elevator

March 18, 2023

[1]:
```
import pandas as pd
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras import models
from sklearn.metrics import r2_score
```

[2]:
```
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from xgboost import XGBRegressor
```

[3]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

[4]:
```
# to ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

0.0.1 With the help of Pandas, read the ".csv" file and performing some task

[5]:
```
data1 = pd.read_csv("elevators.csv")
```

[6]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16599 entries, 0 to 16598
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   ClimbRate     16599 non-null  int64
 1   Sgz           16599 non-null  int64
 2   P             16599 non-null  float64
 3   Q             16599 non-null  float64
 4   CurRoll       16599 non-null  float64
 5   AbsRoll       16599 non-null  int64
 6   DiffClb       16599 non-null  int64
 7   DiffRollRate  16599 non-null  float64
 8   DiffDiffClb   16599 non-null  float64
 9   SaTime1       16599 non-null  float64
 10  SaTime2       16599 non-null  float64
 11  SaTime3       16599 non-null  float64
 12  SaTime4       16599 non-null  float64
 13  DiffSaTime1   16599 non-null  float64
 14  DiffSaTime2   16599 non-null  float64
 15  DiffSaTime3   16599 non-null  float64
 16  DiffSaTime4   16599 non-null  float64
 17  Sa            16599 non-null  float64
 18  Goal          16599 non-null  float64
dtypes: float64(15), int64(4)
memory usage: 2.4 MB
```

[7]:
```
data1.head(4)
```

[7]:
```
   ClimbRate  Sgz     P     Q  CurRoll  AbsRoll  DiffClb  DiffRollRate  \
0       -178   40 -0.11  0.13      1.1       -9      -12        -0.011
1       -122    9  0.27  0.05      0.0      -10       -2        -0.005
2        196  -10 -0.44  0.10      0.6      -10       -7         0.003
3        507   -6  0.14  0.10     -0.2      -10        2         0.001

   DiffDiffClb  SaTime1  SaTime2  SaTime3  SaTime4  DiffSaTime1  DiffSaTime2  \
0          0.0  -0.0007  -0.0007  -0.0007  -0.0007       0.0000          0.0
1          0.4  -0.0007  -0.0007  -0.0007  -0.0007       0.0000          0.0
2          0.2  -0.0006  -0.0006  -0.0006  -0.0006       0.0000          0.0
3         -0.7  -0.0007  -0.0006  -0.0006  -0.0006      -0.0001          0.0

   DiffSaTime3  DiffSaTime4      Sa   Goal
0          0.0          0.0 -0.0007  0.018
1          0.0          0.0 -0.0007  0.017
2          0.0          0.0 -0.0006  0.021
3          0.0          0.0 -0.0006  0.024
```

[8]:
```
data1.tail(4)
```

[8]:
```
       ClimbRate  Sgz     P     Q  CurRoll  AbsRoll  DiffClb  DiffRollRate  \
16595         12  -31  0.20 -0.08     -0.3      -13       12        -0.003
16596       -243  -23  0.08 -0.11     -0.7       -7       13         0.007
16597       -226   15 -0.17 -0.01     -0.5       -7        6         0.001
16598       -193   -2 -0.36  0.04     -0.8      -16      -11        -0.026

       DiffDiffClb  SaTime1  SaTime2  SaTime3  SaTime4  DiffSaTime1  \
16595          1.3  -0.0008  -0.0008  -0.0008  -0.0008          0.0
16596         -0.2  -0.0005  -0.0005  -0.0005  -0.0005          0.0
16597         -0.8  -0.0005  -0.0005  -0.0005  -0.0005          0.0
```

```
16598          3.6  -0.0014  -0.0014  -0.0014  -0.0014          0.0

       DiffSaTime2  DiffSaTime3  DiffSaTime4      Sa   Goal
16595          0.0          0.0          0.0 -0.0008  0.017
16596          0.0          0.0          0.0 -0.0005  0.033
16597          0.0          0.0          0.0 -0.0005  0.016
16598          0.0          0.0          0.0 -0.0014  0.018
```

