**AIX20006 Assignment – Regression Modeling and Evaluation**

**22100748 최윤영**

**22100175 김주성**

**Suppose that you are asked to develop a regression model for analyzing the input-out mapping of the given dataset. Imagine that you are specifically required to use the MLP model to complete the regression task. Deliverables for this problem are as follows:**

**Q. Create an MLP-based regression model using the provided dataset and explain the regression model structure in detail. You may need to come up with by yourself how to divide the dataset**

1. This is the import section where the required libraries are imported.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.neural\_network import MLPRegressor

from sklearn.metrics import r2\_score, mean\_squared\_error

from sklearn.preprocessing import StandardScaler

1. This part removes the outliers from the data using the interquartile range (IQR) method. The first and third quartiles are calculated using the quantile() method, then the IQR is calculated as the difference between them. Any row containing a value outside the range of Q1 - 1.5IQR and Q3 + 1.5IQR is removed.

# Remove outliers using IQR method

Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1

data = data[~((data < (Q1 - 1.5 \* IQR)) | (data > (Q3 + 1.5 \* IQR))).any(axis=1)]

1. This section splits the data into input (X) and output (y) variables, where the input contains all columns except the last one(longitude and latitude), and the output contains the last column(eastward wind).

# Split the dataset into input and output variables

X = data.iloc[:, :-1]

y = data.iloc[:, -1]

1. This part standardizes the input variables to have zero mean and unit variance, using the StandardScaler() method.

# Standardize the input variables

scaler = StandardScaler()

X = scaler.fit\_transform(X)

1. This part splits the data into training and testing sets using the train\_test\_split() method from the sklearn.model\_selection module. **It splits the data into training and testing sets with a 70:30 ratio**, and sets a random state for reproducibility.

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

1. This part builds the multi-layer perceptron (MLP) regression model using the MLPRegressor() method from the sklearn.neural\_network module. The model architecture consists of three hidden layers with 30, 20, and 10 neurons respectively, with the activation function as ReLU. The solver used is 'adam', and regularization parameter alpha is set to 0.001. The maximum number of iterations is set to 5000, and a random state is set for reproducibility.

# Build the MLP regression model

mlp = MLPRegressor(hidden\_layer\_sizes=(30, 20, 10),max\_iter=5000, activation='relu', solver='adam', alpha=0.001, random\_state=1)

1. Train the model using the 'fit' method.

mlp.fit(X\_train, y\_train)

1. Use the 'predict' method to obtain predicted values for the data.

y\_pred = mlp.predict(X\_test)

1. This section evaluates the MLP regression model by predicting the output using the testing input data (X\_test) and calculating the R-squared and mean squared error (MSE) values using the r2\_score() and mean\_squared\_error() methods from the sklearn.metrics module respectively.

r\_squared = r2\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

1. This part calculates the loss and accuracy of the trained MLP regression model. mlp.loss\_curve\_ returns the training loss curve of the MLP model, which is a plot of the model's loss over each iteration of the training process. mlp.score(X\_test, y\_test) calculates the R-squared score of the model on the testing data. Both loss and accuracy are assigned to these respective values.

# Calculate the loss and accuracy

loss = mlp.loss\_curve\_

accuracy = mlp.score(X\_test, y\_test)

1. This section of the code creates two plots. The first plot is a scatter plot of **the actual versus predicted values** of the testing data. y\_test is plotted on the x-axis, while y\_pred is plotted on the y-axis. A black dotted line is drawn diagonally across the plot to show where **perfect predictions would lie**. This plot allows us to visualize how well the model is predicting the target variable for the testing data.

# Plot the actual vs predicted values

plt.scatter(y\_test, y\_pred)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=4)

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.title('Actual vs Predicted')

plt.show()

1. The second plot is a scatter plot of the residuals versus the predicted values of the testing data. The residuals are calculated as the difference between the actual and predicted values (residual = y\_test - y\_pred). The predicted values are plotted on the x-axis, while the residuals are plotted on the y-axis. A black dotted line is drawn horizontally across the plot to indicate where the residuals are equal to zero. This plot allows us to visualize the distribution of the residuals and detect any patterns that the model may have missed.

