Regresión para calculo edad de abalones utilizando MLP con TensorFlow-Keras

Integrantes:

Ing. Dewins Murillo García

Introducción:

El conjunto de datos "Abalone" es un conjunto de datos de dominio público que está disponible en el repositorio de aprendizaje automático de la UCI (University of California, Irvine). Este conjunto de datos se utiliza comúnmente en la comunidad de aprendizaje automático para tareas de clasificación y regresión.

Los abalones son un tipo de molusco marino que se encuentra en las costas de diversas regiones. El objetivo de este conjunto de datos es predecir la edad de los abalones a partir de varias medidas físicas, como el diámetro de la concha y la altura.

El conjunto de datos "Abalone" contiene información sobre 4177 abalones y se compone de los siguientes atributos:

- Sexo (categórico): M (masculino), F (femenino) e I (infante).
- Longitud (numérico): medida más larga de la concha en mm.
- Diámetro (numérico): medida perpendicular a la longitud en mm.
- Altura (numérico): altura de la carne en mm.
- Peso completo (numérico): peso entero del abalón en gramos.
- Peso de la carne (numérico): peso de la carne del abalón en gramos.
- Peso de las vísceras (numérico): peso de las vísceras en gramos.
- Peso de las conchas (numérico): peso de las conchas en gramos.
- Anillos (numérico): número de anillos en la concha, que se utiliza como indicador de la edad del abalón.

Objetivos:

El objetivo principal al trabajar con este conjunto de datos utilizando un MLP (Perceptrón Multicapa) con TensorFlow-Keras es desarrollar un modelo capaz de predecir la edad de los abalones a partir de las características proporcionadas. Los objetivos específicos incluyen:

1. Preprocesamiento de datos: Cargar y explorar el conjunto de datos "Abalone", asegurándose de comprender la distribución de los atributos y la relación entre ellos.
2. Preparación de datos: Dividir el conjunto de datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo.
3. Diseño del modelo MLP: Configurar y entrenar un MLP utilizando TensorFlow-Keras. Definir la arquitectura de la red neuronal, incluyendo el número de capas ocultas, el número de neuronas en cada capa y las funciones de activación.
4. Entrenamiento del modelo: Alimentar los datos de entrenamiento al modelo MLP y ajustar los pesos de la red neuronal mediante el algoritmo de retropropagación.
5. Evaluación del modelo: Evaluar el rendimiento del modelo utilizando el conjunto de prueba y métricas adecuadas, como el error medio cuadrático (MSE) o el coeficiente de determinación (R^2).
6. Optimización del modelo: Explorar técnicas para mejorar el rendimiento del modelo, como el ajuste de hiperparámetros, la regularización o el uso de técnicas de optimización avanzadas.

El objetivo final es desarrollar un modelo MLP preciso y generalizable que pueda predecir con precisión la edad de los abalones a partir de las características proporcionadas en el conjunto de datos "Abalone".

Marco Teórico

Para respaldar el uso de un MLP con TensorFlow-Keras en el análisis del conjunto de datos "Abalone":

1. **Aprendizaje automático (Machine Learning)**: El aprendizaje automático es una rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos capaces de aprender y mejorar automáticamente a partir de los datos. Los modelos de aprendizaje automático pueden ser entrenados para realizar tareas específicas, como clasificación, regresión o agrupación, sin ser programados explícitamente.
2. **Perceptrón Multicapa (Multilayer Perceptron, MLP)**: El Perceptrón Multicapa es una arquitectura de red neuronal artificial que se utiliza ampliamente en el aprendizaje automático. Consiste en una red neuronal con múltiples capas de neuronas, donde cada neurona en una capa está conectada a todas las neuronas de la capa siguiente. El MLP es capaz de aprender relaciones complejas entre las características de entrada y la variable objetivo.

