



# 딥러닝 1차 과제

과목명	딥러닝
담당 교수님	이광엽 교수님
전공	컴퓨터공학과
학번	2022307022
이름	서현은
제출일	2023.10.11

문제 1>

1. 다음 경로를 이용하여 winequality-red.csv를 읽어 red 데이터를 만든다.

① 'red' 데이터 처음 세줄(3행)을 print 출력하고 열(column)과 행(row)이 각각 몇 개인지 적으시오.

[Source]

```
import pandas as pd

# 'winequality-red.csv' 파일을 인터넷에서 읽어와 'red' 데이터프레임 생성
red = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv', sep=';')

print(red.head(3)) # 'red' 데이터프레임의 처음 세 줄 출력
print()
print("행: {}".format(len(red.index))) # 열(column)의 개수 출력
print("열: {}".format(len(red.columns))) # 행(row)의 개수 출력
```

[Output]

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	W
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	W
0	11.0	34.0	0.9978	3.51	0.56	
1	25.0	67.0	0.9968	3.20	0.68	
2	15.0	54.0	0.9970	3.26	0.65	

	alcohol	quality
0	9.4	5
1	9.8	5
2	9.8	5

행: 1599  
열: 12

② 열(column)가운데 임의로 5개만 선택하여 'red' 데이터를 다시 만들고 처음 5행을 print 출력한다.

[Source]

```
import pandas as pd

red = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv', sep=';')

selected_columns = [1, 3, 5, 7, 9] # 임의로 선택할 열(column) 인덱스
red_selected = red.iloc[:, selected_columns] # 선택된 열로 새로운 "red" 데이터 생성

print(red_selected.head()) # 처음 5행 출력
```

[Output]

	volatile acidity	residual sugar	free sulfur dioxide	density	sulphates
0	0.70	1.9	11.0	0.9978	0.56
1	0.88	2.6	25.0	0.9968	0.68
2	0.76	2.3	15.0	0.9970	0.65
3	0.28	1.9	17.0	0.9980	0.58
4	0.70	1.9	11.0	0.9978	0.56

③ ②번에서 선택한 5개 열 가운데 임의로 한 개를 선택하여 본인 영어이름으로 column name을 바꾸고 이 column을 index로 한다. 처음 5행을 print 출력한다.

[Source]

```
import numpy as np
import pandas as pd

red = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv', sep=';')

selected_columns = [1, 3, 5, 7, 9]          #임의로 선택할 열(column) 인덱스
rename_col_index = 2                       #선택할 열(column) 인덱스

red = red.iloc[:, selected_columns]        #선택한 열만 포함하는 데이터프레임 생성
rename_col = red.columns[rename_col_index] #선택한 열의 이름 가져오기
red.rename(columns={rename_col: 'SeoHyuneun'}, inplace=True) #열의 이름을 변경
red = red.set_index('SeoHyuneun')         #열을 인덱스로 지정
print(red.head())
```

[Output]

	volatile acidity	residual sugar	density	sulphates
SeoHyuneun				
11.0	0.70	1.9	0.9978	0.56
25.0	0.88	2.6	0.9968	0.68
15.0	0.76	2.3	0.9970	0.65
17.0	0.28	1.9	0.9980	0.58
11.0	0.70	1.9	0.9978	0.56

문제 2>

1. column이 3열, row가 5행인 pandas data를 임의로 만든다. 이때, 데이터값은 0~100사이의 임의의 값으로 한다. column name은 임의로 한다. 만들어진 pandas data를 전체 print 출력한다.

2. 위 pandas data를 이용하여 다음처럼 split하여 4개의 데이터를 만들고 각각을 전체 print 출력한다. 데이터 이름은 임의로 한다.

