Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Inteligencia artificial avanzada para la ciencia de datos I
TC3006C.101

Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Rodolfo Sandoval Schipper A01720253

9 sept 2023

Índice general

1.- Implementación e introducción del Modelo

En esta entrega, se propone crear un modelo de clasificación utilizando el algoritmo Random Forest. Para llevar a cabo este proceso, optamos por utilizar Python como lenguaje de programación y el framework de sklearn, ya que es conocido por su facilidad de uso y una amplia gama de bibliotecas disponibles para la manipulación y análisis de datos. Estaremos utilizando un dataset numérico para mejorar el rendimiento para crear el modelo de clasificación. A continuación se muestra el proceso, el dataset utilizado, el rendimiento, y la configuración del modelo.

El objetivo es hacer un análisis de separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation).

Hacer un diagnóstico y explicar el grado de bias o sesgo: bajo medio alto.

Implementar diagnóstico y explicación el grado de varianza: bajo medio alto, diagnóstico y explicación del nivel de ajuste del modelo: underfit o overfit, e implementar técnicas de regularización para ajustar el modelo con los hiper parámetros óptimos. A continuación se entrega la liga del archivo collab en caso de no poder acceder al archivo en el repositorio: https://colab.research.google.com/drive/1j9ytO9ttK4igz3-l_9ni-yysltrV73IT?usp=sharing

1.1 Dataset Mobile Price Classification

Bob ha iniciado su propia empresa de telefonía móvil. Quiere dar una dura lucha a las grandes empresas como Apple, Samsung, etc.

No sabe cómo estimar el precio de los móviles que fabrica su empresa. En este competitivo mercado de la telefonía móvil no se pueden simplemente dar por sentado las cosas. Para solucionar este problema recopila datos de ventas de teléfonos móviles de varias empresas.

Bob quiere descubrir alguna relación entre las características de un teléfono móvil (por ejemplo: RAM, memoria interna, etc.) y su precio de venta. Pero no es tan bueno en Machine Learning. Entonces necesita tu ayuda para resolver este problema.

En este problema no es necesario predecir el precio real, sino un rango de precios que indique qué tan alto es el precio.

El dataset que estaremos utilizando es el siguiente: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv

Los archivos del conjunto de datos están adjuntados al repositorio cómo train móvil y test_movil.

Ejemplo del dataset:

id	battery_power		blue	clock_speed		dual_sim	fc	four_g	int_memory		m_dep	mobile_wt
...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen		wifi	
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16
226	1412	3476	12	7	2	0	1	0				
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12
746	857	3895	6	0	7	1	0	0				
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4
1270	1366	2396	17	10	10	0	1	1				
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20
295	1752	3893	10	0	7	1	1	0				
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18
749	810	1773	15	8	7	1	0	1				

Cantidad de registros: 2000

Variables:

battery_power: Capacidad total de energía que una batería puede almacenar en un momento dado, medida en mAh (miliamperios-hora).

blue: Indica si el dispositivo tiene Bluetooth o no.

clock_speed: Velocidad a la que el microprocesador ejecuta las instrucciones.

dual_sim: Indica si el dispositivo tiene soporte para dos tarjetas SIM o no.

fc: Mega píxeles de la cámara frontal.

four_g: Indica si el dispositivo es compatible con la tecnología 4G.

int_memory: Memoria interna del dispositivo en gigabytes.

m_dep: Profundidad del dispositivo móvil en centímetros.

mobile_wt: Peso del teléfono móvil.

n_cores: Número de núcleos del procesador.

pc: Mega píxeles de la cámara principal.

px_height: Altura de resolución en píxeles.

px_width: Ancho de resolución en píxeles.

ram: Memoria de acceso aleatorio (RAM) en megabytes.

sc_h: Altura de pantalla del dispositivo en centímetros.

sc_w: Ancho de pantalla del dispositivo en centímetros.

talk_time: Tiempo más largo que una sola carga de batería durará cuando se utiliza.

three_g: Indica si el dispositivo es compatible con la tecnología 3G.

touch_screen: Indica si el dispositivo tiene pantalla táctil o no.

wifi: Indica si el dispositivo tiene capacidad de conexión Wi-Fi o no.

