10 folds.

Los mejores hiperparametros encontrados fueron:

```
{'subsample': 0.5517241379310345, 'n_estimators': 120, 'max_depth': 6,
'learning_rate': 0.4214572864321608, 'lambda': 7.894736842105263,
'gamma': 5.684210526315789, 'eta': 0.6000000000000001}
```

- **Red Neuronal:**

Se usó como modelo un Perceptrón Multicapa (MLP) donde cada capa están densamente conectadas entre sí .

Un modelo MLP tiene varias fases:

Primeramente se comienza con la propagación hacia adelante (Forward Propagation), en donde los datos de entrada pasan por las capas de la red hasta llegar a la capa de salida. En cada capa, cada neurona realiza una suma ponderada de los datos de entrada con su pesos correspondientes, y se aplica una función de activación. Después se calculan los errores de las predicciones con respecto a los valores reales utilizando la función de pérdida. Luego se realiza una propagación hacia atrás (propagación desde la capa de salida hacia la capa de entrada) donde se usa la técnica de Backpropagation para realizar un cálculo de derivadas parciales para ajustar el peso de las neuronas minimizando los errores. Este proceso se repite durante un número determinado de épocas.

En este caso se creó un modelo con el que primeramente se usó como capa de entrada de 3192097 características, por lo que la capa de entrada tendrá la misma cantidad de neuronas. Se usaron capas ocultas. Como capa de salida se usó como función de activación la función sigmoid.

Para optimizar el modelo se usó como optimizador Adam y como función de perdida sparse_categorical_crossentropy.

Luego se utilizó como contenedor del modelo KerasClassifier, para el cual se realizó una búsqueda de hiperparametros usando Randomized Search. Se mejoraron los siguientes hiperparametros:

1. Epochs: número de veces que se realiza el proceso de forward (propagación hacia adelante)y la técnica de backpropagation (propagación hacia atrás para recalcular los pesos).
2. Batches: es un conjunto de datos que se procesan en la red neuronal en cada época.

Por otra parte, se usó K-fold Cross Validation con 10 folds

Los mejores hiperparametros encontrados fueron:

```
{'epochs': 5, 'batch_size': 35}
```

- **Ensamble:**

En este caso se usó un ensamblaje de tipo voting en donde se combinan las predicciones de múltiples modelos diferentes de aprendizaje automático en el cual la predicción final se decide de acuerdo con la predicción más votada.

En nuestro caso, se usaron los modelos de Naive Bayes, la Red Neuronal y Random Forest.

- Detallar el caso del mejor modelo.

El ensamble con los modelos Naive bayes, Red neuronal y Random forest.

- Adicionalmente para la red neuronal describir la arquitectura implementada y justificar su elección.

Se determina el número de clases únicas de salida (en 'y_train'), lo que indica el número de neuronas de salida.

La arquitectura implementada consiste en una capa de entrada con 3.192.097 características, una capa densa con una cantidad de neuronas con el mismo número de clases únicas de salida, utilizando la función de activación sigmoid.

Compilando el modelo con un optimizador Adam, una pérdida de sparse_categorical_crossentropy y la métrica de F1.

La justificación de la elección de dicha arquitectura fue por la simplicidad de la arquitectura que consiste en solo una capa densa.

- Mencionar si se probaron técnicas o algoritmos adicionales a las solicitadas en el TP2

Se armó un modelo SVM para ver qué score tenía en kaggle y para ser usado en el ensamblado. Pero no se logró un buen score en kaggle y tenía una performance pobre, por ende se decidió descartar dicha opción.

Se intentó hacer un ensamble utilizando diferentes modelos, entre ellos se trató de usar una combinación de los siguientes modelos: SVM, Naive Bayes, Random Forest, Red neuronal y XGBoost. Y el ensamble que dio una mejor performance fueron los modelos Random Forest, Naive Bayes y Red neuronal.

Conclusiones generales

Las siguientes preguntas son una guía de ayuda para que puedan responder en el informe, pero no son las únicas que esperamos que respondan:

- ¿Fué útil realizar un análisis exploratorio de los datos?

