Tecnológico de Monterrey

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Main Professor:

Jorge Adolfo Ramírez Uresti

Team members

~

Sebastian Jimenez Alcantara A01754993

Due date:

September 11th, 2024

Data Dictionary
link : https://www.kaggle.com/datasets/larsen0966/penguins

- **species**: Especie de pingüino, una variable categórica con tres posibles valores: Adélie, Chinstrap, Gentoo.
- **island**: Isla donde se encontró el pingüino, variable categórica con tres valores: Biscoe, Dream, Torgersen.
- **bill_length_mm**: Longitud del pico del pingüino en milímetros, un valor numérico.
- **bill_depth_mm**: Profundidad del pico del pingüino en milímetros, un valor numérico.
- **flipper_length_mm**: Longitud de la aleta en milímetros, es un valor entero.
- **body_mass_g**: Masa corporal del pingüino en gramos, un valor entero.
- **sex**: Sexo del pingüino, una variable categórica con dos valores: Male (Macho), Female (Hembra).
- **year**: Año en que se recolectaron los datos, es un número que representa el año.

Justificación del Dataset para el Algoritmo

Variables Claramente Definidas y Relevantes: El dataset contiene características (variables) bien definidas, como **bill_length_mm** (longitud del pico en milímetros) y **flipper_length_mm** (longitud de la aleta en milímetros). Estas características son biológicamente relevantes para distinguir entre diferentes especies de pingüinos, ya que las diferentes formas de estos son significativas entre especies, lo que hace que estas variables sean útiles para un algoritmo de clasificación.
Al reducir el problema a dos características clave, se simplifica el modelo sin perder poder predictivo, ya que estas dos variables probablemente capturen una parte importante de la variabilidad entre especies.

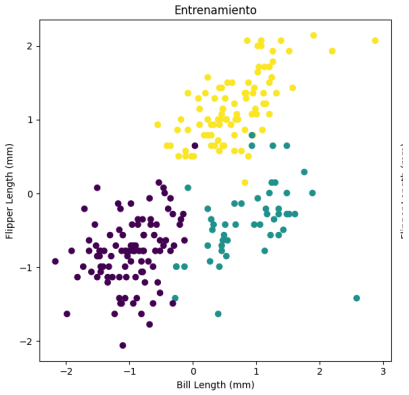
Algoritmo de Clasificación Adecuado: El algoritmo de redes neuronales es una buena elección, ya que puede manejar relaciones no lineales entre las características. Las diferencias en la longitud del pico y de la aleta entre especies no necesariamente siguen un patrón lineal, por lo que una red neuronal con capas ocultas puede aprender relaciones más complejas.
Además, la salida del modelo es categórica, lo que se ajusta bien a la tarea de predecir la especie de pingüino. El uso de una función de activación **softmax** en la última capa es ideal para problemas de clasificación multiclase como este.

Suficiente Variedad en las Clases (Multiclase): El objetivo es clasificar entre tres especies de pingüinos (Adélie, Chinstrap, Gentoo), lo que proporciona un desafío de clasificación multiclase. Los datos parecen un poco desequilibrados en términos de representación de clases, lo que no es lo óptimo para entrenar un modelo de machine learning lo que puede causar que esté sesgado hacia una clase particular.

Datos Continuos y Bien Estructurados: Las dos variables seleccionadas para el análisis son continuas y numéricas, lo cual es ideal para el algoritmo de redes neuronales. Los modelos de redes neuronales suelen manejar bien datos numéricos continuos cuando estos son escalados correctamente (lo que se hace con el **StandardScaler** en el

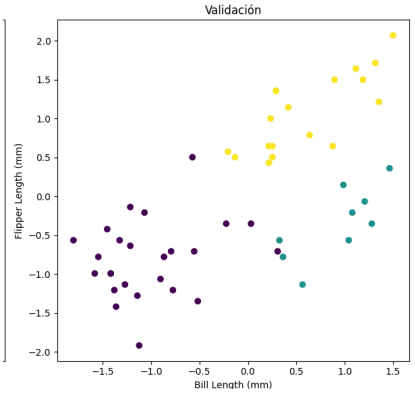
preprocesamiento). Este escalado también contribuye a la estabilidad y eficacia del proceso de optimización durante el entrenamiento del modelo.

Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation)



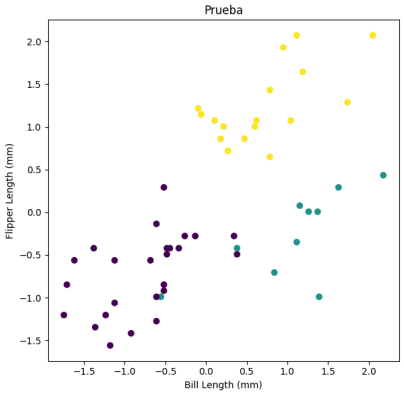
Gráfica 1: Datos de Entrenamiento

Esta gráfica muestra cómo están distribuidos los datos de entrenamiento. Los tres colores diferentes representan las tres clases de especies de pingüinos, y parece que hay una separación bastante clara entre las clases basadas en las dos variables (**bill_length_mm** y **flipper_length_mm**).



Gráfica 2: Datos de Validación

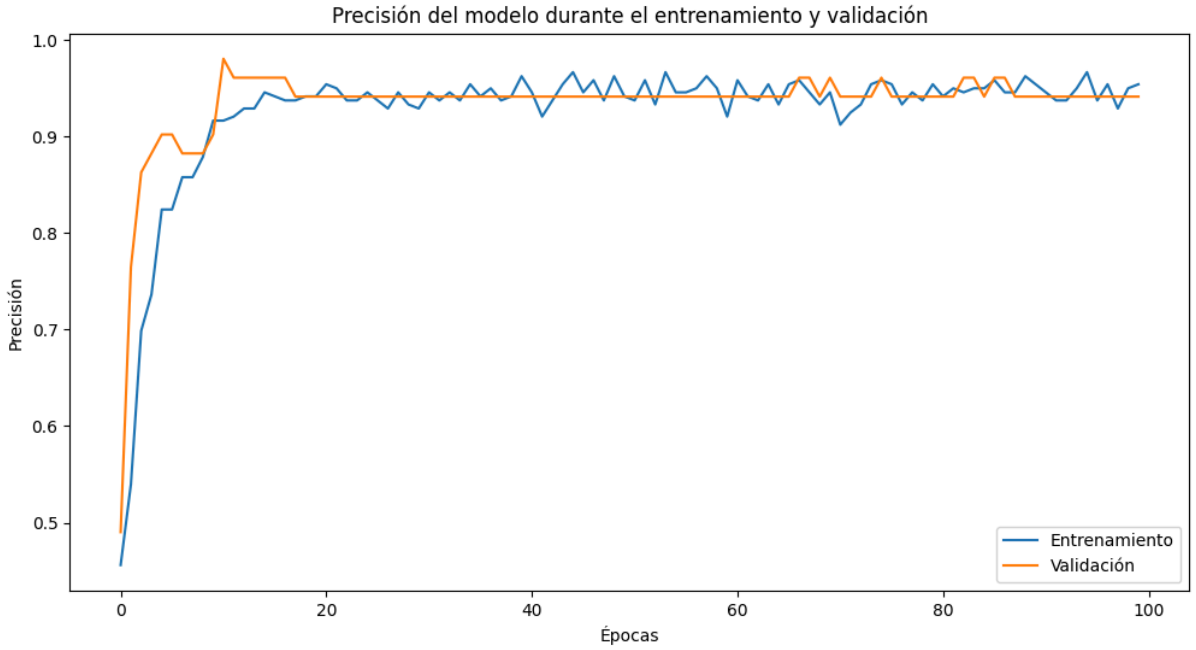
Esta gráfica muestra la distribución de los datos de validación, que se utilizan para monitorear el rendimiento del modelo durante el entrenamiento. Aquí también se observa una clara separación entre las clases.



Gráfica 3: Datos de Prueba

Esta gráfica muestra los datos de prueba, que se utilizan para medir la capacidad del modelo de generalizar en datos no vistos. Al igual que en las gráficas anteriores, las clases de pingüinos están bien separadas.

Las gráficas de dispersión en las tres etapas (entrenamiento, validación y prueba) indican que los datos están bastante bien distribuidos y separados, lo cual sugiere que el modelo puede funcionar bien para diferenciar entre las especies basadas en estas dos características.



Gráfica 4: Precisión del modelo durante el entrenamiento y validación

La precisión para el conjunto de **entrenamiento** (línea azul) comienza baja, pero se incrementa rápidamente y luego se estabiliza después de unas 20 épocas.