Clases: 0(Precio Minimo), 1(Precio Bajo), 2(Precio Medio), 3(Precio Alto)

2.- Analisis con (Train/Test/Validation)

Primero cargamos los datos iniciales. Estos datos contienen características (atributos) y etiquetas (los valores que se intentan predecir). Luego hicimos la división en conjuntos de entrenamiento y de prueba. El código utiliza la función train test split de scikit-learn para dividir el conjunto de datos en dos partes: entrenamiento y prueba. La proporción de división se controla mediante el argumento test size, que se estableció en 0.2, lo que significa que el 80% de los datos se usarán para entrenar y el 20% se reservará para probar el modelo. Luego

utilizamos el conjunto de entrenamiento y de validación, los conjuntos de entrenamiento (X_train y y_train) se utilizan para entrenar el modelo. Esto significa que el modelo ajustará sus parámetros y aprenderá a hacer predicciones utilizando estos datos. Esta validación se logró hacer con Grid de sklearn donde se regulariza el modelo utilizando un conjunto de diferentes parámetros a probar para darle el mejor ajuste posible con de acuerdo a las mediciones del entrenamiento con los datos reales. Esto está desplegado por medio de las gráficas donde se diagnostica el sesgo y la varianza del resultado. La información y los resultados se pueden consultar. En la siguiente liga: https://colab.research.google.com/drive/1j9ytO9ttK4igz3-l_9ni-yysltrV73IT?usp=sharing

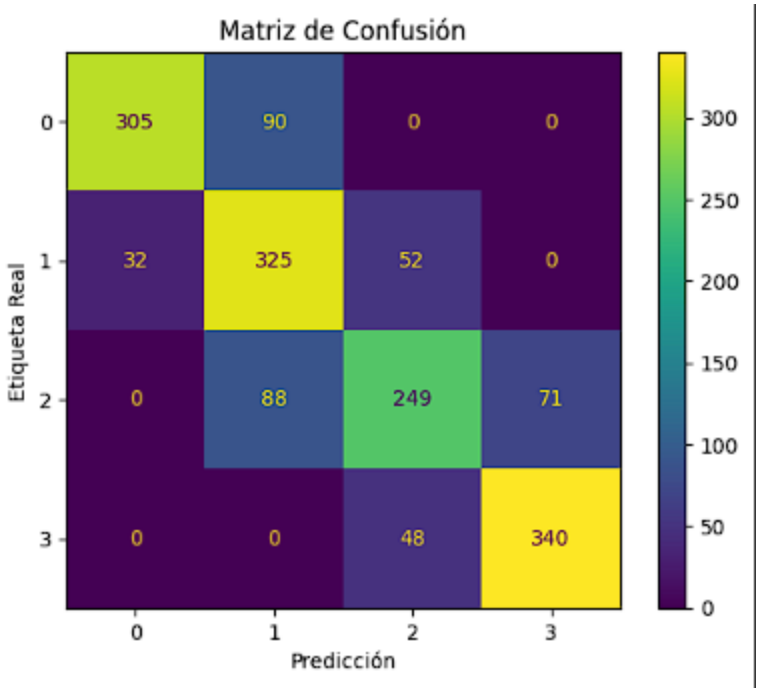
Evaluamos el modelo con el conjunto prueba:

El porcentaje de precisión es de 75%. Sin embargo, era necesario obtener este porcentaje para ajustar el modelo con el mejor fitting posible. Los resultados/predicciones del modelo son los siguientes:

Predicciones en el conjunto de prueba:

```
[3 3 2 3 1 3 3 1 3 0 3 3 0 0 2 0 1 1 3 2 1 3 1 2 3 0 2 0 3 0 2 0 3 0 0 1 3
 1 3 2 0 2 0 0 0 1 1 2 1 2 2 0 2 1 3 1 3 1 1 3 3 3 1 1 1 1 1 2 1 1 1 1 2 3
 3 0 2 0 2 3 1 3 3 0 3 0 3 1 3 0 1 1 2 0 2 1 0 2 1 3 1 0 0 2 1 2 0 1 2 3 3
 3 1 3 3 3 3 1 3 0 0 3 2 1 1 0 3 2 3 1 0 1 1 2 3 1 1 0 3 2 1 3 2 2 3 3 3 2
 2 3 3 3 0 1 2 2 3 3 3 3 2 2 3 3 2 3 1 0 3 0 0 0 1 0 0 1 1 0 1 2 0 0 1 0 2
 2 2 1 0 0 0 0 0 3 1 1 2 1 2 3 1 2 3 3 3 1 2 0 0 1 1 2 1 2 3 3 1 2 0 3 2 2
 2 0 0 1 0 3 0 1 0 2 1 1 3 0 3 0 3 1 1 0 0 2 1 2 2 3 1 1 3 0 0 3 3 3 1 3 1
 0 3 2 1 3 3 3 3 1 0 1 1 3 1 1 3 1 0 3 1 1 2 0 0 3 2 3 2 2 1 3 2 2 3 2 2 1
 1 1 2 3 1 0 0 3 0 3 1 1 1 0 1 3 1 3 1 2 1 2 1 0 0 1 3 1 0 0 0 3 1 1 3 3 1
 3 3 2 3 1 3 3 2 2 3 3 3 0 3 1 3 1 3 1 3 3 0 1 1 3 1 3 1 3 0 0 1 0 3 0 1 1
 1 1 1 3 2 0 1 0 1 2 2 1 3 1 2 2 2 1 3 1 1 3 2 1 2 0 1 1 1 3 2 1 0 2 0 0 1
 0 0 0 0 2 2 3 2 3 0 3 1 3 1 1 1 1 2 0 2 2 3 3 1 2 1 3 1 3 2 1 2 2 2 1 0 0
 1 1 1 1 1 3 3 1 3 3 0 1 3 1 1 0 3 2 3 1 3 1 2 3 3 3 1 2 0 2 2 0 1 1 0 0 1
 1 2 3 3 3 2 2 1 1 2 2 3 3 1 1 3 2 1 2 1 0 2 3 0 1 0 3 1 1 3 2 2 0 3 0 2 2
 0 3 0 3 3 0 1 1 3 3 1 0 2 3 3 0 2 1 3 0 3 3 0 2 3 2 3 0 1 2 3 1 3 2 3 1 0
 1 0 3 1 0 3 2 3 1 0 3 3 3 2 3 3 1 1 1 3 3 3 1 0 1 1 1 2 2 1 0 2 2 3 2 0 3
 1 3 3 0 1 3 1 2 1 0 0 0 2 1 0 1 0 2 2 0 2 3 1 0 3 1 1 3 2 0 0 0 0 0 3 0 3
 1 3 2 1 3 3 1 1 1 3 2 2 1 0 3 0 1 1 2 1 1 1 1 1 2 1 3 1 3 2 1 1 3 2 0 1 3
 0 3 3 0 2 1 1 3 0 3 2 0 3 2 3 0 0 3 0 1 2 3 2 1 1 2 1 1 3 1 1 1 2 2 1 0 0
 1 0 0 3 0 1 1 0 1 1 0 2 0 3 2 3 0 0 1 2 2 1 0 1 2 0 1 2 0 0 3 3 1 3 1 2 3
 0 1 0 1 2 0 3 2 0 3 0 1 1 2 3 3 2 3 0 3 2 0 1 0 2 3 1 0 2 1 2 1 0 3 2 1 3
 1 1 1 1 1 3 1 2 1 2 0 0 1 1 0 2 0 0 1 1 3 3 3 3 0 1 2 1 1 0 0 3 1 0 2 0 2
 2 2 2 2 0 1 1 3 0 0 3 1 3 0 0 2 2 2 1 2 3 1 0 0 2 3 0 3 0 0 1 2 2 1 2 0 3
 2 1 2 2 3 0 1 1 2 0 3 2 0 1 3 1 1 3 1 2 3 1 1 1 2 3 3 0 3 3 0 2 3 2 2 2 3
 2 1 1 2 0 2 1 1 1 3 2 1 2 0 1 1 3 1 0 2 1 3 1 0 0 2 2 2 2 0 3 3 2 1 3 1 1
 3 1 2 1 2 2 3 0 3 0 2 2 0 2 1 2 2 1 1 3 3 1 0 1 1 3 1 3 0 2 2 0 2 3 3 3 0
 2 1 0 1 2 3 3 0 2 2 2 1 2 0 1 2 0 1 0 0 3 2 2 0 1 0 0 3 2 3 2 0 0 1 1 1 2
 2]
```

