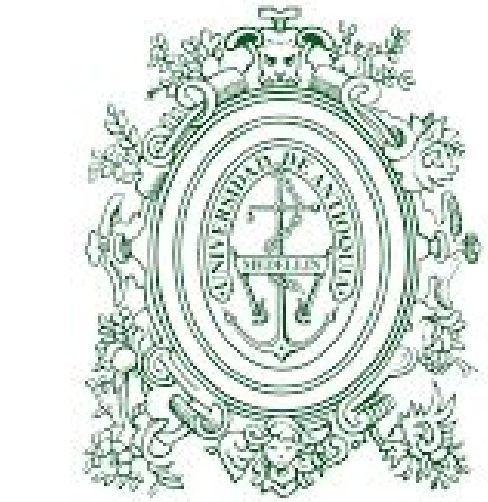
PROYECTO FASE 2



Estudiantes:
Karol Daniela Alzate Mejía
Cristian Camilo Herrera Altare
Joan Esneider Duque Jaramillo

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL PARA LAS CIENCIAS DE LA INGENIERÍA
Codigo: 2508919 - 01
Pro: Raul Ramos Pollan
Departamento de ingeniería
Universidad de Antioquia

1. Realizar el preprocesado y limpieza de datos

Para el preprocesado se utilizó la librería de pandas para leer los archivos .csv donde venía la información del dataset. El dataset estaba dividido en dos partes, pero para comprobar los tipos de datos y corroborar que no existieran valores atípicos o nulos, se justaron ambos datasets. Después de realizado este procedimiento se encontró que el dataset estaba limpio, entonces se procedió a utilizar el Label Enconder para pasar aquellas características que tenían un dato tipo object, para ser numérico.

2. Encontrar los mejores hiperparámetros para DOS algoritmos predictivos

Para hallar lor mejores hiperparámetros, primero se decidió usar como estrategia de validación el KFolds, luego, se utilizó la herramienta de GridSearch que permite visualizar una promedio total con el número de KFolds de la métrica de desempeño que, en nuestro caso es el RMSE, para configurar los diferentes hiperparámetros de los diferentes modelos con el fin de visualizar con cuál de estos hiperparámetros el modelo tuvo mejor desempeño.

Antes de usar el GridSearch, se realizó un pipeline con Standard Scaler, para escalar los datos, y el modelo. Los modelos que se utilizaron para este punto fueron Random Forest Regressor y MLP Regressor.

Para MLP se utilizaron lo siguientes hiperparámetros como prueba.

```
parameters = {'mlpregressor__hidden_layer_sizes': [30, 40, 45],
              'mlpregressor__activation': ('identity', 'logistic', 'tanh', 'relu'),
              'mlpregressor__solver':('lbfgs', 'sgd', 'adam'),
              'mlpregressor__learning_rate':('constant', 'invscaling')}
```

Obteniendo esta tabla como resultado, donde se muestra que el elemento 6 de la tabla fue el que mejor desempeño mostró frente a los demás

.

	mean_fit_time	params	mean_test_mean_squared_error	rank_test_mean_squared_error	mean_train_mean_squared_error
0	0.080768	{'mlpregressor__activation': 'identity', 'mlpr...	-2.002375	2	-1.602666
1	0.709054	{'mlpregressor__activation': 'identity', 'mlpr...	-2.006431	8	-1.604793
2	0.769835	{'mlpregressor__activation': 'identity', 'mlpr...	-4.313965	38	-4.001295
3	0.080699	{'mlpregressor__activation': 'identity', 'mlpr...	-2.002420	5	-1.602666
4	0.637888	{'mlpregressor__activation': 'identity', 'mlpr...	-49.390572	68	-50.639832
5	0.748348	{'mlpregressor__activation': 'identity', 'mlpr...	-3.764400	28	-3.491021
6	0.099507	{'mlpregressor__activation': 'identity', 'mlpr...	-2.002331	1	-1.602666
7	0.703575	{'mlpregressor__activation': 'identity', 'mlpr...	-2.006017	7	-1.604479
8	1.037747	{'mlpregressor__activation': 'identity', 'mlpr...	-2.701039	19	-2.332393
9	0.129904	{'mlpregressor__activation': 'identity', 'mlpr...	-2.002411	4	-1.602666

Y este fue el conjunto de hiperparámetros del elemento 6.

```
MLPRegressor(activation='identity', alpha=0.0001,
             batch_size='auto', beta_1=0.9, beta_2=0.999,
             early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=40, learning_rate='constant',
             learning_rate_init=0.001, max_iter=200,
             momentum=0.9, n_iter_no_change=10,
             nesterovs_momentum=True, power_t=0.5,
             random_state=None, shuffle=True, solver='lbfgs',
             tol=0.0001, validation_fraction=0.1,
             verbose=False, warm_start=False))],
```

