Proyecto Final IA - Clasificador Binario para predecir si un vuelo se va a retardar en USA

Departamento de Ingeniería Electrónica, Electrónica y Computación
Pontificia Universidad Javeriana
Sergio Andrés Castellanos Landazábal, sergiocastellanos@javeriana.edu.co

***Resumen*—El propósito de este proyecto es comparar dos clasificadores binarios con sus mejores configuraciones, para ver cual realiza mejor la predicción de si un vuelo se va a atrasar en USA. Utilizando un dataset de kaggle como conjunto de entrenamiento y de prueba. Este dataset tiene datos como el numero de pasajeros, en que parte del día sale el vuelo, de que aeropuerto sale, etc.**

***Index Terms*—Classification Algorithms, Neural Networks, Support Vector Machine.**

I. INTRODUCCIÓN

ESte proyecto utilizará el dataset "2019 Airline Delays w/Weather and Airport Detail.[el] cual se puede encontrar en kaggle. Este dataset como menciona su nombre contiene datos de algunos vuelos que hubieron en Estados Unidos en el 2019 y los primeros meses del 2020. Sin embargo, debido a que son muchos datos en este proyecto se decidió utilizar el small dataset que se encuentra ahí mismo, este dataset sólo contiene los vuelos que sucedieron en Enero, por tanto es mucho más pequeño.

El dataset cuenta con 26 características y la etiqueta que dice si el vuelo se retrasó más de 15 minutos. Pero para este proyecto sólo se utilizarán 11 de estas características:

- DAY OF WEEK
- DEP BLOCK
- CONCURRENT FLIGHTS
- NUMBER OF SEATS
- CARRIER NAME
- AIRLINE AIRPORT FLIGHTS MONTH
- PLANE AGE
- DEPARTING AIRPORT
- PRCP: Precipitación máxima del día.
- TMAX: Temperatura máxima del día
- AWND: Viento máximo del día.

El objetivo es entrenar dos clasificadores binarios con este dataset, el primer método es una red neuronal y el otro una máquina de soporte vectorial, y observar cual tiene el mejor rendimiento para esta solución. Sin embargo, primero hay que realizar un preprocesamiento de los datos antes de entrenar los modelos.

II. DESARROLLO

II-A. Preprocesamiento

En primer lugar, se toman 2000 datos del dataset, ya que así sea small sigue siendo muy grande. Posteriormente, se obtienen las 11 características que se mencionaron anteriormente y la etiqueta. Este dataset no cuenta con valores NAN, sin embargo, las características DEP BLOCK, CARRIER NAME y DEPARTING AIRPORT contienen nombres, por tanto, hay que darle un valor númerico a estas, para ello se crean diccionarios conteniendo cada nombre.

Ya con los datos limpios, se realiza la normalización de los datos. Para esto se utilizan dos métodos Min-Max Scaler con un rango de -1 a 1 y el Standar Scaler de la librería de SKLearn.

Luego a los datos normalizados con los dos métodos se les realiza PCA, reduciendo a 10 características. Observando su varianza se determina que los datos normalizados por el método de Min-Max Scaler son los mejores para realizar el entrenamiento y prueba de los clasificadores.

Finalmente, se separa el dataset en dos, la parte de entrenamiento y la de prueba, mediante la función train_test_split.

```
0.9934848079099791
varianza explicada por cada caracteristica después de PCA: [0.34616362 0.18163149 0.12661575 0.11482854 0.07754263 0.
 0.04520745 0.02692308 0.01919217]
0.9914118944549161
varianza explicada por cada caracteristica después de PCA: [0.28283934 0.19023665 0.11441445 0.10729434 0.09412136 0.
 0.05759235 0.03913932 0.02770535]
```

Figura 1. Varianza PCA

II-B. Métricas de rendimiento

Con el fin de evaluar los clasificadores se utilizaran las siguientes métricas:

- F1 Score
- Coeficiente de Matthews
- Curva de ROC

El resultado nos mostrara que clasificador es el mejor para nuestra solución.