[9]:
```
data1.isnull().any()
```

[9]:
```
ClimbRate       False
Sgz             False
P               False
Q               False
CurRoll         False
AbsRoll         False
DiffClb         False
DiffRollRate    False
DiffDiffClb     False
SaTime1         False
SaTime2         False
SaTime3         False
SaTime4         False
DiffSaTime1     False
DiffSaTime2     False
DiffSaTime3     False
DiffSaTime4     False
Sa              False
Goal            False
dtype: bool
```

[10]:
```
data1.isnull().sum()
```

[10]:
```
ClimbRate       0
Sgz             0
P               0
Q               0
CurRoll         0
AbsRoll         0
DiffClb         0
DiffRollRate    0
DiffDiffClb     0
SaTime1         0
SaTime2         0
SaTime3         0
SaTime4         0
DiffSaTime1     0
```

```
DiffSaTime2     0
DiffSaTime3     0
DiffSaTime4     0
Sa              0
Goal            0
dtype: int64
```

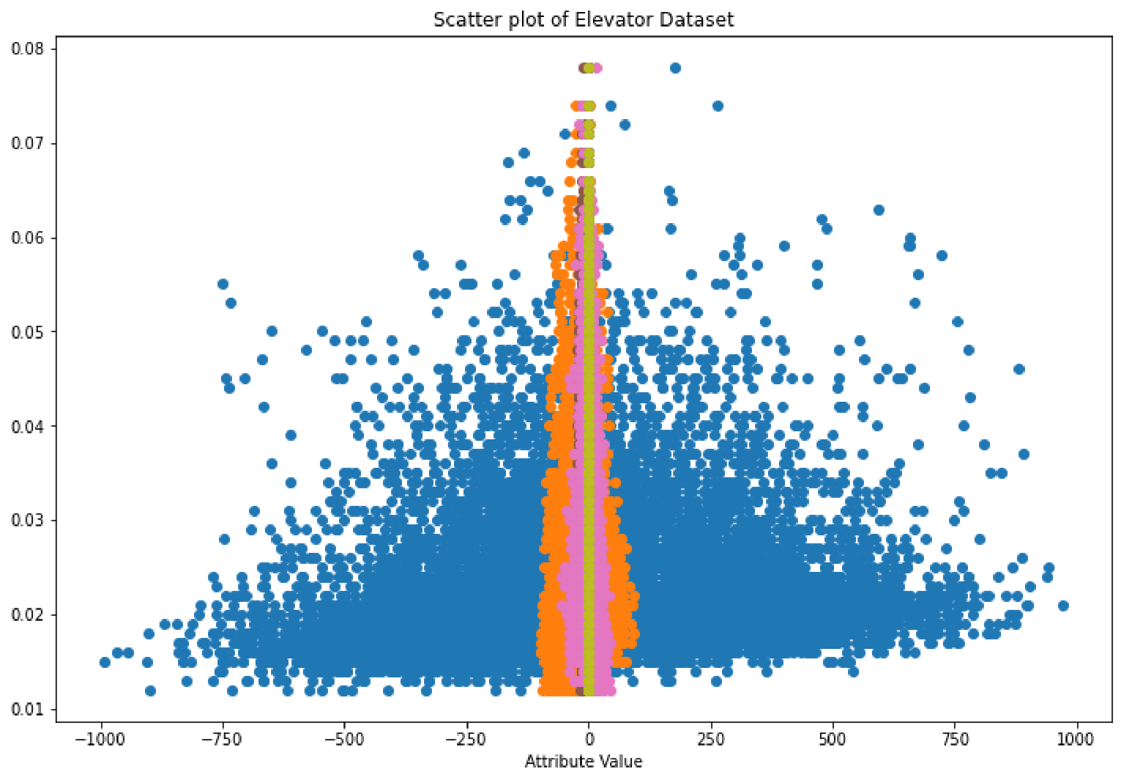
[11]:
```
import matplotlib.pyplot as plt
```

