2024-1 Final report

[Introduction to Artificial Intelligence Project]

Project title:

Sleep stage classification using mice brain signals

Team Members

송민영(22000374), 심성환(22000397),

채정원(22100725), 하지민(22100766)

Date: 2024. 06. 02

Contents

1. Problem description

This project aims to analyze brain signal data from mice to classify their sleep patterns. Sleep is classified into different stages in humans and rodents. In humans, sleep is divided into five stages: WAKE, N1, N2, N3, and REM, while in rodents, it is classified into three stages: Non-REM, REM, and WAKE. The goal of this project is to develop an accurate and efficient AI model to classify the sleep stages of mice. To achieve this, various machine learning and deep learning models were employed to analyze the brain signal data of mice and accurately classify their sleep stages. The project particularly focused on testing various models and selecting the one that provided the highest accuracy.

2. Background and conventional approach

Sleep is characterized by the absence of wakefulness and by the loss of consciousness of one's surroundings. It is essential to the restoration and recovery of vital bodily and mental functions. In this way, sleep occupies a significant portion of life and plays a critical role in health. Specifically, it helps the body to maintain homeostasis and ensures the quality of daytime activities.

Given its importance, sleep is a crucial part of our lives. People classify sleep stages to understand and diagnose sleep-related pathologies, with brain signals being the most accurate data for

indicating sleep stages. In humans, sleep occurs in five stages: wake, N1, N2, N3, and REM. Similarly, sleep is important for rodents, which are nocturnal animals that sleep during the day and are active at night. Therefore, light cycle data is generally needed to classify the sleep cycle of rodents. Unlike humans, rodents do not experience deep sleep but instead have non-consolidated sleep. Thus, their sleep stage is classified into 3 stages: Non-REM(NREM), REM, and wake.

3. Proposed method

A. Data collection and preparation

```
X_train shape: (22992, 13, 176)
Y_train shape: (22992,)
X_test shape: (5748, 13, 176)
Y_test shape: (5748,)
```

For the project, data for training and testing were used. The training dataset comprised 22,992 samples, each with 13 time points. The number of frequency points in the training data was 176. The testing dataset consisted of 5,748 samples, also with 13 time points and 176 frequency points.

Given that the data was three-dimensional, reshaping it was essential for some models. For Convolutional Neural Network (CNN) model, reshaping the training data to a four-dimensional layer was necessary to use Conv2D. For the second model, Multilayer perceptron (MLP), flattening the data from

three-dimensional to two-dimensional was required. The Random Forest, SVM, KNN and Decision Tree model also required reshaping the data to two-dimensions. Additionally, flattening the input data was needed for the sequential model. Since the Y_train and Y_test data were already one-dimensional, no reshaping was needed for these labels.

B. AI approach

We used models of CNN, MLP, Random Forest, Sequential Model, KNN, Decision Tree, and SVM as training models, and we wanted to select five models with high values among them. First, CNN(Convolutional Neural Network)is effective at extracting patterns from images or time-series data. Second, MLP(Multilayer Perceptron) has simple structure, applicable to various types of data. Third, Random Forest is high predictive performance and strong overfitting prevention. Forth, Sequential Model is strong at recognizing patterns in time-series data. Fifth, KNN(K-Nearest Neighbors)is easy to understand and implement, effective with small datasets. Sixth, Decision Tree is easy to interpret, fast learning. Seventh, SVM(Support vector Machine) works effectively in high-dimensional space, strong generalization ability.

We found the models required to classify data on mouse sleep patterns and selected seven models that fit them. Since each model had different training methods, we decided to test various models and select a model with high accuracy because we weren't sure which model produced high accuracy.

C. Evaluation criteria

Initially, the data were classified as train_X, train_Y, test_X, and test_Y, of which train_X and train_Y were used to learn from the model. The remaining test_X and test_Y were used as data to evaluate the performance learned from the model. Therefore, after completing the learning, the result of the data in test_X is predicted and compared with test_Y to check the accuracy of how similar it is. And the result was outputted in percentages to compare the accuracy by model. The following picture is a code that checks accuracy in the Random Forest model.