Matriz de confusión con los datos reales (objetivo) y las predicciones:



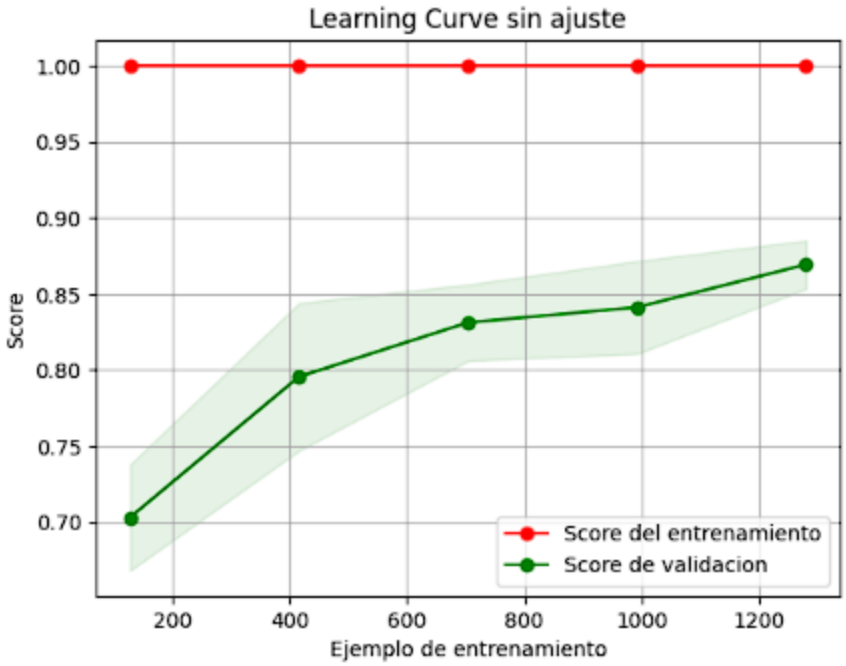
2.1 Observaciones

El modelo parece funcionar bien en la clasificación de las clases principales (0, 1, 2 y 3). Esto significa que es capaz de predecir con precisión los valores que son más comunes en el conjunto de datos. La presencia de pocos datos fuera de la relación principal indica que el modelo puede tener dificultades para clasificar las clases minoritarias o menos frecuentes. Esto puede deberse a un pequeño desequilibrio de clases en los datos, donde algunas clases tienen muchas más instancias que otras. En tales casos, es común que el modelo se desempeñe peor en las clases minoritarias debido a la falta de ejemplos de entrenamiento. Esto nos ayuda también a conseguir mejores hiper parámetros para configurar el modelo y obtener mejor rendimiento.