[12]:
```
fig, ax = plt.subplots(figsize=(12,8))

ax.scatter(data1["ClimbRate"], data1["Goal"])
ax.scatter(data1["Sgz"], data1["Goal"])
ax.scatter(data1["P"], data1["Goal"])
ax.scatter(data1["Q"], data1["Goal"])
ax.scatter(data1["CurRoll"], data1["Goal"])
ax.scatter(data1["AbsRoll"], data1["Goal"])
ax.scatter(data1["DiffClb"], data1["Goal"])
ax.scatter(data1["Sa"], data1["Goal"])
ax.scatter(data1["Goal"], data1["Goal"])

ax.set_xlabel("Attribute Value")

ax.set_title("Scatter plot of Elevator Dataset")
plt.show()
```



[13]:
```
X = data1.drop('Goal', axis=1).values
y = data1['Goal'].values
```

0.0.2 Create a linear regression model

[14]:
```
linear_model = LinearRegression()
```

[15]:
```
# Define the cross-validation method
cv = KFold(n_splits=5, shuffle=True, random_state=42)
```

[16]:
```
# Evaluate the model using 5-fold cross-validation
linear_scores = cross_val_score(linear_model, X, y, cv=cv,␣
 ↪scoring='neg_mean_squared_error')
```

[17]:
```
# Calculate the evaluation metrics from the scores
rmse = np.sqrt(-linear_scores.mean())
mse = -linear_scores.mean()
```

[18]:
```
mae = -cross_val_score(linear_model, X, y, cv=cv,␣
 ↪scoring='neg_mean_absolute_error').mean()
r2 = cross_val_score(linear_model, X, y, cv=cv, scoring='r2').mean()
```

[19]:
```
# Print the evaluation metrics for the Linear Regression model
print('Linear Regression Model:')
print('R-squared:', r2)
```

```
Linear Regression Model:
R-squared: 0.8129792647274817
```

0.0.3 Create an Artificial Neural Network model

[20]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[21]:
```
# Scale the features using standard scaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[22]:
```
# Create an Artificial Neural Network model
model = Sequential()
model.add(Dense(units=16, activation='relu', input_dim=X_train.shape[1]))
model.add(Dropout(rate=0.2))
model.add(Dense(units=8, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