# Plot the residual vs predicted values

residual = y\_test - y\_pred

plt.scatter(y\_pred, residual)

plt.plot([y\_pred.min(), y\_pred.max()], [0, 0], 'k--', lw=4)

plt.xlabel('Predicted')

plt.ylabel('Residual')

plt.title('Residual vs Predicted')

plt.show()

1. The code prints out the R-squared score, mean squared error (MSE), and accuracy of the MLP model on the testing data. R-squared measures how well the model fits the data, while MSE measures the average squared difference between the actual and predicted values. Accuracy represents the proportion of correct predictions that the model made on the testing data.

print(f"R-squared: {r\_squared:.3f}")

print(f"Mean Squared Error: {mse:.3f}")

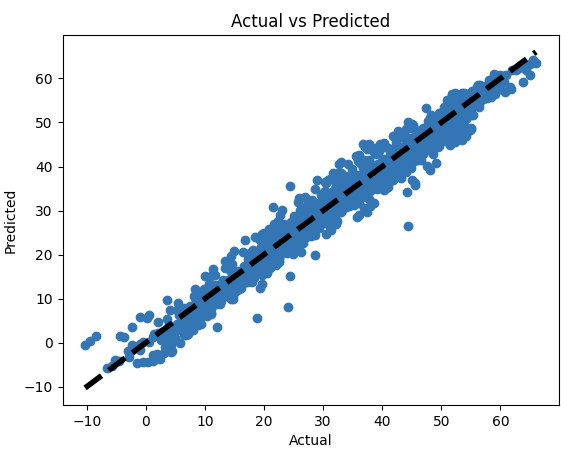
print(f"Accuracy: {accuracy:.3f}")

**Q. Evaluate the model performance using the given dataset. Specifically, you should provide the R-square value, the actual by predicted plot, the residual by predicted plot, and the model representation error value.**

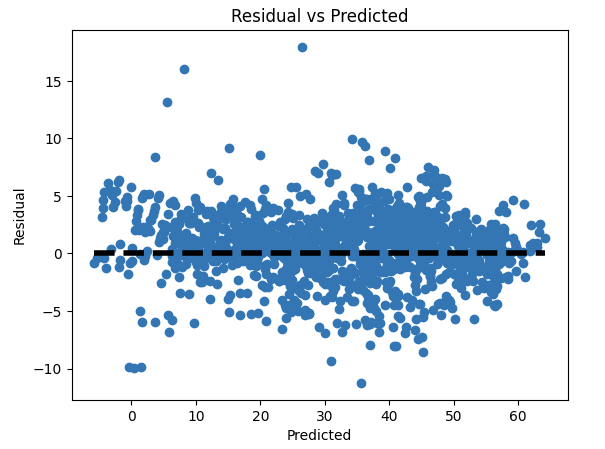
1. R-square value

R-squared: 0.965

1. The actual by predicted plot



1. The residual by predicted plot



1. The model representation error value(MSE)

Mean Squared Error: 8.464

**Q. Discuss with your teammates the results of the model evaluation.**

1. R-square value

An R-squared value of 0.965 means that 96.5% of the variability in the target variable(predicted) can be explained by the input features used in the model. The model is able to capture most of the patterns in the data and make accurate predictions of the response variable based on the input features. However, R-squared is not a perfect measure of model performance and should be used in conjunction with other evaluation metrics.

1. The actual by predicted plot

The black dotted line represents the situation where the prediction is perfect, while the blue dots represent the actual predicted results. As can be seen from the graph, the blue dots tend to closely follow the black line. It means that the predicted values are very close to the actual values. This indicates that the model is able to accurately predict the target variable for the given input features. In other words, the model is able to capture the underlying patterns and relationships in the data, and is able to generalize well to unseen data. A nearly perfect match in the actual vs predicted plot is a desirable outcome in regression analysis, as it indicates that the model is performing well on the given dataset.

1. The residual by predicted plot

It means that the model's predictions are consistently close to the actual values, with little to no bias. In other words, the difference between the predicted and actual values is small, indicating that the model is making accurate predictions. A residual is the difference between the actual value and the predicted value, and the plot of residuals vs predicted values is used to assess the quality of a regression model. If the residuals are scattered randomly around the horizontal line at y=0, it suggests that the model is unbiased and making consistent predictions.

1. Model representation error

To determine the model representation error, we calculated the Mean Squared Error (MSE), which is a measure of how well the model's predictions match the actual values. Specifically, it is the average of the squared differences between the predicted and actual values. It is difficult to assess whether the model has been trained well or not based solely on the numerical value of MSE. However, we can estimate that the actual error is around 2.90, which is the square root of 8.464, on average. Considering that the range of the actual wind data is between -10 and 60, an error of 2.90 can be interpreted as a good prediction by the model.

* **A number of failed works that we did before increasing the model's accuracy**

텍스트, 도표, 지도, 그래프이(가) 표시된 사진

자동 생성된 설명라인, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

R-squared: -2.029

Model representation error: 353.252

**처음에 이러한 결과물을 얻었고, 이에 대해 정확도를 높이기 위한 방법들로 아래와 같은 방식들이 있었다.**

데이터 전처리 과정 개선: 모델에 입력되는 데이터의 전처리 과정에서 누락된 값이나 이상치를 적절하게 처리해주거나, 데이터 스케일링을 적용할 수 있습니다. 이를 통해 모델이 더 정확한 예측을 할 수 있도록 돕습니다.

