Keras iris Modeling

```
import warnings
warnings.filterwarnings('ignore')
```

- 실습용 데이터 설정
  - iris.csv

```
import seaborn as sns

iris = sns.load_dataset('iris')
```

- pandas DataFrame

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

I. Data Preprocessing

1) iris.Species 빈도분석

- Species : setosa, virginica, versicolor

```
iris.species.value_counts()
```

```
versicolor    50
setosa        50
virginica     50
Name: species, dtype: int64
```

2) DataFrame to Array & Casting

```
iris_AR = iris.values

iris_AR
```

- object to float

```
AR_X = iris_AR[:, 0:4].astype(float)
AR_y = iris_AR[:, 4]

AR_X.shape, AR_y.shape
```

```
((150, 4), (150,))
```

3) One Hot Encoding with sklearn & Keras

- LabelEncoder( )
  - ['setosa', 'virginica', 'virsicolor'] to [0, 1, 2]

```
from sklearn.preprocessing import LabelEncoder

encoder =  LabelEncoder()
AR_yLBE = encoder.fit_transform(AR_y)

AR_yLBE
```

```
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,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

- One-Hot Encoding - to_categorical( )

```
import tensorflow as tf

AR_yOHE = tf.keras.utils.to_categorical(AR_yLBE)

AR_yOHE
```

- tensorFlow Version

```
tf.__version__
```

```
'2.4.1'
```

- Keras Version

```
tf.keras.__version__
```

```
'2.4.0'
```

4) Train & Test Split with sklearn Package

- 7 : 3

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(AR_X, AR_yOHE,
                                                    test_size = 0.3,
                                                    random_state = 2045)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((105, 4), (45, 4), (105, 3), (45, 3))
```

II. Keras Modeling

1) Keras models & layers Import

```
from tensorflow.keras import models
from tensorflow.keras import layers
```

2) Model Define

- 모델 신경망 구조 정의

```
Model_iris = models.Sequential()

Model_iris.add(layers.Dense(16, activation = 'relu', input_shape = (4,)))
Model_iris.add(layers.Dense(8, activation = 'relu'))
Model_iris.add(layers.Dense(3, activation = 'softmax'))
```

- 모델 구조 확인
  - Layers & Parameters

```
Model_iris.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 16)                80        
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 136       
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 27        
=================================================================
Total params: 243
Trainable params: 243
Non-trainable params: 0
_________________________________________________________________
```

- 모델 레이어 시각화

```
from tensorflow.keras import utils

utils.plot_model(Model_iris)
```