[23]:
```
# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=64, validation_split=0.2)
```

```
Epoch 1/100
166/166 [==============================] - 1s 3ms/step - loss: 0.1744 -
val_loss: 0.0295
Epoch 2/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0550 -
val_loss: 0.0253
Epoch 3/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0800 -
val_loss: 0.0133
Epoch 4/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0203 -
val_loss: 0.0155
Epoch 5/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0245 -
val_loss: 0.0091
Epoch 6/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0043 -
val_loss: 0.0041
Epoch 7/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0127 -
val_loss: 0.0044
Epoch 8/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0266 -
val_loss: 0.0024
Epoch 9/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0449 -
val_loss: 0.0020
Epoch 10/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0093 -
val_loss: 2.4312e-04
Epoch 11/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0126 -
val_loss: 0.0013
Epoch 12/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0772 -
val_loss: 0.0041
Epoch 13/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0235 -
val_loss: 0.0029
Epoch 14/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0331 -
```

```
val_loss: 0.0039
Epoch 15/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0020 -
val_loss: 0.0043
Epoch 16/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0015 -
val_loss: 0.0035
Epoch 17/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0081 -
val_loss: 0.0030
Epoch 18/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0074 -
val_loss: 0.0020
Epoch 19/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0021 -
val_loss: 4.9643e-04
Epoch 20/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0013 -
val_loss: 3.3950e-04
Epoch 21/100
166/166 [==============================] - 0s 2ms/step - loss: 1.6999e-04 -
val_loss: 2.9759e-04
Epoch 22/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0035 -
val_loss: 4.7954e-04
Epoch 23/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0018 -
val_loss: 2.8217e-04
Epoch 24/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0018 -
val_loss: 7.5992e-04
Epoch 25/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0030 -
val_loss: 5.5846e-04
Epoch 26/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0039 -
val_loss: 9.6191e-04
Epoch 27/100
166/166 [==============================] - 0s 2ms/step - loss: 1.0688e-04 -
val_loss: 9.6797e-04
Epoch 28/100
166/166 [==============================] - 0s 2ms/step - loss: 3.4975e-04 -
val_loss: 9.8595e-04
Epoch 29/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0010 -
val_loss: 0.0011
Epoch 30/100
166/166 [==============================] - 0s 2ms/step - loss: 9.4853e-04 -
```

```
val_loss: 7.6167e-04
Epoch 31/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0087 -
val_loss: 9.0117e-05
Epoch 32/100
166/166 [==============================] - 0s 2ms/step - loss: 5.0141e-04 -
val_loss: 2.2567e-04
Epoch 33/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0010 -
val_loss: 3.9481e-04
Epoch 34/100
166/166 [==============================] - 0s 2ms/step - loss: 3.8329e-04 -
val_loss: 5.6111e-04
Epoch 35/100
166/166 [==============================] - 0s 2ms/step - loss: 4.0043e-04 -
val_loss: 2.6151e-04
Epoch 36/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0010 -
val_loss: 7.9462e-05
Epoch 37/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0013 -
val_loss: 1.4629e-04
Epoch 38/100
166/166 [==============================] - 0s 2ms/step - loss: 6.9973e-04 -
val_loss: 9.2451e-05
Epoch 39/100
166/166 [==============================] - 0s 2ms/step - loss: 5.6992e-04 -
val_loss: 1.3804e-04
Epoch 40/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0013 -
val_loss: 5.0679e-05
Epoch 41/100
166/166 [==============================] - 0s 2ms/step - loss: 3.7951e-04 -
val_loss: 6.6548e-05
Epoch 42/100
