<Homework 5 프로그램 소개 문서>

구현 해야 하는 목록들

1. Quality 열을 라벨로 사용(원 핫 인코딩 사용), 나머지를 입력 데이터로 설정
2. train, test 비율은 8:2
3. 모델 구조는 아래 그림을 참고하되, 성능 향상을 위해 변경해도 좋음
4. 손실 함수는 cross-entropy를 사용하고 Early-Stopping 적용 (val_loss로 적용)
5. 학습은 k-fold 교차 검증을 사용 (k=5)
6. 각 fold에 대한 정확도 시각화
7. test 셋에 대한 정확도 출력

```
import pandas as pd
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping, ModelCheckpoint
from google.colab import drive
import numpy as np
import matplotlib.pyplot as plt

drive.mount('/content/drive')

#데이터셋 읽기
df = pd.read_csv('./drive/MyDrive/Colab Notebooks/data_2023/dataset/winequality-red.csv')

print(df['quality'].unique())
```

```
Mounted at /content/drive
[5 6 7 4 8 3]
```

먼저 코랩에 드라이브를 연동해주고, 사용할 데이터 셋을 읽을 수 있도록 코드를 구현하였습니다. 해당 csv 파일은 data_2023/dataset 폴더 내에 winequality-red.csv로 저장되어 있습니다.

<1번 구현>

```
# 'quality' 열에 대한 원-핫 인코딩
y = pd.get_dummies(df['quality'])

# 'quality' 열을 제외한 나머지를 입력 데이터로 사용
X = df.drop('quality', axis=1)

# 데이터셋의 차원을 확인 (디버깅을 위해)
print(X.shape, y.shape)
```

```
(1599, 11) (1599, 6)
```

Quality 열에 대한 원-핫 인코딩을 사용하였고, 나머지를 입력 데이터로 사용하였습니다.

<2번 구현>

```
# 데이터 정규화
# 성능 향상을 위해 정규화 부분을 추가해줌
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# train과 test set 나누기
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

따로 ppt에 구현사항에는 없지만, 성능 향상을 위해 정규화 부분을 추가해주었습니다.
train과 test를 8:2 비율로 나눴습니다.

<3번, 4번, 5번 구현>

```
# k-폴드 교차 검증 설정
k = 5
kf = KFold(n_splits=k, shuffle=True, random_state=42)
fold_no = 1  # fold_no 변수 초기화
acc_per_fold = []

# KFold 교차 검증 수행
for train, val in kf.split(X_train, y_train):
    # 훈련 데이터와 검증 데이터 분할
    X_train_fold, X_val_fold = X_train[train], X_train[val]
    y_train_fold, y_val_fold = y_train.iloc[train], y_train.iloc[val]
    # 모델 구축
    model = Sequential()
    model.add(Dense(64, input_shape=(X_train_fold.shape[1],), activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(y_train_fold.shape[1], activation='softmax'))
    model.summary()

    # 모델 컴파일
    model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])

    # 체크포인트 및 얼리 스토핑 설정
    checkpointer = ModelCheckpoint('best_model_fold_{}.h5'.format(fold_no), save_best_only=True, monitor='val_loss', mode='min')
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)

    # 모델 학습
    history = model.fit(X_train_fold, y_train_fold, epochs=370, batch_size=36, validation_data=(X_val_fold, y_val_fold), callbacks=[early_stopping, checkpointer], verbose=1)

     # 성능 평가
    scores = model.evaluate(X_val_fold, y_val_fold, verbose=0)
    acc_per_fold.append(scores[1] * 100)
    fold_no += 1
```

k-fold 교차 검증을 k=5로 설정하여 수행하였고, 손실 함수는 cross-entropy를 사용하고 Early-Stopping 적용하였습니다.
Epoch는 370, batch size는 36으로 설정하였습니다.

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_24 (Dense)            (None, 64)                768

 dense_25 (Dense)            (None, 32)                2080

 dense_26 (Dense)            (None, 6)                 198

=================================================================
Total params: 3046 (11.90 KB)
Trainable params: 3046 (11.90 KB)
Non-trainable params: 0 (0.00 Byte)
```

모델 구조는 다음과 같습니다.

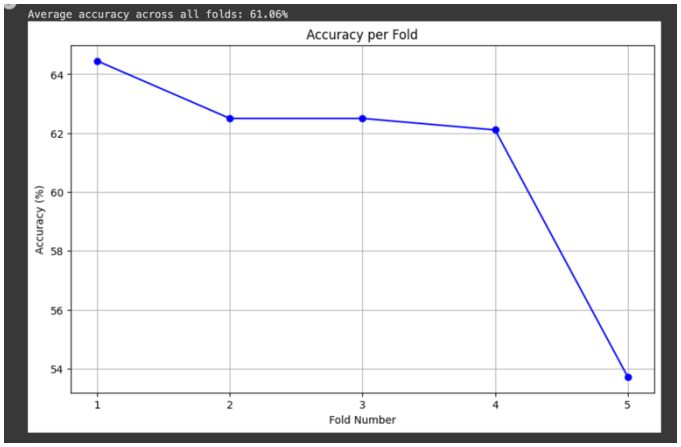
<6번 구현>

```
# 교차 검증 결과 분석
average_accuracy = np.mean(acc_per_fold)
print(f'Average accuracy across all folds: {average_accuracy:.2f}%')

# 정확도 시각화
plt.figure(figsize=(10, 6))
plt.plot(range(1, k+1), acc_per_fold, marker='o', linestyle='-', color='b')
plt.title('Accuracy per Fold')
plt.xlabel('Fold Number')
plt.ylabel('Accuracy (%)')
plt.xticks(range(1, k+1))
plt.grid(True)
plt.show()
```

시각화 결과는 다음과 같습니다.

Average accuracy across all folds = 61.06% 가 나왔습니다.




```
# 전체 데이터셋에 대한 모델 성능 평가
# 전체 데이터셋에 대해 동일한 모델 아키텍처를 사용하여 학습 및 평가를 수행함
model = Sequential()
model.add(Dense(64, input_shape=(X_scaled.shape[1],), activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(y.shape[1], activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
model.fit(X_scaled, y, epochs=100, batch_size=32, verbose=1)
```

전체 데이터셋에 대한 모델 성능 평가는 위와 같습니다.

Epoch는 100, batch size는 32로 설정했습니다.

<7번 구현>

```
# 최종 테스트 세트에서 성능 평가
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Test Set Accuracy: {test_accuracy*100:.2f}%')
```

```
Test Set Accuracy: 61.87%
```

최종 테스트 셋에 대한 성능 평가 구현코드 입니다.

Test Set Accuracy = 61.87% 가 나왔습니다.