Es fundamental hacer el exploratorio de datos, en este caso verificamos si habían campos nulos, vimos la cantidad de críticas positivas y negativas. Esta información es valiosa para saber la calidad de los datos antes de entrenar los modelos.

- ¿Las tareas de preprocesamiento ayudaron a mejorar la performance de los modelos?

Este preprocesamiento nos ayuda a mejorar el rendimiento de los modelos, ya que al eliminar los caracteres especiales, los stopwords y palabras de un solo carácter nos ayuda a reducir el ruido en los datos Al convertir el texto a minúsculas y eliminar los acentos nos ayuda a la normalización del texto (evita la duplicación de las palabras). Y con la lematización, al reducir las palabras a su forma base nor ayuda a generalizar mejor y reducir la variabilidad de las palabras,

- ¿Cuál de todos los modelos obtuvo el mejor desempeño en TEST?

Red neuronal

- ¿Cuál de todos los modelos obtuvo el mejor desempeño en Kaggle?

El ensamble con los modelos: Random forest, Red neuronal y Naive Bayes .

- Justificación por la cual no se seleccionó los modelos que dieron un mayor score en kaggle

El ensamble con los modelos: Random forest, Red neuronal y Naive Bayes fue el modelo con mayor score. Pero a diferencia de este, se utilizó un modelo de red neuronal con una arquitectura distinta al modelo seleccionado para la red neuronal.
La arquitectura de la red neuronal usada en el modelo de el ensamble con mayor score en kaggle consiste en:

1. Una capa de entrada de 3,192,097 características.
2. Una primera capa oculta de tipo densa con una cantidad de neuronas con el mismo número de clases únicas de salida, con una función de activación sigmoidal y una regularización de l2 para prevenir el sobre ajuste.
3. Una primera capa de Dropout que penalizan el valor de los pesos de la red.
4. Una segunda capa oculta de tipo densa con una cantidad de neuronas con el mismo número de clases únicas de salida, con una función de activación sigmoid y una regularización de l2 para prevenir el sobre ajuste.
5. Una segunda capa de Dropout que penalizan el valor de los pesos de la red.
6. Una capa de salida de tipo densa con el mismo número de clases únicas de salida, y con la función de activación de softmax.

Y tiene el mismo compilado que la red neuronal seleccionada.

```python
def create_model():
    cant_clases=len(np.unique(y_train))

    modelo= keras.Sequential([
    keras.Input(shape=(3192097,)),
    keras.layers.Dense(cant_clases, activation='sigmoid',
                       kernel_regularizer=keras.regularizers.l2(0.001)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(cant_clases,  activation='sigmoid',
                       kernel_regularizer=keras.regularizers.l2(0.001)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(cant_clases, activation='softmax')
    ])

    modelo.compile(
      optimizer="adam",
      loss="sparse_categorical_crossentropy",
      # metricas para ir calculando en cada iteracion o batch
      metrics=['f1_score'],
    )

    return modelo
```

Este tiene un score en kaggle más alto que el de modelo seleccionado de la red neuronal.

df_pred_rn_keras_tf_v7.csv
Complete · Ricardo Contreras · 2h ago
0.76933

Pero al calcular las métricas del ensamble (que contiene dicha arquitectura de la red neuronal mencionada) y de la red neuronal, vemos que no tienen buenas métricas y tienen unas métricas peores que los modelos de ensamble y de la red neuronal seleccionadas para el presente trabajo.

```
Modelo Red Neuronal (RN con mayor score en kaggle):
Precisión: 0.930161943319838
Recall: 0.3829166666666667
F1-score: 0.5425029515938606
Accuracy: 0.6770833333333334
```

```
Modelo Ensamble voting_rn_nb_rf (ensamble con mayor score en kaggle):
Precisión: 0.8971797884841363
Recall: 0.63625
F1-score: 0.7445148707947343
Accuracy: 0.7816666666666666
```

Cuadro de comparación:

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test	Kaggle
Red Neuronal Seleccionado	0.80867	0.87222	0.75375	0.82166	0.75014
Red Neuronal con mayor score en kaggle	0.5425	0.93016	0.3829	0.6770	0.76933
Ensamble Seleccionado	0.79853	0.88250	0.72916	0.81604	0.75906
Ensamble con mayor score en kaggle	0.7816	0.8971	0.63625	0.7816	0.77301

Como la idea es tener un buen balance entre las métricas y además tener unas buenas métricas con datos que nunca ha visto el modelo.