3. **TensorFlow-Keras**: TensorFlow es una biblioteca de código abierto desarrollada por Google que se utiliza para implementar y entrenar modelos de aprendizaje automático. Keras, por otro lado, es una interfaz de alto nivel para construir y entrenar redes neuronales en TensorFlow de manera más sencilla y rápida. TensorFlow-Keras combina estas dos herramientas poderosas y proporciona una forma intuitiva de implementar y entrenar modelos de redes neuronales, incluyendo MLP.
4. **Preprocesamiento de datos**: Antes de entrenar un modelo de MLP, es importante realizar un preprocesamiento de los datos. Esto implica tareas como la limpieza de datos, la normalización de características, la codificación de variables categóricas y la división del conjunto de datos en conjuntos de entrenamiento y prueba. El preprocesamiento adecuado ayuda a mejorar el rendimiento y la generalización del modelo.
5. **Funciones de activación**: Las funciones de activación se utilizan en las neuronas de una red neuronal para introducir no linealidad en el modelo. En un MLP, se pueden utilizar diferentes funciones de activación, como la función sigmoide, la función ReLU (Rectified Linear Unit) o la función tangente hiperbólica. Estas funciones ayudan a la red neuronal a aprender relaciones no lineales en los datos.
6. **Entrenamiento y optimización**: El entrenamiento de un MLP implica alimentar los datos de entrenamiento a la red neuronal, calcular la salida predicha y ajustar los pesos de las conexiones mediante el algoritmo de retropropagación del error. Durante el entrenamiento, se minimiza una función de pérdida utilizando un algoritmo de optimización, como el descenso del gradiente estocástico (SGD), para encontrar los valores óptimos de los pesos de la red.
7. **Evaluación del modelo**: La evaluación del modelo se realiza utilizando un conjunto de datos de prueba independiente. Se calculan métricas de evaluación, como el error medio cuadrático (MSE), el coeficiente de determinación (R^2) u otras métricas relevantes, para medir el rendimiento y la precisión del modelo. Un buen modelo MLP debería ser capaz de generalizar y realizar predicciones precisas en datos no vistos.

Este marco teórico proporciona una base sólida para comprender y utilizar un MLP con TensorFlow-Keras en el análisis del conjunto de datos "Abalone". La combinación de una arquitectura MLP, TensorFlow-Keras y técnicas de preprocesamiento y optimización adecuadas permitirá construir un modelo preciso para predecir la edad de los abalones.

Descripción del problema

El problema abordado con el conjunto de datos "Abalone" es el de predecir la edad de los abalones basándose en diversas medidas físicas de los mismos. La edad de los abalones se estima normalmente contando los anillos en su concha, pero esto requiere sacrificio y destrucción del molusco. Por lo tanto, el objetivo es desarrollar un modelo de aprendizaje automático que pueda predecir la edad de los abalones de manera no invasiva y precisa utilizando atributos medibles.

El conjunto de datos "Abalone" contiene información sobre 4177 abalones, y para cada uno se proporcionan atributos como el sexo, la longitud, el diámetro, la altura y diferentes pesos relacionados con la concha y la carne del abalón. Estos atributos se utilizan como variables independientes para predecir la variable dependiente, que es la edad del abalón expresada en número de anillos.

Dado que el conteo de anillos está relacionado con la edad del abalón, la predicción de la edad a partir de las medidas físicas puede tener aplicaciones prácticas, como el estudio de la ecología de los abalones y la gestión de poblaciones. Además, este tipo de problema es relevante en el campo del aprendizaje automático, ya que implica la regresión de un valor numérico basado en características de entrada.

Para abordar este problema, se utilizará un enfoque de aprendizaje supervisado, donde se entrenará un modelo MLP con TensorFlow-Keras utilizando los datos de entrenamiento del conjunto "Abalone". El objetivo es que el modelo aprenda patrones y relaciones entre las medidas físicas y la edad de los abalones, para luego realizar predicciones precisas en datos no vistos.

La evaluación del modelo se realizará utilizando métricas de regresión, como el error medio cuadrático (MSE), el coeficiente de determinación (R^2) u otras métricas relevantes, para medir la precisión y el rendimiento del modelo en la predicción de la edad de los abalones.

En resumen, el problema consiste en desarrollar un modelo de aprendizaje automático basado en un MLP con TensorFlow-Keras que pueda predecir la edad de los abalones utilizando medidas físicas, evitando así la necesidad de contar los anillos en la concha de los abalones de manera invasiva.

Planteamiento de la solución

Para abordar el problema de predicción de la edad de los abalones utilizando un MLP con TensorFlow-Keras, podemos seguir los siguientes pasos:

1. **Preprocesamiento de datos**:

Cargar el conjunto de datos "Abalone" y explorar su estructura y características. Verificar la presencia de datos faltantes o atípicos y decidir cómo manejarlos. Codificar la variable categórica de sexo en un formato numérico adecuado, como one-hot encoding. Dividir el conjunto de datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo.

