Julian David Rangel - 89721

Siniestros viales

Arboles de decisión

```
#Definición del modelo
arbol1 = DecisionTreeClassifier()
#arbol1 = GridSearchCV(dt, param_grid, cv=5, scoring='f1')
#Entrenamiento y evaluación del modelo
arbol1 = arbol1.fit(X_train,y_train)

# Calcular métricas de desempeño
y_pred = arbol1.predict(X_test)
print("\n", metrics.classification_report(y_test, y_pred, digits=2))

# Visualizar matriz de confusión
# Y_pred13 = np_utils.to_categorical(y_pred13)
# cm = matriz_confusion(Y_test, y_pred, 'si', 'Matriz de confusión clasificador AD')
```

```
              precision    recall  f1-score   support

           0       0.53      0.55      0.54     20310
           1       0.06      0.07      0.06       972
           2       0.77      0.75      0.76     38462

    accuracy                           0.67     59744
   macro avg       0.45      0.45      0.45     59744
weighted avg       0.67      0.67      0.67     59744
```

Se realiza pruebas en la definición del modelo para mejorar los valores de precisión, recall y f1-score y se establece lo siguiente:

- Se establece el parámetro de max_depth en 10 para limitar la profundidad máxima del árbol de decisión, para prevenir el sobreajuste y mejorar la generalización del modelo.
- Se establece el parámetro de criterion en gini, ya que se determino que en el modelo era mejor que entropy ya que mide la impureza de una partición. Cuanto menor sea el valor de Gini, mejor será la calidad de la partición.
- Se establece el parámetro de random_state en 42 para controlar la aleatoriedad del algoritmo.

```python
#Definición del modelo
arbol1 = DecisionTreeClassifier(
    max_depth         = 10,
    criterion         = 'gini',
    random_state      = 42
)
#arbol1 = GridSearchCV(dt, param_grid, cv=5, scoring='f1')
#Entrenamiento y evaluación del modelo
arbol1 = arbol1.fit(X_train,y_train)

# Calcular métricas de desempeño
y_pred = arbol1.predict(X_test)
print("\n", metrics.classification_report(y_test, y_pred, digits=2))

# Visualizar matriz de confusión
# Y_pred13 = np_utils.to_categorical(y_pred13)
# cm = matriz_confusion(Y_test, y_pred, 'si', 'Matriz de confusión clasificador AD')
```

```
              precision    recall  f1-score   support

           0       0.84      0.42      0.56     20310
           1       0.18      0.01      0.02       972
           2       0.75      0.97      0.85     38462

    accuracy                           0.77     59744
   macro avg       0.59      0.47      0.47     59744
weighted avg       0.77      0.77      0.74     59744
```

Se evidencia mejora en cuanto a el macro avg y weighted avg de precisión, recall y f1-score.

Random Forest

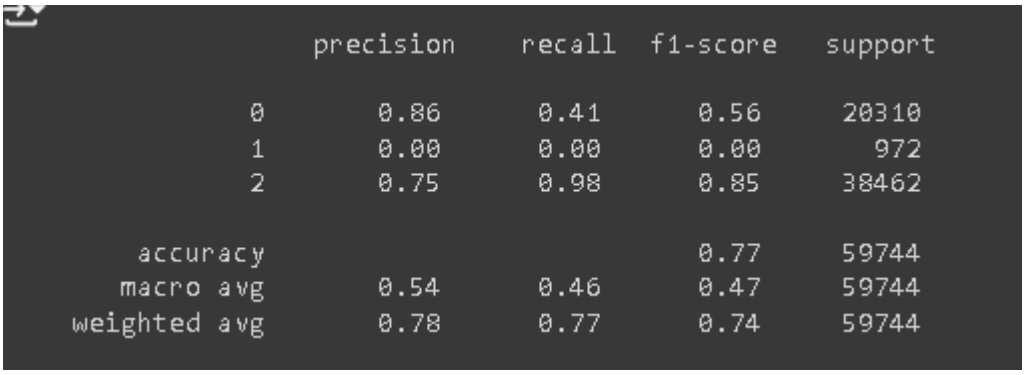
```
#Definición del modelo
rfc = RandomForestClassifier(n_estimators=200)

#Entrenamiento y evaluación del modelo
rfc.fit(X_train, y_train)

# Calcular métricas de desempeño
y_pred2 = rfc.predict(X_test)
print("\n", metrics.classification_report(y_test, y_pred2, digits=2))
```

```
              precision    recall  f1-score   support

           0       0.78      0.45      0.57     20310
           1       0.00      0.00      0.00       972
           2       0.76      0.95      0.84     38462

    accuracy                           0.76     59744
   macro avg       0.51      0.47      0.47     59744
weighted avg       0.76      0.76      0.74     59744
```

Se realiza pruebas en la definición del modelo para mejorar los valores de precisión y se establece lo siguiente:

- Se establece el parámetro de max_depth en 20 para limitar la profundidad máxima del árbol de decisión, para prevenir el sobreajuste y mejorar la generalización del modelo.