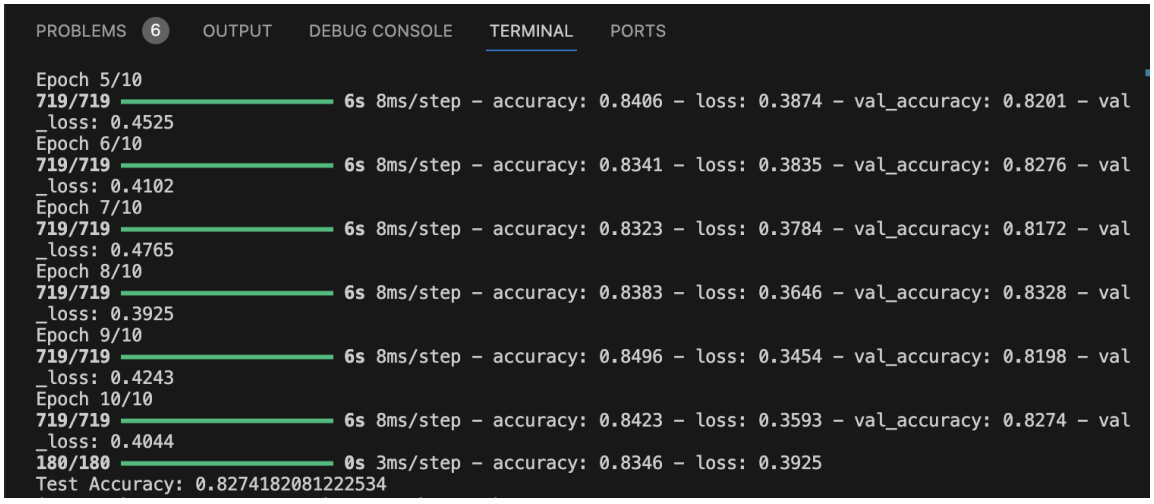
```
# Decision Tree Training
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, Y_train)

# Prediction of Data X_test
Y_pred = clf.predict(X_test)

# Evaluation
accuracy = accuracy_score(Y_test, Y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

Picture : Code of Evaluation(Random Forest)

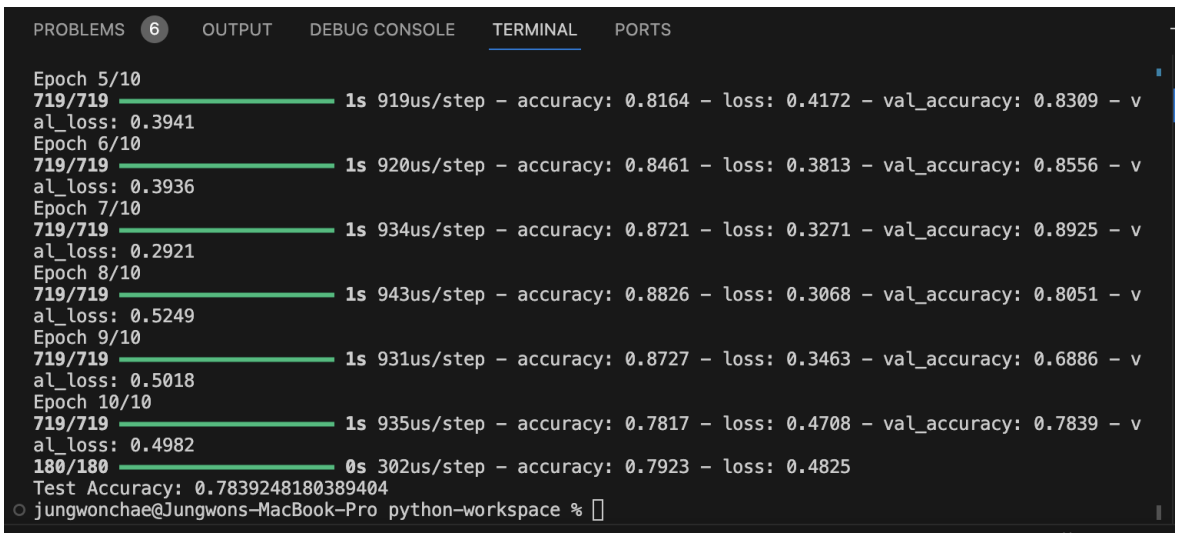
4. Results

We run experiments with different models to analyze the sleep stages of rats. Each member of the team run the experiment using the model of their choice. We will develop an AI analytics model with a model that best divides sleep stages. Using the model above, we analyzed the sleep stages of each team member.

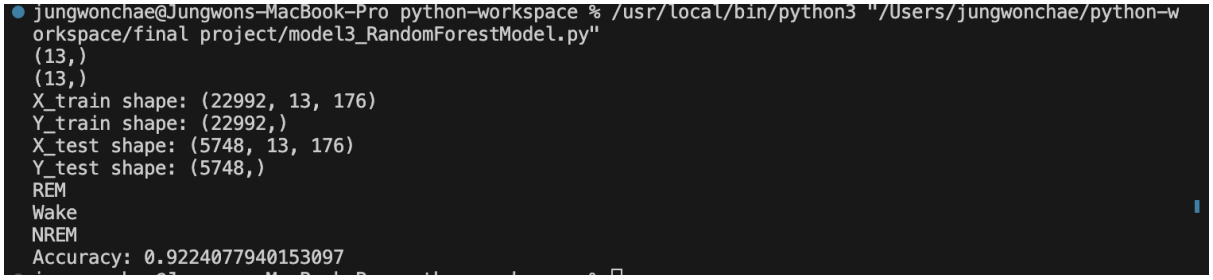
First, a model using CNN. The CNN model has an accuracy of about 0.849, which is good enough for immediate use.