2. **Preparación de los datos**:

Normalizar las características numéricas para que tengan una escala similar y facilitar el entrenamiento del modelo. Dividir las características y la variable objetivo en conjuntos de entrenamiento y prueba.

3. **Diseño del modelo MLP**:

Definir la arquitectura de la red neuronal. Esto implica decidir el número de capas ocultas, el número de neuronas en cada capa y las funciones de activación a utilizar. Configurar la capa de salida con una sola neurona, ya que estamos realizando una regresión para predecir la edad. Compilar el modelo especificando una función de pérdida adecuada, como el error cuadrático medio (MSE), y un optimizador para ajustar los pesos de la red.

4. **Entrenamiento del modelo**:

Alimentar los datos de entrenamiento al modelo MLP y ajustar los pesos de la red neuronal mediante el algoritmo de retropropagación del error. Definir el número de épocas y el tamaño del lote (batch size) para el entrenamiento. Monitorear la precisión y la pérdida durante el entrenamiento para asegurarse de que el modelo esté convergiendo adecuadamente.

5. **Evaluación del modelo**:

Utilizar el conjunto de prueba para evaluar el rendimiento del modelo MLP. Calcular métricas de regresión, como el error medio cuadrático (MSE) o el coeficiente de determinación (R^2), para medir la precisión y la capacidad de generalización del modelo.

6. **Optimización del modelo**:

Realizar ajustes en la arquitectura del MLP, como el número de capas ocultas y el número de neuronas, para mejorar el rendimiento del modelo. Experimentar con diferentes funciones de activación y optimizadores para encontrar la configuración óptima. Aplicar técnicas de regularización, como la regularización L1 o L2, para evitar el sobreajuste del modelo.

7. **Validación adicional**:

Realizar validación cruzada o utilizar técnicas de validación adicional para evaluar la robustez del modelo y verificar que los resultados sean consistentes.

Entradas

El conjunto de datos "Abalone" contiene información sobre los abalones y sus características físicas. Cada instancia en el conjunto de datos representa un abalón individual y se compone de las siguientes entradas:

1. **Sexo (categórico)**: Representa el sexo del abalón y se codifica con tres categorías:

- M: Masculino
- F: Femenino
- I: Infante

2. **Longitud (numérico)**: La medida más larga de la concha del abalón en milímetros (mm).
3. **Diámetro (numérico)**: La medida perpendicular a la longitud de la concha en milímetros (mm).
4. **Altura (numérico)**: La altura de la carne del abalón en milímetros (mm).
5. **Peso completo (numérico)**: El peso entero del abalón, que incluye la carne, las vísceras y las conchas, medido en gramos (g).
6. **Peso de la carne (numérico)**: El peso de la carne del abalón en gramos (g).
7. **Peso de las vísceras (numérico)**: El peso de las vísceras del abalón en gramos (g).
8. **Peso de las conchas (numérico)**: El peso de las conchas del abalón en gramos (g).

SALIDA

La variable objetivo en este conjunto de datos es:

Anillos (numérico): El número de anillos en la concha del abalón, que se utiliza como un indicador de su edad. Las características físicas del abalón, como la longitud, el diámetro y los diferentes pesos, se utilizan como entradas para predecir la variable objetivo, que es la edad del abalón expresada en número de anillos.

