인공지능

머신러닝 응용 프레임워크

변 영 철

머신러닝 응용 프레임워크

성별 인식 코드 추상화하기

Step1 클래스 모듈 만들기

결과적으로 클래스로 구현한 결과는 다음과 같습니다. MyDataFrame, MyModel 등의 클래스로 구현하였습니다.

```
import numpy as np # 수학 연산 수행을 위한 모듈
import pandas as pd # 데이터 처리를 위한 모듈
import seaborn as sns # 데이터 시각화 모듈
import matplotlib.pyplot as plt # 데이터 시각화 모듈

# 다양한 분류 알고리즘 패키지를 임포트함.
from sklearn.linear_model import LogisticRegression  # Logistic Regression 알고리즘
#from sklearn.cross_validation import train_test_split # 데이타 쪼개주는 모듈

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier  # for K nearest neighbours
from sklearn import svm  #for Support Vector Machine (SVM) Algorithm
from sklearn import metrics #for checking the model accuracy
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm

class MyDataFrame: #gildong
```

```
data_frame = 0

def load_csv(self, f):
    # CSV 파일 읽어오기
    self.data_frame = pd.read_csv(f)

def file_info(self):
    self.data_frame.info()

def show_head(self):
    print(self.data_frame.head())

def show_col_name(self):
    for col in self.data_frame.columns:
        print(col)

def show_cols(self):
    print(self.data_frame.columns)

def show_hist(self):
    self.data_frame.hist(edgecolor='black', linewidth=1.2)
    fig = plt.gcf()
    fig.set_size_inches(12, 10)
    plt.show()

def plot(self, a, b, c):
    # 읽어온 데이터 표시하기
```

```
            cl = self.data_frame[c].unique()

            col = ['orange', 'blue', 'red', 'yellow', 'black', 'brown']

            fig = self.data_frame[self.data_frame[c] == cl[0]].plot(kind='scatter',
x=a, y=b, color=col[0], label=cl[0])

            for i in range(len(cl) - 1):
                self.data_frame[self.data_frame[c] == cl[i +
1]].plot(kind='scatter', x=a, y=b, color=col[i + 1], label=cl[i + 1], ax=fig)

            fig.set_xlabel(a)
            fig.set_ylabel(b)
            fig.set_title(a + " vs. " + b)
            fig = plt.gcf()
            fig.set_size_inches(10, 6)
            plt.show()

        def show_boxplot(self, a, b):
            f, sub = plt.subplots(1, 1, figsize=(8, 6))
            sns.boxplot(x=self.data_frame[a], y=self.data_frame[b], ax=sub)
            sub.set(xlabel=a, ylabel=b)

        def show_violenplot(self, a, b):
            plt.figure(figsize=(8, 6))
            plt.subplot(1, 1, 1)
            sns.violinplot(x=a, y=b, data=self.data_frame)
```

```
    def show_heatmap(self):
        plt.figure(figsize=(12, 8))
        sns.heatmap(self.data_frame.corr(), annot=True,
cmap='cubehelix_r')
        plt.show()

    def prepare_data(self, q, target, r):
        train, test = train_test_split(self.data_frame, test_size=r)
        # train=70% and test=30%
        print(train.shape)
        print(test.shape)

        # 학습용 문제, 학습용 정답
        train_X = train[q]  # 키와 발크기만 선택
        train_y = train[target]  # 정답 선택

        # 테스트용 문제, 테스트용 정답
        test_X = test[q]  # taking test data features
        test_y = test[target]  # output value of test data
        return train_X, train_y, test_X, test_y

    def drop(self, col):
        self.data_frame.drop(col, axis=1, inplace=True)

    def show_unique(self, col):
        print(self.data_frame[col].unique())
```

```
    def to_numeric(self, col, m, new_col):
        self.data_frame[new_col] = self.data_frame[col].map(m)


class MyModel: #youngja
    train_X = []
    train_y = []

    test_X = []
    test_y = []

    def __init__(self, i, j, k, l):
        self.train_X = i
        self.train_y = j
        self.test_X = k
        self.test_y = l

    def run_SVM(self):
        gildong = svm.SVC()
        gildong.fit(self.train_X, self.train_y)  # 가르친 후
        prediction = gildong.predict(self.test_X)  # 얼마나 맞히는지
테스트

        rate1 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate1))
```

```
    def run_LR(self):
        cheolsu = LogisticRegression()
        cheolsu.fit(self.train_X, self.train_y)
        prediction = cheolsu.predict(self.test_X)

        rate2 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate2))


    def run_DT(self):
        youngja = DecisionTreeClassifier()
        youngja.fit(self.train_X, self.train_y)
        prediction = youngja.predict(self.test_X)

        rate3 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate3))

    def run_NN(self):
        minsu = KNeighborsClassifier(n_neighbors=3)  # this examines 3
neighbours for putting the new data into a class
        minsu.fit(self.train_X, self.train_y)
        prediction = minsu.predict(self.test_X)

        rate4 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate4))
```

```
class MyML:   # minsu
    file_name = 0
    test_data_ratio = 0
    q_cols = 0
    target = 0

    heatmap_flag = 0

    def set_heatmap(self, flag):
        self.heatmap_flag = flag;

    def set_file(self, f):
        self.file_name = f

    def set_data_ratio(self, v):
        self.test_data_ratio = v

    def set_q_cols(self, i):
        self.q_cols = i

    def set_target(self, t):
        self.target = t

    def doML(self):
        gildong = MyDataFrame()
        gildong.load_csv(self.file_name);
        gildong.file_info()
```

```
        gildong.plot('Height', 'FeetSize', 'Sex')

        if self.heatmap_flag:
            gildong.show_heatmap()

        a, b, c, d = gildong.prepare_data(self.q_cols, self.target,
self.test_data_ratio)

        youngja = MyModel(a, b, c, d)
        youngja.run_SVM()
        youngja.run_LR()
        youngja.run_DT()
        youngja.run_NN()
```

클래스 라이브러리를 이용한 붓꽃 인식

다음은 위의 클래스를 라이브러리로 이용하여 붓꽃을 인식하는 코드를 작성한 예입니다. 먼저 아래와 같이 두 클래스를 임포트합니다.

```
from myai import MyDataFrame
from myai import MyModel
```

그런 다음 MyDataFrame 클래스로 gildong 객체를 생성합니다.

```
gildong = MyDataFrame()
```

Iris.csv 파일을 읽어 들인 후 몇 가지 정보를 출력해봅니다.

```
gildong.load_csv('Iris.csv');

gildong.file_info()
```

그 결과 다음과 같은 결과가 표시됩니다.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #    Column            Non-Null Count   Dtype
---   ------            --------------   -----
 0    Id                150 non-null     int64
```

