

```

1  import torch
2  import numpy as np
3  from sklearn.ensemble import StackingClassifier, RandomForestClassifier
4  from sklearn.linear_model import LogisticRegression
5  from sklearn.svm import SVC
6  from sklearn.neighbors import KNeighborsClassifier
7  from sklearn.metrics import accuracy_score
8  import joblib
9
10 # 1. 전처리된 데이터 불러오기
11 X_train_np = np.load("/data/eyes_data/X_train.npy")
12 y_train_np = np.load("/data/eyes_data/y_train.npy")
13 X_test_np = np.load("/data/eyes_data/X_test.npy")
14 y_test_np = np.load("/data/eyes_data/y_test.npy")

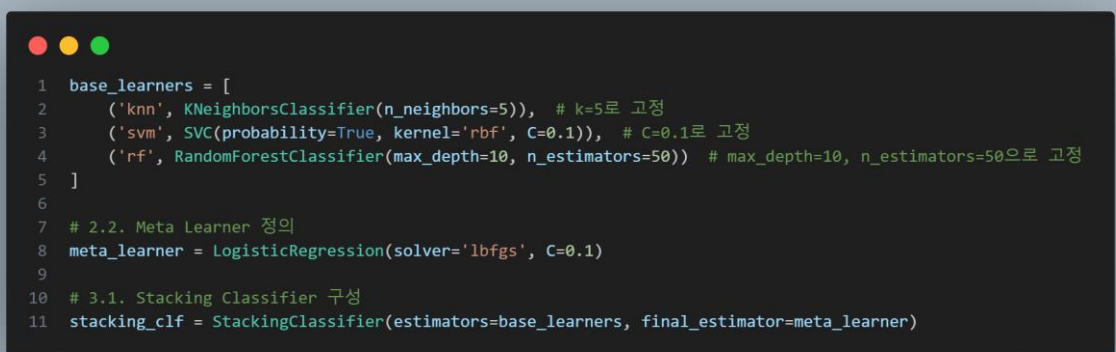
```

1. 데이터 로드

- 학습 데이터: X_train(입력 특성)과 y_train(레이블)
- 테스트 데이터: X_test와 y_test를 각각 Numpy 배열로 불러옵니다.

2. 데이터 구조

- X_train_np와 X_test_np: 이미지 데이터를 전처리 후 벡터 형태로 변환한 입력 데이터
- y_train_np와 y_test_np: 대응하는 클래스 레이블



```

1  base_learners = [
2      ('knn', KNeighborsClassifier(n_neighbors=5)), # k=5로 고정
3      ('svm', SVC(probability=True, kernel='rbf', C=0.1)), # C=0.1로 고정
4      ('rf', RandomForestClassifier(max_depth=10, n_estimators=50)) # max_depth=10, n_estimators=50으로 고정
5  ]
6
7  # 2.2. Meta Learner 정의
8  meta_learner = LogisticRegression(solver='lbfgs', C=0.1)
9
10 # 3.1. Stacking Classifier 구성
11 stacking_clf = StackingClassifier(estimators=base_learners, final_estimator=meta_learner)

```

1. Base Learners (기본 모델)

- **KNeighborsClassifier**: k=5로 설정된 k-Nearest Neighbors

- **SVC (Support Vector Classifier)**: 커널 rbf와 C=0.1을 사용한 서포트 벡터 머신
- **RandomForestClassifier**: max_depth=10, n_estimators=50의 설정으로 결정 트리 앙상블

2. Meta Learner (최종 분류기)

- **LogisticRegression**: C=0.1로 학습된 로지스틱 회귀 모델
- Base Learners가 예측한 결과를 Meta Learner가 입력받아 최종 예측을 수행합니다.

3. Stacking Classifier

- Base Learners에서 예측한 결과를 Meta Learner에 전달하는 **Stacking 앙상블** 기법입니다.

```

1  # 4. 모델 학습
2  print("Training Stacking Classifier...")
3  stacking_clf.fit(X_train_np, y_train_np)
4
5  # 5.1. Train 데이터 평가
6  y_train_pred = stacking_clf.predict(X_train_np)
7  train_accuracy = accuracy_score(y_train_np, y_train_pred)
8  print(f"Train Accuracy: {train_accuracy * 100:.2f}%")
9
10 # 5.2. Test 데이터 평가
11 y_pred = stacking_clf.predict(X_test_np)
12 test_accuracy = accuracy_score(y_test_np, y_pred)
13 print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
14
15 # 6. 학습된 모델 저장
16 model_path = "/data/eyes_data/best_stacking_model.pkl"
17 joblib.dump(stacking_clf, model_path)
18 print(f"Model saved to {model_path}")

```

1. 모델 학습

- fit() 함수를 사용해 학습 데이터를 기반으로 **Stacking Classifier**를 학습시

킵니다.

2. Train 데이터 평가

- 학습 데이터에 대한 예측 결과를 사용하여 **Train Accuracy**를 계산합니다.

3. Test 데이터 평가

- 테스트 데이터에 대한 예측 결과를 사용하여 **Test Accuracy**를 계산하고 성능을 확인합니다.

4. 모델 저장

- joblib 라이브러리를 사용해 학습된 **Stacking Classifier** 모델을 **.pkl** 파일로 저장합니다.
- 저장된 모델은 이후에 재사용하거나 배포 시 활용할 수 있습니다.

5. 경로 설정

- 모델은 `/data/eyes_data/` 경로에 `best_stacking_model.pkl`이라는 이름으로 저장됩니다.