Inteligencia artificial avanzada para la ciencia de datos
Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Kathia Bejarano Zamora
A01378316

11 de Septiembre 2023

Instituto Tecnológico y de Estudios Superiores de Monterrey.
Campus Estado de México.

"Yo, como integrante de la comunidad estudiantil del Tecnológico de Monterrey, soy consciente de que la trampa y el engaño afectan mi dignidad como persona, mi aprendizaje y mi formación, por ello me comprometo a actuar honestamente, respetar y dar crédito al valor y esfuerzo con el que se elaboran las ideas propias, las de los compañeros y de los autores, así como asumir mi responsabilidad en la construcción de un ambiente de aprendizaje justo y confiable."

PERCEPTRON

Un perceptrón es una neurona artificial, es fundamental para las redes neuronales. Basicamente lo que se busca es simular las neuronas de un ser humano, SI recordamos el funcionamiento tenemos las ramificaciones, que se encargan de recibir toda la información y trasladarla a los núcleo que hace todo el tratado de esta información y finalmente sale la información ya tratada. Lo que hace la neurona artificial es imitar lo descrito anteriormente pero basada en una función matemática, cada neurona recibirá datos, los pesa, calcula la suma y obtiene un resultado.

A continuación muestro una imagen en donde se ve de manera gráfica una neurona y su comparativa con la neurona aritficial.

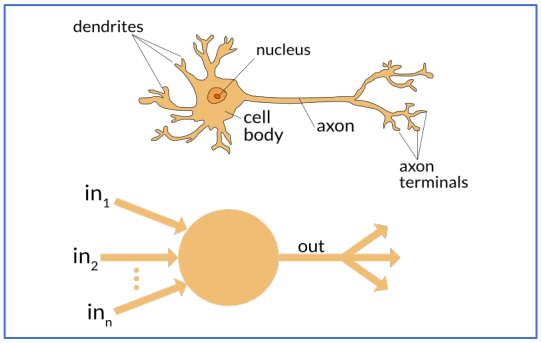


Imagen 1. Perceptron

https://appliedgo.net/perceptron/

El perceptrón se forma por distintos componenetes, que a pesar de ya haberlos visto de manera gráfica es importante describirlos uno a uno.

Input (Entrada): En este caso las entradas serán las características del conjunto de datos.

Pesos(w): Son valores preliminares y con el paso del aprendizaje se van ajustando, basicamente son quienes nos irán dictando la mejora y una buen apredicción

Función de activación: Es la función que se realiza dentro de la neurona que muy comunmente es la sigmoidal.

Output(salida): Suma ponderada que pasó por la activación y será el valor obtenido apartir de este flujo.

A continuación hablaremos de manera más especóifica sobre el trabajo realizado.

BIAS, ACCURACY VARIANCE

Al ser un modelo tenemos métricas de validación que nos ayudan a interpretar nuestros resultados. A continuación describire los 3 fundamentales.

El primero de ellos es el "bias", o sesgo en español, es una medición de la exactitud y representa ese error sistemático del modelo. Su principal parámetro es el error. Su interpretación se da de la siguiente manera:

Si lo obtenido es alto, el modelo es simple y existe una posibilidad de underfitting. Lo que se busca es un sesgo lo más bajo posible.

El segundo de ellos es "accuracy" o precisión, es el cocientre entre los verdaderos positivos y las predicciones positivas producidas en el modelo. (algo que se ve muy bien en la primer entrega de mi perceptron manual). En este caso el valor esperado debe ser 1.

Finalmente tenemos a la varianza, concepto de estadística, representa la variabilidad de una serie de datos, con respecto su promedio.

SET DE DATOS

Para la selección del set de datos utilizamos el set de datos Iris, que provee directamente sklearn. Este set de datos se escogió principalmente por los datos numéricos que contiene.

-El análisis del dataset se genera a partir de las especies de iris tales: setosa, versicolor y virginica.

-El conjunto consta de 150 muestras, por lo que es un buen npumero para el análisis por medio del preceptrón.

-Existen cuatro características en el conjunto de datos.

-Tenemos muy claramente definidos nuestros targets que se mencionaron anteriormente.

