Sleep stage classification using mice brain signals

TEAM 18
송민영 (22000374) / 심성환 (22000397) / 채정원 (22100725) / 하지민 (22100766)

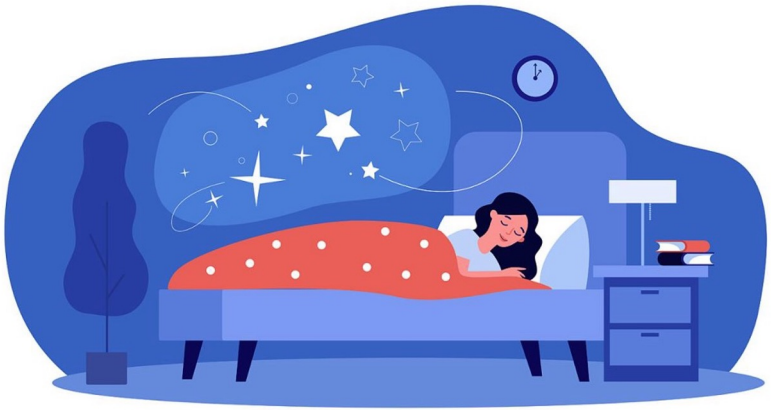
Problem description

To develop an accurate and efficient AI model to classify the sleep stages of mice

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.

Background



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

Data collection

CNN

```
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)
```

Random Forest, MLP, Decision Tree, SVC, KNN

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

Sequential

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

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.

1) CNN

Use Case: Excellent for processing images and time-series data.

Components: Convolutional layers, pooling layers, fully connected layers.

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

2) MLP

Use Case: Suitable for various types of classification and regression tasks.

Components: Input layer, hidden layers, output layer.

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

3) Random Forest

Use Case: High performance in classification and regression tasks.

Components: Ensemble of multiple decision trees.

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

4) Sequential Model

Use Case: Ideal for sequential data processing, particularly time-series data.

Components: Stacked layers in a sequential manner (e.g., LSTM, GRU).

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

5) KNN

Use Case: Simple pattern recognition, classification, and regression.

Components: Comparison of a data point with its K nearest neighbors.

6) Decision Tree

Use Case: Clear rule-based classification and regression.

Components: Tree structure consisting of nodes and edges.

7) SVM

Use Case: Effective for classification and regression in high-dimensional spaces.

Components: Optimal hyperplane to separate data.

Evaluation

1) Classify Train, Test Data
   Evaluate base on Test Data

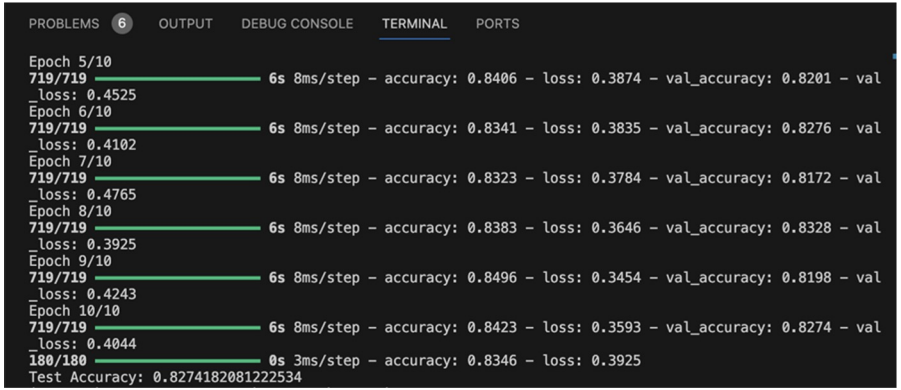
```
X_train = np.load('X_train_mouse02.npy')
Y_train = np.load('Y_train_mouse02.npy')

# For testing data (day 5)
# testing data -> 5748
X_test = np.load('X_test_mouse02.npy')
Y_test = np.load('Y_test_mouse02.npy')
```

2)