II-C. Red Neuronal

En primer lugar, se desarrolla el primer clasificador una red neuronal. Esta red neuronal tiene varios hiper parámetros que se deben cambiar, con el fin de encontrar los que mejor solucionen el problema de clasificación.

Los primeros hiperparámetros son que activation and solver va a utilizar la red neuronal. Estos se iteran con dos for y se halla el score de cada combinación de estos parámetros, guardando el conjunto de parámetros con mejor score. Dando como resultado identity activation y adan solver, como se observa a continuación:

```
[ ]  activations =['identity', 'logistic', 'tanh', 'relu']
     solvers = ["lbfgs", "sgd","adam"]
     max_score= 0

     for i in activations:
       for j in solvers:
         clf_MLP = MLPClassifier(activation=i, solver=j, max_iter=10000, random_state=1)
         clf_MLP.fit(X_train, np.ravel(y_train))
         score= clf_MLP.score(X_test, y_test)
         if(max_score < score):
           max_score = score
           best_a= i
           best_s= j

     print("max_score",max_score)
     print("best_Activation",best_a)
     print("best_Solver",best_s)

     max_score 0.838
     best_Activation identity
     best_Solver adam
```

Figura 2. Activation and Solver

Posteriormente, se encuentra el mejor hidden layer size o el número de capas sin contar la de entrada y la de salida de nuestra red neuronal, esto mediante un for de 1 a 100. Dando como resultado 99 capas.

```
max_score=0
for i in range(1,100):
  clf_MLP = MLPClassifier(activation=best_a, solver=best_s, hidden_layer_sizes=i, max_iter=10000, random_state=1)
  clf_MLP.fit(X_train, np.ravel(y_train))
  score= clf_MLP.score(X_test, y_test)
  if(max_score < score):
    max_score = score
    best_l= i

print("max_score",max_score)
print("Best layer",best_l)

max_score 0.842
Best layer 99
```

Figura 3. Layer hidden size

Luego, se itera el learning rate init de 0.001 a 100 de a pasos de 0.1, se empieza desde 0.001 ya que es el valor por default.

A continuación, con los mejores hiper parámetros establecidos se configura y se entrena la red neuronal para luego realizar las predicciones con los datos de pruebas.

```
clf_MLP = MLPClassifier(activation=best_a, solver=best_s, hidden_layer_sizes=best_l, learning_rate_init=best_lr, max
clf_MLP.fit(X_train, np.ravel(y_train))
print('Score:',max_score)
y_test_predicted = clf_MLP.predict(X_test)
y_test_scores = clf_MLP.predict_proba(X_test)
```

Figura 4. Configuración de la red neuronal

Finalmente, se hallan las métricas de evaluación. El F1 score da como resultado 0.841, el coeficiente de Matthews es igual a 0.5 y el área bajo la curva de ROC es de 78 % como se observa en figura 5 hay mayores valores True Positive.

```
Score: 0.842
Coeficiente de Matthews: 0.5056921465603261
F1 Score:  0.842
```

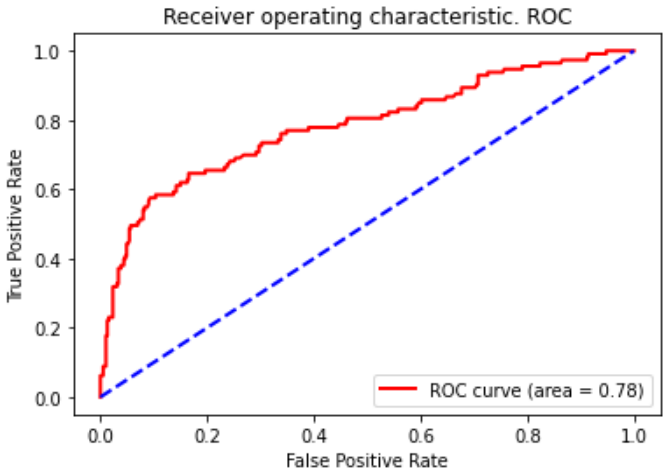

Figura 5. Métricas de evaluación de la red neuronal

II-D. Máquina de soporte vectorial (SVM)

Por el otro lado, se realiza la máquina de soporte vectorial, la cual al igual que en el método de redes neuronales también tiene un hiper parámetro que iterar.