```
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

이제, 데이터 프레임에 어떤 컬럼이 있는지 확인해 봅니다.

```
gildong.show_cols()
```

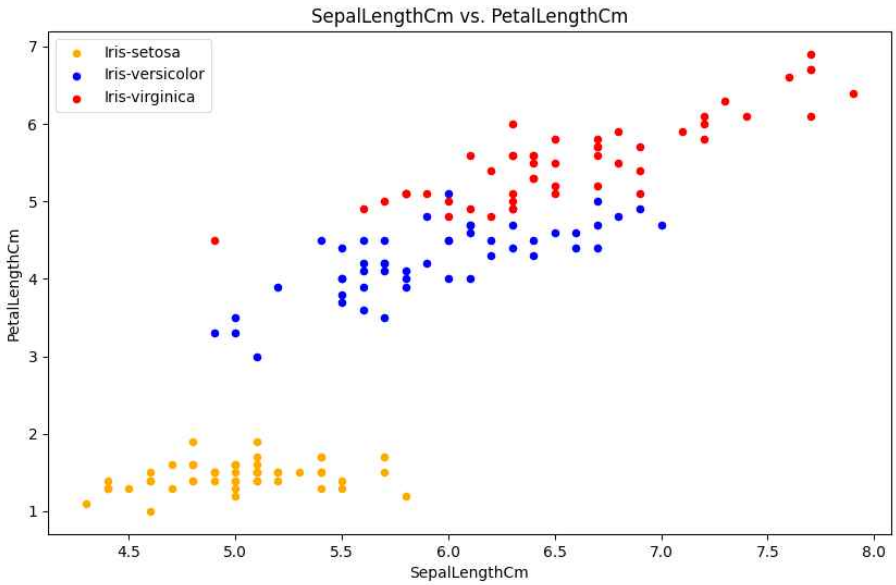
실행 결과 다음과 같이 6가지 컬럼이 있음을 보여줍니다.

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm', 'Species'],          dtype='object')
```

X축을 'SepalLengthCm'으로, Y 축을 'PetalLengthCm'으로, 그리고 'Species' 종류에 따라 다른 색으로 표시하도록 해봅니다.

```
gildong.plot('SepalLengthCm', 'PetalLengthCm', 'Species')
```

그 결과 다음과 같이 표시됩니다.



Setosa의 경우 상대적으로 꽃받침의 길이(SepalLengthCm)와 꽃잎의 길이(PetalLengthCm)가 짧습니다. 이에 비해 Virginica는 꽃받침과 꽃잎의 길이가 깁니다. Versicolor는 그 중간에 위치하는 것을 볼 수 있습니다.

이번에는 이러한 데이터를 이용하여 학습과 테스트를 수행해보겠습니다. 먼저 데이터를 나눕니다. 이때, 입력은 꽃받침 길이와 너비, 꽃잎의 길이와 너비를 모두 사용하고, 맞출 값은 꽃의 유형입니다. 학습데이터는 80%, 테스트 데이터는 20%로 지정하였습니다.

```
a, b, c, d = gildong.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'], 'Species', 0.2)
```

나눈 데이터를 이용할 모델을 생성하고, 학습 및 테스트를 수행한 후 인

식률을 확인합니다.

```
youngja = MyModel(a, b, c, d)
youngja.run_SVM()
youngja.run_LR()
youngja.run_DT()
youngja.run_NN()
```

다음은 실행 결과입니다. 나눈 데이터의 크기와 4가지 모델에 따른 인식률이 표시되었습니다.

```
(120, 6)
(30, 6)
인식률: 100.0
인식률: 100.0
인식률: 100.0
인식률: 100.0
```

이어서 히트맵을 표시해봅니다. 하지만 다음과 같은 오류가 발생하였습니다.

```
ValueError: could not convert string to float: 'Iris-setosa’
```

아마도 문자열을 실수(숫자)로 바꿀 수 없다고 합니다. Species 컬럼에 숫자가 아닌 문자열이 있는 것 같습니다. 그래서 Species 컬럼에 어떤 값

이 있는지 확인해봅니다. 이때 동작하지 않은 히트맵 출력 코드는 주석처리합니다.

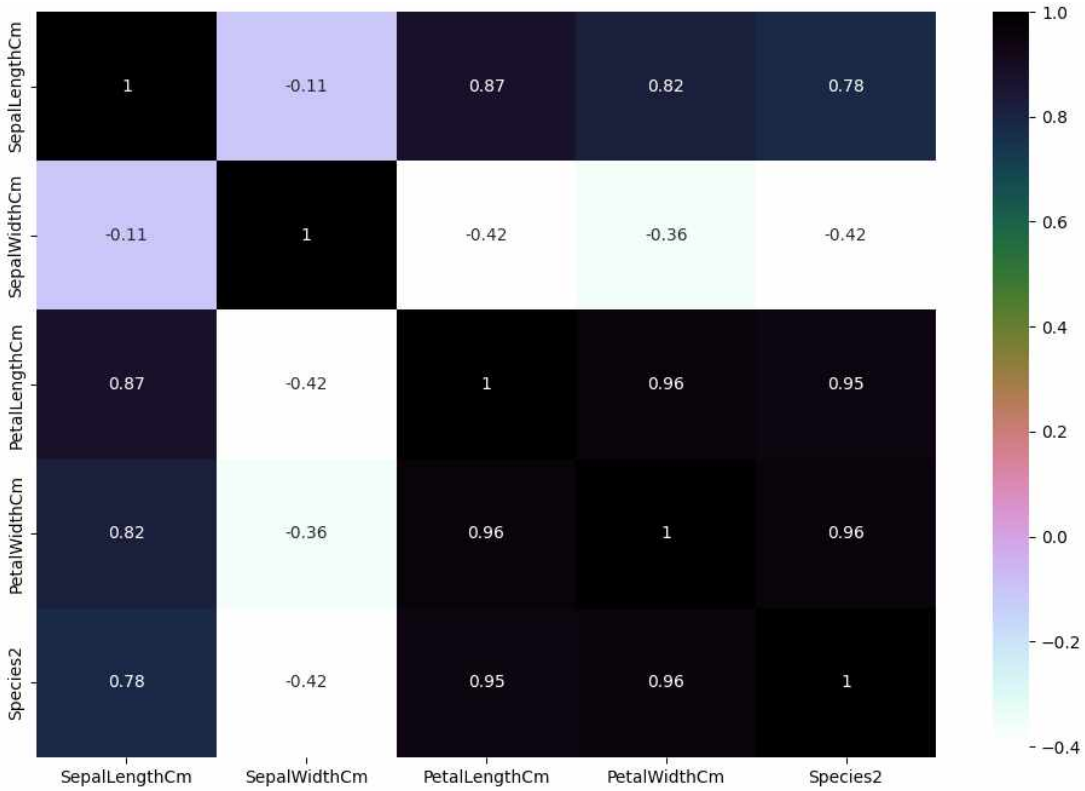
```
gildong.show_unique('Species')
#gildong.show_heatmap()
```

결과를 보니 다음과 같이 3가지가 있네요.

```
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

각각의 문자열에 대해 다음과 같이 0, 1, 2로 대치하도록 합니다. 그리고 Id, Species 컬럼을 삭제합니다. 마지막으로 히트맵을 다시 출력하도록 합니다.