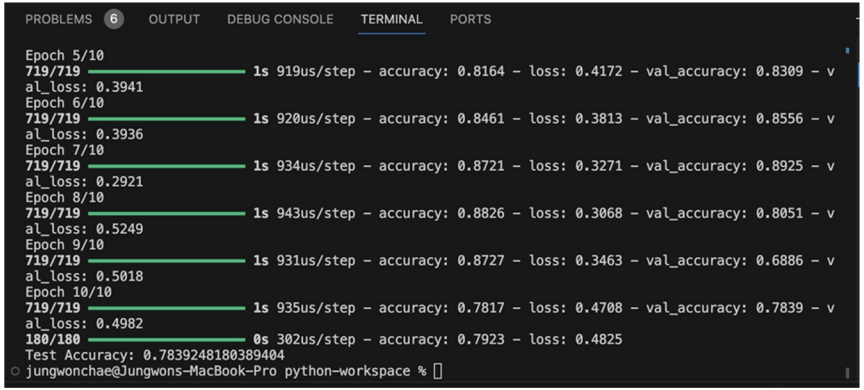
```
# Decision Tree Training
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, Y_train)

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

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

Accuracy result by model

1) CNN

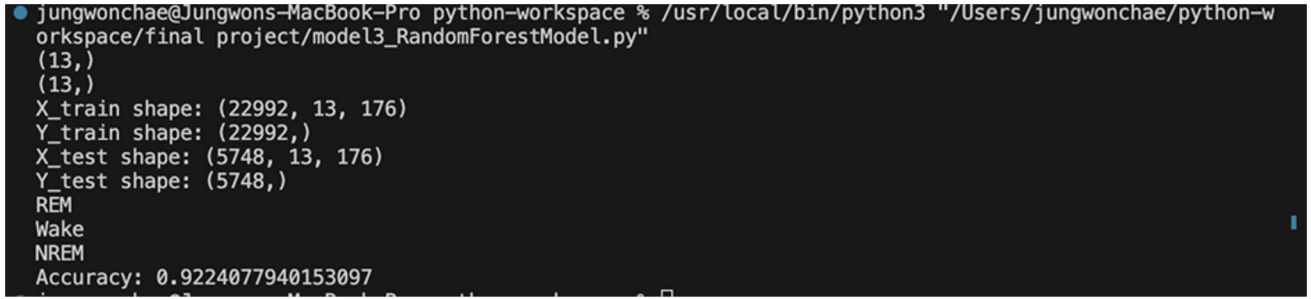


2) MLP



Accuracy result by model

3) Random Forest



Accuracy result by model

4) Sequential

```
Epoch 5/10
719/719 ━━━━━━━━━━━━━━━━━━━━ 1s 918us/step - accuracy: 0.8207 - loss: 0.4197 - val_accuracy: 0.8206 - v
al_loss: 0.3571
Epoch 6/10
719/719 ━━━━━━━━━━━━━━━━━━━━ 1s 911us/step - accuracy: 0.8163 - loss: 0.3918 - val_accuracy: 0.8241 - v
al_loss: 0.3341
Epoch 7/10
719/719 ━━━━━━━━━━━━━━━━━━━━ 1s 915us/step - accuracy: 0.8267 - loss: 0.3780 - val_accuracy: 0.8285 - v
al_loss: 0.4282
Epoch 8/10
719/719 ━━━━━━━━━━━━━━━━━━━━ 1s 907us/step - accuracy: 0.8153 - loss: 0.4268 - val_accuracy: 0.8314 - v
al_loss: 0.3971
Epoch 9/10
719/719 ━━━━━━━━━━━━━━━━━━━━ 1s 917us/step - accuracy: 0.8187 - loss: 0.4189 - val_accuracy: 0.8185 - v
al_loss: 0.4424
Epoch 10/10
719/719 ━━━━━━━━━━━━━━━━━━━━ 1s 886us/step - accuracy: 0.8162 - loss: 0.4178 - val_accuracy: 0.8304 - v
al_loss: 0.4115
180/180 ━━━━━━━━━━━━━━━━━━━━ 0s 280us/step - accuracy: 0.8329 - loss: 0.4053
Test Accuracy: 0.8303757905960083
jungwonchae@Jungwons-MacBook-Pro python-workspace %
```

Accuracy result by model

5) KNN

```
Best K value: 9
Accuracy: 0.87
```

6) Decision Tree

```
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
```

Accuracy result by model

7) SVM

```
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
```

Conclusion

1) Random Forest > SVM > KNN > CNN > MLP ~ Sequential ~ Decision

1) High accuracy is important.

   However, choose the best model for specific situations.