3.- Diagnóstico del sesgo y/o varianza

Probar modelo sin hiper parámetros para ver cómo ajustarlo:

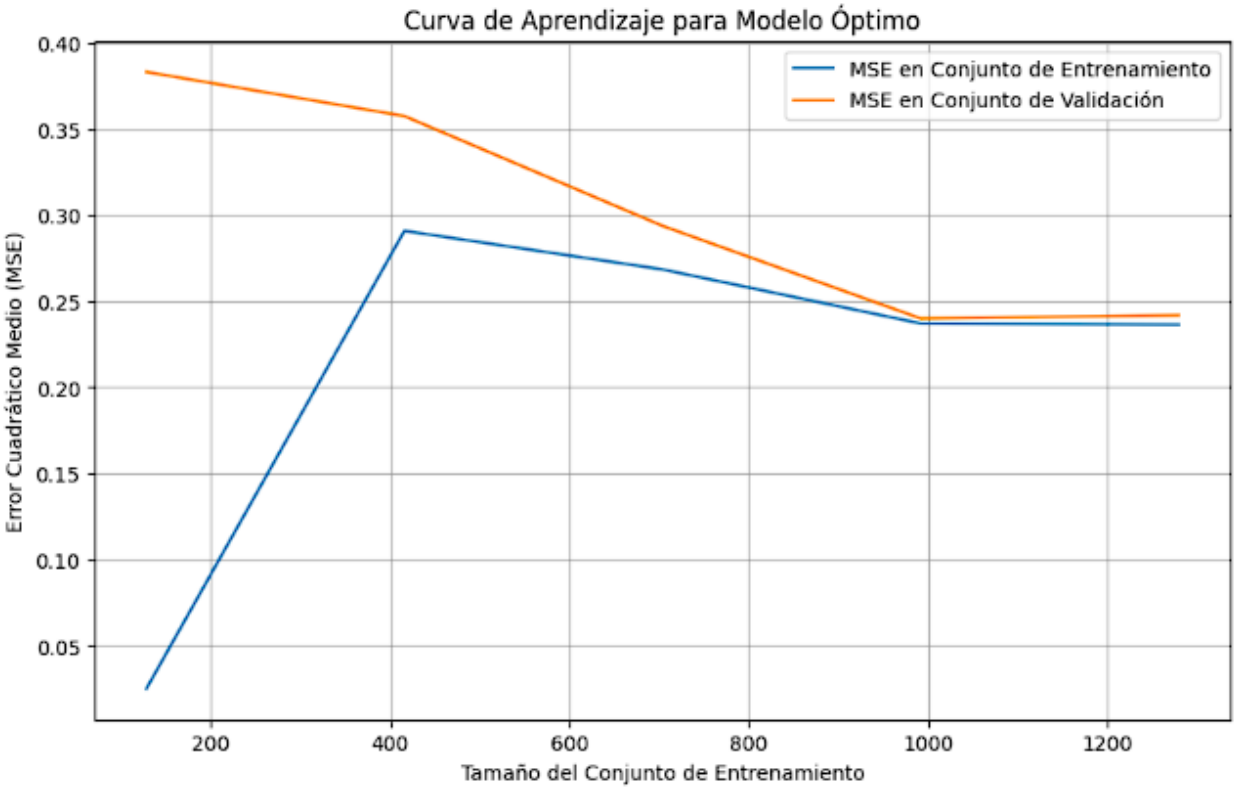
Primero debemos obtener los resultados del modelo sin hacer un ajuste de los hiper parámetros. Esto no es buena práctica ya que los resultados y el rendimiento del modelo pueden tener un nivel alto de sesgo y/o varianza. Para este caso, implementamos el modelo y graficamos estos niveles del entrenamiento y su puntaje de validación. Nuestro resultado con el modelo sin el ajuste es el siguiente:

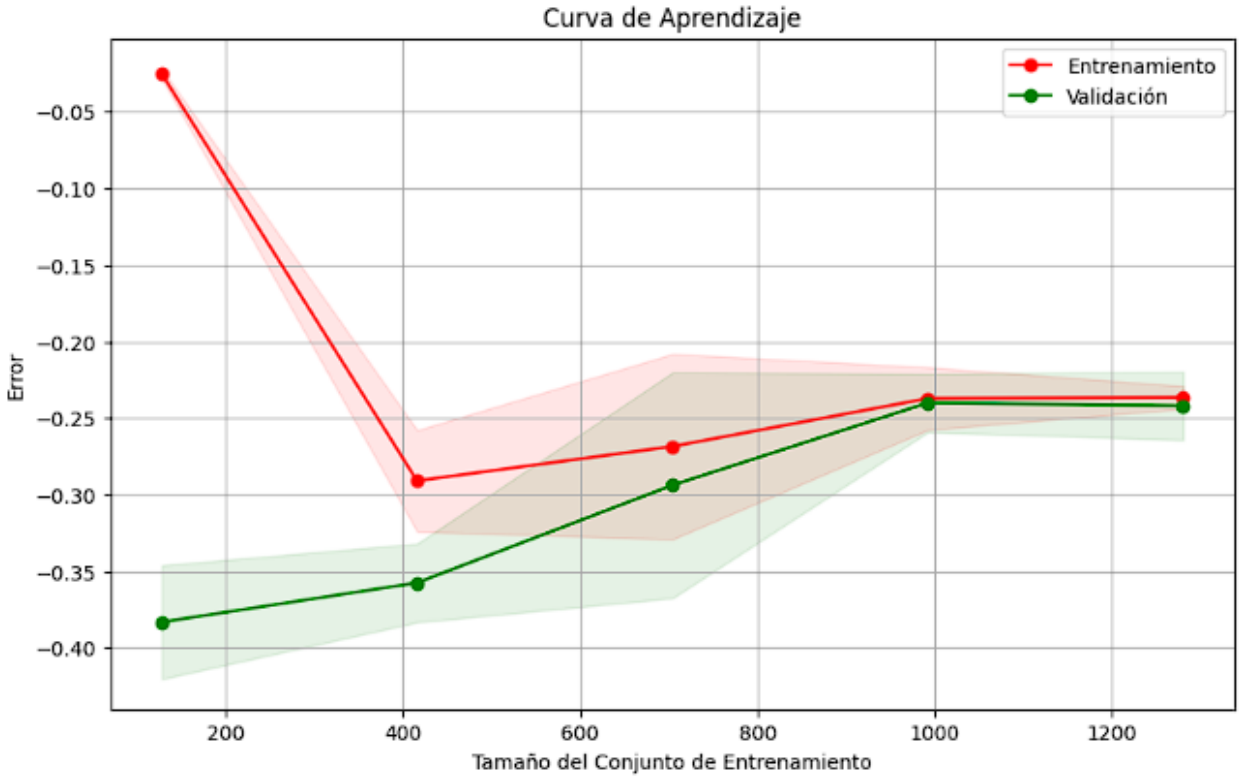


Cómo podemos ver el sobreajuste ocurre cuando el modelo es demasiado complejo y memoriza los datos de entrenamiento en lugar de aprender patrones generales que puedan aplicarse a nuevos datos (generalización). Un puntaje de entrenamiento perfecto (1.0) en la curva de aprendizaje es una señal de que el modelo está aprendiendo los datos de entrenamiento en exceso y no está generalizando bien. Tal que, debemos ajustar el modelo para tener el mínimo MSE de acuerdo a ambos puntajes. Esto se logra simplificando el modelo, ya que con los valores default el modelo tiene un ajuste de **overfitting.**

Para ajustar el modelo, lo único que tenemos que hacer es buscar los hiper parámetros. Para el siguiente modelo hemos establecido min_impurity_decrease en 0.02, lo que significa que se requerirá una reducción significativa en la impureza para que se produzca una división en un nodo. Hemos establecido max_features para calcular la raíz cuadrada del número total de características limitando el número máximo de características que se considerarán al buscar la mejor división en un nodo a la raíz cuadrada del número total de características en los datos de entrenamiento. Esta es una técnica común para reducir la complejidad del modelo y evitar el sobreajuste en modelos Random Forest.