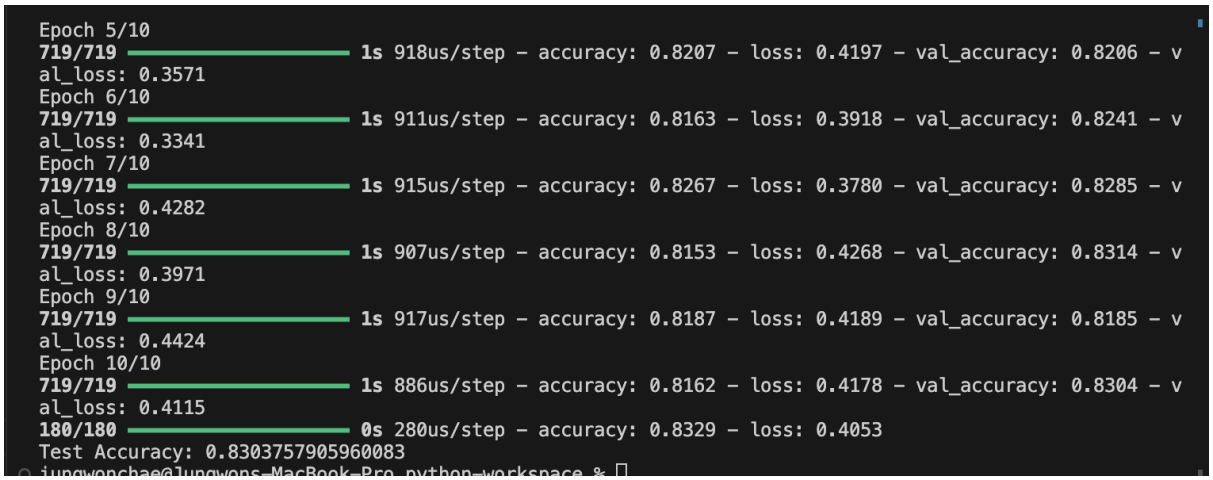
Next model is MLP model. The MLP model has an accuracy of about 0.833, which is higher than the CNN model.



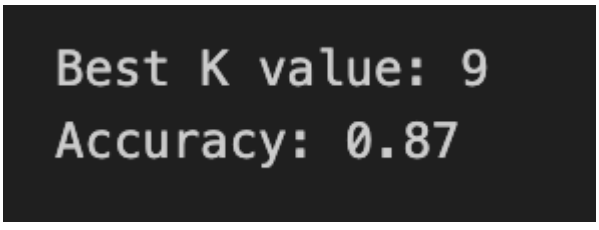
And Random Forest Model has an accuracy of about 0.922, this model has higher accuracy than the models we analyzed earlier.



Sequential model has an accuracy of about 0.833, this model has similar accuracy to the MLP model, but lower accuracy than the rest of the models.



KNN model has an accuracy of about 0.87, this model has higher accuracy than the CNN, MLP, and Sequential models, but lower accuracy than the Random Forest model.



Decision model has an accuracy of about 0.83, this model has similar accuracy to the MLP, Sequential models, but lower accuracy than the CNN, Random Forest, and KNN models.