A continuación muestro un preview del set de datos.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Imagen 1.1 Preview Iris

https://www.kaggle.com/code/kostasmar/exploring-the-iris-data-set-scikit-learn

JUSTIFICACIÓN DE USO DE IRIS

El conjunto de datos IRIS es uno de los conjuntos de datos más populares en el aprendizaje automático y la clasificación. Es una base de datos de referencia para la clasificación de especies de flores iris en función de medidas como la longitud y el ancho de los pétalos y sépalos.
Hay varias razones por las que personalmente considero el conjunto IRIS muy bueno para el este tipo de ejercicio:

1. Simplicidad: El conjunto de datos IRIS es pequeño, fácil de entender y visualizar. Con solo cuatro características y tres clases de destino.

2. Problema de clasificación multiclase: El conjunto de datos IRIS es un problema de clasificación multiclase, lo que significa que es adecuado para probar algoritmos de clasificación que pueden manejar múltiples clases.

En cuanto a implementarlo en un código de perceptrón, al ser un algoritmo de aprendizaje supervisado de clasificación binaria (dos clases en este caso setosa o no setosa) utilizado para separar linealmente datos en dos categorías. Y considerando que el conjunto de datos IRIS es un problema de clasificación multiclase, lo que significa que hay más de dos clases, aun así se optó por utilizarlo debirdo a que al realizar la transformación correspondiente, es muy útil para enseñar conceptos de clasificación y aprender sobre cómo manejar problemas de múltiples clases en el contexto de algoritmos de aprendizaje automático.

GENERALIZACIÓN Y MI DATASET

La "generalización" en el ámbito del aprendizaje automático se refiere a la habilidad de un modelo entrenado para realizar pronósticos precisos en datos que no ha encontrado durante su fase de entrenamiento. En otras palabras, un modelo con buena capacidad de generalización puede aplicar eficazmente el conocimiento adquirido de un conjunto de datos de entrenamiento a nuevas situaciones o datos que no ha visto previamente.

En el contexto específico del conjunto de datos IRIS y la generalización, existen varios aspectos a tener en cuenta:

1. Tamaño del conjunto de datos: El conjunto de datos IRIS comprende 150 muestras de iris distribuidas en tres clases diferentes. Al dividir este conjunto de datos adecuadamente en conjuntos de entrenamiento y prueba, podemos evaluar la habilidad de generalización del modelo en datos que no se utilizaron durante el proceso de entrenamiento. Es crucial garantizar que haya suficientes datos en el conjunto de prueba para obtener una evaluación confiable de la capacidad de generalización, lo cual es fundamental.

2. Validación cruzada: La validación cruzada es una técnica que involucra la división de los datos en múltiples conjuntos de entrenamiento y prueba de manera iterativa para evaluar la capacidad de generalización de un modelo de manera más sólida. Utilizamos la validación cruzada para obtener una estimación más precisa de cómo el modelo se desempeñará en datos no vistos.

Lograr la generalización en el contexto del conjunto de datos IRIS implica entrenar un modelo que no solo se desempeñe bien en los datos utilizados para el entrenamiento, sino que también sea capaz de hacer predicciones precisas en nuevos datos de prueba que no estuvieron involucrados en el proceso de entrenamiento.

Para alcanzar una adecuada generalización, es esencial considerar factores como el tamaño del conjunto de datos, la elección del modelo, la aplicación de técnicas de regularización y el uso de métodos como la validación cruzada para evaluar el rendimiento del modelo en datos no observados.

```
Características (X):
>>> print(X[:5])  # Imprime las primeras 5 filas de características
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
print("\nEtiquetas (y):")
print(y[:5])  # Imprime las primeras 5 etiquet>>> print("\nEtiquetas (y):")

Etiquetas (y):
>>> print(y[:5])  # Imprime las primeras 5 etiquetas
[1 1 1 1 1]
```

