Checkpoint 4

Redes neuronales

Estructura de la red utilizada

La red tiene una capa de entrada de 107 neuronas valor igual a la cantidad de columnas de nuestro set de entrenamiento, una capa oculta de 80 neuronas con una funcion de activacion tipo relu y finalmente una capa de salida con una unica neurona que utliza la funcion de activacion sigmoidea para hacer la prediccion final.

Como la funcion sigmoidea devuelve un valor entre 0 y 1 el criterio usado para decidir si la reserva fue cancelada es que este valor sea mayor a 0.6.

Regularizacion

Para el entrenamiento de la red se uso early stopping.

El early stopping, o detención anticipada, es una técnica utilizada durante el entrenamiento de modelos para evitar el sobreajuste y mejorar la generalización. Consiste en detener el entrenamiento del modelo cuando se observa que el rendimiento en un conjunto de validación comienza a empeorar, a pesar de que el rendimiento en el conjunto de entrenamiento sigue mejorando.

Los parametros utilizados para el early stopping fueron

min_delta = 0.001

patience = 5

monitor = ‘loss’

Hiperparametros

Hiperparametro	Explicacion
neuronas	Es el hiperparámetro que determina el número de neuronas en una capa oculta de una red neuronal.
learning_rate	Controla la tasa de aprendizaje del algoritmo de optimización utilizado durante el entrenamiento de la red neuronal. Determina la magnitud del ajuste que se realiza en los pesos de la red en cada iteración.

beta_1	Es especifico del algoritmo de optimizacion Adam. Controla la tasa de decaimiento del promedio móvil exponencial de los gradientes pasados.
beta_2	Es especifico del algoritmo de optimizacion Adam. Controla la tasa de decaimiento del promedio móvil exponencial de los cuadrados de los gradientes pasados.
epsilon	Es especifico del algoritmo de optimizacion Adam. Se utiliza para evitar la división por cero cuando se calcula la actualización de los pesos. Es un valor muy pequeño que se suma al denominador en el cálculo.

Mejores hiperparametros obtenidos

Para la obtencion de los mejores hiperparemtros hicimos una busqueda con RandomCv con 5 folds y el numero de combinaciones default (10).

neuronas	80
learning_rate	0.001
epsilon	1e-07
beta_2	0.999
beta_1	0.9

Mejores metricas obtenidas

- f1-score train = 0.80
- f1-score validacion: 0.81

Conclusiones

Modelo	f1-score entrenamiento	f1-score validacion (kaggle)
Arbol de decision	**0.8300**	**0.8200**
K-nearest neighbours	**0.7700**	**0.7800**
SVM	**0.7500**	**0.8700**
Random Forest	**0.8800**	**0.8689**
XgBoost	**0.8800**	**0.8712**
Ensamble hibrido voting	**0.8775**	**0.8719**
Ensamble hibrido stacking	**0.8825**	**0.8719**
Red neuronal	**0.8000**	**0.8125**

Decision Tree

El entrenamiento de este modelo fue sencillo de realizar, con un tiempo de entrenamiento aceptable y con un buen rendimiento.

K-Nearest Neighbors

Con knn se obtuvieron las metricas mas bajas de entre todos los modelos entrenados. Se podria mejorar el rendimiento del entrenamiento haciendo un analisis mas profundo en cuanto a outliers ya que el modelo es muy sensible en este aspecto.

Support Vector Machine

Para este modelo, era necesario reducir la dimensión de los datos(en este caso, utilizando PCA), para reducir los el tiempo de entrenamiento.Aun asi, el modelo se demoraba mucho tiempo utilizando los kernel de “poly” o “linear”.

Ademas, al buscar los hiperparámetros optimos, el modelo continuaba siendo lento incluso al agregar nuevos parámetros.

Por estas razones, este modelo fue uno de los que obtuvo un rendimiento inferior al realizar predicciones.

Random Forest

Tuvo un entrenamiento similar al caso del arbol de desicion (en sentido de que los parametros eran muy similar), teniendo una prediccion superior a este y un tiempo de espera aceptable.

XgBoost

Este modelo fue uno de los que mas tiempo de entrenamiento consumio y a la vez el que mejor metrica obtuvo en validacion. Esto tiene sentido ya que xgboost es un ensamble y estos se comportan mejor ante datos nuevos.

Ensamble Hibrido: Voting

Para este modelo se utilizo los mejores modelos que se obtuvieron (omitiendo SVM ya que demoraba demasiado), dando como resultado similar a Xgboost

Ensamble Hibrido: Stacking

Se utilizaron los mismos modelos del caso voting, teniendo el mejor modelo en la salida, con un tiempo de respuesta un poco alto (a comparación de Voting)

Redes neuronales

El entrenamiento de la red neuronal termino dando buenas métricas pero no fue la que mejor resultado dio.

Nota: Es posible que al tener todo en un mismo notebook por problemas de sincronizacion haya submits a kaggle que se generaron que tengan el nombre de un modelo que no corresponde.