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:

18 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.

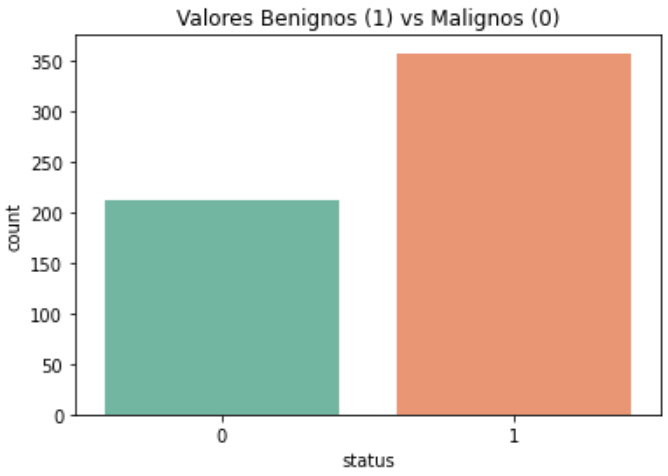
- sklearn.model_selection.train_test_split Para la división de un dataset en dos bloques, destinados al entrenamiento y validación del modelo.

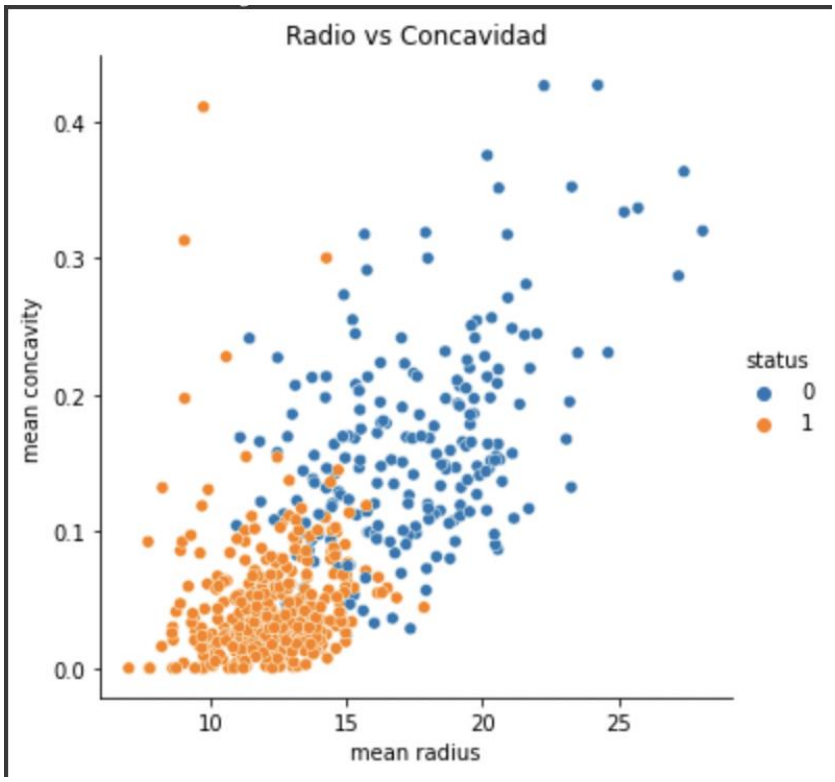
Dataset:

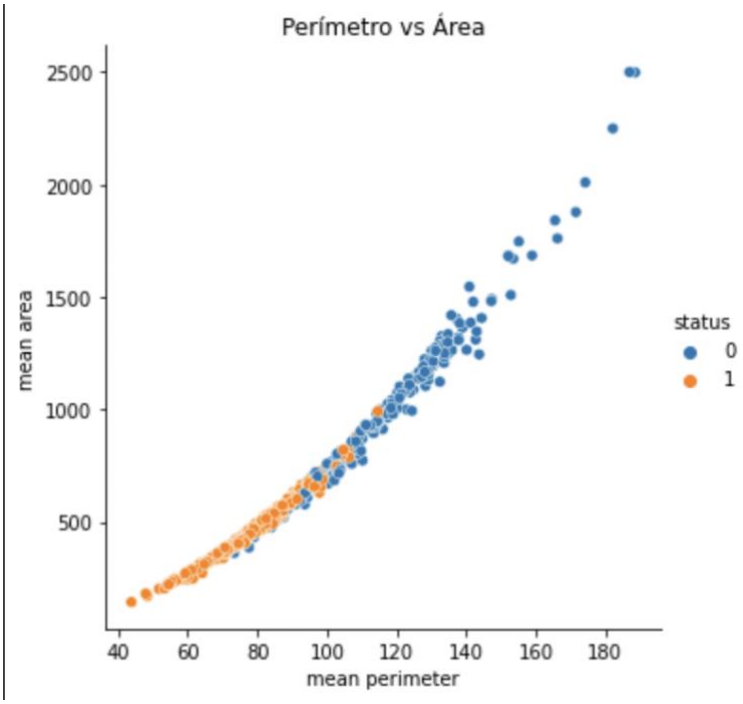
Breast Cancer, obtenido de: dataset y scikit-learn.

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.

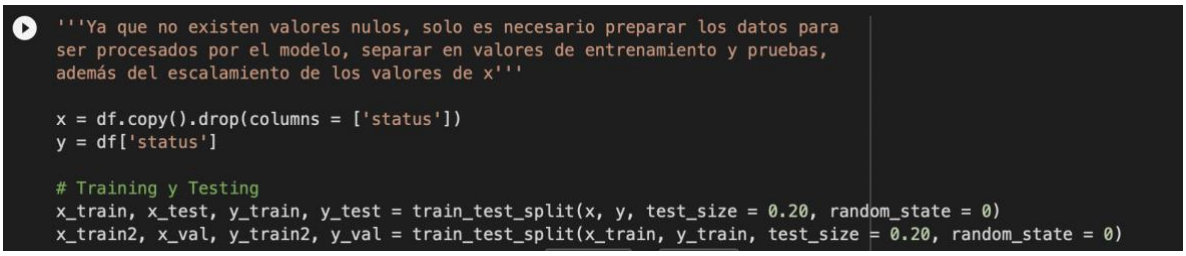
Antes de comenzar a implementar el modelo fue necesario hacer un proceso sobre el entendimiento de los datos, como la obtención de los tipos de datos con los que se cuenta en el dataset y la búsqueda de valores nulos, en este rubro se encontró que no había ni un solo dato con valores nulos, lo cual es 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 (En la imagen siguiente se muestra como se realizó la separación del dataset) para que posteriormente sean escalados para un mejor desempeño del modelo y finalmente realizar las configuraciones necesarias a el árbol de decisión.



```
'''Ya que no existen valores nulos, solo es necesario preparar los datos para
ser procesados por el modelo, separar en valores de entrenamiento y pruebas,
además del escalamiento de los valores de x'''

x = df.copy().drop(columns = ['status'])
y = df['status']

# Training y Testing
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
x_train2, x_val, y_train2, y_val = train_test_split(x_train, y_train, test_size = 0.20, random_state = 0)
```

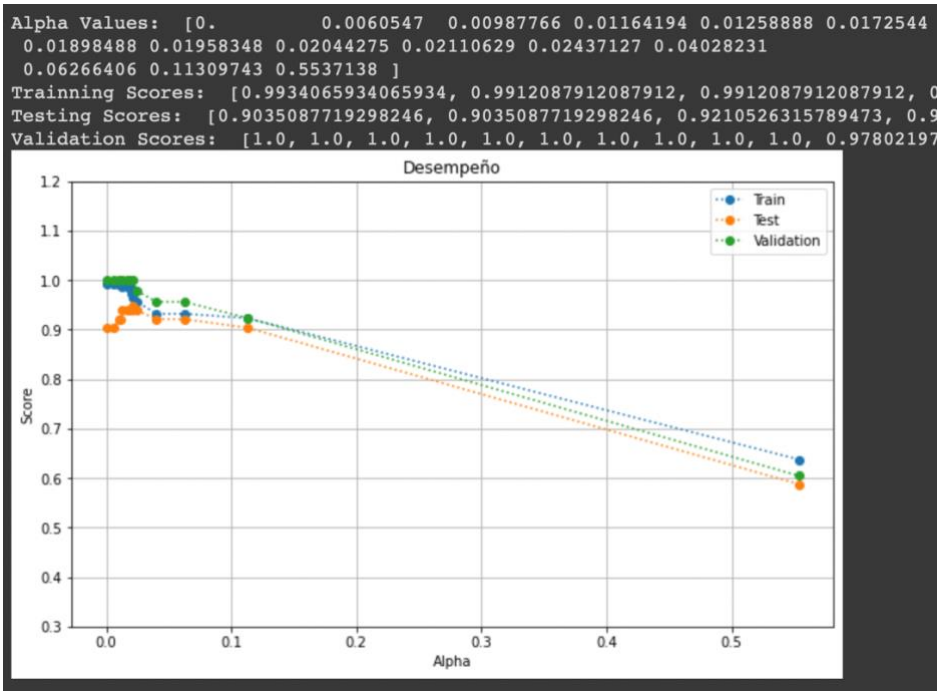
Teniendo ya separados los datos se procede a hacer pruebas de árboles de decisión utilizando diferentes parámetros para comparar y decidir cual resulta con un mayor score.

```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
Validation Scores:  [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.97802197
```



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.

```
********** Test **********
================================================================================
Valores De Entrada:
[[-0.20175604  0.3290786  -0.13086754 ...  1.3893291   1.08203284
   1.54029664]
 [-0.25555773  1.46763319 -0.31780437 ... -0.83369364 -0.73131577
  -0.87732522]
 [-0.02619262 -0.8407682  -0.09175081 ... -0.49483785 -1.22080864
  -0.92115937]
 ...
 [ 1.71811488  0.09318356  1.7286186  ...  1.57630515  0.20317063
  -0.15406178]
 [ 1.18859296  0.34352115  1.19333694 ...  0.56019755  0.26991966
  -0.27320074]
 [ 0.26263752 -0.58080224  0.28459338 ... -0.19383705 -1.15564888
   0.11231497]]
================================================================================
Valores Reales:
[0 1 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 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 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]
```

mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	real values	prediction	status
0.864118	0.733639	0.856697	1.120328	1.553585	...	-0.062808	1.103531	0.874443	1.219091	1.389329	1.082033	1.540297	0	0	Acertado
-1.016540	-0.769012	-0.726495	-0.695677	-1.002451	...	-0.423883	-0.157482	-0.951752	-0.644332	-0.833694	-0.731316	-0.877325	1	1	Acertado
-0.941988	-0.857154	-0.575023	-0.805295	-0.999514	...	-0.328279	-1.285756	-0.981828	-1.027447	-0.494838	-1.220809	-0.921159	1	1	Acertado
-1.013566	-0.807228	-0.906497	-0.637214	-0.999514	...	-0.104976	-1.681759	-0.229905	-0.556576	-0.599974	-0.426177	-0.378290	1	1	Acertado
-1.301861	-0.796034	-0.504734	-1.251074	-0.592839	...	-0.572410	-1.568489	-1.344779	-1.099005	-0.985727	-1.457609	-1.225189	1	1	Acertado
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.223683	1.038596	0.448190	-1.832050	-0.588435	...	0.238685	-0.361014	-0.474443	0.064230	-0.069697	-2.067886	-0.860466	0	1	No Acertado
0.166183	-0.626442	-0.486059	1.134943	0.522949	...	-0.438907	0.665495	-0.352828	-0.627784	-0.521045	0.265152	0.116811	1	1	Acertado
0.903774	1.618914	2.002591	-0.312014	-0.313892	...	1.020588	1.506170	0.276827	1.320386	1.576305	0.203171	-0.154062	0	0	Acertado
0.425926	0.893194	1.188431	0.184921	-0.478324	...	1.044489	0.740713	0.011365	0.543625	0.560198	0.269920	-0.273201	0	0	Acertado
0.655927	-0.132206	0.330177	-0.516634	0.884112	...	-0.041468	-0.838871	-0.224020	-0.410652	-0.193837	-1.155649	0.112315	1	0	No Acertado

```
********** Validation **********
================================================================================
Valores De Entrada:
[[ 0.03893575  0.72143462  0.18947794 ...  0.98165998 -0.09561074
   1.66955118]
 [ 1.18859296 -0.13789728  1.10275081 ...  0.65368558  0.47970232
  -0.79302878]
 [-1.25881771 -0.15956111 -1.2487827  ... -0.74863486  0.42884592
   0.09826556]
 ...
 [ 0.02194574  1.12101192  0.04330487 ...  0.9617363   2.89220292
   1.07385635]
 [-0.38015112  0.5433098  -0.40344943 ... -1.02250881 -1.15247035
  -0.83068119]
 [ 1.1234646   0.64681476  1.05334019 ... -0.34694285 -0.85051046
  -1.00657976]]
================================================================================
Valores Reales:
[0 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1
 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0
 0 1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0]
================================================================================
Predicción:
[0 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1
 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0
 0 1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0]
```

mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	real values	prediction	status
1.932832	1.621487	0.981454	0.510121	1.505136	...	0.133692	1.395555	2.437790	2.129741	0.981660	-0.095611	1.669551	0	0	Acertado
-0.369562	-0.074560	0.245104	-0.688369	-1.235885	...	0.902790	-0.400835	-0.048135	0.432803	0.653686	0.479702	-0.793029	0	0	Acertado
-0.807754	-0.746366	-0.870185	1.566107	0.214639	...	-0.846759	-0.400835	-0.638559	-0.700997	-0.748635	0.428846	0.098266	1	1	Acertado
-0.716547	-0.989945	-1.005135	-1.477618	-0.614861	...	-0.298062	-1.117621	-0.946521	-1.164997	-1.246420	-1.661034	-0.864962	1	1	Acertado
1.042568	0.616546	0.747762	0.689164	0.364390	...	0.648416	0.537181	2.148136	1.303336	1.102735	2.170678	2.237147	0	0	Acertado
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
-1.118851	-0.983769	-0.897938	0.992440	0.066357	...	-0.897975	0.630098	-1.034136	-1.138620	-0.963351	-0.100379	-0.097864	1	1	Acertado
0.810584	0.966538	1.191802	0.305501	-0.550263	...	1.274962	0.855753	0.804481	0.822938	1.697380	1.089979	-0.315911	0	0	Acertado
0.540927	0.309016	0.413175	1.555146	0.244002	...	-0.119317	1.037162	1.007174	0.536103	0.961736	2.892203	1.073856	0	0	Acertado
-0.739745	-0.725263	-0.520037	-0.421632	0.029653	...	-0.515048	-1.706979	-1.059767	-1.081303	-1.022509	-1.152470	-0.830681	1	1	Acertado
-0.335062	0.293575	0.235766	-0.143933	-1.360677	...	0.592078	-1.957411	-0.363289	-0.070161	-0.346943	-0.850510	-1.006580	0	0	Acertado

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

```
# Classification Report Test
print(metrics.classification_report(y_test, prediccion_test,
                                    target_names = ['Maligno', 'Benigno']))
```

```
              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
```

```
[210] # Classification Report Validation
print(metrics.classification_report(y_val, prediccion_val,
                                    target_names = ['Maligno', 'Benigno']))
```

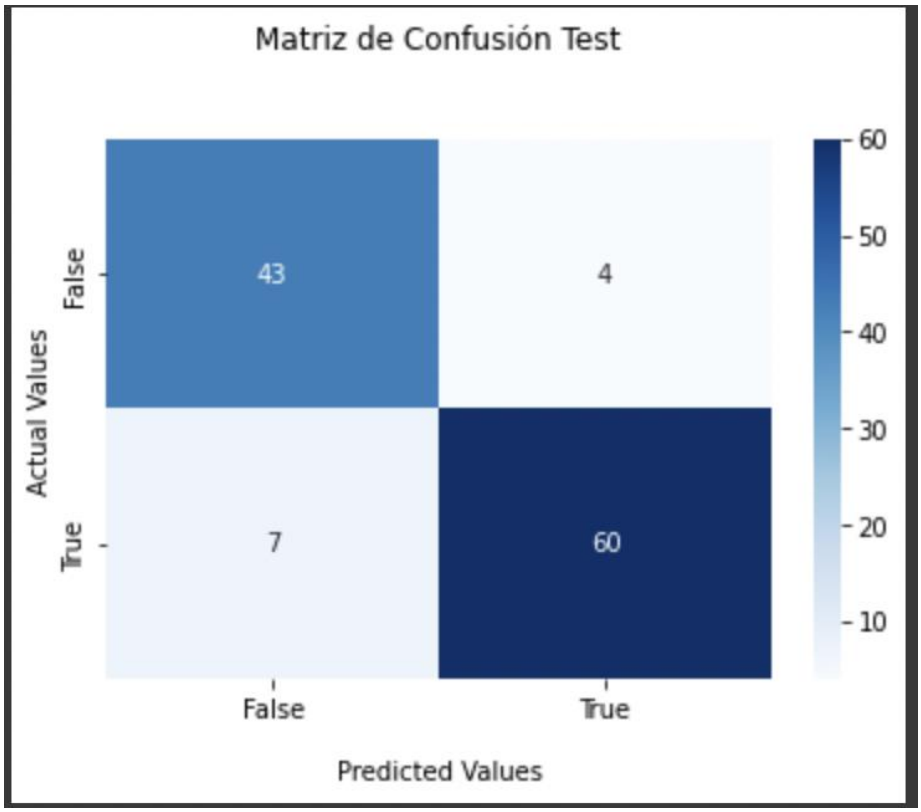
```
              precision    recall  f1-score   support

      Maligno       1.00      1.00      1.00        36
      Benigno       1.00      1.00      1.00        55

     accuracy                           1.00        91
    macro avg       1.00      1.00      1.00        91
 weighted avg       1.00      1.00      1.00        91
```

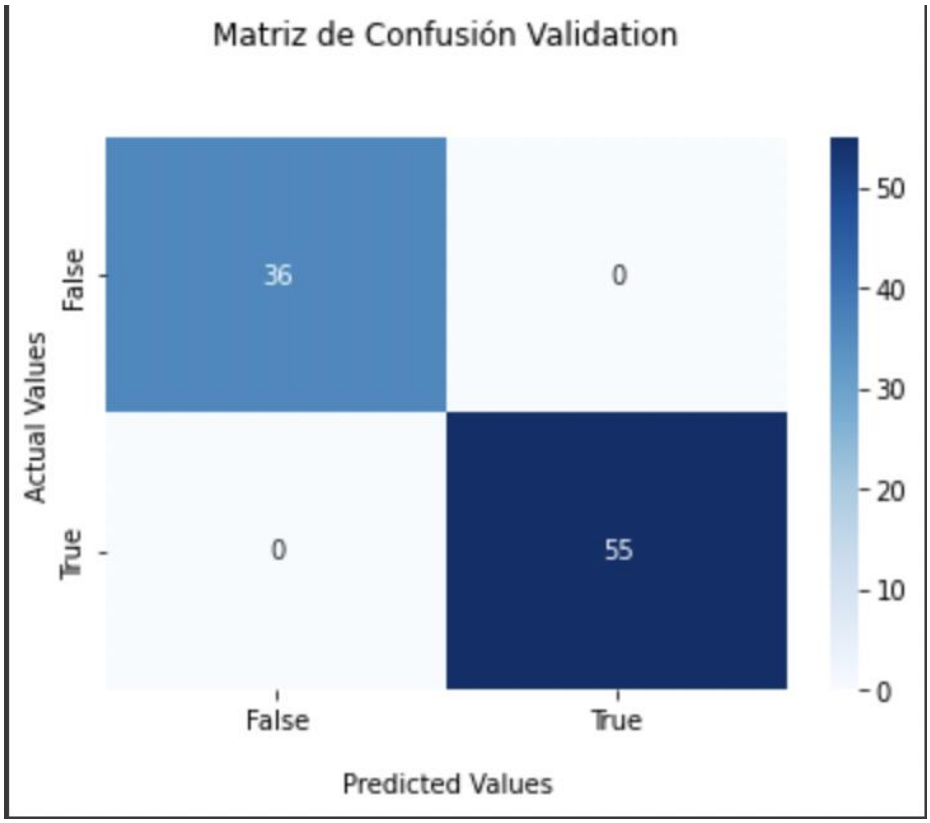
```
# Test
print('Accuracy Score Test: ', metrics.accuracy_score(y_test, prediccion_test))
print('Precision Score Test: ', metrics.precision_score(y_test, prediccion_test))
print('Recall Score Test: ', metrics.recall_score(y_test, prediccion_test))
```

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

```
[223] # Validation
print('Accuracy Score Validation: ', metrics.accuracy_score(y_val, prediccion_val))
print('Precision Score Validation: ', metrics.precision_score(y_val, prediccion_val))
print('Recall Score Validation: ', metrics.recall_score(y_val, prediccion_val))
```

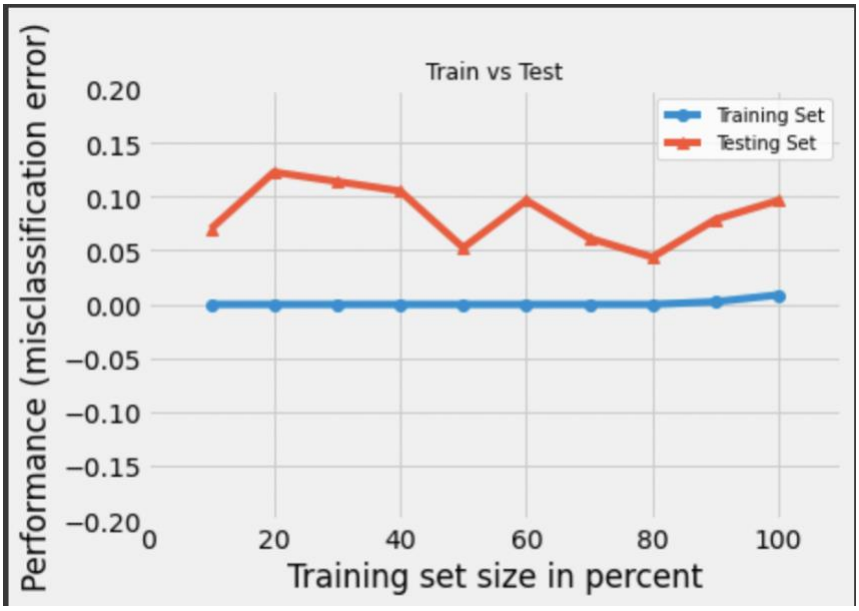
```
Accuracy Score Validation:  1.0
Precision Score Validation:  1.0
Recall Score Validation:  1.0
```

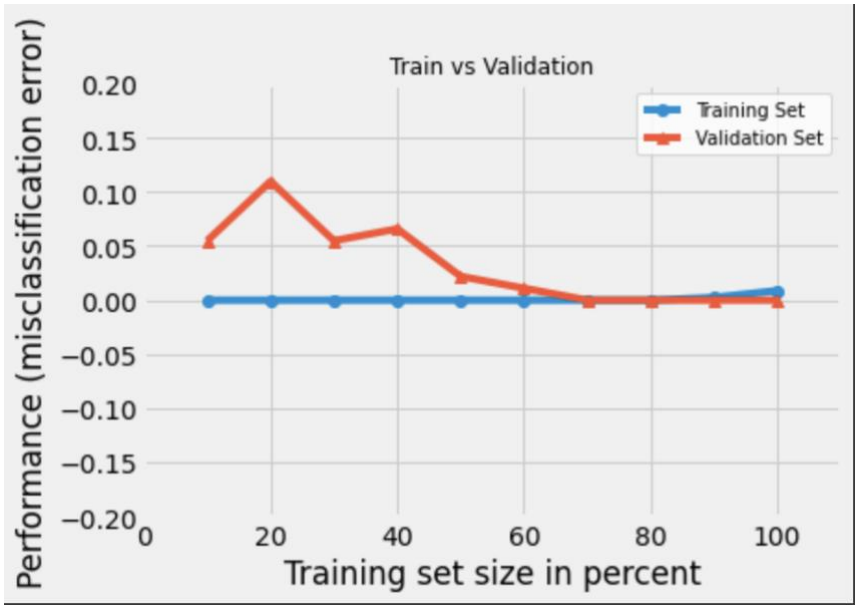

Matriz de Confusión Test
43
4
7
60
False
True
Actual Values
False
True
Predicted Values
60
50
40
30
20
10


Matriz de Confusión Validation
36
0
0
55
False
True
Actual Values
False
True
Predicted Values
50
40
30
20
10
0

Finalmente se hace una comparación de desempeño, es decir train vs test y test vs validation, donde es posible observar que en ambas gráficas se tiene un fit deseado ya que los valores de error son bastante bajos, lo que indica que el modelo está bastante ajustado y se están realizando buenas predicciones, de acuerdo con estas gráficas, podemos observar que se cuenta con un fit deseado ya que se tiene un error aceptable tanto en training, testing y validation; de igual forma podemos observar que tanto los valores del sesgo y la varianza son bastante bajos, lo cual nos confirma que se está teniendo un fit deseado y existe un buen balance en el modelo.

Como conclusión, podemos decir que si hubo un cambio positivo en el desempeño del modelo gracias a la diferente combinación de parámetros, es decir que se buscó siempre un ajuste en los parámetros dados, para así obtener mejores valores en las métricas y un mejor comportamiento en el performance de las predicciones y el desempeño del modelo.






```
Mean Squared Error Test:  0.096491228070175 43
Mean Absolute Error Test:  0.096491228070175 43
Mean Squared Error Validation:  0.0
Mean Absolute Error Validation:  0.0
```