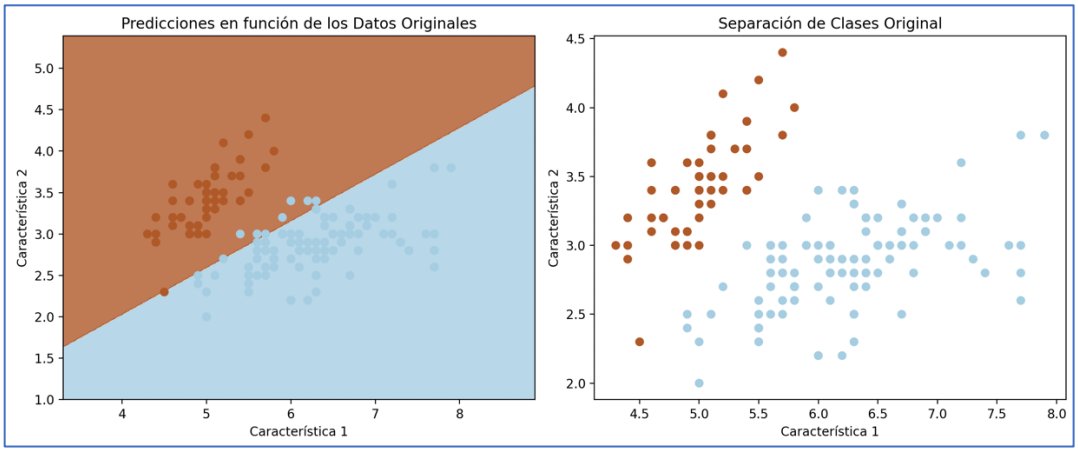
Imagen 1.2

"PerceptronIris. Py

A continuación, explicaré cómo se realizó la separación y visualización de los datos:

1. Preparación de los Datos: Primero, se cargan los datos del conjunto de datos IRIS utilizando la biblioteca scikit-learn. Luego, se realiza una transformación en la etiqueta "y" para convertirla en un problema de clasificación binaria, donde "1" representa la clase "setosa" y "0" representa las otras dos clases. Además, se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función `train_test_split` para poder evaluar el rendimiento del modelo en datos no vistos.

En este código, agrego un gráfico de dispersión (subplot) que muestra la separación de las clases originales en el conjunto de datos IRIS (antes de la predicción del Perceptrón). El primer gráfico de dispersión muestra las predicciones del Perceptrón en función de las dos primeras características, y el segundo muestra cómo estaban originalmente separadas las clases en el conjunto de datos.



Gráfica 1.1

"PerceptronIris. Py

Estos gráficos ayudan a visualizar cómo el Perceptrón aprendió a separar las clases y cómo se compara con la separación original en los datos. Esto puede proporcionar una comprensión más clara de cómo el modelo está funcionando y si está logrando una buena separación de clases.

CÓDIGO

Para esta primer implementación del código utilizamos sklearn para básicamente realizar todo el análisis. Importante mencionar que para el llamado de la función Perceptron definimos el número de iteraciones y alpha.

El número de iteraciones nos ayuda a las corridas y el alpha es ese ajuste de pesos que se mencionó en la introducción.

A diferencia de la entrega previa, este código ya corre con el set de datos bien seleccionado por lo que pudimos ver muchas mejoras.
Continuando con la explicación del código:

2. Entrenamiento del Perceptrón: Se crea un objeto Perceptrón con parámetros como el número máximo de iteraciones (max_iter) y la tasa de aprendizaje (alpha). Luego, el modelo se ajusta a los datos de entrenamiento utilizando set.fit(X_train, y_train).

3. Predicciones y Evaluación: Se realizan predicciones tanto en los datos de entrenamiento como en los de prueba utilizando el modelo entrenado. Se calcula la precisión (accuracy) en ambos conjuntos de datos para evaluar el rendimiento del modelo.

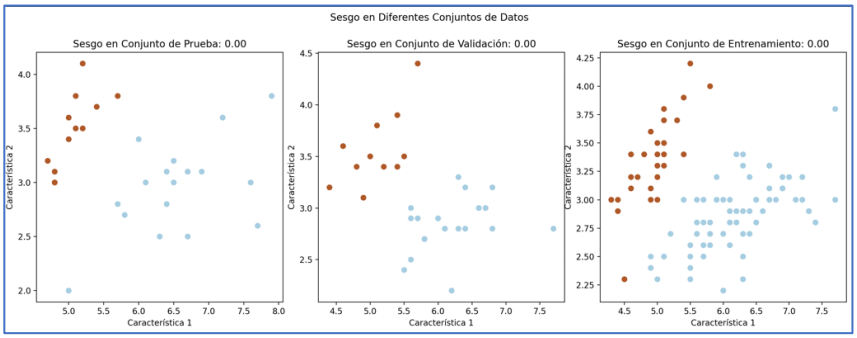
4. Validación Cruzada: Se utiliza la validación cruzada (cross_val_predict) para obtener predicciones en múltiples conjuntos de prueba. Esto proporciona una evaluación más robusta del modelo.