3) Model Compile

- 모델 학습방법 설정

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

4) Model Fit

- 모델 학습 수행

```
History_iris = Model_iris.fit(X_train, y_train,
                              epochs = 500,
                              batch_size = 7,
                              validation_data = (X_test, y_test))
```

```
Epoch 1/500
15/15 [==============================] - 1s 33ms/step - loss: 1.3366 - accuracy: 0.3866 -
Epoch 2/500
15/15 [==============================] - 0s 4ms/step - loss: 1.2292 - accuracy: 0.4235 - v
Epoch 3/500
15/15 [==============================] - 0s 4ms/step - loss: 1.2713 - accuracy: 0.3459 - v
Epoch 4/500
15/15 [==============================] - 0s 4ms/step - loss: 1.0946 - accuracy: 0.4237 - v
Epoch 5/500
15/15 [==============================] - 0s 5ms/step - loss: 1.0906 - accuracy: 0.4963 - v
Epoch 6/500
15/15 [==============================] - 0s 4ms/step - loss: 1.0156 - accuracy: 0.5945 - v
Epoch 7/500
15/15 [==============================] - 0s 4ms/step - loss: 0.9813 - accuracy: 0.5395 - v
Epoch 8/500
15/15 [==============================] - 0s 4ms/step - loss: 0.9284 - accuracy: 0.6038 - v
Epoch 9/500
15/15 [==============================] - 0s 4ms/step - loss: 0.9650 - accuracy: 0.5160 - v
Epoch 10/500
15/15 [==============================] - 0s 4ms/step - loss: 0.9403 - accuracy: 0.4567 - v
Epoch 11/500
15/15 [==============================] - 0s 4ms/step - loss: 0.8496 - accuracy: 0.5793 - v
Epoch 12/500
15/15 [==============================] - 0s 4ms/step - loss: 0.8452 - accuracy: 0.5071 - v
Epoch 13/500
15/15 [==============================] - 0s 4ms/step - loss: 0.7977 - accuracy: 0.5118 - v
Epoch 14/500
15/15 [==============================] - 0s 4ms/step - loss: 0.7485 - accuracy: 0.6066 - v
Epoch 15/500
15/15 [==============================] - 0s 4ms/step - loss: 0.7439 - accuracy: 0.6867 - v
Epoch 16/500
15/15 [==============================] - 0s 4ms/step - loss: 0.6907 - accuracy: 0.7835 - v
Epoch 17/500
15/15 [==============================] - 0s 4ms/step - loss: 0.6946 - accuracy: 0.8471 - v
Epoch 18/500
15/15 [==============================] - 0s 4ms/step - loss: 0.6898 - accuracy: 0.9302 - v
Epoch 19/500
15/15 [==============================] - 0s 4ms/step - loss: 0.6280 - accuracy: 0.9199 - v
Epoch 20/500
15/15 [==============================] - 0s 4ms/step - loss: 0.6414 - accuracy: 0.9084 - v
Epoch 21/500
15/15 [==============================] - 0s 4ms/step - loss: 0.5516 - accuracy: 0.9197 - v
Epoch 22/500
15/15 [==============================] - 0s 4ms/step - loss: 0.5492 - accuracy: 0.9319 - v
Epoch 23/500
15/15 [==============================] - 0s 4ms/step - loss: 0.5046 - accuracy: 0.9624 - v
Epoch 24/500
```

```
15/15 [==============================] - 0s 4ms/step - loss: 0.5029 - accuracy: 0.9562 - v
Epoch 25/500
15/15 [==============================] - 0s 4ms/step - loss: 0.4600 - accuracy: 0.9839 - v
Epoch 26/500
15/15 [==============================] - 0s 4ms/step - loss: 0.4450 - accuracy: 0.9622 - v
Epoch 27/500
15/15 [==============================] - 0s 4ms/step - loss: 0.4390 - accuracy: 0.9684 - v
Epoch 28/500
15/15 [==============================] - 0s 4ms/step - loss: 0.3992 - accuracy: 0.9665 - v
Epoch 29/500
15/15 [==============================] - 0s 4ms/step - loss: 0.3879 - accuracy: 0.9556 - v
```

5) 학습 결과 시각화

```
import matplotlib.pyplot as plt

plt.figure(figsize = (9, 6))
plt.ylim(0, 1.2)
plt.plot(History_iris.history['loss'])
plt.plot(History_iris.history['val_loss'])
plt.plot(History_iris.history['accuracy'])
plt.plot(History_iris.history['val_accuracy'])
plt.legend(['loss', 'val_loss', 'accuracy', 'val_accuracy'])
plt.grid()
plt.show()
```

6) Model Evaluate

- Loss & Accuracy

```
loss, accuracy = Model_iris.evaluate(X_test, y_test)

print('Loss = {:.2f}'.format(loss))
print('Accuracy = {:.2f}'.format(accuracy))
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.0329 - accuracy: 0.9778
Loss = 0.03
Accuracy = 0.98
```