La precisión para el conjunto de **validación** (línea naranja) también aumenta rápidamente y sigue un patrón muy similar al conjunto de entrenamiento. Ambos se estabilizan alrededor del 95% de precisión.

Grado de sesgo (bias): Bajo

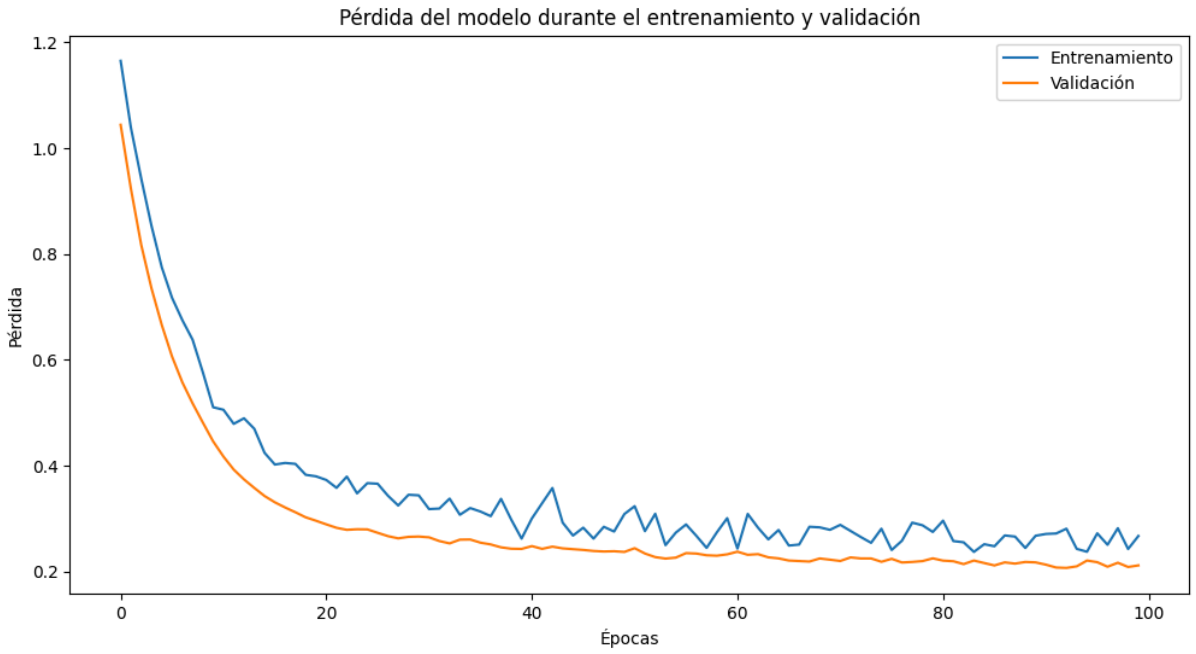
El **sesgo** se refiere a la capacidad del modelo de aprender los patrones en el conjunto de entrenamiento. En este caso, dado que la precisión en el entrenamiento es alta (cerca del 95% o más), podemos decir que el modelo ha aprendido bien los patrones y tiene un **sesgo bajo**.

Grado de varianza: Bajo

La **varianza** se refiere a cómo de bien el modelo generaliza a datos no vistos (conjunto de validación). Si hubiera una gran diferencia entre la precisión en entrenamiento y validación, eso indicaría una alta varianza (sobreajuste). Sin embargo, en este caso, las líneas de precisión en entrenamiento y validación son muy cercanas, lo que indica que el modelo generaliza bien y tiene una **varianza baja**.

Nivel de ajuste del modelo: Buen ajuste (fit)

El modelo no está subajustado (underfit) porque tiene una precisión alta en ambos conjuntos. Tampoco está sobreajustado (overfit) ya que las precisiones en entrenamiento y validación son muy similares. Por lo tanto, el modelo tiene un **buen ajuste** (fit).



Gráfica 4: Pérdida del modelo durante el entrenamiento y validación

La **pérdida** en el conjunto de **entrenamiento** (línea azul) comienza alta pero disminuye rápidamente hasta estabilizarse en niveles bajos.