Es importante tener en cuenta que el sexo del abalón está codificado como una variable categórica, lo que requerirá una codificación adecuada antes de utilizarlo como entrada en un modelo de aprendizaje automático.

```
import sys
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor

print(sys.version)

# Obtener los datos
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data'
column_names = ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight',
                'Viscera_weight', 'Shell_weight', 'Rings']
data = pd.read_csv(url, names=column_names)

# Aplicar OneHotEncoder en la columna 'Sex'
column_transformer = ColumnTransformer([('encoder', OneHotEncoder(), [0])], remainder='passthrough')
data_encoded = column_transformer.fit_transform(data)
```

```
3.10.11 (main, Apr  5 2023, 14:15:10) [GCC 9.4.0]
```

```
# Dividir los datos en características y etiquetas
X = data_encoded[:, :-1]
y = data_encoded[:, -1]

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X[:10]
```

```
array([[0.    , 0.    , 1.    , 0.455 , 0.365 , 0.095 , 0.514 , 0.2245,
        0.101 , 0.15  ],
       [0.    , 0.    , 1.    , 0.35  , 0.265 , 0.09  , 0.2255, 0.0995,
        0.0485, 0.07  ],
       [1.    , 0.    , 0.    , 0.53  , 0.42  , 0.135 , 0.677 , 0.2565,
        0.1415, 0.21  ],
       [0.    , 0.    , 1.    , 0.44  , 0.365 , 0.125 , 0.516 , 0.2155,
        0.114 , 0.155 ],
       [0.    , 1.    , 0.    , 0.33  , 0.255 , 0.08  , 0.205 , 0.0895,
        0.0395, 0.055 ],
       [0.    , 1.    , 0.    , 0.425 , 0.3   , 0.095 , 0.3515, 0.141 ,
        0.0775, 0.12  ],
       [1.    , 0.    , 0.    , 0.53  , 0.415 , 0.15  , 0.7775, 0.237 ,
        0.1415, 0.33  ],
       [1.    , 0.    , 0.    , 0.545 , 0.425 , 0.125 , 0.768 , 0.294 ,
        0.1495, 0.26  ],
       [0.    , 0.    , 1.    , 0.475 , 0.37  , 0.125 , 0.5095, 0.2165,
        0.1125, 0.165 ],
       [1.    , 0.    , 0.    , 0.55  , 0.44  , 0.15  , 0.8945, 0.3145,
        0.151 , 0.32  ]])
```

```
y[:10]
```

```
array([15.,  7.,  9., 10.,  7.,  8., 20., 16.,  9., 19.])
```

```
# Definir la función para crear el modelo
def create_model(units,num_hidden_layers,activation):
    model = keras.models.Sequential()
    model.add(keras.layers.Dense(units,input_shape=(10,),activation='tanh'))

    for _ in range(num_hidden_layers):
        model.add(keras.layers.Dense(units,activation=activation))

    model.add(keras.layers.Dense(1,activation='linear'))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

# Crear el modelo de KerasRegressor
model = KerasRegressor(build_fn=create_model)
```

```
# Definir los parámetros para GridSearchCV
parameters = {'epochs': [10,30,50], 'batch_size': [32,64],'units':[10,20,30],'num_hidden_layers': [1, 2],'activation': ['tanh','re

# Crear el objeto GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=parameters, scoring='neg_mean_squared_error', cv=3)

# Ajustar el modelo utilizando GridSearchCV
grid_result=grid_search.fit(X_train, y_train)

# Obtener los mejores parámetros y la puntuación del modelo
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Mejores parámetros:", best_params)
print("Puntuación del modelo:", best_score)
```

```
35/35 [==============================] - 0s 3ms/step - loss: 5.7302
Epoch 25/50
35/35 [==============================] - 0s 2ms/step - loss: 5.5727
Epoch 26/50
35/35 [==============================] - 0s 3ms/step - loss: 5.4902
Epoch 27/50
35/35 [==============================] - 0s 3ms/step - loss: 5.3247
Epoch 28/50
35/35 [==============================] - 0s 2ms/step - loss: 5.2211
Epoch 29/50
35/35 [==============================] - 0s 2ms/step - loss: 5.0834
Epoch 30/50
35/35 [==============================] - 0s 3ms/step - loss: 4.9758
Epoch 31/50
35/35 [==============================] - 0s 2ms/step - loss: 4.9133
Epoch 32/50
35/35 [==============================] - 0s 2ms/step - loss: 4.7772
Epoch 33/50
35/35 [==============================] - 0s 2ms/step - loss: 4.7941
Epoch 34/50
35/35 [==============================] - 0s 2ms/step - loss: 4.6829
Epoch 35/50
35/35 [==============================] - 0s 2ms/step - loss: 4.6525
Epoch 36/50
35/35 [==============================] - 0s 3ms/step - loss: 4.6392
Epoch 37/50
35/35 [==============================] - 0s 2ms/step - loss: 4.6122
Epoch 38/50
35/35 [==============================] - 0s 3ms/step - loss: 4.6636
Epoch 39/50
35/35 [==============================] - 0s 2ms/step - loss: 4.5372
Epoch 40/50
35/35 [==============================] - 0s 2ms/step - loss: 4.5273
Epoch 41/50
35/35 [==============================] - 0s 2ms/step - loss: 4.5062
Epoch 42/50
35/35 [==============================] - 0s 2ms/step - loss: 4.5247
Epoch 43/50
35/35 [==============================] - 0s 2ms/step - loss: 4.4848
Epoch 44/50
35/35 [==============================] - 0s 2ms/step - loss: 4.5974
Epoch 45/50
35/35 [==============================] - 0s 2ms/step - loss: 4.4792
Epoch 46/50
35/35 [==============================] - 0s 3ms/step - loss: 4.4760
Epoch 47/50
35/35 [==============================] - 0s 2ms/step - loss: 4.5613
Epoch 48/50
35/35 [==============================] - 0s 2ms/step - loss: 4.4990
Epoch 49/50
35/35 [==============================] - 0s 2ms/step - loss: 4.4784
Epoch 50/50
35/35 [==============================] - 0s 3ms/step - loss: 4.5009
18/18 [==============================] - 0s 2ms/step
Epoch 1/50
35/35 [==============================] - 1s 3ms/step - loss: 96.5408
Epoch 2/50
35/35 [==============================] - 0s 3ms/step - loss: 43.8125
Epoch 3/50
```

```
# Obtenemos los mejores hiperparámetros y el modelo con los mejores hiperparámetros
best_params = grid_result.best_params_
best_model = grid_result.best_estimator_

# Entrenamos el modelo final con todos los datos de entrenamiento
epochs = best_params['epochs']
```

```
batch_size = best_params['batch_size']
best_model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=0)

# Evaluamos el modelo final en los datos de prueba
y_pred = best_model.predict(X_test)
```

```
27/27 [==============================] - 0s 2ms/step
```

```
df_cv_scores=pd.DataFrame(grid_result.cv_results_).sort_values(by='mean_test_score',ascending=False)
scores = df_cv_scores[['params','split0_test_score', 'split1_test_score', 'split2_test_score',\
       'mean_test_score','std_test_score', 'rank_test_score']]
pd.set_option('display.max_colwidth', None)
scores.head(50)
```