7) Model Predict

- Probability

```
import numpy as np
np.set_printoptions(suppress = True, precision = 5)
```

```
Model_iris.predict(X_test)
```

```
array([[0.99982, 0.00018, 0.     ],
       [0.99755, 0.00245, 0.     ],
       [0.00327, 0.99669, 0.00004],
       [0.     , 0.00009, 0.99991],
       [0.99998, 0.00002, 0.     ],
       [0.00017, 0.9998 , 0.00002],
       [0.     , 0.00419, 0.99581],
       [0.99998, 0.00002, 0.     ],
       [0.     , 0.00038, 0.99962],
       [0.99979, 0.00021, 0.     ],
       [0.00019, 0.88924, 0.11057],
       [0.     , 0.00267, 0.99733],
       [0.00007, 0.99821, 0.00172],
       [1.     , 0.     , 0.     ],
       [0.99998, 0.00002, 0.     ],
       [0.00015, 0.99693, 0.00292],
       [0.00008, 0.98867, 0.01126],
       [0.9999 , 0.0001 , 0.     ],
       [0.00004, 0.99977, 0.00019],
       [0.99999, 0.00001, 0.     ],
       [0.99986, 0.00014, 0.     ],
       [0.99986, 0.00014, 0.     ],
       [0.     , 0.00085, 0.99915],
       [0.99994, 0.00006, 0.     ],
       [0.     , 0.0559 , 0.9441 ],
       [0.     , 0.00534, 0.99466],
       [0.99997, 0.00003, 0.     ],
       [0.00003, 0.99847, 0.0015 ],
       [0.00003, 0.99821, 0.00177],
       [0.     , 0.0302 , 0.96979],
       [0.99998, 0.00002, 0.     ],
       [0.00004, 0.99904, 0.00092],
       [0.99998, 0.00002, 0.     ],
       [0.00004, 0.70252, 0.29744],
       [0.00008, 0.99659, 0.00333],
       [0.00011, 0.99987, 0.00002],
       [0.     , 0.00011, 0.99989],
       [0.00014, 0.99976, 0.0001 ],
       [0.99989, 0.00011, 0.     ],
       [0.00012, 0.99814, 0.00175],
       [0.     , 0.00014, 0.99986],
       [0.9998 , 0.0002 , 0.     ],
       [0.     , 0.00121, 0.99879],
       [0.     , 0.00049, 0.99951],
       [0.     , 0.01449, 0.98551]], dtype=float32)
```

- Class

```
y_hat = Model_iris.predict_classes(X_test)

y_hat
```

```
array([0, 0, 1, 2, 0, 1, 2, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       2, 0, 2, 2, 0, 1, 1, 2, 0, 1, 0, 1, 1, 1, 2, 1, 0, 1, 2, 0, 2, 2,
       2])
```

- Probability to Class

```
np.argmax(Model_iris.predict(X_test), axis = 1)
```

```
    array([0, 0, 1, 2, 0, 1, 2, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
           2, 0, 2, 2, 0, 1, 1, 2, 0, 1, 0, 1, 1, 1, 2, 1, 0, 1, 2, 0, 2, 2,
           2])
```

- One-Hot Encoding to Array
  - np.argmax( ) : 다차원 배열의 차원에 따라 가장 큰 값의 인덱스를 반환
  - axis = 1 : 열기준

```
y = np.argmax(y_test, axis = 1)

y
```

```
    array([0, 0, 1, 2, 0, 1, 2, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
           2, 0, 2, 2, 0, 1, 1, 2, 0, 1, 0, 2, 1, 1, 2, 1, 0, 1, 2, 0, 2, 2,
           2])
```

- Confusion Matrix & Claasification Report

```
from sklearn.metrics import confusion_matrix, classification_report

confusion_matrix(y, y_hat)
```

```
    array([[17,  0,  0],
           [ 0, 14,  0],
           [ 0,  1, 13]])
```

```
print(classification_report(y, y_hat,
                            target_names = ['setosa',
                                            'virginica',
                                            'versicolor']))
```

```
                  precision    recall  f1-score   support

        setosa       1.00      1.00      1.00        17
     virginica       0.93      1.00      0.97        14
    versicolor       1.00      0.93      0.96        14

      accuracy                           0.98        45
     macro avg       0.98      0.98      0.98        45
  weighted avg       0.98      0.98      0.98        45
```

