Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey.

Escuela de ingeniería y ciencias.

Inteligencia Artificial Avanzada para la Ciencia de Datos

Momento de Retroalimentación Módulo 2

Portafolio Análisis

Análisis y Reporte sobre el Desempeño del Modelo.

Alumno:

Jorge Chávez Badillo A01749448

Profesor:

Iván Amaya

Fecha:

9 de septiembre de 2022

Desempeño del Modelo

Para el desarrollo de este reporte de desempeño decidí utilizar la segunda implementación de modelo de machine learning, en este caso el algoritmo utilizado fueron los árboles de decisión para un dataset para clasificación, los árboles de decisión son un tipo de aprendizaje máquina supervisado donde los datos son separados continuamente de acuerdo a ciertos parámetros, las características generales de los árboles son que están compuestos por nodos y hojas, donde las hojas son las decisiones o las clasificaciones finales mientras que los nodos son la parte donde la información es separada.

Para esta solución se utilizaron diferentes librerías para poder realizar el procesamiento de datos y la implementación del modelo de machine learning.

Librerías Utilizadas:

- graphviz Software de dibujo de diagramas, en este caso utilizada para el árbol de decisión.
- numpy Librería para el cálculo numérico y análisis de datos.
- pandas Librería para la manipulación y tratamiento de datos.
- seaborn Para la visualización de datos.
- sklearn.tree Módulo de sklearn para el entrenamiento de árboles de decisión.
- sklearn.metrics Funciones de pérdida, puntiación y utilidad para medir el rendimiento de los modelos de clasificación.
- matplotlib.pyplot Utilizada para la creación de gráficos en dos dimensiones.
- sklearn.tree.DecisionTreeClassifier Para la modelación y congiguración de parámetros de los árboles de decisión.
- sklearn.preprocessing.StandardScaler Librería para la estandarización de datos.
- mlxtend.plotting.plot_learning_curves Utilizada para observar gráficos de desempeño de los modelos.

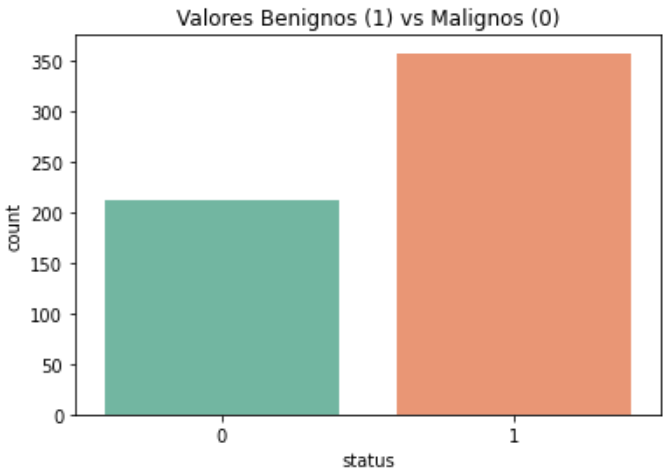
- sklearn.model_selection.train_test_split Para la división de un dataset en dos bloques, destinados al entrenamiento y validación del modelo.

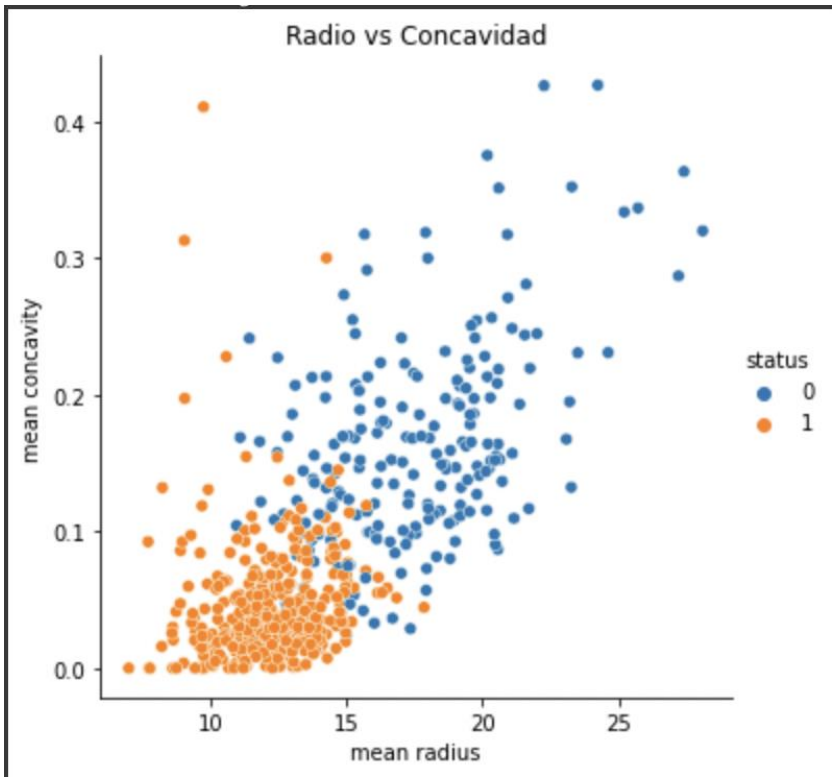
Dataset:

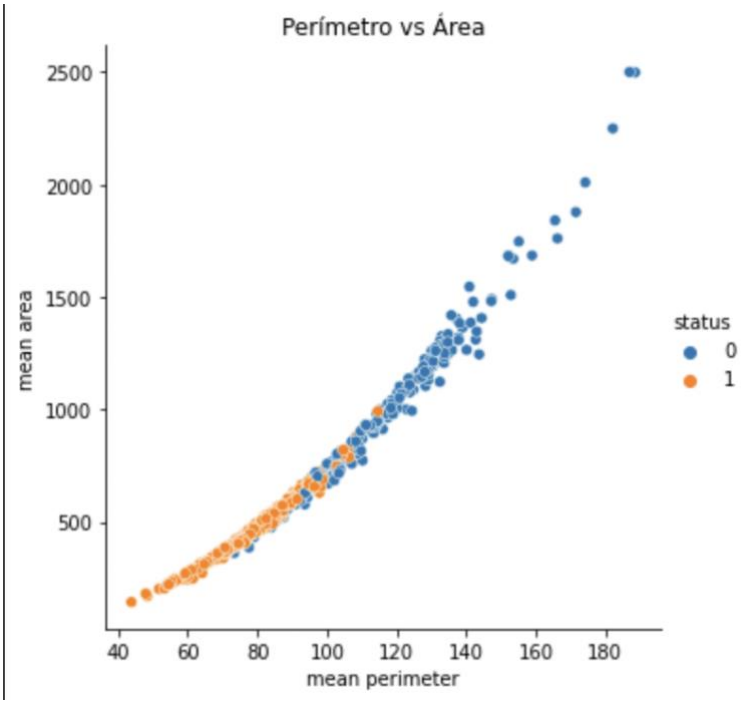
Breast Cancer, obtenido de: dataset y scikit-learn.

El dataset utilizado fue seleccionado ya que este se utiliza con la implementación de modelos de clasificación, en contexto con el dataset, este cuenta con la información necesaria para poder obtener si un tumor es benigno o maligno, este dataset cuenta con diferentes atributos numéricos sobre los tumores, por lo tanto es posible aplicar un modelo de predicción para obtener resultados precisos y eficientes.

Antes de comenzar a implementar el modelo fue necesario hacer un proceso sobre el entendimiento de los datos, como la obtención de los tipos de datos con los que se cuenta en el dataset y la búsqueda de valores nulos, en este rubro se encontró que no había ni un solo dato con valores nulos, lo cual es totalmente bueno ya que nos facilita algunos pasos para la limpieza de datos, sin embargo, es aún necesario hacer un análisis estadístico general ara que de esta manera se logre encontrar diferentes correlaciones entre los datos.




Radio vs Concavidad
mean concavity
0.0
0.1
0.2
0.3
0.4
10
15
20
25
mean radius
status
0
1


Perímetro vs Área
mean area
500
1000
1500
2000
2500
40
60
80
100
120
140
160
180
mean perimeter
status
0
1

Posteriormente lo que se tiene que hacer es dividir o separar los datos en la variable x con las características que consumirá nuestro modelo y y para los valores esperados, todo ello con el fin de entrenar el modelo y posterormente realizar las predicciones, de igual manera, es importante separar la información en datos de entrenamiento, pruebas y validación para que posteriormente sean escalados para un mejor desempeño del modelo y finalmente realizar las configuraciones necesarias a el árbol de decisión.