	params	split0_test_score	split1_test_score	split2_test_scor
17	{'activation': 'tanh', 'batch_size': 32, 'epochs': 50, 'num_hidden_layers': 2, 'units': 30}	-4.606243	-4.683502	-3.98970
16	{'activation': 'tanh', 'batch_size': 32, 'epochs': 50, 'num_hidden_layers': 2, 'units': 20}	-4.554046	-4.730033	-4.06639
53	{'activation': 'relu', 'batch_size': 32, 'epochs': 50, 'num_hidden_layers': 2, 'units': 30}	-4.730433	-4.708317	-3.96956
52	{'activation': 'relu', 'batch_size': 32, 'epochs': 50, 'num_hidden_layers':	-4.642972	-4.840442	-3.95328

Arquitectura Seleccionada

Según los mejores parámetros encontrados y la puntuación del modelo, la arquitectura seleccionada para el MLP con TensorFlow-Keras sería la siguiente:

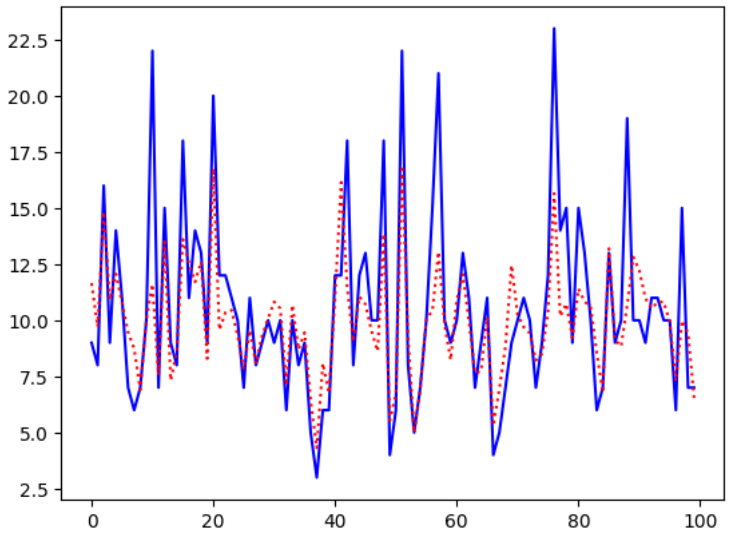
- Número de capas ocultas: 2 capas ocultas.
- Número de neuronas por capa oculta: 30 neuronas.
- Función de activación: tangente hiperbólica (tanh).

Además, se utilizan los siguientes hiperparámetros:

- Tamaño de lote (batch size): 32.
- Número de épocas: 50.