GRÁFICAS DE EVIDENCIA BIAS

Para este código podemos observar que realmenet el códgio es muy bueno. Continuación de la descripción de mi código:

5. Visualización: Se crean tres gráficos para visualizar los resultados:

a. Métricas de Evaluación: Se muestra un gráfico de barras que representa tres métricas: varianza de las predicciones, sesgo (bias), y precisión promedio. Estas métricas ayudan a evaluar el rendimiento y la calidad del modelo.



Gráfica 1.2

Comaprativa Sesgo diferentes conjuntos

Las tres gráficas representan el sesgo en los conjuntos de datos de entrenamiento, prueba y validación. Algunas observaciones sobre estas gráficas son las siguientes:

1. Sesgo en el Conjunto de Prueba (Gráfica 1):

- Esta gráfica muestra cómo el modelo está sesgado en el conjunto de prueba, que son datos que no se utilizaron durante el entrenamiento.
- Al igual que en el conjunto de entrenamiento, un sesgo cercano a cero indica que el modelo está haciendo predicciones cercanas a la media real de las etiquetas de clase en el conjunto de prueba.
- Es importante que el sesgo en el conjunto de prueba no sea significativamente diferente del sesgo en el conjunto de entrenamiento para que se este generalizando bien

2. Sesgo en el Conjunto de Validación (Gráfica 2):

- Esta gráfica muestra cómo el modelo está sesgado en el conjunto de validación, que generalmente se utiliza para ajustar hiperparámetros y evaluar el rendimiento final del modelo.

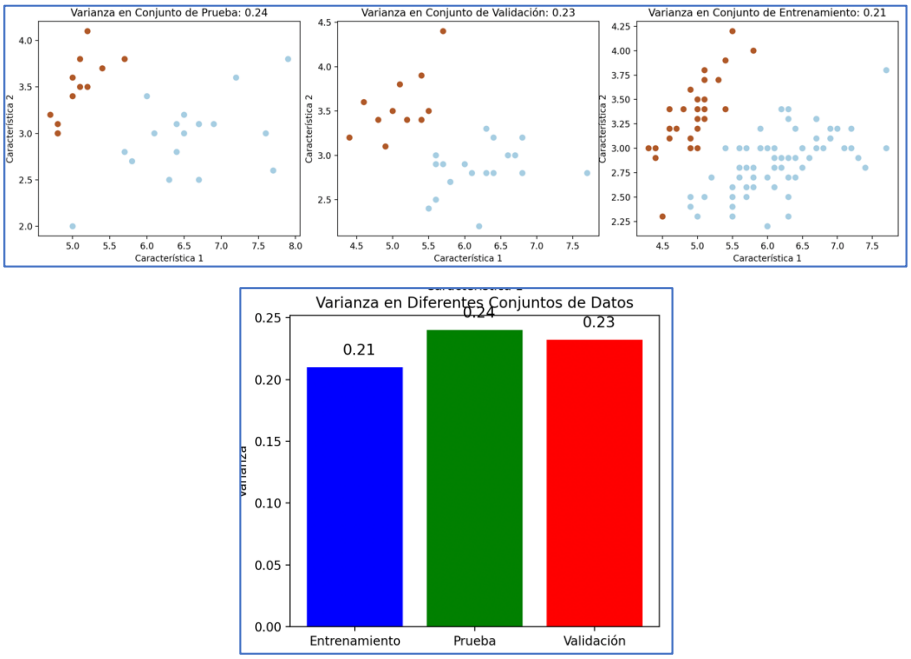
- Al igual que en los otros conjuntos, un sesgo cercano a cero indica que el modelo está haciendo predicciones cercanas a la media real de las etiquetas de clase en el conjunto de validación.

3. Sesgo en el Conjunto de Entrenamiento (Gráfica 3):

- Esta gráfica muestra cómo el modelo está sesgado en el conjunto de entrenamiento.

- El sesgo se calcula como la diferencia entre la media de las predicciones del modelo en el conjunto de entrenamiento y la media real de las etiquetas de clase.

GRÁFICAS DE EVIDENCIA VARIANZA



Gráfica 1.3

Comaprativa Varianza diferentes conjuntos

La varianza en los tres conjuntos de datos (entrenamiento, prueba y validación) se refiere a la dispersión o variabilidad de las predicciones del modelo en cada uno de esos conjuntos.