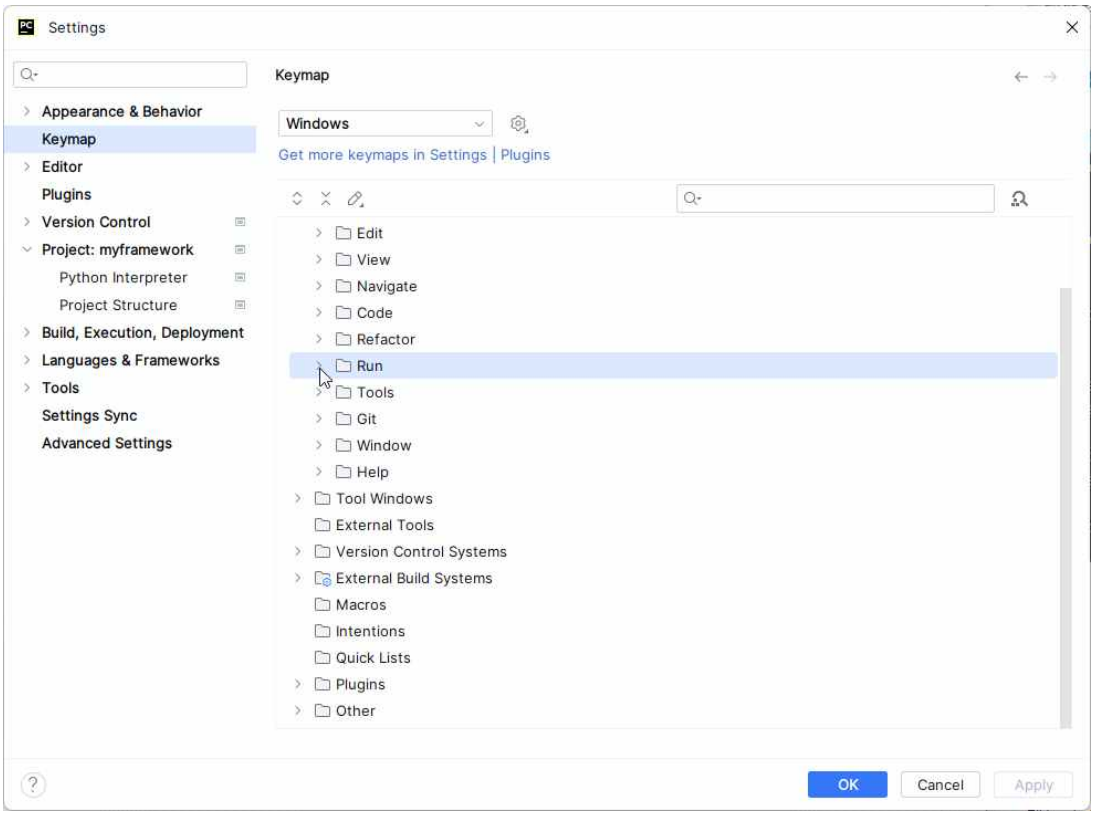
```
gildong.to_numeric('Species', {'Iris-setosa':0, 'Iris-versicolor':1,
'Iris-virginica':2}, 'Species2')
gildong.drop('Id')
gildong.drop('Species')

gildong.show_heatmap()
```



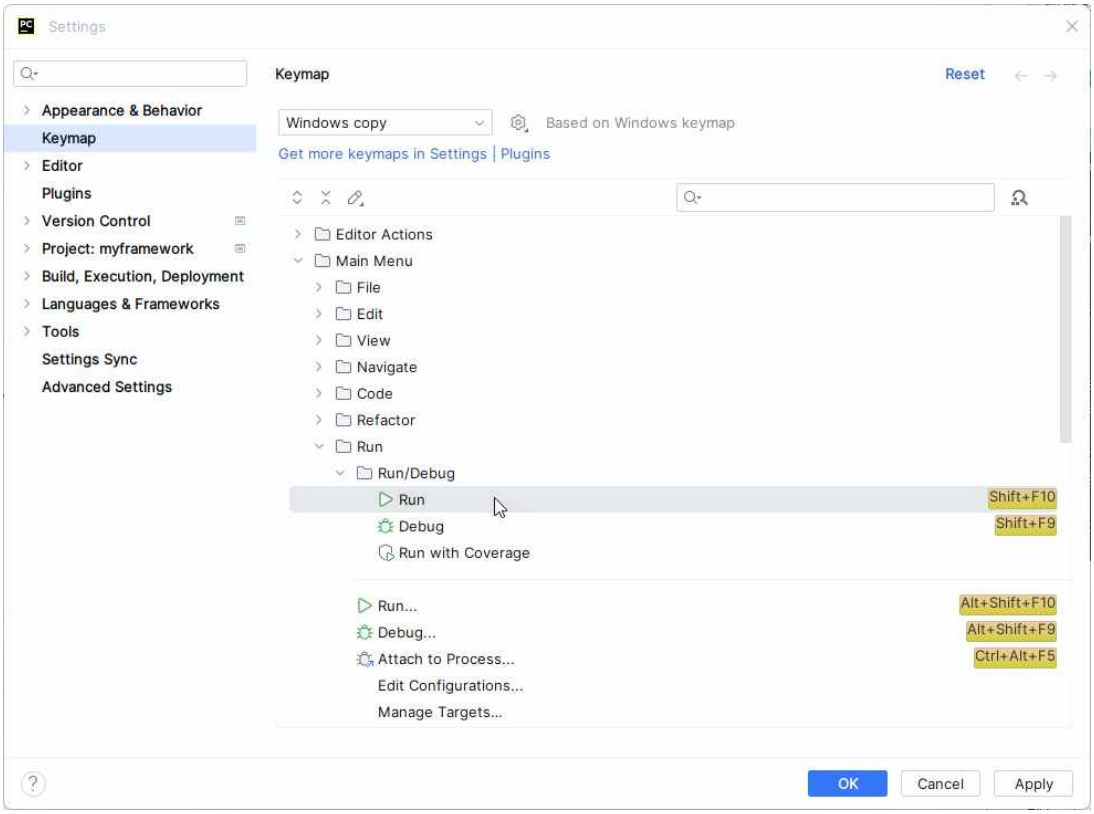
결과를 보면, 0.96으로 꽃잎의 길이와 너비가 상관관계가 가장 큽니다. 그리고 꽃의 종류(Species2)와 상관관계가 가장 큰 것은 꽃잎의 너비로 0.96임을 알 수 있습니다. 꽃잎의 길이도 0.95로 상당히 큽니다. 꽃받침 길이도 0.78로 비교적 크나 상대적으로 꽃받침 길이는 -0.42로 작습니다. 음수의 경우에는 음의 상관관계를 갖고 있음을 뜻합니다.