- ① 1, 2열, 1, 2, 3행 데이터
- ② 3열, 1, 2, 3행 데이터
- ③ 1, 2열, 4, 5행 데이터
- ④ 3열, 4, 5행 데이터

[Source]

```
import pandas as pd

num_columns = 3 #column 개수
num_rows = 5 #row 개수

column_names = ['A', 'B', 'C'] #column 이름 리스트

#임의의 값을 가진 데이터프레임 생성
data = [
    [10, 20, 30],
    [40, 50, 60],
    [70, 80, 90],
    [15, 25, 35],
    [45, 55, 65]
]

df = pd.DataFrame(data, columns=column_names)
#1번 (전체 데이터프레임 출력)
print(df)
print("=====")

#2번
df_1 = df.iloc[:3,:2] #열: A,B / 행: (1~3)
df_2 = df.iloc[:3,2:3] #열: C / 행: (1~3)
df_3 = df.iloc[3,:2] #열: A,B / 행: (4~5)
df_4 = df.iloc[3,2:] #열: C / 행: (4~5)
print(df_1)
print("-----") #구분선
print(df_2)
print("-----")
print(df_3)
print("-----")
print(df_4)
```

[Output]

```

    A  B  C
0  10  20  30
1  40  50  60
2  70  80  90
3  15  25  35
4  45  55  65
=====
    A  B
0  10  20
1  40  50
2  70  80
-----
    C
0  30
1  60
2  90
-----
    A  B
3  15  25
4  45  55
-----
    C
3  35
4  65
```

3 . 1번의 데이터에서 한 column에 있는 5개 숫자의 합(sum)과 평균(average)를 만든다. 3열에 대하여 각각 수행한다. 이 과정을 for문을 이용하여 코딩한다. 계산된 합과 평균을 2행 3열의 pandas 데이터로 만든다.

4. 3번의 합과 평균 pandas 데이터를 1번 pandas 데이터와 결합하여 3열 7행의 새로운 pandas 데이터를 만든다. 이때, pandas의 concat 메서드를 사용한다. 새로운 pandas 전체를 print 출력한다.

[Source]

```
import pandas as pd
import numpy as np

data = [
    [10, 20, 30],
    [40, 50, 60],
    [70, 80, 90],
    [15, 25, 35],
    [45, 55, 65]
]

df = pd.DataFrame(data)

result_data = []

for col in range(3):
    column_sum = df[col].sum() #열의 합 계산
    column_avg = df[col].mean() #열의 평균 계산

    result_data.append([column_sum, column_avg]) #결과 데이터 추가

result_df = pd.DataFrame(result_data) #결과 데이터프레임 생성
result_df.columns = ['Sum', 'Average'] #열 이름 설정

#3번
print(result_df) #결과 출력
print("=====") #구분선

#4번
#두 개의 데이터프레임을 결합하여 새로운 데이터프레임 생성
new_df = pd.concat([df, result_df.T]) #result_df 전치
print(new_df)
```

[Output]

	Sum	Average	
0	180	36.0	
1	230	46.0	
2	280	56.0	
=====			
	0	1	2
0	10.0	20.0	30.0
1	40.0	50.0	60.0
2	70.0	80.0	90.0
3	15.0	25.0	35.0
4	45.0	55.0	65.0
Sum	180.0	230.0	280.0
Average	36.0	46.0	56.0

문제 3>

1. 교재 코드 3-16부터 3-20까지 실행한다. 이때 코드3-17에서 교재와 다른 데이터 이미지 5장을 시각화하여 출력한다. 또한, 코드3-20 결과를 출력한다.

[코드 3-16]

```
#라이브러리 호출 및 데이터 준비
%matplotlib inline
from sklearn.datasets import load_digits
digits = load_digits() #숫자 데이터셋(digits)은 사이킷런에서 제공

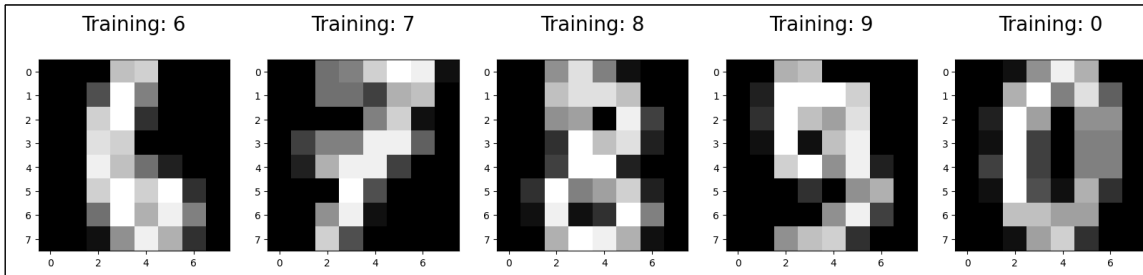
print("Image Data Shape" , digits.data.shape) #digit 데이터셋의 형태(이미지 1797개, 8x8이미지의 64차원을 가짐)
print("Label Data Shape", digits.target.shape) #레이블 이미지 1797개가 있음
```

```
Image Data Shape (1797, 64)
Label Data Shape (1797,)
```