El hiper parámetro a iterar es el kernel que utiliza el algoritmo para realizar la clasificación, hay cuatro kernels, linear, poly, rbf y sigmoid. Al igual que en los puntos anteriores, se itera con un for y se selecciona el kernel con mayor score. Como se observa en la figura 6 se selecciona el kernel poly.

```
kernels =['linear', 'poly', 'rbf', 'sigmoid']
max_score=0
for i in kernels:
  Svm = svm.SVC(kernel= i)
  Svm.fit(X_train, np.ravel(y_train))
  score= Svm.score(X_test, y_test)
  if(max_score < score):
    max_score = score
    best_k= i
print("max_score",max_score)
print("Best Kernel:",best_k)

max_score 0.826
Best Kernel: poly
```

Figura 6. Selección de Kernel

A continuación, se configura, se entrena el clasificador y se prueba con los respectivos datos de prueba, como se observa en la figura 7.

```
Svm = svm.SVC(kernel= best_k)
Svm.fit(X_train, np.ravel(y_train))
print('Score:',max_score)
y_test_predicted = Svm.predict(X_test)
y_test_scores = Svm.decision_function(X_test)
```

Figura 7. Configuración SVM

Finalmente, se hallan las métricas de rendimiento. El F1 score da como resultado 0.841, el coeficiente de Matthews es igual a 0.5 y el área bajo la curva de ROC es de 78 % como se observa en figura 5 hay mayores valores True Positive.

```
Score: 0.826
Coeficiente de Matthews: 0.45549110090554507
F1 Score:  0.826
```

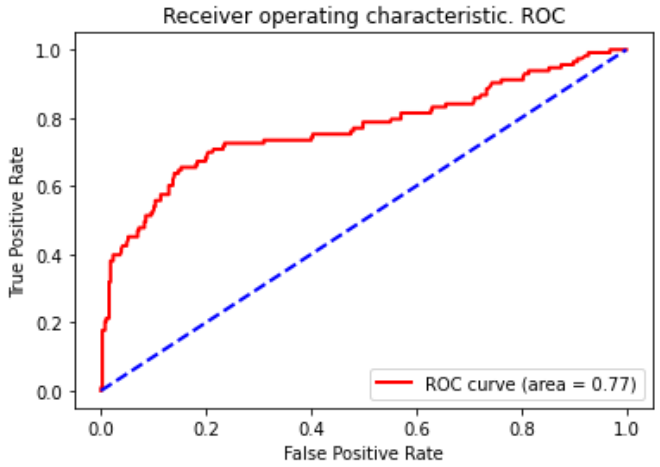

Figura 8. Métricas de evaluación de SVM

III. CONCLUSIONES

En conclusión se puede observar que el mejor modelo para la solución es la red neuronal, ya que el F1 score y el porcentaje de área bajo la curva de ROC es mayor que la del modelo SVM, sin embargo, SVM también tiene muy buenos resultados.

Además, es importante seleccionar los mejores hiper parámetros de cada clasificador, para que así la solución sea siempre la mejor, ya que de estos depende la calidad del modelo y depende del dataset y la problemática unos funcionan mejor que otros.

Finalmente, para mejorar los resultados obtenidos se pueden utilizar más datos en el proceso de entrenamiento y de prueba, e incluir la característica de los meses.

IV. REFERENCIAS

1 "sklearn.neural_network.MLPClassifier — scikit-learn 1.0.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (accessed Nov. 28, 2021).

2 "sklearn.svm.SVC — scikit-learn 1.0.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (accessed Nov. 28, 2021).

3 "2019 Airline Delays w/Weather and Airport Detail — Kaggle." https://www.kaggle.com/threnjen/2019-airline-delays-and-cancellations?select=train_test_small.csv (accessed Nov. 28, 2021).