Lo mismo se realizó para el modelo de Random Forest Regression. Se probó con el siguiente conjunto de hiperparámetros.

```
parameters = {'randomforestregressor__n_estimators': [10, 50, 150, 200],
              'randomforestregressor__criterion':('mse','mae'),
              'randomforestregressor__max_features':('auto', 'sqrt', 'log2')}
```

Luego, se obtuvo la tabla con los diferentes resutados mostrando que el elemento 3 fue el que mejores resultados obtuvo.

	mean_fit_time	params	mean_test_mean_squared_error	rank_test_mean_squared_error	mean_train_mean_squared_error
0	0.069991	{'randomforestregressor__criterion': 'mse', 'r...	-1.803671	5	-0.307172
1	0.219483	{'randomforestregressor__criterion': 'mse', 'r...	-1.672644	3	-0.210119
2	0.584834	{'randomforestregressor__criterion': 'mse', 'r...	-1.622687	2	-0.206668
3	0.993656	{'randomforestregressor__criterion': 'mse', 'r...	-1.603938	1	-0.200926
4	0.047498	{'randomforestregressor__criterion': 'mse', 'r...	-3.696987	22	-0.658872
5	0.187899	{'randomforestregressor__criterion': 'mse', 'r...	-3.123149	19	-0.418053
6	0.450005	{'randomforestregressor__criterion': 'mse', 'r...	-3.003816	9	-0.385295
7	0.662196	{'randomforestregressor__criterion': 'mse', 'r...	-3.016599	11	-0.385076
8	0.079592	{'randomforestregressor__criterion': 'mse', 'r...	-3.788152	23	-0.658967
9	0.242499	{'randomforestregressor__criterion': 'mse', 'r...	-3.072002	16	-0.424801

Y aquí están los hiperparámetros del índice 3.

```
RandomForestRegressor(bootstrap=True, criterion='mse',
                      max_depth=None, max_features='auto',
                      max_leaf_nodes=None,
                      min_impurity_decrease=0.0,
                      min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0,
                      n_estimators=200, n_jobs=None,
                      oob_score=False, random_state=None,
                      verbose=0, warm_start=False))],
```

3. Encontrar los mejores hiperparámetros para DOS combinaciones de algoritmo no supervisado + algoritmo predictivo

Para este problema de regresión no se encontró una combinación de algoritmos no supervisado + algoritmo predictivo por lo que se decidió utilizar otro algoritmo de regresión Logistic Regression y se realizó el mismo proceso anterior. Este es el conjunto de prueba hiperparámetros para Logistic Regression.

```
parameters = {'logisticregression__solver':('newton-cg', 'sag','saga', 'lbfgs'),
              'logisticregression__penalty':('l2','none')}
```

Obteniendo la siguiente tabla donde muestra que el mejor conjunto de hiperparámetros lo tiene el elemento 5.

	mean_fit_time	params	mean_test_mean_squared_error	rank_test_mean_squared_error	mean_train_mean_squared_error
0	0.257496	{'logisticregression__penalty': 'l2', 'logisti...	-5.797468	4	-1.172326
1	0.348799	{'logisticregression__penalty': 'l2', 'logisti...	-5.797468	4	-1.172607
2	0.583903	{'logisticregression__penalty': 'l2', 'logisti...	-5.939241	7	-1.219592
3	0.201010	{'logisticregression__penalty': 'l2', 'logisti...	-5.797468	4	-1.172326
4	21.790826	{'logisticregression__penalty': 'none', 'logis...	-8.827848	8	-0.019418
5	0.770993	{'logisticregression__penalty': 'none', 'logis...	-5.159494	1	-0.453433
6	0.995498	{'logisticregression__penalty': 'none', 'logis...	-5.739241	3	-0.626149
7	0.449894	{'logisticregression__penalty': 'none', 'logis...	-5.643038	2	-0.040795

Y estos son sus hiperparámetros.

```
LogisticRegression(C=1.0, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1,
                   l1_ratio=None, max_iter=150,
                   multi_class='multinomial', n_jobs=None,
                   penalty='none', random_state=None,
                   solver='sag', tol=0.0001, verbose=0,
                   warm_start=False))],
```

Esta es una tabla con la comparación de los tres modelos donde claramente se visualiza que el Random Forest Regressor fue el que menor eror tuvo, por lo tanto es el modelo que mejor se ajusta a este problema.

mean_fit_time	params	mean_test_mean_squared_error	rank_test_mean_squared_error	mean_train_mean_squared_error
0.993656	{'randomforestregressor__criterion': 'mse', 'r...	-1.603938	1	-0.200926
0.099507	{'mlpregressor__activation': 'identity', 'mlpr...	-2.002331	1	-1.602666
0.770993	{'logisticregression__penalty': 'none', 'logis...	-5.159494	1	-0.453433