III. Model Save & Load

1) File System

- Save to Colab File System

```
!ls -l
```

```
total 12
-rw-r--r-- 1 root root 7979 Mar  8 01:55 model.png
drwxr-xr-x 1 root root 4096 Mar  1 14:35 sample_data
```

```
Model_iris.save('Model_iris.h5')

!ls -l
```

```
total 48
-rw-r--r-- 1 root root 34592 Mar  8 01:56 Model_iris.h5
-rw-r--r-- 1 root root  7979 Mar  8 01:55 model.png
drwxr-xr-x 1 root root  4096 Mar  1 14:35 sample_data
```

- Download Colab File System to Local File System

```
from google.colab import files

files.download('Model_iris.h5')
```

- Load from Colab File System

```
from keras.models import load_model

Model_local = load_model('Model_iris.h5')
```

```
Model_local.predict_classes(X_test)
```

```
array([0, 0, 1, 2, 0, 1, 2, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       2, 0, 2, 2, 0, 1, 1, 2, 0, 1, 0, 1, 1, 1, 2, 1, 0, 1, 2, 0, 2, 2,
       2])
```

2) Google Drive

- Mount Google Drive

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

- Check Mounted_Drive

```
!ls -l '/content/drive/My Drive/Colab Notebooks/datasets'
```

```
total 1736479
-rw------- 1 root root      20066 Mar  4 04:45 cat.1700.jpg
-rw------- 1 root root   69155672 Mar  4 04:46 creditCardFraud.zip
-rw------- 1 root root   90618980 Mar  4 04:51 dogs_and_cats_small.zip
drwx------ 2 root root       4096 Mar  4 05:34 image
-rw------- 1 root root    8204887 Mar  4 04:45 Images_500.zip
-rw------- 1 root root   12929865 Mar  4 04:42 Logo_Data.zip
-rw------- 1 root root   18272469 Mar  4 04:50 MNIST.csv
-rw------- 1 root root   22824989 Mar  7 07:09 Online_Retail.zip
-rw------- 1 root root        741 Mar  4 04:44 PII.csv
-rw------- 1 root root 1141460846 Mar  4 04:50 waferImages.zip
-rw------- 1 root root  414658234 Mar  4 04:49 yolo_weight.zip
```

```
import pandas as pd

DF = pd.read_csv('/content/drive/My Drive/Colab Notebooks/datasets/PII.csv')

DF.head(3)
```

	Name	Gender	Age	Grade	Picture	BloodType	Height	Weight
0	송태섭	남자	21	3	무	B	179.1	63.9
1	최유정	여자	23	1	유	A	177.1	54.9
2	이한나	여자	20	1	무	A	167.9	50.2

- Save to Mounted Google Drive Directory

```
Model_iris.save('/content/drive/My Drive/Colab Notebooks/models/001_Model_iris.h5')
```

```
!ls -l '/content/drive/My Drive/Colab Notebooks/models'
```

```
total 67597
-rw------- 1 root root    34592 Mar  8 01:56 001_Model_iris.h5
-rw------- 1 root root 27683600 Jan 15 05:41 002_dogs_and_cats_small.h5
-rw------- 1 root root 41499744 Jan 15 06:53 003_dogs_and_cats_augmentation.h5
```

- Load from Mounted Google Drive Directory

```
from keras.models import load_model

Model_google = load_model('/content/drive/My Drive/Colab Notebooks/models/001_Model_iris.h5')
```

```
Model_google.predict_classes(X_test)
```

```
    array([0, 0, 1, 2, 0, 1, 2, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
           2, 0, 2, 2, 0, 1, 1, 2, 0, 1, 0, 1, 1, 1, 2, 1, 0, 1, 2, 0, 2, 2,
           2])
```

#

#

#

The End

#

#

#