은닉층의 수나 노드의 개수 등을 조정해보는 것이 가능합니다.

데이터 추가 및 조정: 더 많은 데이터를 추가하거나, 데이터를 선별적으로 제외하여 모델에 불필요한 정보가 들어가지 않도록 조정할 수 있습니다.

교차 검증: 데이터를 더욱 효과적으로 활용하기 위해 교차 검증을 적용할 수 있습니다. 이를 통해 모델이 일반화 능력을 갖추도록 합니다.

앙상블: 여러 개의 모델을 합쳐서 더 좋은 예측 결과를 도출할 수 있습니다. 이를 통해 각 모델의 장점을 활용하여 더욱 정확한 예측을 할 수 있습니다.

**이러한 방법들 중 우리는 아래와 같은 부분들을 수정하기로 논의하였고,**

**What we improved:**

**이상치(outlier) 처리 : IQR 기반으로 범위를 설정하고 범위 밖의 값을 outlier로 간주하여 삭제**

**표준화(Standardization) : 각 feature의 평균과 표준편차를 사용하여 데이터를 표준화하여 scaling**

**모델링**

**MLPRegressor의 Hidden layer의 구성 : (30, 20, 10)으로 구성**

**activation function : relu 사용**

**solver : adam 사용**

**alpha : L2 regularization (0.001로 설정)**

**결론적으로 아래와 같이 현저히 높아진 정확도의 결과물을 얻을 수 있었다.**

**지도, 텍스트, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명**스크린샷, 도표, 그래프, 라인이(가) 표시된 사진

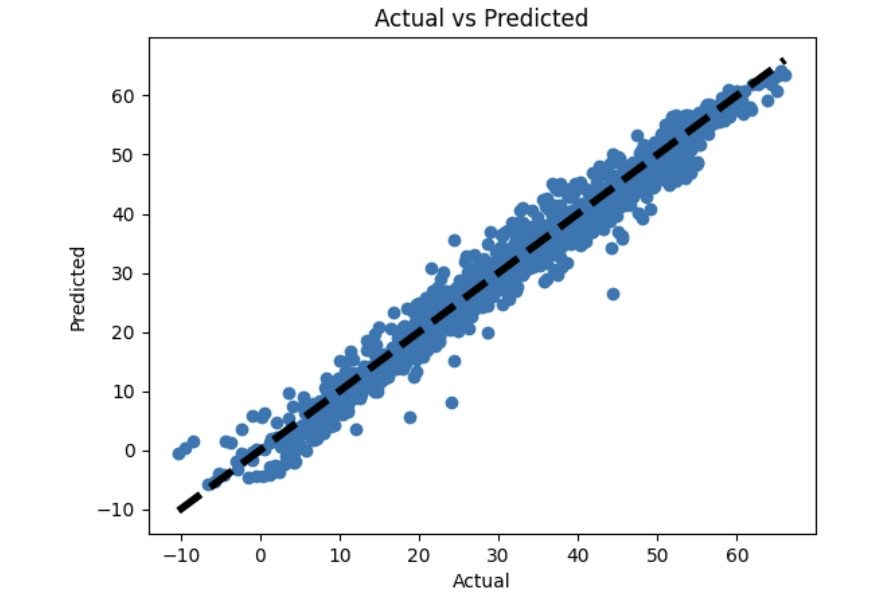
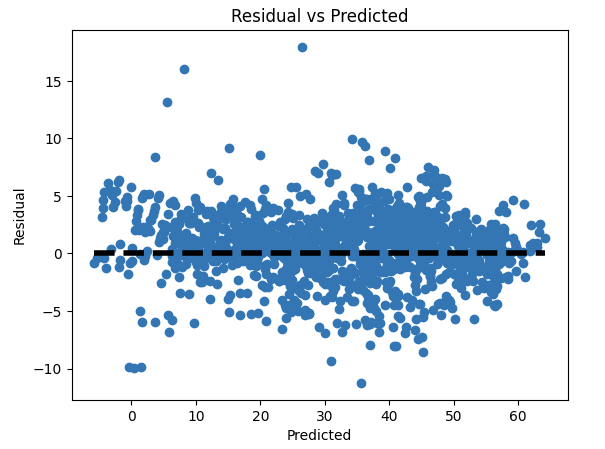
자동 생성된 설명