```
PS D:\한동대학교\24-1학기\AI Project 입문\과제\팀플> python team.py
(13,)
(176,)
X_train shape: (22992, 13, 176)
Y_train shape: (22992,)
X_test shape: (5748, 13, 176)
Y_test shape: (5748,)
REM
Wake
NREM
Accuracy: 0.83
```

SVM model has an accuracy of about 0.904, this model has accuracy higher than the CNN, MLP, Sequential, KNN, and Decision models, but lower accuracy than the Random Forest model.

```
clf = svm.SVC(kernel='linear')

clf.fit(X_train_scaled, Y_train)

▸ SVC

predictions = clf.predict(X_test_scaled)

print("Accuracy:", accuracy_score(Y_test, predictions))

Accuracy: 0.9036186499652052
```

5. Conclusion

According to the results, the model with the highest accuracy is the Random Forest model with an accuracy of about 0.922, followed by the SVM model with an accuracy of about 0.904, the KNN model with an accuracy of about 0.87, the CNN with an accuracy of about 0.849, and the MLP, Sequential, and Decision Tree models with an accuracy of about 0.833.

It is best to use the Random Forest model, which has the highest accuracy. However, in some cases, SVM, KNN, and other models has accuracy enough to be used. Models with an accuracy of about 0.8 or better can still be useful. Model selection should be based not only on accuracy, but also on other factors such as run time, resource usage, and model complexity.

Therefore, it is important to consider models with high accuracy first. However, choose the best model for specific situations. Other models can be just as useful, and the choice should be based on a combination of factors.

6. Division of work

We divided the tasks to research how to preprocess the data and how to train it. Based on this research, we each implemented various models by splitting the parts among ourselves.

A. 송민영 (22000374)

Research on Data Preprocessing Methods

K-Nearest Neighbors (KNN) Model Test

B. 심성환 (22000397)

Research on Data Training Methods

Support Vector Machine (SVM) Model Test

C. 채정원 (22100725)

Research on Data Training Methods

CNN, MLP, Random Forest, Sequential Model Test

D. 하지민 (22100766)

Research on Data Preprocessing Methods

Decision Tree Model Test

7. Discussion (per each member)

A. 송민영 (22000374)

Before starting the project, I had a limited conceptual understanding of data preprocessing and training models. With the help of my team members, I was able to gradually get a grasp of these concepts. Through the process of testing and comparing various machine learning and deep learning models, I learned a lot, especially deepening my understanding of data processing and model optimization methods. The process of developing an AI model was a highly challenging yet rewarding experience.

B. 심성환 (22000397)

I was worried because I was analyzing the data and selecting models myself for the first time. However, after getting help from my team members and analyzing the data myself, the concepts started to understand. Looking at the data, I learned how to analyze it and what factors are important.

I had a lot of help from the team this time. I learned how to proceed with AI projects and how to properly change the data before when using any model.

I think it’s important to get this experience. I learn about AI analytics from this experience, and it will help me other AI study.

C. 채정원 (22100725)

As it was my first-time doing a project using AI, implementing the models was challenging. However, I learned a lot about different types of models and data preprocessing techniques. Initially, I struggled with reshaping data for various models, such as CNN, MLP, and others. Despite these difficulties, I gained valuable insights into the importance of proper data formatting and model requirements. This experience enhanced my understanding of machine learning workflows and boosted my confidence in handling similar tasks in the future.

D. 하지민 (22100766)

While taking the AI project class, I was always curious about how real data was applied. Through this project, I was able to classify data and learn how to effectively learn the data. In this project, I identified the data and thought about how to preprocess it. As a result, it was time to understand the data and share opinions with the team members on how to process it to learn it effectively. After that, I thought that if I were to do an AI-related project, I would like to learn about the model in detail.

8. Reference

Automatic sleep stage classification: From classical machine learning methods to deep learning

https://www.sciencedirect.com/science/article/pii/S1746809422002737#:~:text=The%20classification%20of%20sleep%20stages%20plays%20a%20crucial%20role%20in,time%2Dconsuming%20and%20subjective%20procedure.

Physiology, Sleep Stages

https://www.ncbi.nlm.nih.gov/books/NBK526132/#:~:text=Sleep%20occurs%20in%20five%20stages,leading%20to%20progressively%20deeper%20sleep.

9. Codes

[Preprocessing Data]

: code insertion needed for every model before running