Nuestra formula para max_features fue : max_features = int(np.sqrt(X_train.shape[1]))
El diagnóstico del sesgo y la varianza en un modelo de aprendizaje automático se puede realizar a través del análisis de la curva de aprendizaje. Esta curva muestra cómo cambia el rendimiento del modelo a medida que se incrementa el tamaño del conjunto de entrenamiento.


Curva de Aprendizaje para Modelo Óptimo
MSE en Conjunto de Entrenamiento
MSE en Conjunto de Validación
Error Cuadrático Medio (MSE)
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
200
400
600
800
1000
1200
Tamaño del Conjunto de Entrenamiento


Curva de Aprendizaje
Entrenamiento
Validación
Error
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
−0.40
200
400
600
800
1000
1200
Tamaño del Conjunto de Entrenamiento

3.1 Observaciones

Sesgo: Se puede observar que tanto el MSE en el conjunto de entrenamiento como en el conjunto de validación se estabilizan a medida que se aumenta el tamaño del conjunto de entrenamiento. Esto indica que el modelo tiene un buen equilibrio entre sesgo y varianza y es un buen modelo generalizador.

Varianza (Variance): Se puede observar que finalizando el entrenamiento no hay una brecha significativa entre el MSE en el conjunto de entrenamiento y el MSE en el conjunto de validación, esto indica una varianza óptima para los resultados del modelo.

Con de acuerdo a los aspectos más técnicos de cómo se implementó los métodos de validación, se reitera que se utiliza learning curve de sklearn para calcular el error cuadrático medio en el conjunto de entrenamiento y en el conjunto de validación. Luego, se convierten los valores de error en positivo multiplicando por -1, ya que learning curve devuelve valores negativos de error.

Para obtener los resultados de acuerdo al código, graficamos la curva de aprendizaje. En el eje x, se muestra el tamaño del conjunto de entrenamiento, y en el eje y, se muestra el MSE. La curva de aprendizaje se divide en dos líneas. Una representa el MSE en el conjunto de entrenamiento y otra que representa el MSE en el conjunto de validación.

Niveles sesgo y varianza:

Con el pequeño ajuste que logramos hacer con nuestro modelo podemos observar que las curvas de entrenamiento y validación son estables y no muestran una variabilidad significativa a medida que aumenta el tamaño del conjunto de entrenamiento. Esto sugiere que el modelo no es muy sensible al número de ejemplos de entrenamiento y es consistente en su rendimiento. Es importante tener en cuenta que las curvas de aprendizaje pueden variar según los parámetros ingresados. A veces, puede ser difícil alcanzar un rendimiento perfecto en ambos conjuntos, pero el objetivo es encontrar un equilibrio entre sesgo y varianza para lograr un buen rendimiento en datos no vistos. Las curvas de aprendizaje son herramientas valiosas para evaluar y ajustar el rendimiento de tu modelo.Tal que observando nuestro modelo podemos decidir que:

Nivel de Sesgo: Las curvas están cercanas y siguen el mismo patrón (low bias) ***sesgo bajo*** esto indica que es capaz de ajustarse a los datos de entrenamiento y generalizar bien a datos no vistos. Esta situación es deseable porque indica que el modelo está capturando adecuadamente la relación entre las características y las etiquetas del problema.

Nivel de Variación: tanto la puntuación de entrenamiento como la puntuación de validación están cercanas y siguen el mismo patrón a medida que aumenta el tamaño del conjunto de entrenamiento, generalmente esto indica que el modelo tiene una ***varianza baja*** (low variance).