166/166 [==============================] - 0s 2ms/step - loss: 1.1320e-04 -
val_loss: 7.8852e-05
Epoch 43/100
166/166 [==============================] - 0s 2ms/step - loss: 1.5127e-04 -
val_loss: 6.8476e-05
Epoch 44/100
166/166 [==============================] - 0s 2ms/step - loss: 7.9831e-05 -
val_loss: 9.3846e-05
Epoch 45/100
166/166 [==============================] - 0s 2ms/step - loss: 6.0555e-04 -
val_loss: 5.6666e-05
Epoch 46/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0017 -
```

```
val_loss: 1.5791e-04
Epoch 47/100
166/166 [==============================] - 0s 2ms/step - loss: 4.7866e-05 -
val_loss: 1.4201e-04
Epoch 48/100
166/166 [==============================] - 0s 2ms/step - loss: 4.0821e-04 -
val_loss: 2.1204e-04
Epoch 49/100
166/166 [==============================] - 0s 2ms/step - loss: 7.1778e-05 -
val_loss: 2.0949e-04
Epoch 50/100
166/166 [==============================] - 0s 2ms/step - loss: 1.3113e-04 -
val_loss: 2.5815e-04
Epoch 51/100
166/166 [==============================] - 0s 2ms/step - loss: 5.1665e-05 -
val_loss: 2.6330e-04
Epoch 52/100
166/166 [==============================] - 0s 2ms/step - loss: 7.8604e-04 -
val_loss: 3.2847e-04
Epoch 53/100
166/166 [==============================] - 0s 2ms/step - loss: 0.0014 -
val_loss: 9.3504e-05
Epoch 54/100
166/166 [==============================] - 0s 2ms/step - loss: 5.7679e-05 -
val_loss: 9.4171e-05
Epoch 55/100
166/166 [==============================] - 0s 2ms/step - loss: 7.1528e-05 -
val_loss: 7.1289e-05
Epoch 56/100
166/166 [==============================] - 0s 2ms/step - loss: 9.7646e-05 -
val_loss: 5.2954e-05
Epoch 57/100
166/166 [==============================] - 0s 2ms/step - loss: 7.1217e-05 -
val_loss: 5.1105e-05
Epoch 58/100
166/166 [==============================] - 0s 2ms/step - loss: 9.7139e-05 -
val_loss: 5.3514e-05
Epoch 59/100
166/166 [==============================] - 0s 2ms/step - loss: 1.1549e-04 -
val_loss: 5.2707e-05
Epoch 60/100
166/166 [==============================] - 0s 2ms/step - loss: 5.5844e-04 -
val_loss: 5.2717e-05
Epoch 61/100
166/166 [==============================] - 0s 2ms/step - loss: 1.6727e-04 -
val_loss: 5.5117e-05
Epoch 62/100
166/166 [==============================] - 0s 2ms/step - loss: 4.7141e-05 -
```

```
val_loss: 5.2049e-05
Epoch 63/100
166/166 [==============================] - 0s 2ms/step - loss: 4.6270e-05 -
val_loss: 5.1782e-05
Epoch 64/100
166/166 [==============================] - 0s 2ms/step - loss: 9.3179e-05 -
val_loss: 6.3752e-05
Epoch 65/100
166/166 [==============================] - 0s 2ms/step - loss: 5.1456e-05 -
val_loss: 5.6050e-05
Epoch 66/100
166/166 [==============================] - 0s 2ms/step - loss: 9.9514e-05 -
val_loss: 5.1018e-05
Epoch 67/100
166/166 [==============================] - 0s 2ms/step - loss: 9.1401e-05 -
val_loss: 5.1718e-05
Epoch 68/100
166/166 [==============================] - 0s 2ms/step - loss: 5.7007e-05 -
val_loss: 5.1046e-05
Epoch 69/100
166/166 [==============================] - 0s 2ms/step - loss: 4.6210e-05 -
val_loss: 5.2570e-05
Epoch 70/100
166/166 [==============================] - 0s 2ms/step - loss: 5.0905e-05 -
val_loss: 5.2221e-05
Epoch 71/100
166/166 [==============================] - 0s 2ms/step - loss: 4.6437e-05 -
val_loss: 5.2528e-05
Epoch 72/100
166/166 [==============================] - 0s 2ms/step - loss: 5.9892e-05 -
val_loss: 5.0361e-05
Epoch 73/100
166/166 [==============================] - 0s 2ms/step - loss: 5.2576e-05 -
val_loss: 5.1145e-05
Epoch 74/100
166/166 [==============================] - 0s 2ms/step - loss: 4.5719e-05 -
val_loss: 5.1671e-05
Epoch 75/100
166/166 [==============================] - 0s 2ms/step - loss: 4.8824e-05 -
val_loss: 5.0350e-05
Epoch 76/100
166/166 [==============================] - 0s 2ms/step - loss: 5.3146e-05 -
val_loss: 5.3667e-05
Epoch 77/100
166/166 [==============================] - 0s 2ms/step - loss: 5.1186e-05 -
val_loss: 5.9484e-05
Epoch 78/100