Por esta justificación no se decidió seleccionar los modelos con mayor score en kaggle.

- ¿Cuál fue el modelo más sencillo de entrenar y más rápido? ¿Es útil en relación al desempeño obtenido?

El modelo de Naive Bayes ya que es un modelo simple y rápido de entrenar y puede ser útil para buscar una solución rápida a un problema o hacer un análisis rápido de un modelo base. Además es un modelo que usa el aprendizaje Bayesiano que es usado mayormente en la clasificación de textos y documentos. Sacando de lado el modelo del ensamble, fue el que mejor desempeño tuvo en Kaggle, así que es útil en relación al desempeño obtenido.

- ¿Cree que es posible usar su mejor modelo de forma productiva?

Creemos que un dataset de 30000 datos es un conjunto de datos pequeño comparado con los conjuntos de datos utilizados en forma productiva, por lo que a la hora de ponerlo en marcha para predecir una nueva observación en un conjunto de datos muy grande con el que el modelo no entreno,, no creemos que

sea la mejor opción para usarlo en forma productiva. El modelo que mejor desempeño tuvo en Kaggle fue el del ensamblaje y dado que los modelos que se utilizaron son complejos, pensamos que no se ajustaría bien a otros datos en forma productiva (alta varianza). Por otra parte, los modelos de ensamblaje tardaron mucho en ser entrenados. Un ejemplo de ello es Random Forest, que duró 7 horas de entrenamiento. Creemos que podemos encontrar modelos más rápidos a la hora de ponerlos en marcha en producción. Otro punto a tener en cuenta es la forma en la se optimizaron los modelos, esto es visible dado que en el concurso de kaggle hay grupos que tuvieron un mejor rendimiento utilizando modelos similares con el mismo conjunto de datos no visible para todos.

- ¿Cómo podría mejorar los resultados?

Tal vez podría mejorar los resultados si se puede eliminar de los textos las palabras menos frecuentes, es decir eliminar aquellas palabras que aparecen menos de mil veces en los textos.

También queremos destacar que fue fácil caer en overfitting a la hora de entrenar modelos. Si bien pasaba que se alcanzaba un score alto en entrenamiento, el rendimiento en Kaggle era bajo comparado con el entrenamiento. Un ejemplo de ello era entrenar el modelo de Naive Bayes que en entrenamiento logró alcanzar un score de 0,88 mientras que en Kaggle se alcanzaba un score de 0,73. Lo mismo pasa con el modelo de Red Neuronal puesto que tiene el mejor desempeño en entrenamiento pero en Kaggle, no es el mejor score que obtuvimos de todos los modelos.

Tareas Realizadas

Teniendo en cuenta que el trabajo práctico tuvo una duración de 9 (nueve) semanas, le pedimos a cada integrante que indique cuántas horas (en promedio) considera que dedicó semanalmente al TP

Integrante	Principales Tareas Realizadas	Promedio Semanal (hs)
Ricardo Contreras	Armado de informe. Preprocesamiento. Modelo Naive Bayes. Modelo XGboost. Modelo Redes Neuronales. Ensamble Preprocesamiento del dataset de métricas. Métricas.	17

Claudia Ramos	Armado del reporte Modelo Naive Bayes Modelo Random Forest	10
Adriana Macarena Iglesias Tripodi	Preprocesamiento Modelo Naive Bayes Modelo XGBoost Modelo Random Forest	10
Anita Vernieri	Preprocesamiento Modelo Naive Bayes Modelo Random Forest Modelo Redes Neuronales	10