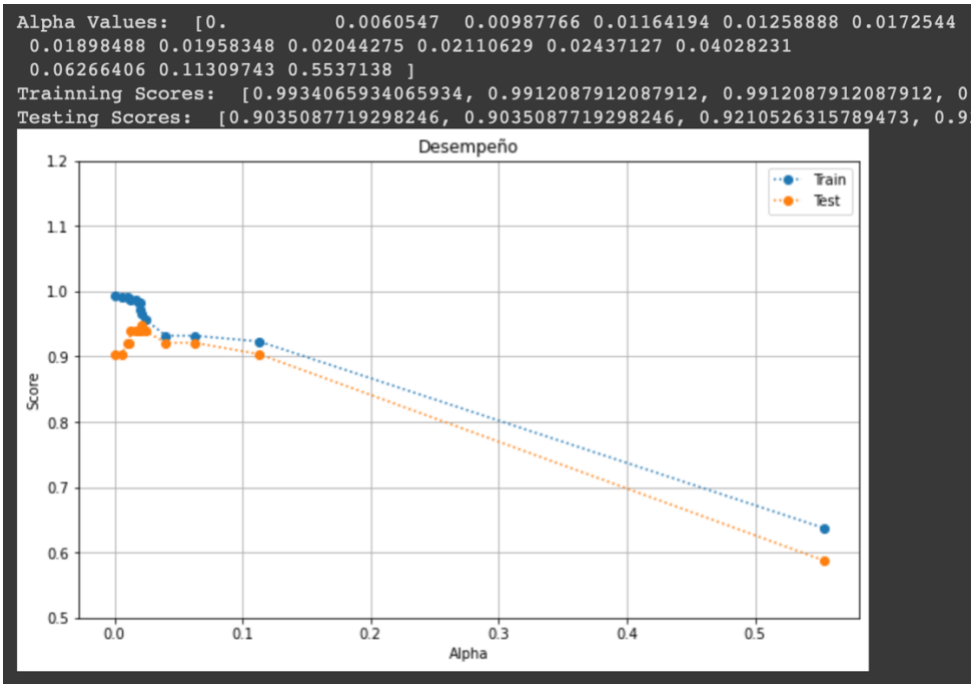
Teniendo ya separados los datos se procede a hacer pruebas de árboles de decisión utilizando diferentes parámetros para comparar y decidir cual resulta con un mayor score.

```python
# Configuración del Modelo de Árbol de Decisión

# Árbol 1
decisionTree1 = DecisionTreeClassifier(random_state = 0,
                                       max_depth = 3)
decisionTree1.fit(x_train, y_train)
print('=' * 10, 'Árbol 1', '=' * 10)
print('Score: ', decisionTree1.score(x_train, y_train))
print()

# Árbol 2
decisionTree2 = DecisionTreeClassifier(criterion = 'entropy',
                                       random_state = 0,
                                       max_depth = 6)
decisionTree2.fit(x_train, y_train)
print('=' * 10, 'Árbol 2', '=' * 10)
print('Score: ', decisionTree2.score(x_train, y_train))
print()

# Árbol 3
decisionTree3 = DecisionTreeClassifier(criterion = 'entropy',
                                       random_state = 0,
                                       max_depth = 6,
                                       min_samples_split = 4,
                                       min_impurity_decrease=0.01)
decisionTree3.fit(x_train, y_train)
print('=' * 10, 'Árbol 3', '=' * 10)
print('Score: ', decisionTree3.score(x_train, y_train))
print()

# Árbol 4
decisionTree4 = DecisionTreeClassifier(criterion = 'entropy',
                                       random_state = 0,
                                       max_depth = 4,
                                       min_impurity_decrease=0.01)
decisionTree4.fit(x_train, y_train)
print('=' * 10, 'Árbol 4', '=' * 10)
print('Score: ', decisionTree4.score(x_train, y_train))
print()

# Árbol 5
decisionTree5 = DecisionTreeClassifier(criterion = 'entropy',
                                       random_state = 0,
                                       max_depth = 6,
                                       min_samples_leaf=8,
                                       min_impurity_decrease=0.02)
decisionTree5.fit(x_train, y_train)
print('=' * 10, 'Árbol 5', '=' * 10)
print('Score: ', decisionTree5.score(x_train, y_train))
print()
```

```
========== Árbol 1 ==========
Score:  0.9714285714285714

========== Árbol 2 ==========
Score:  0.9934065934065934

========== Árbol 3 ==========
Score:  0.9912087912087912

========== Árbol 4 ==========
Score:  0.9868131868131869

========== Árbol 5 ==========
Score:  0.9648351648351648
```

