Neural Network Model - 다중분류

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

실습용 데이터 설정

- iris.csv

```
import seaborn as sns

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

- pandas DataFrame

```
DF.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
```

```
DF.head(3)
```

	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

I. 탐색적 데이터 분석

1) 빈도분석

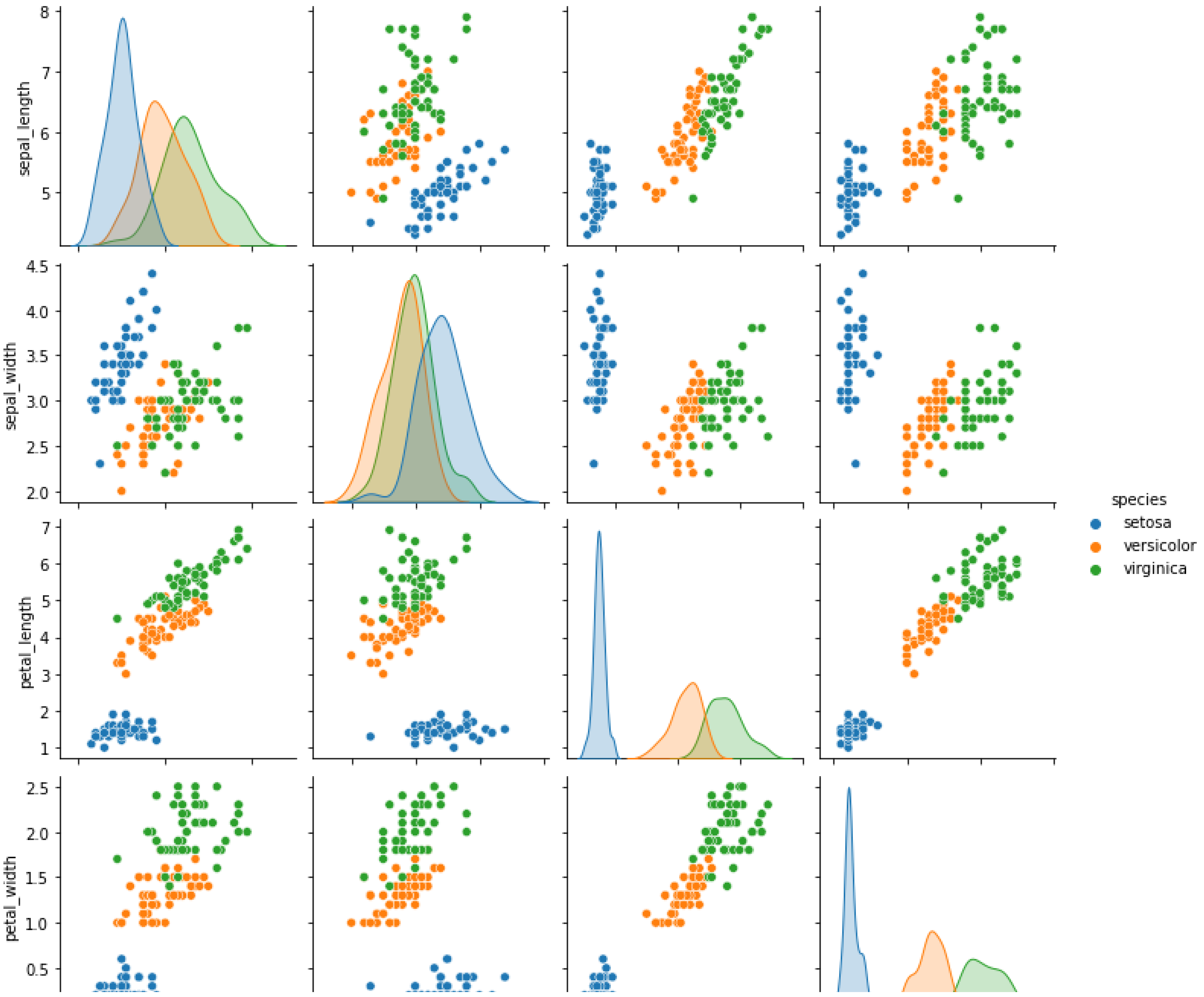
```
DF.species.value_counts()
```

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

2) 분포 시각화

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(hue = 'species', data = DF)
plt.show()
```



II. Data Preprocessing

1) Data Set

```
X = DF[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = DF['species']
```

2) Train & Test Split

- 7 : 3

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 2045)

print('Train Data : ', X_train.shape, y_train.shape)
print('Test Data : ', X_test.shape, y_test.shape)
```

```
Train Data :  (105, 4) (105,)
Test Data :  (45, 4) (45,)
```

III. Modeling

1) Train_Data로 모델 생성

- hidden_layer_sizes : 은닉층 노드의 개수

- activation : 활성화 함수
- solver : 최적화 기법
- max_iter : 학습 반복 횟수

```
from sklearn.neural_network import MLPClassifier

Model_NN = MLPClassifier(hidden_layer_sizes = (5),
                         activation = 'logistic',
                         solver ='adam',
                         max_iter = 5000,
                         random_state = 2045)

Model_NN.fit(X_train, y_train)
```

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=5, learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=5000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=2045, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

2) Test_Data에 Model 적용

```
y_hat = Model_NN.predict(X_test)
```

3) Confusion Matrix

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_hat)
```

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

4) Accuracy

```
from sklearn.metrics import accuracy_score

print('%.8f' % accuracy_score(y_test, y_hat))
```

```
1.00000000
```

5) Classification Report

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_hat,
                            target_names = ['setosa', 'versicolor', 'virginica'],
                            digits = 5))
```

```
              precision    recall  f1-score   support

      setosa    1.00000   1.00000   1.00000        17
  versicolor    1.00000   1.00000   1.00000        14
   virginica    1.00000   1.00000   1.00000        14

    accuracy                        1.00000        45
   macro avg    1.00000   1.00000   1.00000        45
weighted avg    1.00000   1.00000   1.00000        45
```

#

#

#

The End

#

#

#