[코드 3-17]

```
#digits 데이터셋의 시각화
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.data[6:11], digits.target[6:11])):
    plt.subplot(1, 5, index + 1)
    plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
    plt.title('Training: %iWn' % label, fontsize = 20)
```



[코드 3-18]

```
#훈련과 테스트 데이터셋 분리 및 로지스틱 회귀 모델 생성
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.25, random_state=0)

from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression() #로지스틱 회귀 모델의 인스턴스 생성
logisticRegr.fit(x_train, y_train) #모델 훈련
```

```
LogisticRegression
LogisticRegression()
```

[코드 3-19]

```
#일부 데이터를 사용한 모델 예측
logisticRegr.predict(x_test[0].reshape(1,-1))
logisticRegr.predict(x_test[0:10])
```

```
array([2, 8, 2, 6, 6, 7, 1, 9, 8, 5])
```

### [코드 3-20]

```
#전체 데이터를 사용한 모델 예측
predictions = logisticRegr.predict(x_test)
score = logisticRegr.score(x_test, y_test)
print(score)
```

```
0.9511111111111111
```

2. 테스트 이미지 3장을 PC 그림판에서 아래와 같이 만들고 `model.predict(X)` 실행하여 3장의 `accuracy`를 출력한다. --> 코드와 테스트 이미지, `accuracy`를 제출한다.

[Source]

```
import numpy as np
import pandas as pd
from PIL import Image

#이미지 경로 리스트
image_paths = ['/content/1.png', '/content/2.png', '/content/3.png']
X = []

#이미지 경로를 순회하며 이미지 불러오기 및 전처리
for path in image_paths:
    img = Image.open(path).convert("L") #이미지 불러오기 및 흑백 변환
    img = np.array(img) #NumPy 배열로 변환
    img = img / 255.0 #정규화

    flattened_img = []
    for i in range(8):
        for j in range(8):
            flattened_img.append(img[i][j]) #픽셀 값을 리스트에 추가

    X.append(flattened_img) #전처리된 이미지를 X 리스트에 추가

predictions = []
label=[1, 2, 3]

#예측값 계산 및 저장
for x in X:
    prediction=logisticRegr.predict([x])
    predictions.append(prediction)

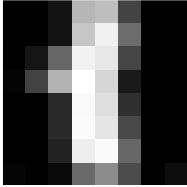
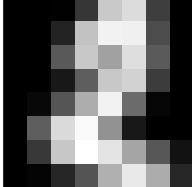
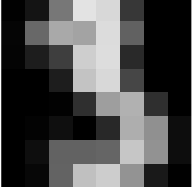
score=logisticRegr.score(X,label)
print("Predictions =", predictions)
print("Accuracy =", score)
```

[Output]

```
Predictions = [array([1]), array([2]), array([3])]
Accuracy = 1.0
```



[Image]

Image			
Path	/content/1.png	/content/2.png	/content/3.png
Prediction	1	2	3

문제 4>

1. [문제2]에서 만든 pandas 데이터에서 임의로 2개 열을 선택한 후 각각을 x, y로 한다. x, y를 코드 3-24을 이용하여 데이터 간의 관계를 시각화하고 결과를 출력하시오.