1. Conjunto de Entrenamiento:
   - La varianza en el conjunto de entrenamiento es relativamente baja, lo que significa que las predicciones del modelo en este conjunto tienden a estar cerca del valor promedio.
   - Esto sugiere que el modelo se ajusta bien a los datos de entrenamiento y que las predicciones no varían significativamente cuando se entrena repetidamente en el mismo conjunto.

2. Conjunto de Prueba:
   - La varianza en el conjunto de prueba es más alta que en el conjunto de entrenamiento, pero aún es moderada. Esto indica que las predicciones en el conjunto de prueba pueden variar un poco más en comparación con el conjunto de entrenamiento, pero no de manera excesiva.
   - El modelo parece generalizar de manera razonable a datos no vistos, ya que la varianza no es muy alta.

3. Conjunto de Validación:
   - La varianza en el conjunto de validación es la más alta de los tres conjuntos.
   - Esto sugiere que las predicciones del modelo en el conjunto de validación pueden variar significativamente, lo que puede indicar que el modelo podría estar sobreajustando un poco los datos de entrenamiento, ya que la varianza es alta en el conjunto de validación.

En resumen, un bajo sesgo y una baja varianza son muy deseables en un modelo de aprendizaje automático. En este caso, parece que el modelo tiene un bajo sesgo en todos los conjuntos, lo que sugiere que se adapta bien a los datos de entrenamiento.

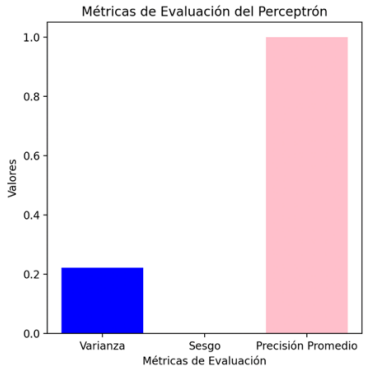
GRÁFICAS DE EVIDENCIA ACCURACY

Continuando con la explicación a detalle del código tenemos:

   b. Predicciones en Función de los Datos Originales: Se realiza un gráfico de dispersión que muestra las predicciones en función de las dos primeras características (característica 1

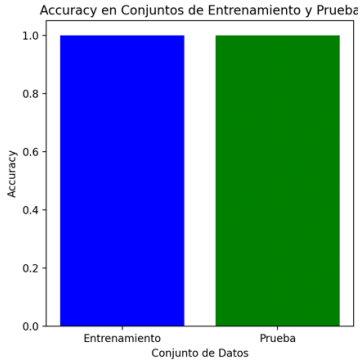
y característica 2) de los datos originales. Esto ayuda a visualizar cómo el modelo separa las clases.

c. Accuracy en Conjuntos de Entrenamiento y Prueba: Se muestra un gráfico de barras que compara la precisión del modelo en los conjuntos de entrenamiento y prueba. Esto ayuda a determinar si el modelo está sobreajustando (overfitting), subajustando (underfitting) o generalizando adecuadamente.



Gráfica 1.4

PerceptronIris.py

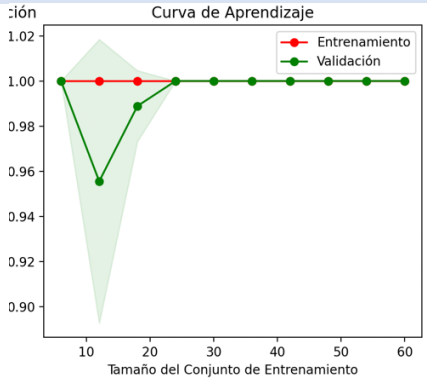


Gráfica 1.5

PerceptronIris.py

Como podemos observar el código es bastante bueno. Un accuracy de 1 (o 100%) significa que todas las predicciones del modelo son correctas, lo cual es el resultado ideal. Tenemos una congruencia bastante buena entre la precisión de los datos de prueba y los datos de entrenamiento. Además de eso podemos observar que no tenemos sesgo, y tenemos un 0.222 de varianza.

FITT UNDERFITTT U OVERFITT



Gráfica 1.6

Curva de aprendizaje

La curva de aprendizaje nos ayuda a evaluar el rendimiento de un modelo de machine learning en función del tamaño del conjunto de entrenamiento. Muestra los cambios en el rendimiento del modelo a medida que se agregan más ejemplos de entrenamiento.