Cómo podemos observar, el árbol de decisión número dos muestra un mejor desempeño, por lo cual ocuparemos la configuración de sus parámetros para la obtención del mejor valor de alpha para finalmente hacer el cálculo de las predicciones utilizando este modelo.

```
Alpha Values:  [0.         0.0060547  0.00987766 0.01164194 0.01258888 0.0172544
 0.01898488 0.01958348 0.02044275 0.02110629 0.02437127 0.04028231
 0.06266406 0.11309743 0.5537138 ]
Trainning Scores:  [0.9934065934065934, 0.9912087912087912, 0.9912087912087912, 0
Testing Scores:  [0.9035087719298246, 0.9035087719298246, 0.9210526315789473, 0.9
```



Gracias a la gráfica anterior podemos obtener el rango de alpha donde se presenta un mayor score, por lo cual se elige un valor dentro de dicho rango para la configuración final del modelo.

```
========== Árbol Final ==========
Score:  0.9912087912087912
```

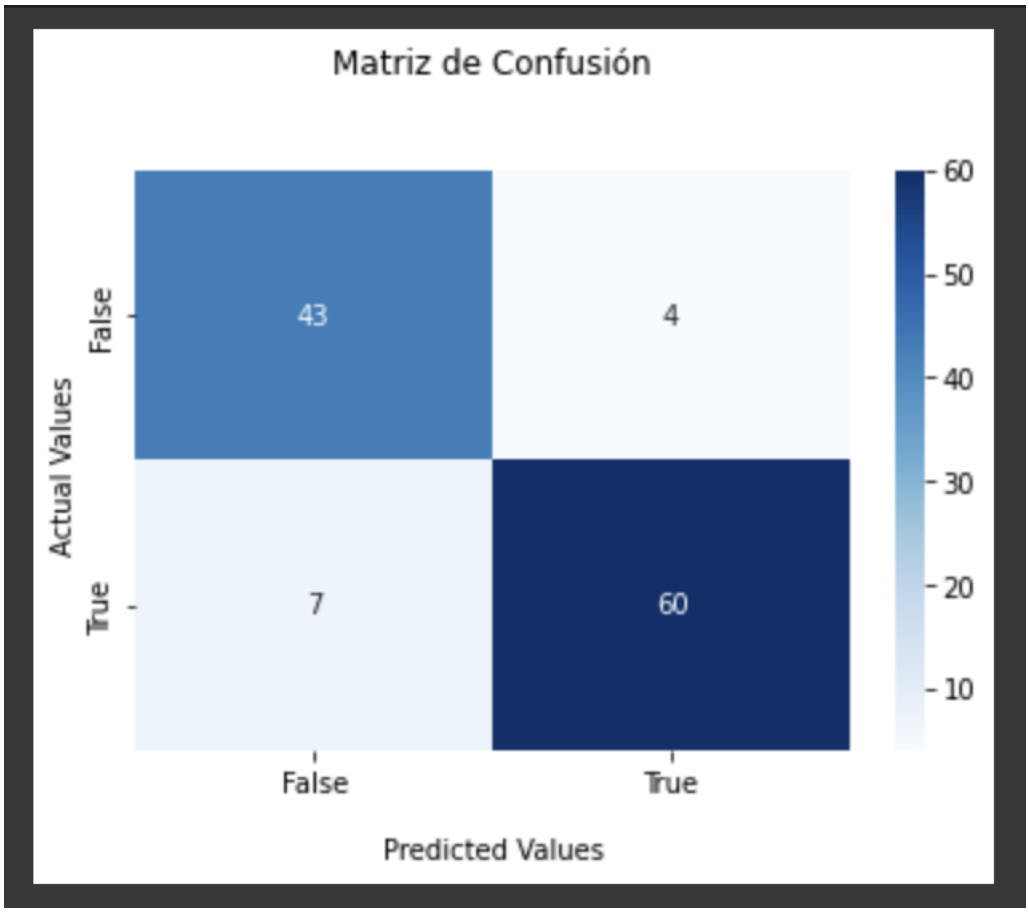
Posteriormente pasamos a las predicciones del modelo, obteniendo los siguientes datos.

```
================================================================================
Valores Reales:
[0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 0 1
 0 0 1]
================================================================================
Predicción:
[0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 1
 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 1
 0 0 0]
```

Las siguientes son algunas de las métricas de precisión y desempeño que tuvo el modelo comparado con los valores reales de y.

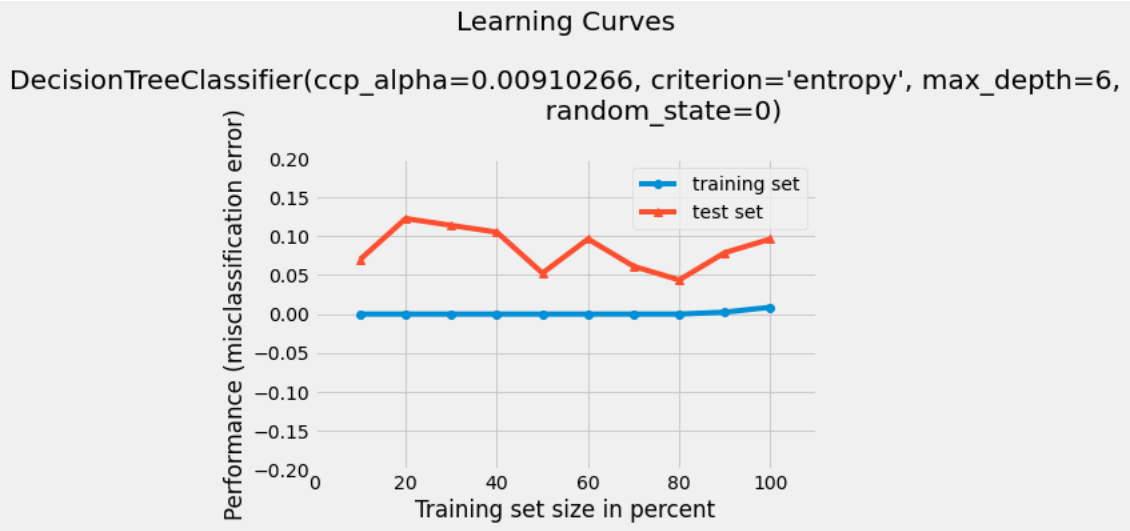
```
Accuracy Score Test:  0.9035087719298246
Precision Score Test:  0.9375
Recall Score Test:  0.8955223880597015
```

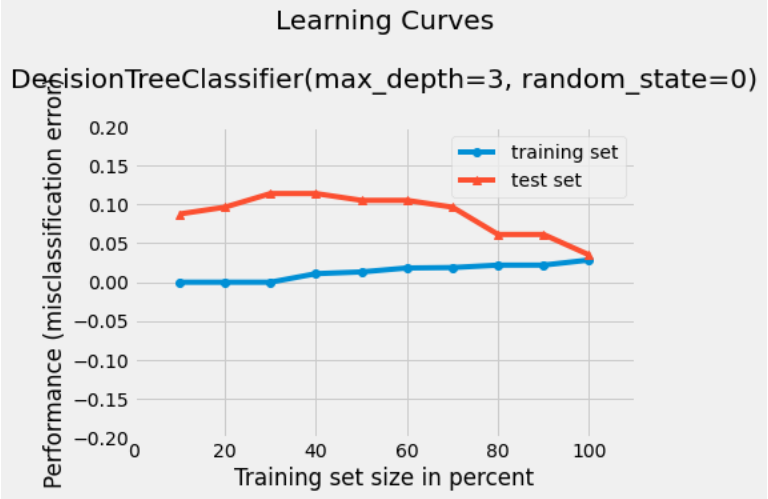
	precision	recall	f1-score	support
Maligno	0.86	0.91	0.89	47
Benigno	0.94	0.90	0.92	67
accuracy			0.90	114
macro avg	0.90	0.91	0.90	114
weighted avg	0.91	0.90	0.90	114



Finalmente hacemos una comparación entre uno de los modelos con menor score versus el árbol de decisión propuesto como el que mejor desempeño mostraba y como podemos observar, en el primer gráfico tenemos que se tiene un fit deseado, sin embargo, en el segundo gráfico podemos observar que con el modelo no optimizado tenemos cierto grado de underfitting lo cual fue ajustado por medio de los parámetros del árbol obteniendo así mejores predicciones, todo ello con respecto a los datos para pruebas (testing).

Como conclusión, podemos decir que si hubo un cambio positivo en el desempeño del modelo gracias a la diferente combinación de parámetros, lo cual nos da mejores valores en las métricas y un mejor comportamiento en el performance de las predicciones.


Learning Curves
DecisionTreeClassifier(ccp_alpha=0.00910266, criterion='entropy', max_depth=6, random_state=0)
Performance (misclassification error)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
0
20
40
60
80
100
Training set size in percent
training set
test set


Learning Curves
DecisionTreeClassifier(max_depth=3, random_state=0)
Performance (misclassification error)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
0
20
40
60
80
100
Training set size in percent
training set
test set