실습 4 - IRIS

```
1 import pandas as pd
2 import seaborn as sns
3
4 # sns 데이터셋에 내장된 데이터를 확인할 수 있다.
5 sns.get_dataset_names()
```

데이터확인

```
[ ] 1 df = sns.load_dataset('iris')
    2 df.head()
```

```
[ ] 1 print(type(df['petal_length'][0]),
    2       type(df['sepal_width'][0]),
    3       type(df['petal_length'][0]),
    4       type(df['petal_width'][0]),
    5       type(df['species'][0]))
```

```
<class 'numpy.float64'> <class 'numpy.float64'> <class 'numpy.float64'> <class 'numpy.float64'> <class 'str'>
```

```
[ ] 1 # sepal_length : 꽃받침 길이
    2 # sepal_with : 꽃받침 너비
    3 # petal_length : 꽃잎 길이
    4 # petal_with : 꽃잎 너비
    5 # species : 꽃 종류(setosa, versicolor, virginica)
    6
    7 df.info()
```

```
[ ] 1 # missing data가 있는지 다시한번 그래프를 그려서 확인
    2 import missingno as mino
    3 mino.matrix(df)
    4
```

```
[ ] 1 df.describe()
```

boxplot

```
[ ]  1 df.head(2)
```

```
[ ]  1 data = df.iloc[:,:-1]
     2 data.boxplot()
```

```
[ ]  1 # sepal_with의 경우 outlier가 발견됨
     2 # 그런데 위에 평균값과 비교해서 볼때 엄청나게 범위를 벗어난 값은 아니기때문에 그냥 두어도 무방할것이라고 예상함
```

산점도

```
[ ]  1 sns.pairplot(df, hue='species')
```

데이터 전처리

y값 인코딩

label로 사용되는 "species"컬럼이 범주형이기 때문에 숫자형으로 변환해주어야 한다.

```
[ ]  1 df.head()
```

```
[ ]  1 X = df.iloc[:,0:4] # sepal_length, sepal_width, petal_length, petal_width
     2 y = df.iloc[:,-1]  # species
     3
     4 # X = df[['sepal_length','sepal_width','petal_length','petal_width']]
     5 # y = df['species']
```

```
     1 from sklearn.preprocessing import LabelEncoder
     2
     3 encoding = LabelEncoder() # 선언
     4 y_encoding = encoding.fit_transform(y)
     5 y_encoding
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

모델링

```
[ ]  1 from sklearn.model_selection import train_test_split
     2
     3 X_train, X_test, y_train, y_test = train_test_split(X, y_encoding, test_size = 0.2,
     4                                                     random_state = 2) # test_size : default = 0.25
     5 print('훈련 : ', len(X_train), len(y_train))
     6 print('테스트 : ', len(X_test), len(y_test))
     7
```

```
[ ]  1 from tensorflow.keras.models import Sequential
     2 from tensorflow.keras.layers import Dense
     3
     4 model = Sequential()
     5
     6 model.add(Dense(8, input_dim = 4, activation = 'relu'))
     7 model.add(Dense(3, activation = 'softmax'))
     8
     9 model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

```
     1 model.fit(X_train,y_train, epochs = 10, batch_size = 10)
```

```
[ ]  1 test_loss, test_acc = model.evaluate(X_test,y_test,verbose=2)
     2
     3 print('test_loss',test_loss)
     4 print('test_acc',test_acc)
```

```
1/1 - 0s - loss: 0.6087 - accuracy: 0.7333 - 134ms/epoch - 134ms/step
test_loss 0.6086758971214294
test_acc 0.7333333492279053
```

모델 성능향상

```
import pandas as pd
import numpy as np
import seaborn as sns

from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization

df = sns.load_dataset('iris')

X = df.iloc[:,0:4] # sepal_length, sepal_width, petal_length, petal_width
y = df.iloc[:,-1]  # species

encoding = OneHotEncoder(sparse = False)
y_encoding = encoding.fit_transform(np.array(y).reshape(-1,1))
# encoding = LabelEncoder()
# y_encoding = encoding.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoding, test_size = 0.2,
                                                    random_state=2) # test_size : default = 0.25

model = Sequential( )

model.add(Dense(1024, input_dim = 4, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(X_train,y_train, epochs = 1000, batch_size = 10)
```

```
[ ]
test_loss, test_acc = model.evaluate(X_test,y_test,verbose=2)

print('test_loss',test_loss)
print('test_acc',test_acc)
```

```
1/1 - 0s - loss: 0.2356 - accuracy: 0.8333 - 440ms/epoch - 440ms/step
test_loss 0.23555192351341248
```

one-hot encoding을 사용하는 경우 참고

```
# 원핫 인코딩 사용
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

X = df.iloc[:,0:4]
y = df.iloc[:,-1]

encoding = OneHotEncoder(sparse = False)
y_encoding = encoding.fit_transform(np.array(y).reshape(-1,1))
# 원핫 인코딩을 사용하려면 범주형 변수는 배열이 반드시 2차원 배열이어야 하기 때문에 reshape 진행

X_train, X_test, y_train, y_test = train_test_split(X, y_encoding)

model = Sequential( )

model.add(Dense(1024, input_dim = 4, activation = 'relu'))
model.add(Dense(512,  activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(X_train,y_train, epochs = 10, batch_size = 10)
```

```
[ ]
test_loss, test_acc = model.evaluate(X_test,y_test,verbose=2)

print('test_loss',test_loss)
print('test_acc',test_acc)
```

```
2/2 - 0s - loss: 0.2018 - accuracy: 0.8947 - 178ms/epoch - 89ms/step
test_loss 0.20183640718460083
test_acc 0.8947368264198303
```

```
1
```

+ 코드 + 텍스트

텍스트 셀 추가