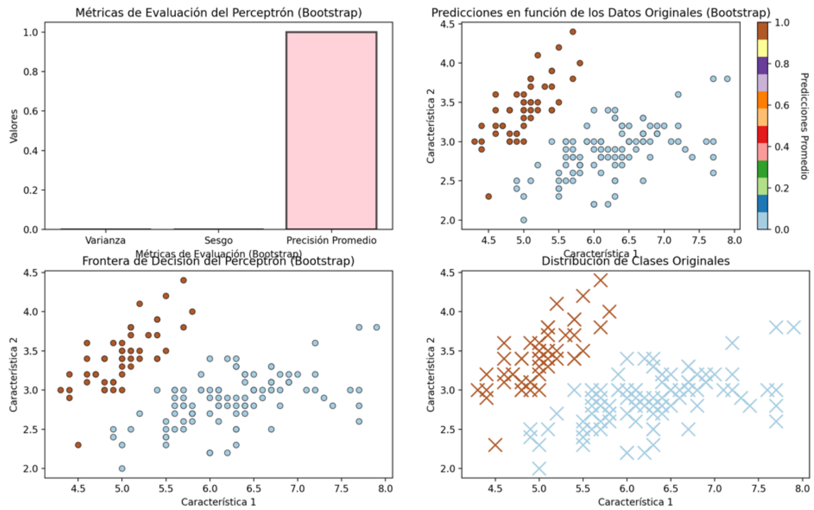
En nuestra curva de entrenamiento (Roja), el modelo se entrena con un conjunto de datos de entrenamiento de diferentes tamaños. Y no tenemos problema alguno pues se ve bien. Para la curva de validación (Verde) se muestra el rendimiento del modelo en un conjunto de datos de validación independiente. Inicialmente, el rendimiento mejora a medida que el modelo generaliza mejor con más datos de entrenamiento. Sin embargo, después de cierto punto, la curva de validación parece estabilizarse o incluso mostrar signos de degradación. Esto sugiere que agregar más datos de entrenamiento puede no mejorar significativamente la capacidad de generalización del modelo.

En resumen, el modelo es bueno pero no lo suficiente pues puede llegar a caer en redundancia y no llegar a generalizar del todo bien.

CÓDIGO MEJORADO

Fue bastante díficil mejora nuestros resultados, debido a la gran interpretación obtenida, sin embargo tomamos la decisión de hacer cambios puntuales y a continuación los describiré.
Primero implementamos un método que se denomina bootstrap en donde basicmanete lo que se hace es que toma muchas muestras (en est ecaso pusimos 100) aleatorias a partir del conjuitno de datos dado y empieza a entrenar sobre estos, de tal manera que se aplica el perceptron a cada una de las muestras y finalmente para la obtención de las métricas de evaluación se hace un promedio.
Además de eso añadí 3 claras diferencias a la hora de llamar a mi función de perceptron que son: definir el número de iteraciones a 1000, definir mi alpha con .001, y mi random state 42, número óptimo para la estandarización de la sálida.
Finalmente agregué lo que se conoce como una busqueda en cuadricula para encontrar los mejores hiperparámetros.

GRÁFICAS DE EVIDENCIA



Gráfica 2.

PerceptronIrisMejora.py

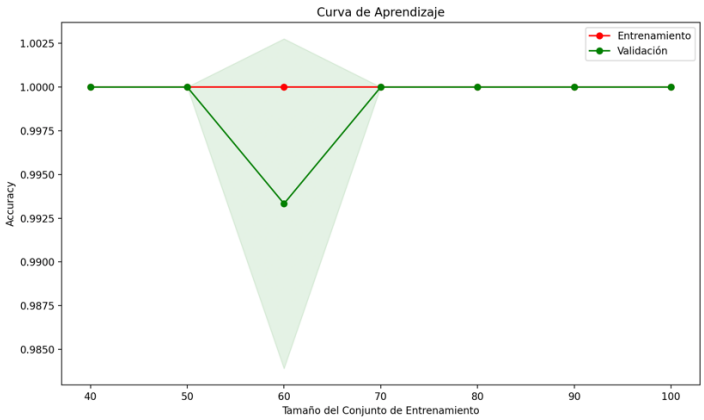
Podemos observar un comportamiento de manera muy clara y es que nuestra varianza se redujo sin embargo para entender exactamente qué paso decidí imprimir las varianzas de nuestras pruebas y obtenemos lo siguiente:

```
[0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.0564 0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.    ]
>>> print(np.mean(variance_bootstrap))
0.00037600000000000003
```

Imagen 2.1

"PerceptronIrisMejora. Py

Hubo una clara reducción en la varianza lo cual nos indica que los datos están saliendo muy cercanos e incluso iguales. Además de eso analizaremos a ocnitnuación la curva de aprendizaje, misma con la que podemos hacer una comparativa directa en la grafica del código anterior.