```
import os


import pandas as pd
from skimage import io
import numpy as np
import matplotlib.pyplot as plt


time = np.load('time.npy')
freq = np.load('time.npy')


print(time.shape)
print(freq.shape)


# For training data (day 1, 2, 3, 4)
X_train = np.load('X_train_mouse02.npy')
Y_train = np.load('Y_train_mouse02.npy')
```

```
# For testing data (day 5)
X_test = np.load('X_test_mouse02.npy')
Y_test = np.load('Y_test_mouse02.npy')

# Check shapes
print("X_train shape:", X_train.shape)
print("Y_train shape:", Y_train.shape)
print("X_test shape:", X_test.shape)
print("Y_test shape:", Y_test.shape)

from matplotlib import colormaps

# Plot spectrogram images of each class
encountered_labels = set()
classes = ['REM', 'Wake', 'NREM']
spectro_of_3_classes = [[] for _ in range(3)]

for i, label in enumerate(Y_test.flatten()):
```

```
    if label in [0, 1, 2] and label not in encountered_labels:
        if(label == 0):
            spectro_of_3_classes[0] = (X_test[i])
        if(label == 1):
            spectro_of_3_classes[1] = (X_test[i])
        if(label == 2):
            spectro_of_3_classes[2] = (X_test[i])
        encountered_labels.add(label)
    if len(encountered_labels) == 3:
        break


# Visualize spectrograms
fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharex=False, sharey=False)


for i in range(3):
    # Visualize the spectrogram
    spectro_of_3_classes[i] = np.transpose(spectro_of_3_classes[i])
    im = axes[i].imshow(spectro_of_3_classes[i], aspect='auto', origin='lower', extent=[time[0], time[-1], freq[0], freq[-1]])
```

```
    fig.colorbar(im, ax=axes[i], label='PSD (dB)', shrink=0.8)
    im.set_cmap(colormaps.get_cmap('rainbow'))
    # im.set_clim([5, 28])
    axes[i].set_xlabel("Times", fontsize=14)
    axes[i].set_ylabel("Frequency (Hz)", fontsize=14)
    axes[i].set_title(f"Spectrogram of Class {classes[i]}", fontsize=16)

plt.tight_layout()
plt.show()

print(classes[0])
print(classes[1])
print(classes[2])
```

[Model 1. CNN]

```
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.models import Sequential

model_cnn = Sequential()
```

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1)

X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

model_cnn.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=X_train.shape[1:]))

model_cnn.add(MaxPooling2D(pool_size=(2, 2)))

model_cnn.add(Flatten())

model_cnn.add(Dense(128, activation='relu'))

model_cnn.add(Dense(3, activation='softmax'))

model_cnn.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model_cnn.summary()

history_cnn = model_cnn.fit(X_train, Y_train, epochs=10, batch_size=32, validation_data=(X_test, Y_test))

test_loss, test_accuracy = model_cnn.evaluate(X_test, Y_test)

print("Test Accuracy:", test_accuracy)
```