166/166 [==============================] - 0s 2ms/step - loss: 5.4552e-05 -
```

```
val_loss: 5.1445e-05
Epoch 79/100
166/166 [==============================] - 0s 2ms/step - loss: 4.6151e-05 -
val_loss: 5.3090e-05
Epoch 80/100
166/166 [==============================] - 0s 2ms/step - loss: 4.4909e-05 -
val_loss: 5.4982e-05
Epoch 81/100
166/166 [==============================] - 0s 2ms/step - loss: 4.7868e-05 -
val_loss: 5.7340e-05
Epoch 82/100
166/166 [==============================] - 0s 2ms/step - loss: 9.9064e-05 -
val_loss: 7.2818e-05
Epoch 83/100
166/166 [==============================] - 0s 2ms/step - loss: 7.4136e-05 -
val_loss: 4.3422e-05
Epoch 84/100
166/166 [==============================] - 0s 2ms/step - loss: 4.4107e-05 -
val_loss: 4.1406e-05
Epoch 85/100
166/166 [==============================] - 0s 2ms/step - loss: 3.6049e-05 -
val_loss: 3.3547e-05
Epoch 86/100
166/166 [==============================] - 0s 2ms/step - loss: 3.3023e-05 -
val_loss: 3.3008e-05
Epoch 87/100
166/166 [==============================] - 0s 2ms/step - loss: 3.2978e-05 -
val_loss: 3.0787e-05
Epoch 88/100
166/166 [==============================] - 0s 2ms/step - loss: 3.1276e-05 -
val_loss: 2.9172e-05
Epoch 89/100
166/166 [==============================] - 0s 2ms/step - loss: 3.3443e-05 -
val_loss: 2.8453e-05
Epoch 90/100
166/166 [==============================] - 0s 2ms/step - loss: 3.7676e-05 -
val_loss: 2.6685e-05
Epoch 91/100
166/166 [==============================] - 0s 2ms/step - loss: 3.1389e-05 -
val_loss: 2.6765e-05
Epoch 92/100
166/166 [==============================] - 0s 2ms/step - loss: 2.9465e-05 -
val_loss: 2.4311e-05
Epoch 93/100
166/166 [==============================] - 0s 2ms/step - loss: 2.8084e-05 -
val_loss: 2.2116e-05
Epoch 94/100
166/166 [==============================] - 0s 2ms/step - loss: 2.6934e-05 -
```

```
val_loss: 2.2166e-05
Epoch 95/100
166/166 [==============================] - 0s 2ms/step - loss: 2.8818e-05 -
val_loss: 2.1122e-05
Epoch 96/100
166/166 [==============================] - 0s 2ms/step - loss: 2.5630e-05 -
val_loss: 2.1193e-05
Epoch 97/100
166/166 [==============================] - 0s 2ms/step - loss: 2.9575e-05 -
val_loss: 2.0586e-05
Epoch 98/100
166/166 [==============================] - 0s 2ms/step - loss: 2.5145e-05 -
val_loss: 1.7492e-05
Epoch 99/100
166/166 [==============================] - 0s 2ms/step - loss: 2.2342e-05 -
val_loss: 1.7257e-05
Epoch 100/100
166/166 [==============================] - 0s 2ms/step - loss: 2.2079e-05 -
val_loss: 1.8170e-05
```

```
[23]: <keras.callbacks.History at 0x2a49d8b4d90>
```

```
[24]: # Make predictions on the test set
y_pred = model.predict(X_test)
```

```
104/104 [==============================] - 0s 1ms/step
```

```
[25]: # Calculate the evaluation metrics
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mse = mean_squared_error(y_test, y_pred)
```

```
[26]: mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
[27]: # Print the evaluation metrics for the Artificial Neural Network model
print('Artificial Neural Network Model:')
print('R-squared:', r2)
```

```
Artificial Neural Network Model:
R-squared: 0.6945748908805229
```

0.0.4 Create an XGBoost Regression model

```
[28]: xgb_model = XGBRegressor(objective='reg:squarederror', random_state=42)
```

```
[29]: # Evaluate the model using 5-fold cross-validation
xgb_scores = cross_val_score(xgb_model, X, y, cv=cv,␣
 ↪scoring='neg_mean_squared_error')
```

[30]:
```
# Calculate the evaluation metrics from the scores
xgb_rmse = np.sqrt(-xgb_scores.mean())
xgb_mse = -xgb_scores.mean()
```

[31]:
```
xgb_mae = -cross_val_score(xgb_model, X, y, cv=cv,␣
 ↪scoring='neg_mean_absolute_error').mean()
xgb_r2 = cross_val_score(xgb_model, X, y, cv=cv, scoring='r2').mean()
```

[32]:
```
# Print the evaluation metrics for the XGBoost Regression model
print('XGBoost Regression Model:')
print('R-squared:', xgb_r2)
```

```
XGBoost Regression Model:
R-squared: 0.8833271649748543
```