단축키 설정하기

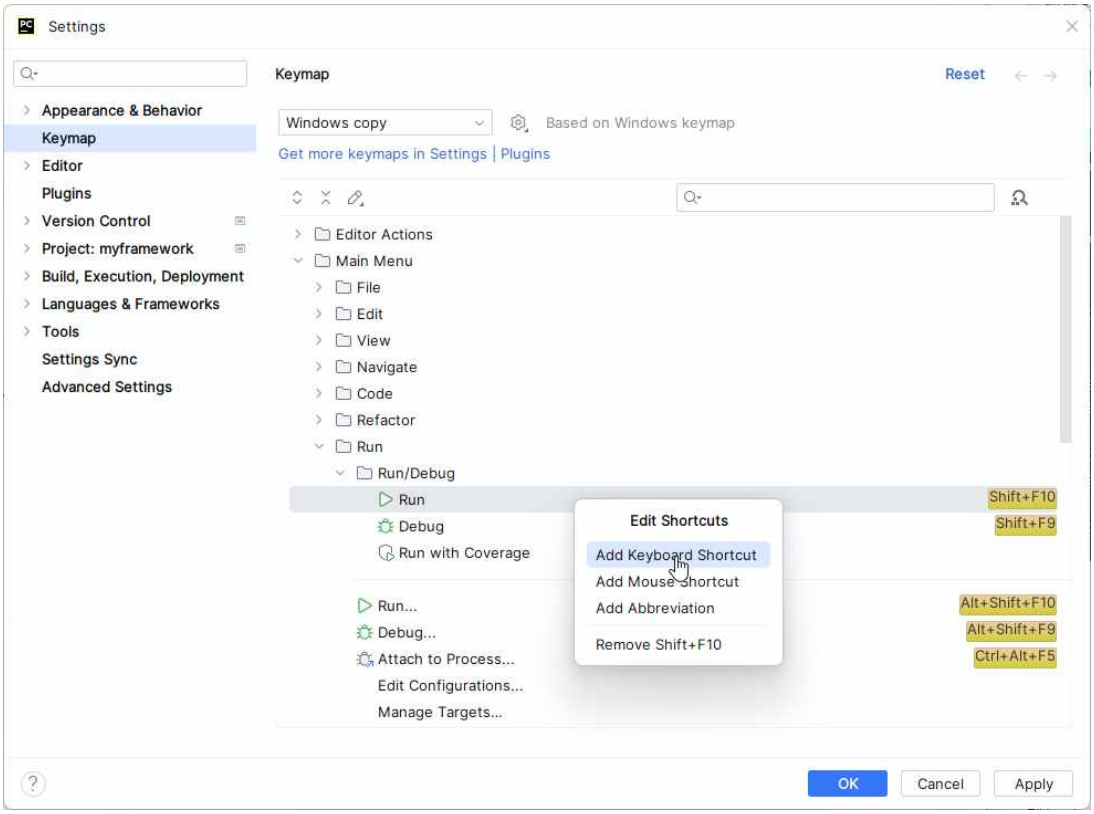
File | Settings 메뉴 선택 후 아래와 같이 왼쪽에서 Keymap을 선택합니다.



그런 다음, 위 그림에서 오른쪽의 Run 아이콘을 클릭하면 다음과 같이 표시됩니다.



여기에서 Run/Debug에 있는 Run 메뉴 항목을 두 번 클릭하면 단축키를 편집할 수 있는 메뉴가 표시됩니다.



Add Keyboard Shortcut을 선택한 후 아래와 같이 단축키로 Alt + X를 클릭하여 OK 버튼을 누르면, 향후 프로그램을 실행하고자 할 때는 단순히 Alt + X키를 누르기만 하면 됩니다. 아주 편리해지는 것이지요.

라이브러리 사용 연습하기

현재 작성되어 있는 MyDataFrame 클래스와 MyModel 클래스를 이용하면 얼마나 손쉽게 머신러닝 코드를 작성할 수 있는지 알아보겠습니다. 먼저 라이브러리를 임포트합니다.

```
from myai import *
```

데이터 프레임 객체 df와 머신러닝 모델 객체 model을 생성합니다.

```
df = MyDataFrame()
model = MyModel()
```

df 객체로 파일을 로드합니다.

```
df.load_csv("Iris.csv")
```

파일에 대한 부가 정보를 표시해 봅니다.

```
df.show_file_info()
df.show_cols()
```

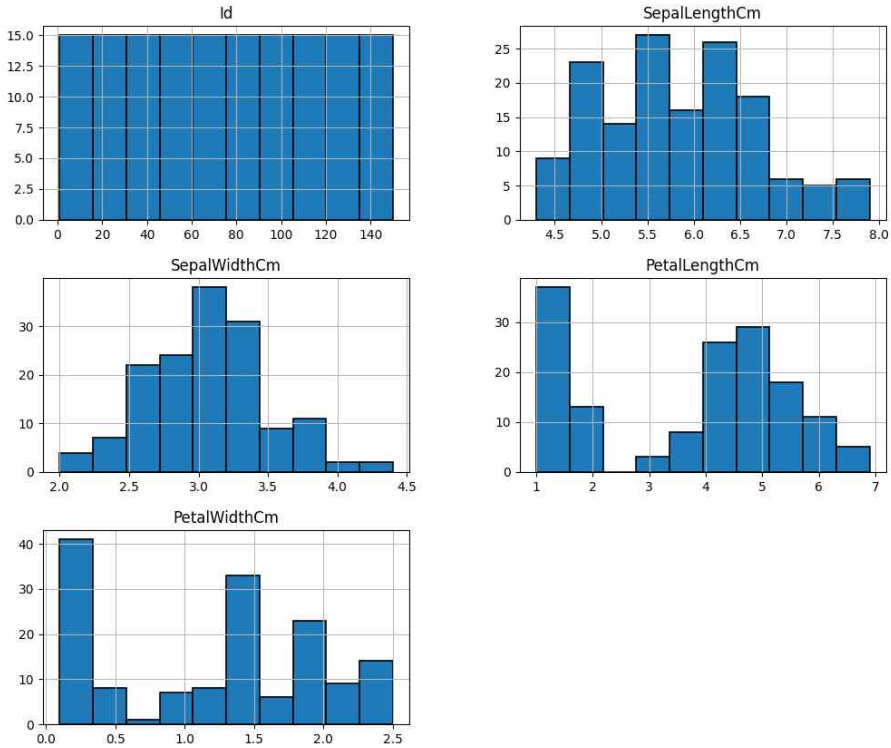
다음은 실행결과입니다.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
```

```
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              150 non-null    int64
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm', 'Species'], dtype='object')

Process finished with exit code 0
```

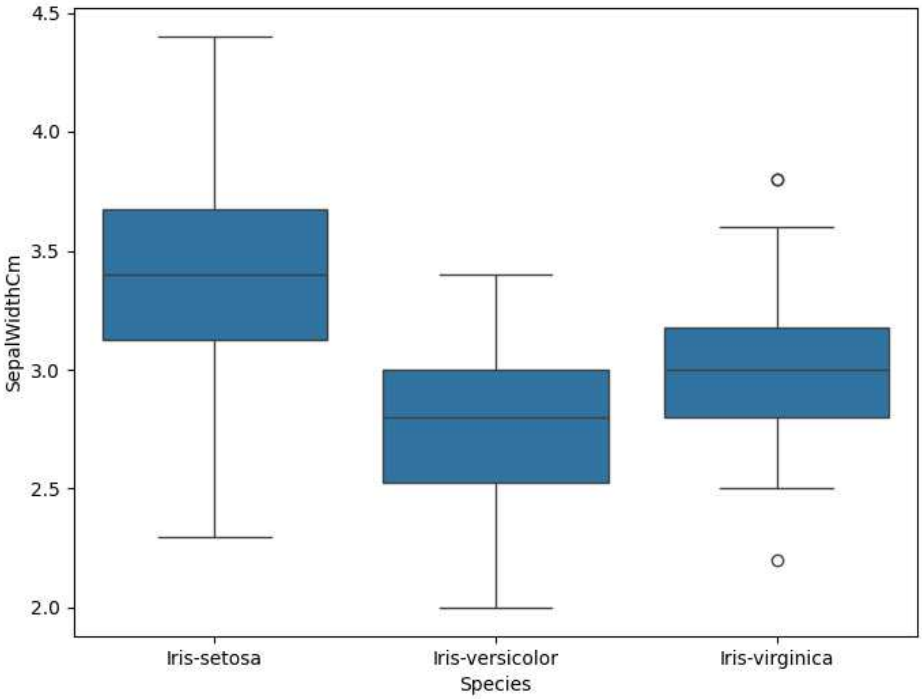
이제 csv 데이터 시각화를 해봅니다.

```
df.show_hist()
```



다음은 박스 플롯을 출력한 결과입니다.

```
df.show_boxplot('Species', 'SepalWidthCm')
```



이처럼 준비된 멤버 함수를 호출하여 시각화를 할 수 있습니다. 계속하기 위하여 일단 시각화 코드는 주석처리하여 코딩을 하는 과정에서 번거롭게 표시되지 않도록 하겠습니다.

```
#df.show_hist()
#df.show_boxplot('Species', 'SepalWidthCm')
```

이제 모델 학습 및 테스트를 위한 데이터를 준비하도록 하겠습니다. 다음은 이를 위한 코드입니다.

```
a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
model.set_data(a, b, c, d)
```

입력을 꽃받침(sepal) 길이와 너비, 꽃잎 길이로 하고, 그에 따라 인식해야 할 타겟으로 꽃의 유형(species)을 설정하여 학습 데이터 입력(a), 학습데이터 타겟(b), 테스트 데이터 입력(c), 테스트 데이터 타겟(d)을 구한 후 인공지능 모델로 보냅니다.

이제 마지막으로 4가지 머신러닝 모델을 이용하여 학습과 테스트를 수행합니다.

```
model.run_LR()
model.run_DT()
model.run_NN()
model.run_SVM()
```

다음은 실행 결과입니다.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #    Column            Non-Null Count   Dtype
---   ------            --------------   -----
 0    Id                150 non-null     int64
 1    SepalLengthCm     150 non-null     float64
```

```
 2    SepalWidthCm    150 non-null    float64
 3    PetalLengthCm   150 non-null    float64
 4    PetalWidthCm    150 non-null    float64
 5    Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm',
       'Species'],
      dtype='object')
(90, 6)
(60, 6)
인식률: 91.7
인식률: 90.0
인식률: 95.0
인식률: 91.7

Process finished with exit code 0
```

결과적으로 우리가 작성한 코드는 다음과 같습니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()

#(1) 데이터 로드
```

```
df.load_csv("Iris.csv")

#(2) 데이터 정보 표시
df.show_file_info()
df.show_cols()

#(3) 데이터 시각화
#df.show_hist()
#df.show_boxplot('Species', 'SepalWidthCm')

#(4) 학습/테스트 데이터 준비하기
a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
model.set_data(a, b, c, d)

#(5) 머신러닝 모델 실행하기
model.run_LR()
model.run_DT()
model.run_NN()
model.run_SVM()
```

전체적으로 보면 다음과 같이 대략 5가지 단계로 실행됩니다.

(1) 데이터 로드
(2) 데이터 정보 표시
(3) 데이터 시각화
(4) 학습/테스트 데이터 준비하기

(5) 머신러닝 모델 실행하기

지금부터를 이에 따라 한 단계 더 추상화하도록 하겠습니다.

한 단계 더 추상화하기

가장 먼저 아래 데이터 로드 부분을 추상화하겠습니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()

#(1) 데이터 로드
df.load_csv("Iris.csv")

df.show_file_info()
df.show_cols()

#df.show_hist()
#df.show_boxplot('Species', 'SepalWidthCm')

a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
model.set_data(a, b, c, d)
```

```
model.run_LR()
model.run_DT()
model.run_NN()
model.run_SVM()
```

아래와 같이 할 수 있겠네요.

```
from myai import *

df = MyDataFrame()
model = MyModel()

def load_csv():
    df.load_csv("Iris.csv")

#(1) 데이터 로드
load_csv()
```

다음, 그 아래 부분에 있는 파일 정보를 보여주는 두 줄의 코드를 show_info 함수로 추상화합니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()

def load_csv():
```

```
        df.load_csv("Iris.csv")

    def show_info():
        df.show_file_info()
        df.show_cols()

    #(1) 데이터 로드
    load_csv()

    #(2) 데이터 정보 표시
    show_info()

    #(3) 데이터 시각화
    #df.show_hist()
    #df.show_boxplot('Species', 'SepalWidthCm')

    #(4) 학습/테스트 데이터 준비하기
    a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
    'PetalLengthCm'], 'Species', 0.4)
    model.set_data(a, b, c, d)

    #(5) 머신러닝 모델 실행하기
    model.run_LR()
    model.run_DT()
    model.run_NN()
    model.run_SVM()
```

데이터 시각화 코드 두 줄을 추상화합니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()

def load_csv():
    df.load_csv("Iris.csv")

def show_info():
    df.show_file_info()
    df.show_cols()

def visualize():
    # df.show_hist()
    # df.show_boxplot('Species', 'SepalWidthCm')
    pass

#(1) 데이터 로드
load_csv()

#(2) 데이터 정보 표시
show_info()

#(3) 데이터 시각화
visualize()
```

```
#(4) 학습/테스트 데이터 준비하기
a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
model.set_data(a, b, c, d)

#(5) 머신러닝 모델 실행하기
model.run_LR()
model.run_DT()
model.run_NN()
model.run_SVM()
```

학습과 테스트 데이터 셋을 준비하는 코드는 다음과 같이 prepare_data라는 함수로 추상화합니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()

def load_csv():
    df.load_csv("Iris.csv")

def show_info():
    df.show_file_info()
    df.show_cols()

def visualize():
```

```
        # df.show_hist()
        # df.show_boxplot('Species', 'SepalWidthCm')
        pass

    def prepare_data():
        a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        model.set_data(a, b, c, d)

#(1) 데이터 로드
load_csv()

#(2) 데이터 정보 표시
show_info()

#(3) 데이터 시각화
visualize()

#(4) 학습/테스트 데이터 준비하기
prepare_data()

#(5) 머신러닝 모델 실행하기
model.run_LR()
model.run_DT()
model.run_NN()
model.run_SVM()
```

마지막으로 모델을 실행하는 부분을 추상화합니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()

def load_csv():
    df.load_csv("Iris.csv")

def show_info():
    df.show_file_info()
    df.show_cols()

def visualize():
    # df.show_hist()
    # df.show_boxplot('Species', 'SepalWidthCm')
    pass

def prepare_data():
    a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
    model.set_data(a, b, c, d)

def run_models():
    model.run_LR()
    model.run_DT()
```

```
        model.run_NN()
        model.run_SVM()
```

```
#(1) 데이터 로드
load_csv()

#(2) 데이터 정보 표시
show_info()

#(3) 데이터 시각화
visualize()

#(4) 학습/테스트 데이터 준비하기
prepare_data()

#(5) 머신러닝 모델 실행하기
run_models()
```

그런데, 사실상 (1), (2), (3), (4), (5)도 모두 묶어 run이라는 함수를 추상화하겠습니다.

```
from myai import *

df = MyDataFrame()
model = MyModel()
```

```
def load_csv():
    df.load_csv("Iris.csv")

def show_info():
    df.show_file_info()
    df.show_cols()

def visualize():
    # df.show_hist()
    # df.show_boxplot('Species', 'SepalWidthCm')
    pass

def prepare_data():
    a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
    model.set_data(a, b, c, d)

def run_models():
    model.run_LR()
    model.run_DT()
    model.run_NN()
    model.run_SVM()

def run():
    # (1) 데이터 로드
    load_csv()
    # (2) 데이터 정보 표시
```

```
    show_info()
    # (3) 데이터 시각화
    visualize()
    # (4) 학습/테스트 데이터 준비하기
    prepare_data()
    # (5) 머신러닝 모델 실행하기
    run_models()
```

```
run()
```

함수를 이용한 코드 추상화를 마쳤습니다. 결과적으로 6개의 함수가 만들어졌습니다. 이제 맨 뒤에 있는 df, model 객체와 6개의 함수를 묶어 MachineLearning 이라는 클래스로 작성해보겠습니다. 가장 먼저, 아래와 같이 클래스 선언합니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    def load_csv():
        df.load_csv("Iris.csv")

    def show_info():
        df.show_file_info()
```

```
        df.show_cols()

    def visualize():
        # df.show_hist()
        # df.show_boxplot('Species', 'SepalWidthCm')
        pass

    def prepare_data():
        a, b, c, d = df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        model.set_data(a, b, c, d)

    def run_models():
        model.run_LR()
        model.run_DT()
        model.run_NN()
        model.run_SVM()

    def run():
        # (1) 데이터 로드
        load_csv()
        # (2) 데이터 정보 표시
        show_info()
        # (3) 데이터 시각화
        visualize()
        # (4) 학습/테스트 데이터 준비하기
        prepare_data()
```

```
        # (5) 머신러닝 모델 실행하기
        run_models()

run()
```

그러면 오류가 합니다. 오류를 고치기 위하여 다음과 같이 곳곳에 self 키워드 코드를 추가합니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    def load_csv(self):
        self.df.load_csv("Iris.csv")

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    def prepare_data(self):
```

```
            a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
            self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()

run()
```

클래스는 객체를 만들라고 있는 것이지요. 다음과 같이 클래스를 작성하여 코드를 완성합니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    def load_csv(self):
        self.df.load_csv("Iris.csv")

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
```

```
            self.model.run_SVM()

        def run(self):
            # (1) 데이터 로드
            self.load_csv()
            # (2) 데이터 정보 표시
            self.show_info()
            # (3) 데이터 시각화
            self.visualize()
            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()

    ml = MachineLearning()
    ml.run()
```

MachineLearning 클래스, 라이브러리로 만들기

MachineLearning 클래스를 라이브러리 모듈로 만들어 보겠습니다. 현 단계에서 MachineLearning 클래스를 라이브러리로 쓸 수 있을까요? 그렇지 않다면 무엇이 문제일까요?

사실 이 클래스는 가만 보면 라이브러리로 사용할 수 없는 부분이 들어 있습니다. 바로 아래 강조된 부분 때문에 그렇습니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    def load_csv(self):
        self.df.load_csv("Iris.csv")

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass
```

```
    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()


ml = MachineLearning()
ml.run()
```

강조된 코드는 현재 사용하는 데이터 Iris.csv 파일과 관련되어 있어서 혹 다른 데이터 파일로 바꿀 경우 우리가 무조건 수정해야 하는 코드입니다.

방법은 다음과 같습니다. MachineLearning 클래스를 상속받는 새로운 클래스 IrisMachineLearning 클래스를 작성한 후 강조된 코드를 IrisMachineLearning으로 옮기는 방법으로 진행합니다. 그러면 자연스럽게 MachineLearning 클래스는 라이브러리 클래스로 사용할 수 있게 됩니다. 다음은 MachineLearning 클래스를 상속받아 IrisMachineLearning 클래스를 선언한 코드입니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    def load_csv(self):
        self.df.load_csv("Iris.csv")

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
```

```
            pass

        def prepare_data(self):
            a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
            self.model.set_data(a, b, c, d)

        def run_models(self):
            self.model.run_LR()
            self.model.run_DT()
            self.model.run_NN()
            self.model.run_SVM()

        def run(self):
            # (1) 데이터 로드
            self.load_csv()
            # (2) 데이터 정보 표시
            self.show_info()
            # (3) 데이터 시각화
            self.visualize()
            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()

class IrisMachineLearning (MachineLearning):
    pass
```

```
ml = MachineLearning()
ml.run()
```

자, 그런데 클래스는 뭐하라고 있는 것인가요? 객체를 만들라고 있는 것이지요. 따라서 IrisMachineLearning 클래스로 객체를 생성해야 합니다.

```
ml = IrisMachineLearning()
ml.run()
```

자, 그렇다면 이제 load_csv 함수를 IrisMachineLearning 클래스로 옮깁니다. 전체를 옮기는 것은 아니고 껍데기는 남겨놓고, 대신 상속받은 클래스에서 재정의하도록 코드를 작성하는 것입니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()
```

```
    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()
```

```
class IrisMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv("Iris.csv")


ml = IrisMachineLearning()
ml.run()
```

프로그램을 실행하면 문제없이 실행되는 것을 볼 수 있습니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my03.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              150 non-null    int64
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm',
```

```
          'Species'],
         dtype='object')
(90, 6)
(60, 6)
인식률: 98.3
인식률: 95.0
인식률: 96.7
인식률: 96.7

Process finished with exit code 0
```

그런데 여기서 한가지! 아래와 같이 load_csv 멤버 함수를 재정의하고 있는데요, 여러분이 실수로 재정의하지 않으면 어떤 일이 일어날까요? 당연히 Iris.csv 파일을 로드하지 않기 때문에 이후 오류가 발생할 수 있겠지요.

```
class IrisMachineLearning (MachineLearning):
    def load_csv(self):
            self.df.load_csv("Iris.csv")


ml = IrisMachineLearning()
ml.run()
```

객체지향 프로그래밍 언어에서는 추상 함수를 선언하여 반드시 재정의하도록 강제할 수 있는데요, 다음과 같이 코드를 작성합니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
```

```
            self.model.run_NN()
            self.model.run_SVM()

    def run(self):
            # (1) 데이터 로드
            self.load_csv()
            # (2) 데이터 정보 표시
            self.show_info()
            # (3) 데이터 시각화
            self.visualize()
            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()

class IrisMachineLearning (MachineLearning):
    def load_csv(self):
            self.df.load_csv("Iris.csv")

ml = IrisMachineLearning()
ml.run()
```

즉, load_csv 함수를 추상 함수를 선언함으로써 이후 상속받는 클래스에서 무조건 재정의하도록, 재정의하지 않으면 오류가 발생하도록 강제할 수 있습니다.

이제 시각화하는 visualize 함수를 살펴보겠습니다.

시각화는 반드시 하는 것이 아니라 여러분이 원할 경우 하면 되는 것입니다. 따라서 visualize 함수는 그냥 두겠습니다.

데이터를 준비하는 prepare_data 멤버 함수를 살펴보겠습니다. 이 코드를 보면 학습과 테스트에 사용할 데이터를 생성하는 코드가 들어 있습니다. 여러분이 csv 데이터 파일을 바꿀 때 이 부분도 같이 수정되어야 합니다. 머신러닝을 위한 데이터 생성은 반드시 해야 하므로 이 함수도 추상 함수로 선언하여 껍데기를 남겨두고 상속받는 IrisMachineLearning 클래스로 옮기겠습니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
```

```
            pass

        @abstractmethod
        def prepare_data(self):
            pass

        def run_models(self):
            self.model.run_LR()
            self.model.run_DT()
            self.model.run_NN()
            self.model.run_SVM()

        def run(self):
            # (1) 데이터 로드
            self.load_csv()
            # (2) 데이터 정보 표시
            self.show_info()
            # (3) 데이터 시각화
            self.visualize()
            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()

class IrisMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv("Iris.csv")
```

```
        def prepare_data(self):
            a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
            self.model.set_data(a, b, c, d)
```

```
ml = IrisMachineLearning()
ml.run()
```

자, 이제 코드가 제법 완성되었습니다. 다시 한번 MachineLearning 클래스를 살펴보기 바랍니다. 이제 이 클래스를 라이브러리로 활용할 수 있을까요? 네 그렇습니다. 라이브러리로 활용할 수 없는 부분을 IrisMachineLearning 클래스로 밀어 넣었기 때문에 이제 가능합니다. 여러분 수고 하셨습니다.

만일, 성별 분류하도록 코드를 수정하려면 어떻게 해야 할까요? 바로 아래와 같이 load_csv 함수에서 male_female.csv 파일을 로드하면 됩니다. 이때 클래스 이름도 IrisMachineLearning이 아닌 MaleFemaleMachineLearning으로 바꾸면 좋을 듯 합니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    @abstractmethod
```

```
def load_csv(self):
    pass

def show_info(self):
    self.df.show_file_info()
    self.df.show_cols()

def visualize(self):
    # self.df.show_hist()
    # self.df.show_boxplot('Species', 'SepalWidthCm')
    pass

@abstractmethod
def prepare_data(self):
    pass

def run_models(self):
    self.model.run_LR()
    self.model.run_DT()
    self.model.run_NN()
    self.model.run_SVM()

def run(self):
    # (1) 데이터 로드
    self.load_csv()
    # (2) 데이터 정보 표시
    self.show_info()
```

```
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()

class MaleFemaleMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv("male_female.csv")

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        self.model.set_data(a, b, c, d)

ml = MaleFemaleMachineLearning()
ml.run()
```

이전 Iris.csv 데이터 파일과 비교해볼 때 컬럼 이름이 바뀌었기 때문에 prepare_data 멤버 함수도 수정해야 합니다. 참고로, male_female.csv 파일의 컬럼 이름을 출력해보면 다음과 같습니다.

```
Index(['Id', 'Height', 'Weight', 'FeetSize', 'Year', 'Sex'], dtype='object')
```

따라서, 이러한 컬럼을 이용하여 우리가 원하는 학습 및 테스트 데이터를

생성하면 됩니다.

```
from  myai  import  *

class  MachineLearning:
      df  =  MyDataFrame()
      model  =  MyModel()

      @abstractmethod
      def  load_csv(self):
            pass

      def  show_info(self):
            self.df.show_file_info()
            self.df.show_cols()

      def  visualize(self):
            #  self.df.show_hist()
            #  self.df.show_boxplot('Species',  'SepalWidthCm')
            pass

      @abstractmethod
      def  prepare_data(self):
            pass

      def  run_models(self):
            self.model.run_LR()
```

```
            self.model.run_DT()
            self.model.run_NN()
            self.model.run_SVM()

        def run(self):
            # (1) 데이터 로드
            self.load_csv()
            # (2) 데이터 정보 표시
            self.show_info()
            # (3) 데이터 시각화
            self.visualize()
            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()

class MaleFemaleMachineLearning (MachineLearning):
        def load_csv(self):
            self.df.load_csv("male_female.csv")

        def prepare_data(self):
            a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
            self.model.set_data(a, b, c, d)


ml = MaleFemaleMachineLearning()
```

```
ml.run()
```

실행 결과는 다음과 같습니다.

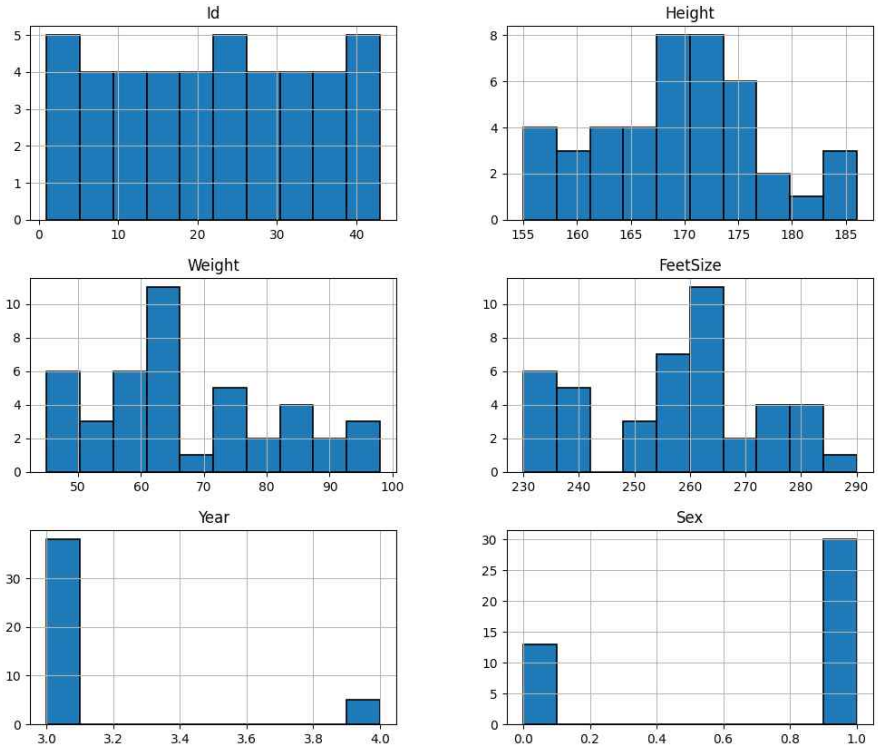
```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my03.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43 entries, 0 to 42
Data columns (total 6 columns):
 #    Column      Non-Null Count   Dtype
---   ------      --------------   -----
 0    Id          43 non-null      int64
 1    Height      43 non-null      int64
 2    Weight      43 non-null      int64
 3    FeetSize    43 non-null      int64
 4    Year        43 non-null      int64
 5    Sex         43 non-null      int64
dtypes: int64(6)
memory usage: 2.1 KB
Index(['Id', 'Height', 'Weight', 'FeetSize', 'Year', 'Sex'], dtype='object')
(25, 6)
(18, 6)
인식률: 88.9
인식률: 77.8
인식률: 88.9
인식률: 83.3
```

```
Process finished with exit code 0
```