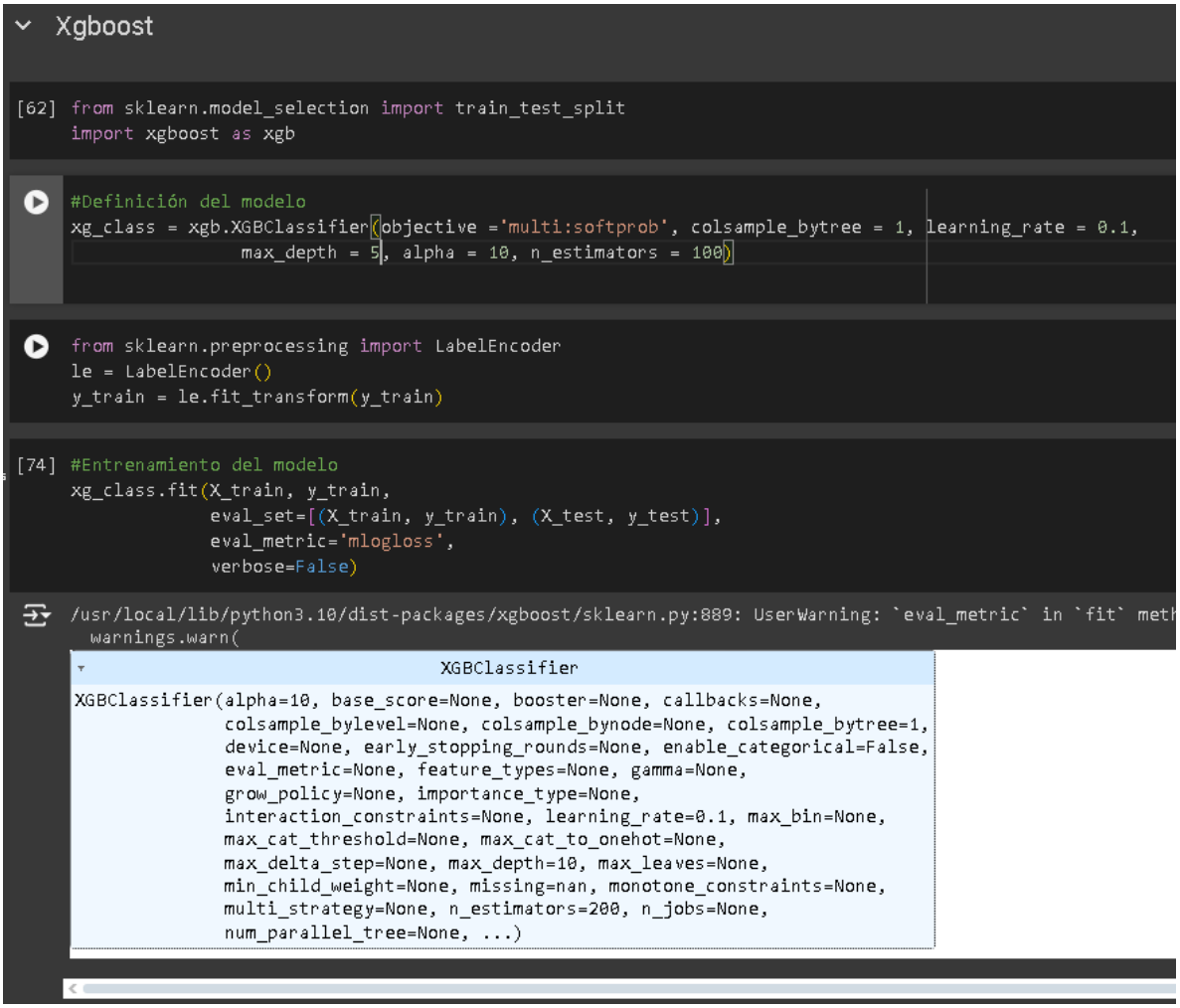
```
#Definición del modelo
rfc = RandomForestClassifier(n_estimators=200, max_depth=20)

#Entrenamiento y evaluación del modelo
rfc.fit(X_train, y_train)

# Calcular métricas de desempeño
y_pred2 = rfc.predict(X_test)
print("\n", metrics.classification_report(y_test, y_pred2, digits=2))
```



	precision	recall	f1-score	support
0	0.86	0.41	0.56	20310
1	0.00	0.00	0.00	972
2	0.75	0.98	0.85	38462
accuracy			0.77	59744
macro avg	0.54	0.46	0.47	59744
weighted avg	0.78	0.77	0.74	59744

Se evidencia mejora en cuanto a el macro avg y weighted avg de precisión

Xgboost




```
#Entrenamiento y validación cruzada mediante k-fold
scores = cross_val_score(xg_class, X_train, y_train, cv=5)
print("Mean cross-validation score: %.2f" % scores.mean())

# Calcular métricas de desempeño
y_pred4 = xg_class.predict(X_test)
print("\n", metrics.classification_report(y_test, y_pred4, digits=2))
```

```
Mean cross-validation score: 0.77

              precision    recall  f1-score   support

           0       0.88      0.41      0.56     20310
           1       0.00      0.00      0.00       972
           2       0.75      0.98      0.85     38462

    accuracy                           0.77     59744
   macro avg       0.54      0.46      0.47     59744
weighted avg       0.78      0.77      0.74     59744

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_class
```

Se realiza pruebas en la definición del modelo para mejorar los valores de precisión y se establece lo siguiente:

- Se establece el parámetro de max_depth en 10 para limitar la profundidad máxima del árbol de decisión, para prevenir el sobreajuste y mejorar la generalización del modelo.
- Se establece el parámetro de n_estimators en 200 para maximizar el numero de arboles a construir, mejorando el rendimiento del modelo, pero aumentando el tiempo de entrenamiento del modelo

```
#Definición del modelo
xg_class = xgb.XGBClassifier(objective ='multi:softprob', colsample_bytree = 1, learning_rate = 0.1,
                max_depth = 10, alpha = 10, n_estimators = 200)
```

```
              precision    recall  f1-score   support

           0       0.83      0.44      0.57     20310
           1       1.00      0.00      0.00       972
           2       0.76      0.97      0.85     38462

    accuracy                           0.77     59744
   macro avg       0.86      0.47      0.47     59744
weighted avg       0.78      0.77      0.74     59744
```

Se evidencia mejora en cuanto a el macro avg y weighted avg de precisión.