Gráfica 2.1

PerceptronIrisMejora.py

Curva de aprendizaje

La curva de aprendizaje muestra la evolución del rendimiento del modelo en el conjunto de entrenamiento (línea roja) y en el conjunto de validación (línea verde) a medida que se incrementa el tamaño del conjunto de entrenamiento. Al comienzo de la curva, tanto el rendimiento en el conjunto de entrenamiento como en el de validación son bajos. Esto sugiere que el modelo está teniendo dificultades para aprender del conjunto de datos, lo que indica subajuste (underfitting). En esta etapa, el modelo es demasiado simple para capturar lo complejo de los datos. A medida que se aumenta el tamaño del conjunto de entrenamiento, el rendimiento en el conjunto de entrenamiento mejora significativamente, mientras que el rendimiento en el conjunto de validación también mejora pero a un ritmo más lento. Esto indica que el modelo se está ajustando cada vez mejor al conjunto de entrenamiento, pero no generaliza tan bien en el conjunto de validación. Esto es un signo de sobreajuste (overfitting). Finalmente la curva de aprendizaje muestra que existe un punto donde el rendimiento en el conjunto de validación es óptimo.

CONCLUSIONES

El Perceptrón, a pesar de su simplicidad, es un algoritmo esencial en el aprendizaje automático, especialmente en tareas de clasificación binaria. Al explorar sus características clave, se pueden destacar los siguientes aspectos:

1. Facilidad de Uso: El Perceptrón se distingue por su sencillez. Opera realizando una combinación lineal de las entradas y aplicando una función de activación de tipo escalón para tomar decisiones de clasificación.

2. Limitaciones Fundamentales: Es importante reconocer que el Perceptrón presenta limitaciones considerables. Su capacidad se restringe a la resolución de problemas que pueden separarse linealmente, lo que significa que no puede manejar situaciones donde la división de clases requiere una frontera no lineal. Esto lo restringe en la resolución de problemas más complejos.

3. Impacto de la Tasa de Aprendizaje: La elección adecuada de la tasa de aprendizaje es esencial para garantizar que el Perceptrón converja de manera efectiva y aprenda patrones de datos de manera apropiada. Una tasa de aprendizaje muy alta puede impedir la convergencia, mientras que una tasa demasiado baja puede ralentizar el proceso de aprendizaje.

En resumen, el Perceptrón es un algoritmo de clasificación elemental pero crucial en el ámbito del aprendizaje automático, especialmente en la tarea de clasificación binaria. No obstante, es vital tener en cuenta sus restricciones, ya que solo puede abordar problemas con separación lineal, lo que lo hace insuficiente para tareas más complejas. Por ende, este algoritmo ha servido como punto de partida en el desarrollo de modelos de aprendizaje automático más avanzados, como las redes neuronales profundas, diseñadas para enfrentar desafíos más complejos y variados en el procesamiento de datos.

REFERENCIAS

1.- *Perceptrons - the most basic form of a neural network*. (2016, 9 junio). Applied Go. https://appliedgo.net/perceptron/

2.- González, L. (2022). ¿Qué es el perceptrón? Perceptrón simple y multicapa. 🤖 *Aprende IA*. https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/

3.- Romero, I. (2021, 4 marzo). *La dicotomía sesgo-varianza en modelos de machine learning - Keepler | Cloud Data Driven Partner*. Keepler | Cloud Data Driven Partner. https://keepler.io/es/2021/03/la-dicotomia-sesgo-varianza-en-modelos-de-machine-learning/

4.- López, J. F. (2022). Varianza. *Economipedia*. https://economipedia.com/definiciones/varianza.html

5.- Kostasmar. (2021). Exploring the Iris data set - Scikit-Learn. *Kaggle*. https://www.kaggle.com/code/kostasmar/exploring-the-iris-data-set-scikit-learn