[Source]

```
import pandas as pd
import matplotlib.pyplot as plt

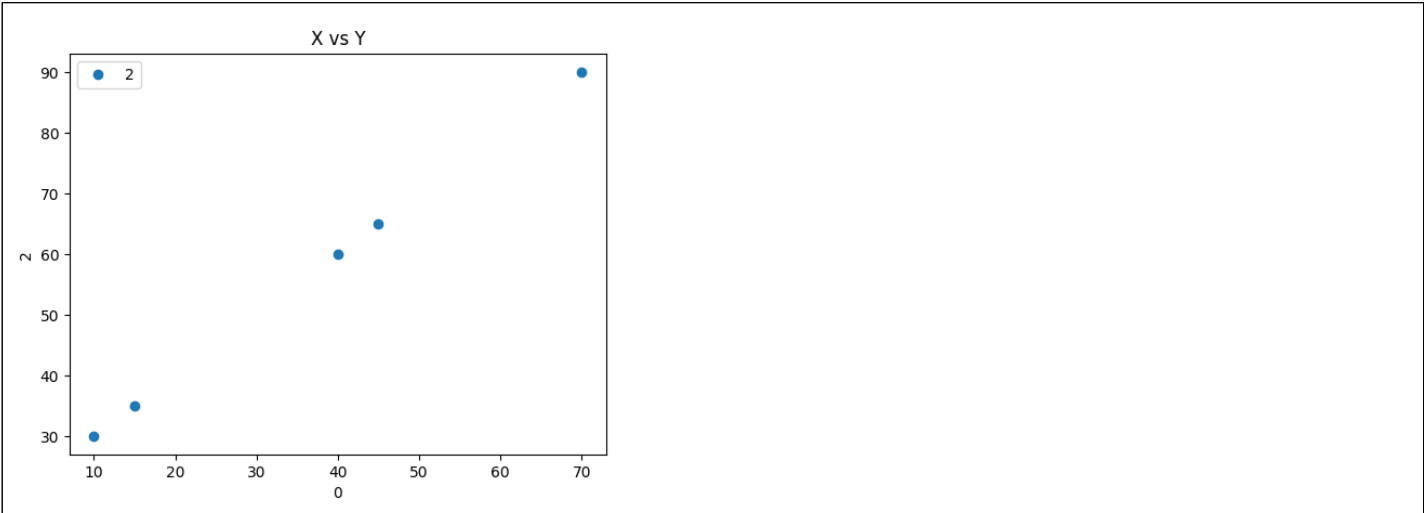
#[문제2]에서 생성한 데이터
data = [
    [10, 20, 30],
    [40, 50, 60],
    [70, 80, 90],
    [15, 25, 35],
    [45, 55, 65]
]

df = pd.DataFrame(data)

#x와 y로 선택할 열 지정
x_col = df.columns[0] #첫 번째 열 선택
y_col = df.columns[2] #세 번째 열 선택

#데이터 간의 관계 시각화 및 출력
df.plot(x=x_col, y=y_col, style='o')
plt.title('X vs Y')
plt.xlabel(x_col)
plt.ylabel(y_col)
plt.show()
```

[Output]



2. 1번 데이터를 이용하여 코드 3-25를 실행한 후 코드 3-27을 이용하여 선형 회귀 예측 결과를 회귀선으로 시각화 하고 결과를 출력하시오. (test\_size는 자유롭게 조절 가능함)

#### [코드 3-25]

#데이터를 독립변수와 종속변수로 분리하고 선형회귀모델 생성

```
from sklearn.linear_model import LinearRegression
```

```
X = df[x_col].values.reshape(-1,1)
```

```
y = df[y_col].values.reshape(-1,1)
```

#데이터의 55%를 훈련 데이터셋으로, 45%를 검증 데이터셋으로 분할

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.45)
```

```
regressor = LinearRegression() #선형회귀 클래스를 가져옴
```

```
regressor.fit(X_train,y_train) #fit() 메서드를 사용하여 모델 훈련
```

LinearRegression

LinearRegression()

#### [코드 3-27]

#테스트 데이터셋을 사용한 회귀선 표현

```
plt.scatter(X_test,y_test,color='gray')
```

```
plt.plot(X_test,y_pred,color='red',linewidth=2)
```

```
plt.show()
```