[Model2. MLP]

```
from keras.layers import Dense
from keras.models import Sequential

model_mlp = Sequential()

X_train_flattened = X_train.reshape(X_train.shape[0], -1)
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

model_mlp.add(Dense(128, input_dim= X_train_flattened.shape[1], activation='relu'))
model_mlp.add(Dense(64, activation='relu'))

model_mlp.add(Dense(3, activation='softmax'))

model_mlp.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model_mlp.summary()
```

```
history_mlp = model_mlp.fit(X_train_flattened, Y_train, epochs=10, batch_size=32, validation_data=(X_test_flattened, Y_test))

test_loss, test_accuracy = model_mlp.evaluate(X_test_flattened, Y_test)
print("Test Accuracy:", test_accuracy)
```

[Model3. Random Forest Model]

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

X_train_flattened = X_train.reshape(X_train.shape[0], -1)
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

rforest = RandomForestClassifier(n_estimators=100, random_state=42)

rforest.fit(X_train_flattened, Y_train)
```

```
Y_pred = rforest.predict(X_test_flattened)

accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy:", accuracy)
```

[Model4. Sequential Model]

```
from keras.models import Sequential
from keras.layers import Dense, Flatten

model_1 = Sequential()

model_1.add(Flatten(input_shape=X_train.shape[1:]))

model_1.add(Dense(128, activation='relu'))

model_1.add(Dense(3, activation='softmax'))

model_1.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

```
model_1.summary()

history_1 = model_1.fit(X_train, Y_train, epochs=10, batch_size=32, validation_data=(X_test, Y_test))

test_loss, test_accuracy = model_1.evaluate(X_test, Y_test)
print("Test Accuracy:", test_accuracy)
```

[Model5. Decision Tree]

```
import numpy as np
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)

# 모델 초기화 및 학습
```

```
#gb = GradientBoostingClassifier()
#gb.fit(X_train_flat, Y_train)

# 예측 및 평가
#Y_pred = gb.predict(X_test_flat)
#accuracy = accuracy_score(Y_test, Y_pred)
#print(f"Test Accuracy: {accuracy:.4f}")

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, Y_train)

Y_pred = clf.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")
print(classification_report(Y_test, Y_pred, target_names=['REM',
'Wake', 'NREM']))
```

```
print("Confusion Matrix:")
print(confusion_matrix(Y_test, Y_pred))
```

[Model6. SVC]

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

clf = svm.SVC(kernel='linear')

X_train_reshaped = X_train.reshape(X_train.shape[0], -1)
X_test_reshaped = X_test.reshape(X_test.shape[0], -1)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_reshaped)
X_test_scaled = scaler.transform(X_test_reshaped)
```

```
clf = svm.SVC(kernel='linear')
clf.fit(X_train_scaled, Y_train)

predictions = clf.predic(X_test_scaled)
print("Accuracy:", accuracy_score(Y_test, predictions))
```

[Model7. KNN]

```
k_values = list(range(1, 11, 2))  # 1, 3, 5, 7, 9
param_grid = {'n_neighbors': k_values}

knn = KNeighborsClassifier()
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')

# 모델 훈련
grid_search.fit(X_train_scaled, Y_train)

# 최적의 K 값 출력
best_k = grid_search.best_params_['n_neighbors']
```

```
print(f'Best K value: {best_k}')

# 최적의 K 값으로 모델 재훈련

knn_best = KNeighborsClassifier(n_neighbors=best_k)

knn_best.fit(X_train_scaled, Y_train)


# 테스트 데이터로 예측 및 정확도 출력

Y_pred = knn_best.predict(X_test_scaled)

accuracy = accuracy_score(Y_test, Y_pred)

print(f'Accuracy: {accuracy:.2f}')
```