4. Realizar las curvas de aprendizaje para cada uno de los casos anteriores

MLPRegressor

```
train_sizes, train_scores, valid_scores = learning_curve(MLPRegressor(activation='identity', alpha=0.0001,
                             batch_size='auto', beta_1=0.9, beta_2=0.999,
                             early_stopping=False, epsilon=1e-08,
                             hidden_layer_sizes=40, learning_rate='constant',
                             learning_rate_init=0.001, max_iter=200,
                             momentum=0.9, n_iter_no_change=10,
                             nesterovs_momentum=True, power_t=0.5,
                             random_state=None, shuffle=True, solver='lbfgs',
                             tol=0.0001, validation_fraction=0.1,
                             verbose=False, warm_start=False), X_mat, y_mat, train_sizes=[50, 80, 110], cv=5)
```

```
print(train_sizes)
print(train_scores)
print(valid_scores)
```

```
[ 50  80 110]
[[0.98841595 0.98171718 0.98171498 0.98171416 0.98171502]
 [0.97018061 0.95077869 0.95269375 0.95269463 0.95266937]
 [0.94826794 0.93941709 0.95202523 0.95200455 0.95200238]]
[[0.70812659 0.70699751 0.74600686 0.5417128  0.54145114]
 [0.82924293 0.89220147 0.86614803 0.72589981 0.65017751]
 [0.85652941 0.88478524 0.85231019 0.75596841 0.75851996]]
```

Random Forest Regressor

```
train_sizes, train_scores, valid_scores = learning_curve(RandomForestRegressor(bootstrap=True, criterion='mse',
                                      max_depth=None, max_features='auto',
                                      max_leaf_nodes=None,
                                      min_impurity_decrease=0.0,
                                      min_impurity_split=None,
                                      min_samples_leaf=1, min_samples_split=2,
                                      min_weight_fraction_leaf=0.0,
                                      n_estimators=200, n_jobs=None,
                                      oob_score=False, random_state=None,
                                      verbose=0, warm_start=False), X_mat, y_mat, train_sizes=[50, 80, 110], cv=5)
```

```
print(train_sizes)
print(train_scores)
print(valid_scores)
```

```
[ 50  80 110]
[[0.98868318 0.97752764 0.97898698 0.98118566 0.98015503]
 [0.99157212 0.98712756 0.98682184 0.98775586 0.986303  ]
 [0.98891296 0.98626827 0.98974896 0.98987057 0.9902073 ]]
[[0.88992503 0.82228005 0.83722499 0.7423948  0.71265937]
 [0.87795891 0.8704512  0.84042227 0.75108436 0.70388978]
 [0.89629269 0.91864468 0.84320694 0.76621178 0.70734057]]
```

Logistic Regression

```
train_sizes, train_scores, valid_scores = learning_curve(LogisticRegression(C=1.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=150,
                                    multi_class='multinomial', n_jobs=None,
                                    penalty='none', random_state=None,
                                    solver='sag', tol=0.0001, verbose=0,
                                    warm_start=False), X_mat, y_mat, train_sizes=[50, 80, 110], cv=5)
```

```
print(train_sizes)
print(train_scores)
print(valid_scores)
```

```
[ 50  80 110]
[[1.         1.         1.         1.         1.        ]
 [0.9875     0.9875     0.9625     0.9625     0.9625    ]
 [0.92727273 0.97272727 0.94545455 0.94545455 0.94545455]]
[[0.15730337 0.19512195 0.18181818 0.14864865 0.10958904]
 [0.23595506 0.19512195 0.18181818 0.16216216 0.23287671]
 [0.25842697 0.20731707 0.11688312 0.21621622 0.21917808]]
```

5. Evaluación diagnóstica

a. Ninguno de los modelos presentados presentó bias/overfitting ya que estos, por un lado tuvieron un buen desempeño tanto en el training como en el test, por lo que no se ve un overfitting, por el lado del bias, en general todos los modelos tuvieron un error de test bajo, por lo que tampoco se presentó este caso.

b. Lo primero sería probar con diferentes hiperparámetros para el Logistic Regression que de los tres modelos fue el que menor desempeño mostró. Lo segundo sería utilizar estrategías de extracción de características para contemplar si los modelos pueden tener una mejora quitando algunas de sus características que estén mandando información redudante y reduzca el desempeño de los modelos. Lo tercero sería buscar otras opciones como el PCA para ver si se obtienen mejores resultados.

c. Se esperaría que tuvieron un error de menos del 5% para pensar en la posibilidad de lanzar un modelo como este. Los retos serían que el modelo en sí sólo se podría lanzar para el conjunto de personas que alimentaron el dataset ya que esté, con personas de otros países, o incluso de diferentes colegios, tendría resultados muy diferentes de manera que el modelo no funconaría.

Repositorio: