Reporte de Análisis de Modelo de Machine Learning

Modelo Seleccionado: Modelo con aplicación de un Framework

Inteligencia Artificial Avanzada para la Ciencia de Datos (Grupo 101)

Módulo 2: Machine Learning

Emilia Victoria Jácome Iñiguez - A00828347

Ingeniería en Transformación Digital de Negocios (ITD)

Profesor Docente

Iván Mauricio Amaya Contreras

Viernes 16 de septiembre de 2022

Tecnológico de Monterrey, Campus Monterrey

Dataset utilizado

Name: Brain stroke prediction dataset full_filled_stroke_data.csv

Source: Kaggle
Original Source: Data files © Original Authors

Length: 4981

Modelo de ML Aplicado

Clasificador de Árbol de Decisión/ Decisión Tree Classifier

Separación y Evaluación del Modelo

Para la separación de las variables entre un conjunto de entrenamiento, un conjunto de prueba y un conjunto de validación, se procedió a utilizar la función train_test_split de la librería de sklearn.model_selection para separar los datos.

Separación de las variables en conjunto de entrenamiento y prueba

```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = df.stroke, train_size = 0.6, test_size = 0.4, random_state=6)

X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.25, random_state=1) # 0.4 x 0.25 = 0.1
```

Figura No.1 Función train_test_split para separar los sets de datos

La proporción utilizada para separar cada subconjunto de datos, fue de 60% – 30% – 10% para entrenamiento – prueba – validación respectivamente. Para lograr 3 grupos en lugar de uno, se utilizó la función de train_test_split dos veces, la primera en función del dataset completo y la segunda con respecto a los datos de prueba.

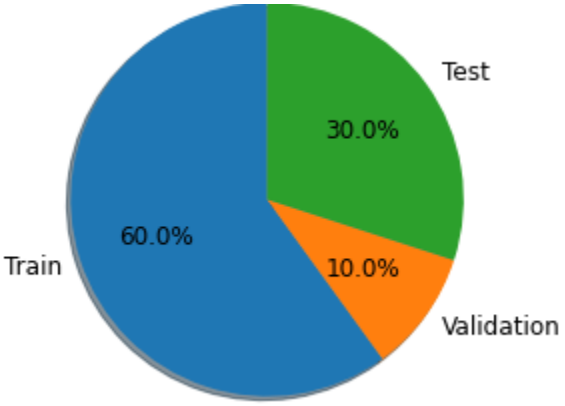


Figura No.2 Proporciones de partición del dataset

Por lo que se tienen 2988 datos para el entrenamiento, 1494 datos para la prueba, y 499 datos para la validación.

```
5]  1 X_train.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 3486 entries, 4212 to 1538
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   gender_tok          3486 non-null   int64
 1   age                 3486 non-null   float64
 2   hypertension        3486 non-null   int64
 3   heart_disease       3486 non-null   int64
 4   ever_married_tok    3486 non-null   int64
 5   work_type_tok       3486 non-null   int64
 6   residence_type_tok  3486 non-null   int64
 7   avg_glucose_level   3486 non-null   float64
 8   bmi                 3486 non-null   float64
 9   smoking_status_tok  3486 non-null   int64
dtypes: float64(3), int64(7)
memory usage: 299.6 KB
```

Figura No.3 Información sobre conjunto de entrenamiento

```
1 X_val.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 374 entries, 139 to 2488
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   gender_tok          374 non-null    int64
 1   age                 374 non-null    float64
 2   hypertension        374 non-null    int64
 3   heart_disease       374 non-null    int64
 4   ever_married_tok    374 non-null    int64
 5   work_type_tok       374 non-null    int64
 6   residence_type_tok  374 non-null    int64
 7   avg_glucose_level   374 non-null    float64
 8   bmi                 374 non-null    float64
 9   smoking_status_tok  374 non-null    int64
dtypes: float64(3), int64(7)
memory usage: 32.1 KB
```

Figura No.4 Información sobre conjunto de validación

```
1 X_test.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1121 entries, 2319 to 3986
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   gender_tok          1121 non-null   int64
 1   age                 1121 non-null   float64
 2   hypertension        1121 non-null   int64
 3   heart_disease       1121 non-null   int64
 4   ever_married_tok    1121 non-null   int64
 5   work_type_tok       1121 non-null   int64
 6   residence_type_tok  1121 non-null   int64
 7   avg_glucose_level   1121 non-null   float64
 8   bmi                 1121 non-null   float64
 9   smoking_status_tok  1121 non-null   int64
dtypes: float64(3), int64(7)
memory usage: 96.3 KB
```

Figura No.5 Información sobre conjunto de prueba

Análisis de Sesgo

El grado de sesgo en este modelo es de **nivel bajo** ya que las **curvas de aprendizaje** para el set de prueba oscila entre 8% y 13% en su error de clasificación lo cuál es muy bajo. Si bien el set de entrenamiento tiene un error de clasificación casi perfecto, esto se debe a que el modelo seleccionado (Árboles de Decisión) tiende a optimizar el entrenamiento y a producir overfitting para minimizar el margen de error del entrenamiento.

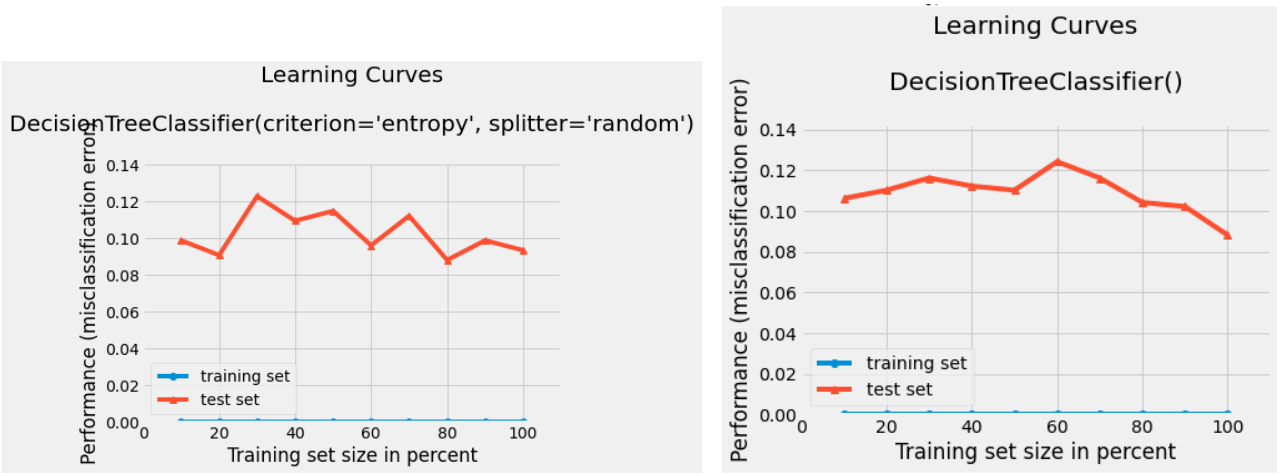


Figura No.6 Curva de aprendizaje basada en el error de clasificación del árbol 1 y árbol 2 respectivamente

Por otra parte, también se graficaron las curvas de aprendizaje en base al Cross-validation score donde se puede ver que conforme se va aumentando los ejemplos de entrenamiento, aumenta a su vez el score llegando a una precisión arriba del 90%, lo cuál indica un bajo nivel de sesgo.

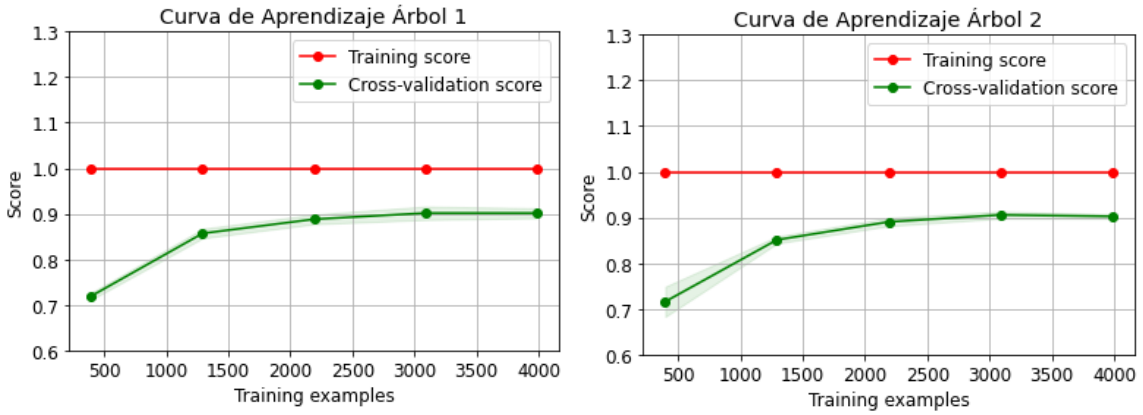


Figura No.7 Curvas de aprendizaje basada en el Cross-validation score del árbol 1 y árbol 2 respectivamente

En adición, se procedió a calcular el error cuadrado promedio de las predicciones para poder corroborar la calidad y el sesgo del modelo.

MSE

```
[52]  1 from sklearn.metrics import mean_squared_error
      2 mean_squared_error(y_test,  y_pred_myTree_1)
```

```
0.09571619812583668
```

MSE

```
[59]  1 from sklearn.metrics import mean_squared_error
      2 mean_squared_error(y_test,  y_pred_myTree_2)
```

```
0.08500669344042838
```

Figura No.8 Cálculo del MSE para cada árbol

Subsecuentemente, los valores de MSE muestran un nivel de error mínimo, menor al 10%, lo que indica que ambos modelos tienen un nivel de sesgo no significativo.

Aparte, hay que considerar que los datos con los que se cuenta contienen mayor cantidad de pacientes que no han tenido un infarto cerebral que pacientes que si han presentado tal y cómo se puede observar en la siguiente imagen.

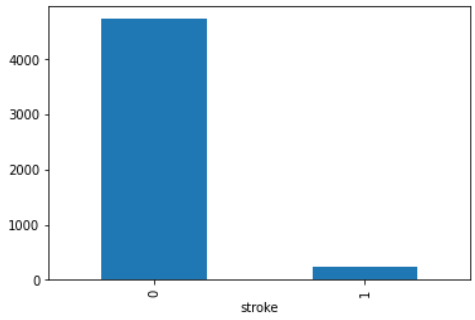


Figura No.9 Conteo de datos por columna de stroke

Este **sesgo estadístico** en los datos afecta directamente al entrenamiento y prueba del modelo ya que tiene más información sobre personas que no se han infartado y muy poca información sobre personas que si, por ende, tiende a detectar con un alto grado de precisión (mayor al 90%) si la persona no se infartó tal y como se muestra en la siguiente imagen.

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1423
1	0.16	0.18	0.17	71
accuracy			0.92	1494
macro avg	0.56	0.57	0.56	1494
weighted avg	0.92	0.92	0.92	1494

Figura No.10 Reporte de Clasificación del Árbol 1

No obstante, el algoritmo tiene un nivel de precisión bajo (menor al 20%) para acertar a una persona que en efecto se infartaron ya que el set de datos de entrenamiento (y el general) contiene muy pocos casos de pacientes infartados. Por lo tanto, se puede decir que el algoritmo contiene un sesgo al etiquetar a un paciente como no infartado cuando en realidad si le ocurrió un infarto y esto repercute por completo la calidad del modelo.

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1423
1	0.16	0.18	0.17	71
accuracy			0.92	1494
macro avg	0.56	0.57	0.56	1494
weighted avg	0.92	0.92	0.92	1494

Figura No.11 Reporte de Clasificación del Árbol 1

De igual manera, en la etapa de prueba del modelo, se puede apreciar el efecto de este sesgo a través de la matriz de confusión:

```
[[1346   77]
 [  56   15]]
```

Figura No.12 Matriz de Confusión del Árbol 1

Se puede ver claramente que los valores de la diagonal principal están muy desbalanceados ya que hay una diferencia desproporcional entre ellos y que la cantidad de falsos negativos es casi 90 veces más grande que el valor de verdaderos positivos.

De igual manera, se justifica la existencia de este sesgo por medio de la similitud de resultados obtenidos entre los dos modelos de árboles de decisión propuestos. He aquí los resultados:

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1423
1	0.16	0.18	0.17	71
accuracy			0.92	1494
macro avg	0.56	0.57	0.56	1494
weighted avg	0.92	0.92	0.92	1494

	precision	recall	f1-score	support
0	0.96	0.95	0.95	1423
1	0.16	0.21	0.18	71
accuracy			0.91	1494
macro avg	0.56	0.58	0.57	1494
weighted avg	0.92	0.91	0.92	1494

Figura No.13 Comparación de Reportes de Clasificación entre Árbol 1 y Árbol 2 respectivamente

Dado que no varía mucho las métricas de desempeño entre el primer modelo y el segundo, se puede concluir que se debe a la limitada data de pacientes que si tuvieron un infarto a pesar de que cada modelo tiene un criterio diferente para definir el valor/peso de sus hojas.

Análisis de Varianza

Al analizar la varianza del modelo, se realizó una prueba para verificar **el cambio en la estimación del modelo** al cambiar la proporción de datos de entrenamiento con la finalidad de determinar el nivel de variabilidad de la función de costo si se cambiara **al producir pequeños cambios en el set de entrenamiento**.

Por lo que se procedió a comparar los valores de las métricas de desempeño para diferentes proporciones del set de entrenamiento:

	Name	Score_a	Score_p	Score_r	Score_f1
0	Tree1	0.895571	0.230952	0.223333	0.191667
1	Tree2	0.897184	0.203333	0.198333	0.166667

Figura No. 14 Métricas de evaluación K-Folds para una proporción de 50% datos de entrenamiento.

	Name	Score_a	Score_p	Score_r	Score_f1
0	Tree1	0.907918	0.195000	0.34	0.233333
1	Tree2	0.913959	0.235952	0.41	0.289062

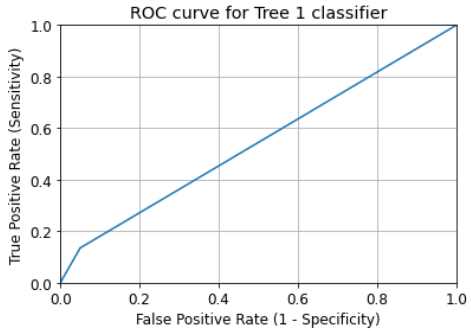
Figura No. 15 Métricas de evaluación K-Folds para una proporción de 60% datos de entrenamiento.

	Name	Score_a	Score_p	Score_r	Score_f1
0	Tree1	0.914296	0.30	0.206667	0.213810
1	Tree2	0.898293	0.19	0.153333	0.130476

Figura No. 16 Métricas de evaluación K-Folds para una proporción de 70% datos de entrenamiento.

Al concluir la prueba, se llegó a la conclusión de que la variación en la calidad de las estimaciones **es mínima** en cuanto a la métrica de precisión con un margen de variación de +-0.01, tomando como referencia el ratio de separación del dataset original (60%-30%-10), lo cual **no es significativo** para marcar una diferencia entre la calidad del modelo o no. Si bien existe una variación más significativa en la especificidad, esta se debe al sesgo de la data anteriormente explicado donde predomina la cantidad de resultados con valor 0 que de 1. Por lo que, aún así la varianza no es significativa para este caso. Por consiguiente, se puede decir que el nivel de **varianza del modelo es bajo**.

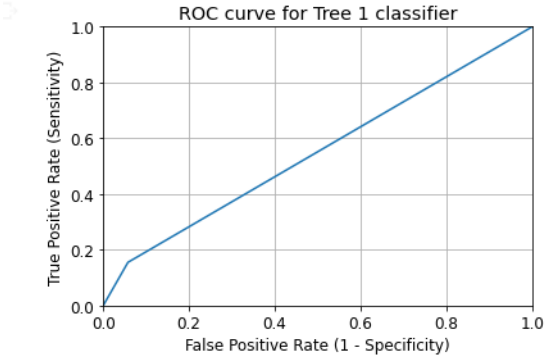
Consecuentemente, esto también puede ser corroborado por las **curvas ROC,** las cuales indican el nivel de desempeño general en base a la tasa de verdaderos positivos frente a la tasa de falsos positivos:




```
1 # IMPORTANT: first argument is true values, second
2 print(metrics.roc_auc_score(y_test, y_pred_prob))
```

```
0.5424016774984052
```

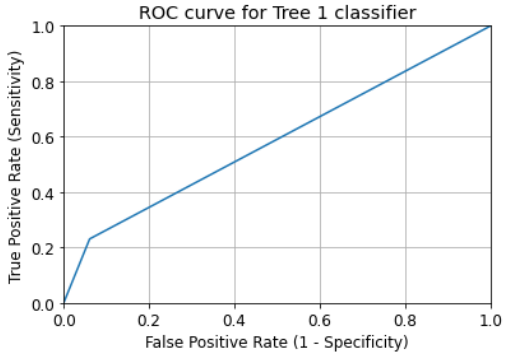
Figura No. 17 Curva ROC y valor AUC para para una proporción de 50% datos de entrenamiento




```
00]  1 # IMPORTANT: first argument is true values, second
     2 print(metrics.roc_auc_score(y_test, y_pred_prob))
```

```
0.5486524204962735
```

Figura No. 18 Curva ROC y valor AUC para para una proporción de 60% datos de entrenamiento




```
[447]  1 # IMPORTANT: first argument is true values, second
       2 print(metrics.roc_auc_score(y_test, y_pred_prob))
```

```
0.5849823702957473
```

Figura No. 19 Curva ROC y valor AUC para para una proporción de 70% datos de entrenamiento

Dichas curvas junto con el valor de su respectiva área **muestran poca variación** con respecto a la separación del dataset original con un margen de +-0.04. Claramente, el valor de AUC tiene una tendencia a aumentar con respecto a la cantidad de datos para el entrenamiento, no obstante, su incremento no es significativo. En conclusión, el hecho de que el modelo tenga poco nivel de variación implica que es un modelo flexible y cuasi óptimo.

Por otro lado, se procedió a calcular el **coeficiente de ajuste** para cada modelo, siendo este el valor de $\mathbf{R^2}$ que mide el nivel de ajuste del modelo a los datos reales.

R2 Score

```
1 from sklearn.metrics import r2_score
2 r2_score(y_test,  y_pred_myTree_1)
```

```
-1.1145764255243336
```

R2 Score

```
[ ]  1 from sklearn.metrics import r2_score
     2 r2_score(y_test,  y_pred_myTree_2)
```

```
-0.8779804618293032
```

Figura No. 20 Cálculo de coeficiente de R^2 para ambos modelos de árboles

A partir de este valor, se puede inferir que existe un alto nivel de varianza en la data predicha ya que el valor de ajuste es negativo para ambos modelos, esto da indicios que el ajuste no es apropiado. Por lo que se puede diagnosticar que son modelos muy precisos pero inconsistentes.

Nivel de Ajuste

En cuanto al nivel de ajuste del modelo, se afirma que existe un caso de **underfitting** donde el modelo es incapaz de llegar a los valores reales debido a que no tiene la suficiente complejidad en su configuración para ajustarse a todos los datos disponibles y no tiene un buen nivel de precisión con su dataset de entrenamiento. Esto en parte, se debe al sesgo proveniente de la fuente de datos y al alto nivel de varianza en su intento por llegar a los valores reales, que a su vez provoca un desbalance en el entrenamiento y afecta directamente la calidad de predicciones que genera el algoritmo propuesto.

Técnicas de Regularización/Ajuste de parámetros

- **Ampliación del conjunto de Entrenamiento:**

Se utilizaron varias técnicas de regularización y/o ajuste para mejorar los resultados del modelo. Entre ellas, la primera que se aplicó, fue la de **ampliar el conjunto de entrenamiento,** restableciendo el ratio de entrenamiento-prueba-validación a 70% – 22.5% – 7.5%.

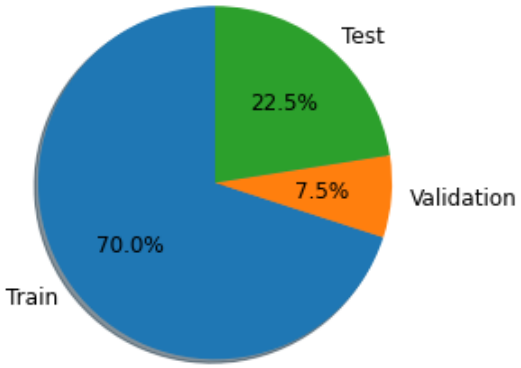


Figura No. 21 Nueva proporción del conjunto de datos

Resizing el conjunto de entrenamiento y prueba

```
[724]  1 X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = df.stroke, train_size = 0.7, test_size = 0.3, random_state=6)
       2 X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.25, random_state=1) # 0.3 x 0.25 = 7.5%
       3 X_train
```

Figura No. 22 Repartición del conjunto de datos de entrenamiento y prueba

Tras ampliar el conjunto de datos de entrenamiento se volvió a graficar la curva de aprendizaje para observar el efecto de esta técnica sobre la calidad del modelo y su nivel de sesgo así como su varianza:

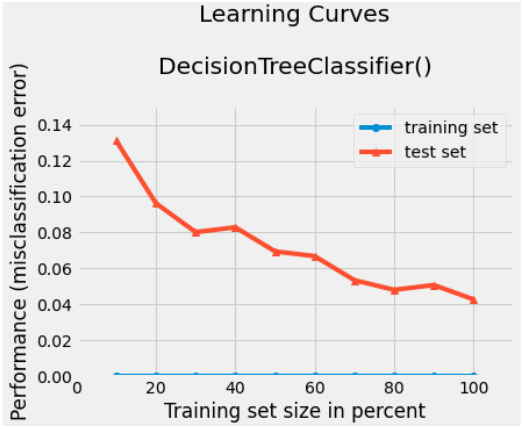


Figura No.23 Curva de aprendizaje basada en el error de clasificación árbol 2 después de cambiar la proporción del set de entrenamiento, prueba y validación

Con este método de refinamiento se puede ver que, aunque el modelo empieza con un error de clasificación mayor al que se tenía en el modelo original, este error logra reducirse hasta llegar a tan solo 4%. Por ende, se puede ver una mejora significativa en el sesgo del modelo.

- **Tree Prunning:**

En segunda instancia, se procedió a aplicar la técnica **de prunning o "podado" del árbol**, la cuál consiste en reducir las partes del árbol que realmente no aportan a clasificar las instancias. Los árboles de decisión son los algoritmos de Machine Learning más propensos al sobreajuste (overfitting) por lo que es bien conocido que esta técnica es de mucha utilidad para erradicar este suceso.

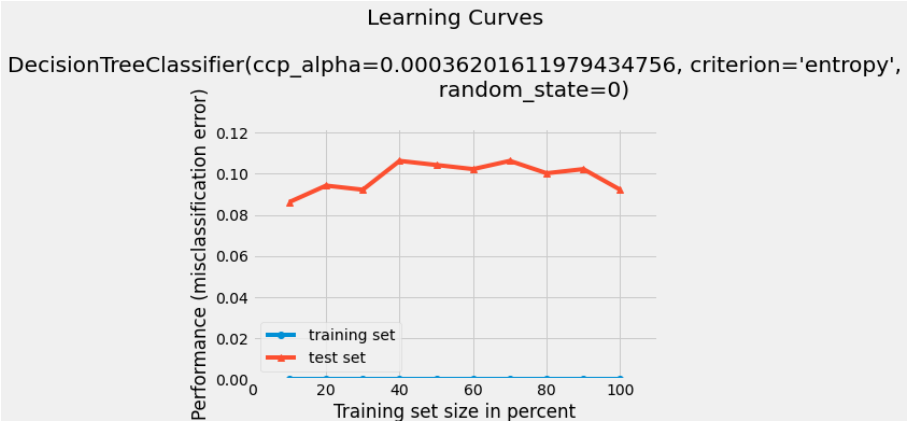


Figura No.24 Curva de aprendizaje basada en el error de clasificación después de la poda del árbol 2

Para este caso en específico, este método de refinamiento no causó mayor diferencia en la curva de aprendizaje del modelo ya que se puede apreciar que el error de clasificación del modelo sigue oscilando entre 8%-13%.

- **Bosques Aleatorios:**

En tercera instancia, se procedió a **aplicar un modelo de bosques aleatorios** el cuál es capaz de entrenar árboles con condiciones diferentes, seleccionadas de forma aleatoria. Por ende, el objetivo es reducir la variación de las salidas de los datos ya que corresponde al promedio de las salidas de los árboles de los que se compone.

Bosques Aleatorios

```
1 from sklearn import ensemble
2
3 myForest = ensemble.RandomForestClassifier()
4 myForest.fit(X_train, y_train)
5 print(myForest.score(X_test, y_test))
6 print(myForest.feature_importances_)
7 myForest.get_params()
8
```

```
0.9527207850133809
[0.03006163 0.23146169 0.0253057  0.02310029 0.02067935 0.051601
 0.03230295 0.27905309 0.23681475 0.06961876]
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Figura No. 25 Aplicación del Modelo de Bosques Aleatorios

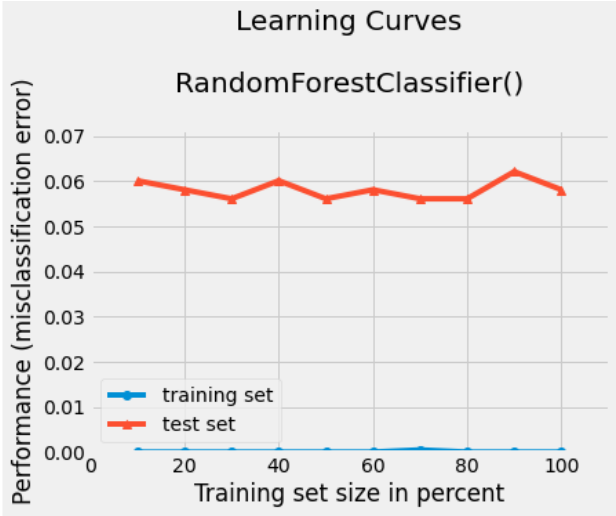


Figura No. 26 Curva de aprendizaje del árbol 2 después de la aplicación de Bosques Aleatorios

En efecto, se aprecia una mejora significativa en la curva de aprendizaje del modelo ya que el error de clasificación del set de prueba ahora oscila entre el 5.5%-6.5%, reflejando así una disminución significativa de la variación.

Comparación

En la siguiente tabla, se puede ver la comparación entre las métricas de desempeño del modelo en cada reajuste. Evidentemente, las métricas del Modelo de Bosques Aleatorios muestran resultados más favorables en cuanto a la calidad de las predicciones debido al alto nivel de precisión, exactitud y F1-Score los cuáles son los valores más altos entre el resto de modelos con sus respectivos ajustes. En lo que concierne al análisis del sesgo, el error cuadrático medio (MSE) es extremadamente bajo en comparación a las métricas de los otros modelos ya que es menor al 5%, lo que se traduce en una reducción considerable del nivel de sesgo en las predicciones. Similarmente, el coeficiente R2 demuestra un decremento en la variación de las predicciones, lo cual se refleja gráficamente en la curva de aprendizaje del modelo.

Métrica	***Modelo Original (Árbol de decisión con coeficiente de GINI)***	***Modelo con resizing***	***Modelo con Prunning***	***Modelo con Bosques Aleatorios:***
Precisión	0.92	0.91	0.92	0.91
Recall	0.92	0.91	0.91	0.95
F1-Score	0.92	0.91	0.91	0.93
R2 Score	-0.79	-0.74	-1.4	-0.6
MSE	0.08	0.09	0.09	0.05

Figura No. 27 Métricas de desempeño y calidad de los modelos

Por ende, se demuestra que el Modelo con reajuste de Bosques Aleatorios es el más favorable para apaciguar los efectos del sesgo y la variación en las predicciones. Consecuentemente, estos reajustes permiten pulir la calidad del modelo y obtener mejores resultados más acotados a la realidad. Dentro del contexto de los datos, esto significa tener mayor asertividad y confianza para predecir si un paciente puede llegar a tener un infarto cerebral o no dadas sus condiciones de vida e historial clínico previo. Subsecuentemente, esto permitirá a los doctores a identificar a las personas más propensas a tener un infarto y actuar prematuramente para prevenir este desafortunado suceso.