딥러닝 무작정 따라하기

1.환경준비

- 라이브러리 Import

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

2.Regression : Advertising

(1) 데이터 전처리

1) 데이터 준비

```
path = 'https://raw.githubusercontent.com/DA4BAM/dataset/master/advertising.csv'
adv = pd.read_csv(path)
adv.head()
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Next steps: Generate code with adv View recommended plots

```
target = 'Sales'
x = adv.drop(target, axis=1)
y = adv.loc[:, target]
```

2) 가변수화

```
코딩을 시작하거나 AI로 코드를 생성하세요.
```

3) 데이터분할

```
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

(2) ML 연습 : 선형회귀

```
# 선형회귀 알고리즘을 불러 옵시다.
from sklearn.linear_model import LinearRegression
```

1) 모델 선언

```
model = LinearRegression()
```

2) 학습

```
model.fit(x_train, y_train)
```

```
LinearRegression
LinearRegression()
```

3) 예측

```
pred = model.predict(x_val)
```

4) 검증

만든 모델은 얼마나 정확한지 검증해 봅시다.

```
print(f'RMSE  : {mean_squared_error(y_val, pred, squared=False)}')
print(f'MAE   : {mean_absolute_error(y_val, pred)}')
print(f'MAPE  : {mean_absolute_percentage_error(y_val, pred)}')
```

```
RMSE  : 1.8716493530685259
MAE   : 1.4155875681427736
MAPE  : 0.14622589325825738
```

(3) 딥러닝 모델링

- 필요한 함수들 불러오기
- 모델 선언
- 학습
- 예측
- 성능 검증

1) 전처리 : Scaling

```
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_val = scaler.transform(x_val)
```

2) 필요한 함수들 불러오기

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

3) 모델 선언

```
nfeatures = x_train.shape[1] #num of columns # 컬럼 갯수 가져오기
nfeatures
```

```
3
```

```
# 메모리 정리(선택, 일반적으로 많이 사용)
clear_session()

# Sequential 타입 모델 선언
model = Sequential( Dense(1, input_shape = (nfeatures,))  )

# 모델요약(선택, 일반적으로 많이 사용)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 4

=================================================================
Total params: 4 (16.00 Byte)
Trainable params: 4 (16.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# 컴파일
model.compile(optimizer='adam', loss='mse')
```

4) 학습

```
model.fit(x_train, y_train)
```

```
5/5 [==============================] - 1s 16ms/step - loss: 230.2092
<keras.src.callbacks.History at 0x7ce2a0da4f10>
```

5) 예측

```
pred = model.predict(x_val)
```

```
2/2 [==============================] - 0s 6ms/step
```

6) 검증

만든 모델은 얼마나 정확한지 검증해 봅시다.

```
print(f'RMSE  : {mean_squared_error(y_val, pred, squared=False)}')
print(f'MAE   : {mean_absolute_error(y_val, pred)}')
print(f'MAPE  : {mean_absolute_percentage_error(y_val, pred)}')
```

```
RMSE  : 15.609694588191559
MAE   : 14.316343239620329
MAPE  : 1.0171074366946442
```

3.Regression : Carseat

(1) 데이터 전처리

- 데이터 준비
- 가변수화
- 스케일링(필요하다면)
- 데이터 분할

1) 데이터 준비

- 카시트 판매량 데이터

변수명	설명	구분
Sales	**각 지역 판매액(단위 : 1000달러)**	**Target**
CompPrice	지역별 경쟁사 판매가격(달러)	feature
Income	가구당 평균 소득액(1000달러)	feature
Advertising	각 지역, 회사의 광고 예산(1000달러)	feature
Population	지역 인구수(단위 : 1000명)	feature
Price	자사 지역별 판매가격(달러)	feature
ShelveLoc	진열상태(범주 : Bad, Medium, Good)	feature
Age	지역 인구의 평균 연령	feature
Education	교육수준(범주 : 10~18)	feature
Urban	매장이 도심에 있는지 여부(범주 : Yes, No)	feature
US	매장이 미국에 있는지 여부(범주 : Yes, No)	feature