4.- Diagnóstico del ajuste del modelo y parámetros

Mencionado anteriormente en el punto ***3. Diagnóstico***, el modelo inicial tenía un sobreajuste (overfit) tal que tuvimos que reducir la complejidad del modelo para obtener un mejor rendimiento. Primero, esto se logró con el ajuste de los parámetros min_impurity_decrease, y con max_features el cual utilizamos para ajustar cómo el modelo recibe las características. Sin embargo, tenemos que ajustar el resto de los hiper parámetros para llegar a un resultado más óptimo y consistente. A continuación se mencionan el resto de la configuración.

La configuración más óptima que se logró tener es de:

‘min_impurity _decrease’ →
Min impurity decrease establece un umbral en la reducción de la impureza. Si la reducción de la impureza obtenida al dividir un nodo es menor que min impurity decrease, entonces la división no se realizará, y el nodo se considerará una hoja. Si la reducción de la impureza es igual o mayor que min impurity decrease, entonces la división se realizará.

'n estimators': **Número de árboles**
'max_depth': **Profundidad de árbol**
'min_samples_split': **Número de división de nodos**

Parametro	Valor
min_impurity_decrease	0.02
max_depth	4
min_samples_split	10
n_estimators	50

Se ajustaron estos parámetros ya que el resultado nos da el mínimo MSE y se sigue el patrón a una distancia cercana del puntaje de validación y de entrenamiento. Concluyendo que este ajuste es ideal para los datos que se están analizando en el modelo. Estos parámetros se ajustan en el modelo de modelo_optimo (ubicado en el código) para realizar el entrenamiento con estos hiper parámetros y fijar el mejor rendimiento posible con de acuerdo a los resultados graficados.

El porque.-
min_impurity_decrease: Este hiper parámetro controla la cantidad mínima de reducción requerida en la impureza (criterio de división) para que se produzca una división en un nodo del árbol. Se configura min_impurity_decrease en 0.02, lo que significa que se requerirá una reducción relativamente grande en la impureza para dividir un nodo. Este valor ayuda a limitar

la profundidad del árbol y prevenir divisiones que no proporcionen una mejora significativa en la calidad de las divisiones.

max_depth: Limitamos la profundidad máxima de cada árbol a 4. Esto significa que ningún árbol en el bosque tendrá más de 4 niveles de profundidad. Limitar la profundidad de los árboles puede ayudar a controlar la complejidad del modelo y evitar un sobreajuste excesivo a los datos de entrenamiento.

min_samples_split: Configuramos min_samples_split en 10, lo que significa que se requerirán al menos 10 muestras en un nodo para que se realice una división. Esto puede ayudar a prevenir divisiones en nodos con un número muy bajo de muestras, lo que nuevamente contribuye a evitar un sobreajuste.

n_estimators: Configuramos n_estimators en 50, lo que indica que estamos construyendo un bosque de 50 árboles de decisión. Tener un número moderado de árboles en el bosque puede ser eficiente en términos computacionales y suele ser suficiente para obtener un buen rendimiento.

Nivel de ajuste: El ajuste con el que terminó el modelo fue con poco **underfit,** un ligero bajo ajuste puede ser una señal de que el modelo es lo suficientemente robusto como para tolerar el ruido o la variabilidad en los datos de entrenamiento. En situaciones en las que los datos de entrenamiento contienen ruido o errores menores, un modelo que no se ajusta de manera excesiva a estos errores puede generalizar mejor a nuevos datos sin ruido. También hay que tomar en cuenta que un modelo con bajo ajuste tiende a ser más robusto y puede funcionar bien en una variedad de conjuntos de datos relacionados. Esto es especialmente valioso en aplicaciones del mundo real donde los datos pueden cambiar con el tiempo o en diferentes contextos.Estas fueron las decisiones que se tuvieron que llevar a cabo para mejorar los resultados ya que inicialmente el overfit era grande. En última instancia, el equilibrio entre sesgo y varianza (y, por lo tanto, la presencia de bajo ajuste) debe ser evaluado en función de los objetivos del problema y las restricciones del mundo real. A veces, un modelo con un ligero bajo ajuste puede ser una elección razonable y práctica.