**# Build the MLP regression model**

**mlp = MLPRegressor(hidden\_layer\_sizes=(30, 20, 10),max\_iter=5000, activation='relu', solver='adam', alpha=0.001, random\_state=1)**

**mlp.fit(X\_train, y\_train)**

**max\_iter=5000으로 증가**

****

**더 나아가, 한번 더 수정의 과정을 거쳐 더욱 향상된 모델로 수정할 수 있었다.**

**파트분배**

**code: 최윤영 40%, 김주성 60%**

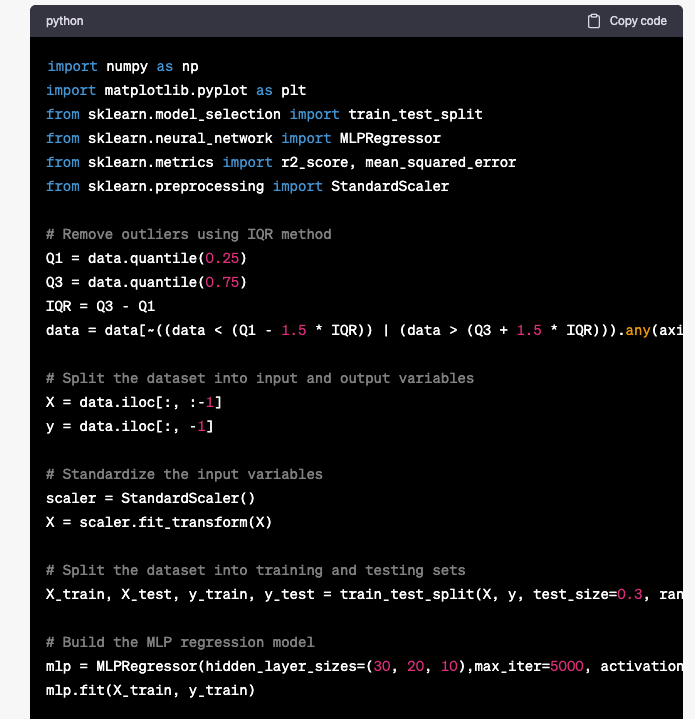
**report: 최윤영 60%, 김주성 40%**

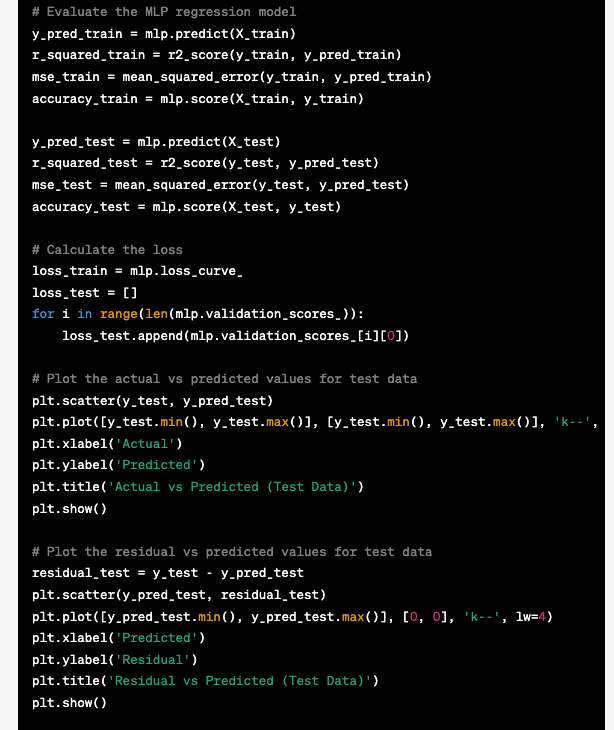
**Citation**

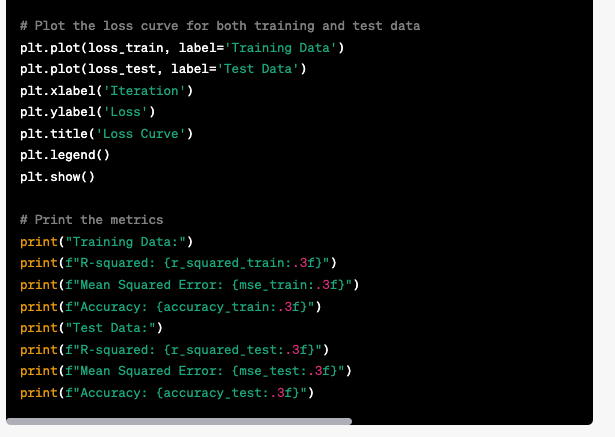
* **sklearn.neural\_network.MLPRegressor**

[**https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPRegressor.html**](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html)

* **ChatGPT**

****

****

****