- 데이터 경로 : https://raw.githubusercontent.com/DA4BAM/dataset/master/Carseats.csv

```
path = 'https://raw.githubusercontent.com/DA4BAM/dataset/master/Carseats.csv'
carseat = pd.read_csv(path)
carseat.head()
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban
0	9.50	138	73	11	276	120	Bad	42	17	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes

Next steps: Generate code with carseat View recommended plots

```
target = 'Sales'
x = carseat.drop(target, axis=1)
y = carseat.loc[:, target]
```

2) 가변수화

```
cat_cols = ['ShelveLoc', 'Education', 'US', 'Urban']
x = pd.get_dummies(x, columns = cat_cols, drop_first = True)
```

3) 데이터분할

```
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

4) Scaling

```
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_val = scaler.transform(x_val)
```

(2) 모델링

- 필요한 함수들 불러오기
- 모델 선언
- 학습
- 예측
- 성능 검증

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

1) 모델 선언

```
x_train.shape
```

```
(320, 18)
```

```
nfeatures = x_train.shape[1] #num of columns
nfeatures
```

```
18
```

```
# 메모리 정리(필수는 아님!)
clear_session()

# Sequential 타입 모델 선언
model = Sequential(Dense(1, input_shape = (nfeatures, )) )

# 모델요약
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 19

=================================================================
Total params: 19 (76.00 Byte)
Trainable params: 19 (76.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# 컴파일
model.compile(optimizer='adam' ,loss='mse')
```

2) 학습

```
model.fit(x_train, y_train)
```

```
10/10 [==============================] - 1s 4ms/step - loss: 65.6738
<keras.src.callbacks.History at 0x7ce2a0471660>
```

3) 예측

```
pred = model.predict(x_val)
```

```
3/3 [==============================] - 0s 7ms/step
```

4) 검증

만든 모델은 얼마나 정확한지 검증해 봅시다.

```
print(f'RMSE  : {mean_squared_error(y_val, pred, squared=False)}')
print(f'MAE   : {mean_absolute_error(y_val, pred)}')
```

```
RMSE  : 8.040140807170927
MAE   : 7.471095470392146
```

4.Regression : 보스턴 집값

(1) 데이터 전처리

1) 데이터 준비

```
path = 'https://raw.githubusercontent.com/DA4BAM/dataset/master/boston.csv'
boston = pd.read_csv(path)
boston.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	36.2

Next steps: Generate code with boston View recommended plots

```
target = 'medv'
x = boston.drop(target, axis=1)
y = boston.loc[:, target]
```

2) 가변수화

3) 데이터분할

```
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

4) Scaling

```
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_val = scaler.transform(x_val)
```

(2) 모델링

- 필요한 함수들 불러오기
- 모델 선언
- 학습
- 예측
- 성능 검증

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

1) 모델 선언

```
x_train.shape
```

```
(404, 12)
```

```
nfeatures = x_train.shape[1] #num of columns
nfeatures
```

```
12
```

```
# 메모리 정리(필수는 아님!)
clear_session()

# Sequential 타입 모델 선언
model = Sequential(Dense(1, input_shape=(nfeatures,)))

# 모델요약
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 13

=================================================================
Total params: 13 (52.00 Byte)
Trainable params: 13 (52.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# 컴파일
model.compile(optimizer='adam', loss='mse')
```

2) 학습

```
model.fit(x_train, y_train)
```

```
13/13 [==============================] - 1s 11ms/step - loss: 637.2553
<keras.src.callbacks.History at 0x7ce337088e80>
```

3) 예측

```
pred = model.predict(x_val)
```

```
4/4 [==============================] - 0s 7ms/step
```

4) 검증

만든 모델은 얼마나 정확한지 검증해 봅시다.

```
print(f'RMSE  : {mean_squared_error(y_val, pred, squared=False)}') # squared=False 는 MSE에 루트 씌움 즉 RMSE
print(f'MAE   : {mean_absolute_error(y_val, pred)}')
print(f'MAPE  : {mean_absolute_percentage_error(y_val, pred)}')
```

```
RMSE  : 23.625339834737034
MAE   : 22.240912904984814
MAPE  : 1.0358623206148376
```

```
코딩을 시작하거나 AI로 코드를 생성하세요.
```