La **pérdida** en el conjunto de **validación** (línea naranja) sigue un patrón similar, y después de unas 20 épocas, ambas pérdidas convergen y se estabilizan.

Grado de sesgo (bias): Bajo

Dado que la pérdida en el conjunto de entrenamiento es baja, el modelo ha aprendido correctamente los patrones de los datos, lo que indica un **sesgo bajo**. Un sesgo alto se manifestaría en una pérdida alta en el conjunto de entrenamiento, lo cual no ocurre aquí.

Grado de varianza: Bajo

La pérdida en el conjunto de validación es muy similar a la del conjunto de entrenamiento, lo que indica que el modelo no está sobreajustado. Un modelo con alta varianza mostraría una diferencia notable entre la pérdida de entrenamiento y validación. Aquí, como las pérdidas son similares, el modelo tiene una **varianza baja**.

Nivel de ajuste del modelo: Buen ajuste (fit)

Las pérdidas en ambos conjuntos son bajas y similares, lo que indica que el modelo ha aprendido bien los patrones en los datos y puede generalizar correctamente. Esto significa que el modelo tiene un **ajuste adecuado (fit)**, y no presenta signos de subajuste o sobreajuste.

Uso de técnicas de regularización o ajuste de parámetros para mejorar el desempeño

Regularización L2

Descripción: En las capas ocultas, se ha aplicado la regularización **L2** mediante el argumento **`kernel_regularizer=l2(0.01)`** en la definición de las capas. La regularización L2 agrega una penalización en los pesos grandes durante el entrenamiento. Esto ayuda a reducir la complejidad del modelo al intentar minimizar los pesos demasiado grandes, lo que evita que el modelo dependa demasiado de características específicas del conjunto de entrenamiento.

Efecto: La regularización L2 previene el sobreajuste al penalizar los pesos excesivamente grandes y hacer que el modelo generalice mejor.

```
model.add(Dense(16, input_dim=2, activation='relu',
kernel_regularizer=l2(0.01))) model.add(Dense(12, activation='relu',
kernel_regularizer=l2(0.01)))
```

Dropout

Descripción: Dropout es una técnica de regularización que apaga aleatoriamente un porcentaje de neuronas en una capa durante el entrenamiento. En este caso, se aplica **Dropout con un 20% de neuronas desactivadas** en las capas ocultas. Esto evita que el modelo dependa demasiado de un subconjunto específico de neuronas, lo que mejora la capacidad de generalización.

Efecto: Dropout ayuda a prevenir el sobreajuste al evitar que el modelo memorice patrones específicos de los datos de entrenamiento.

```
model.add(Dropout(0.2))  # Dropout del 20%
```

Ajuste de hiperparámetros (Batch size y Épocas)

Descripción: En el entrenamiento, se han ajustado los hiperparámetros clave como el **batch size** y el número de **épocas**:

Batch size: Se ha establecido en 10, lo que controla el número de muestras que el modelo procesa antes de actualizar los pesos. Un tamaño de lote más pequeño puede hacer que el modelo aprenda más rápido pero con fluctuaciones más grandes en el gradiente, mientras que un tamaño mayor hace que el aprendizaje sea más estable pero más lento.

Épocas: El número de épocas se ha establecido en 100, lo que determina cuántas veces el modelo verá los datos completos durante el entrenamiento. Un número de épocas adecuado permite que el modelo aprenda sin sobreentrenarse.

Efecto: El ajuste del **batch size** y el número de **épocas** permite que el modelo encuentre un equilibrio entre la velocidad de entrenamiento y la precisión. Más épocas le permiten aprender más, pero también aumentan el riesgo de sobreajuste, mitigado en parte por el Dropout y la regularización L2.

```
history = model.fit(X_train, y_train, epochs=100, batch_size=10,
verbose=1, validation_data=(X_val, y_val))
```

Conclusiones

El uso del dataset de características de los pingüinos y la combinación de técnicas de regularización, como la regularización L2 y Dropout, junto con el ajuste de hiperparámetros, han permitido que el modelo alcance un buen ajuste. El modelo muestra un sesgo bajo y una varianza baja, lo que significa que ha aprendido bien los patrones sin sobreajustarse, y es capaz de generalizar correctamente a nuevos datos. Estas técnicas han sido claves para

evitar tanto el subajuste como el sobreajuste, obteniendo un equilibrio adecuado entre el aprendizaje y la generalización.