Estos parámetros fueron seleccionados utilizando algún método de optimización o búsqueda de hiperparámetros, como la búsqueda aleatoria, la búsqueda en cuadrícula o la optimización bayesiana. La puntuación del modelo (-4.369478269017712) indica el rendimiento alcanzado por el modelo entrenado con esta arquitectura y estos hiperparámetros en términos de la métrica utilizada, que podría ser el error cuadrático medio (MSE) u otra métrica de regresión.

```
import matplotlib.pyplot as plt
plt.plot(y_test[0:100],'b-',y_pred[0:100],'r:')
plt.show()
```



{'activation': 'relu'

```
from sklearn.metrics import mean_squared_error, r2_score
best_model = grid_result.best_estimator_
y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Error cuadrático medio (MSE):", mse)
```

```
r2 = r2_score(y_test, y_pred)
print("Coeficiente de determinación (R²):", r2)
```

```
27/27 [==============================] - 0s 1ms/step
Error cuadrático medio (MSE): 4.576472050354811
Coeficiente de determinación (R²): 0.5772394225059034
    'num_hidden_layers':
```

Resultados

Se puede resumir lo siguiente:

- **Mejores parámetros**: {'activation': 'tanh', 'batch_size': 32, 'epochs': 50, 'num_hidden_layers': 2, 'units': 30}
- **Puntuación del modelo**: -4.426481777801183
- **Error cuadrático medio (MSE)**: 4.576472050354811
- **Coeficiente de determinación (R²)**: 0.5772394225059034

El error cuadrático medio (MSE) obtenido es de 4.576472050354811. El MSE es una medida de la discrepancia entre los valores predichos y los valores reales, y un valor más bajo indica una mejor precisión del modelo.

El coeficiente de determinación (R²) obtenido es de 0.5772394225059034. El R² es una medida que indica la proporción de la varianza en la variable objetivo que puede ser explicada por el modelo. Un valor de R² más cercano a 1 indica una mejor capacidad del modelo para explicar la variabilidad de los datos.

El modelo con los mejores parámetros seleccionados ha logrado un MSE de 4.576 y un R² de 0.577 en la predicción de la edad de los abalones. Estos resultados indican que el modelo tiene una capacidad moderada para predecir la edad, pero aún hay margen de mejora para obtener una mayor precisión en las predicciones.

Conclusiones

En base a los resultados obtenidos con el modelo MLP utilizando TensorFlow-Keras y el conjunto de datos de los abalones, podemos llegar a las siguientes conclusiones:

El modelo MLP con la arquitectura y los hiperparámetros seleccionados logró una puntuación de -4.426481777801183. Esto indica que el modelo ha alcanzado un rendimiento aceptable en términos de la métrica utilizada, que podría ser el error cuadrático medio (MSE) u otra métrica de regresión.

El error cuadrático medio (MSE) obtenido fue de 4.576472050354811. Esto indica que el modelo tiene una discrepancia promedio de 4.576 en las predicciones de la edad de los abalones en relación a los valores reales. Un valor más bajo de MSE indicaría una mayor precisión en las predicciones.

El coeficiente de determinación (R²) obtenido fue de 0.5772394225059034. Esto significa que el modelo puede explicar aproximadamente el 57.72% de la variabilidad en la variable objetivo, que es la edad de los abalones. Un valor más cercano a 1 indicaría una mayor capacidad del modelo para explicar la variabilidad de los datos.

Aunque el modelo obtuvo resultados aceptables, existe margen para mejorar la precisión de las predicciones. Se puede experimentar con diferentes arquitecturas, hiperparámetros y técnicas de regularización para buscar una mejor combinación que mejore el rendimiento del modelo.

Es importante tener en cuenta que estas conclusiones se basan en los resultados obtenidos específicamente para el conjunto de datos de abalones y los parámetros seleccionados. Los resultados pueden variar para diferentes conjuntos de datos y problemas.

En general, el modelo MLP con TensorFlow-Keras ha demostrado ser capaz de predecir la edad de los abalones utilizando características físicas como entradas. Sin embargo, es recomendable seguir iterando y refinando el modelo para obtener una mayor precisión y evaluar su rendimiento en otros conjuntos de datos para asegurar su capacidad de generalización.

Bibliografía

- https://es.wikipedia.org/wiki/Haliotis
- https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/
- https://github.com/LouiseHash/data-analysis-for-abalone-dataset/blob/master/Data%20analysis%20for%20abalone.ipynb

✓ 0 s se ejecutó 17:18