시각화를 하려면 어떻게 해야 할까요? 간단합니다. visualize 멤버 함수를 재정의하면 됩니다.

```
class MaleFemaleMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv("male_female.csv")

    def visualize(self):
        self.df.show_hist()

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['Height', 'Weight', 'FeetSize'],
'Sex', 0.4)
        self.model.set_data(a, b, c, d)


ml = MaleFemaleMachineLearning()
ml.run()
```

다음은 프로그램을 실행한 모습입니다.



이제까지 작성했던 것을 다시 한번 정리해보겠습니다. 우선 전체 코드는 다음과 같습니다.

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()
```

```
    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    @abstractmethod
    def prepare_data(self):
        pass

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
```

```
            self.show_info()
            # (3) 데이터 시각화
            self.visualize()
            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()
```

```
class MaleFemaleMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv("male_female.csv")

    def visualize(self):
        self.df.show_hist()

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['Height', 'Weight', 'FeetSize'],
'Sex', 0.4)
        self.model.set_data(a, b, c, d)

ml = MaleFemaleMachineLearning()
ml.run()
```

우리가 이제까지 한 일은 무엇을까요? MachineLearning 클래스를 라이브러리화하기 위하여 노력했던 것이지요? 어떻게 노력했을까요? MachineLearning 클래스를 라이브러리화할 수 없는 코드를 자식 클래스

인 IrisMachineLearning, MaleFemaleMachineLearnine로 밀어 내려서 이동하여 분리했습니다.

그 결과 이제는 아래에 강조한 IrisMachineLearning 클래스 모듈은 라이브러리로 활용할 수가 있겠지요?

```
from myai import *

class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        pass

    @abstractmethod
    def prepare_data(self):
        pass
```

```
    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()

class MaleFemaleMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv("male_female.csv")

    def visualize(self):
        self.df.show_hist()

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['Height', 'Weight', 'FeetSize'],
```

```
'Sex', 0.4)
            self.model.set_data(a, b, c, d)

ml = MaleFemaleMachineLearning()
ml.run()
```

따라서 IrisMachineLearning 클래스를 라이브러리 파일 myai.py로 옮기겠습니다. 결과적으로 myai.py는 다음과 같이 모두 3개의 클래스로 구성됩니다. 아래에 강조한 부분이 조금 전에 추가한 IrisMachineLearning 모듈입니다.

```
import numpy as np # 수학 연산 수행을 위한 모듈
import pandas as pd # 데이터 처리를 위한 모듈
import seaborn as sns # 데이터 시각화 모듈
import matplotlib.pyplot as plt # 데이터 시각화 모듈

# 다양한 분류 알고리즘 패키지를 임포트함.
from sklearn.linear_model import LogisticRegression  # Logistic Regression 알고리즘
#from sklearn.cross_validation import train_test_split # 데이타 쪼개주는 모듈

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier  # for K nearest neighbours
from sklearn import svm  #for Support Vector Machine (SVM) Algorithm
```

```
from sklearn import metrics #for checking the model accuracy
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree
Algoithm

class MyDataFrame: #gildong
    data_frame = 0

    def load_csv(self, f):
        # CSV 파일 읽어오기
        self.data_frame = pd.read_csv(f)

    def show_file_info(self):
        self.data_frame.info()

    def show_head(self):
        print(self.data_frame.head())

    def show_col_name(self):
        for col in self.data_frame.columns:
            print(col)

    def show_cols(self):
        print(self.data_frame.columns)

    def show_hist(self):
        self.data_frame.hist(edgecolor='black', linewidth=1.2)
        fig = plt.gcf()
```

```
            fig.set_size_inches(12, 10)
            plt.show()

        def plot(self, a, b, c):
            # 읽어온 데이터 표시하기
            cl = self.data_frame[c].unique()

            col = ['orange', 'blue', 'red', 'yellow', 'black', 'brown']

            fig = self.data_frame[self.data_frame[c] == cl[0]].plot(kind='scatter',
x=a, y=b, color=col[0], label=cl[0])

            for i in range(len(cl) - 1):
                self.data_frame[self.data_frame[c] == cl[i +
1]].plot(kind='scatter', x=a, y=b, color=col[i + 1], label=cl[i + 1],
                                                                    ax=fig)
            fig.set_xlabel(a)
            fig.set_ylabel(b)
            fig.set_title(a + " vs. " + b)
            fig = plt.gcf()
            fig.set_size_inches(10, 6)
            plt.show()

        def show_boxplot(self, a, b):
            f, sub = plt.subplots(1, 1, figsize=(8, 6))
            sns.boxplot(x=self.data_frame[a], y=self.data_frame[b], ax=sub)
            sub.set(xlabel=a, ylabel=b)
```

```
        plt.show()

    def show_violenplot(self, a, b):
        plt.figure(figsize=(8, 6))
        plt.subplot(1, 1, 1)
        sns.violinplot(x=a, y=b, data=self.data_frame)
        plt.show()

    def show_heatmap(self):
        plt.figure(figsize=(12, 8))
        sns.heatmap(self.data_frame.corr(), annot=True,
cmap='cubehelix_r')
        plt.show()

    def prepare_data(self, input_cols, target_col, ratio):
        train, test = train_test_split(self.data_frame, test_size=ratio)
        # train=70% and test=30%
        print(train.shape)
        print(test.shape)

        # 학습용 문제, 학습용 정답
        train_X = train[input_cols]  # 키와 발크기만 선택
        train_y = train[target_col]  # 정답 선택

        # 테스트용 문제, 테스트용 정답
        test_X = test[input_cols]  # taking test data features
        test_y = test[target_col]  # output value of test data
```

```
        return train_X, train_y, test_X, test_y

    def drop(self, col):
        self.data_frame.drop(col, axis=1, inplace=True)

    def show_unique(self, col):
        print(self.data_frame[col].unique())

    def to_numeric(self, col, m, new_col):
        self.data_frame[new_col] = self.data_frame[col].map(m)


class MyModel: #youngja
    train_X = []
    train_y = []

    test_X = []
    test_y = []

    def set_data(self, i, j, k, l):
        self.train_X = i
        self.train_y = j
        self.test_X = k
        self.test_y = l

    def run_SVM(self):
        gildong = svm.SVC()
```

```
        gildong.fit(self.train_X, self.train_y)  # 가르친 후
        prediction = gildong.predict(self.test_X)  # 얼마나 맞히는지
테스트

        rate1 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate1))


    def run_LR(self):
        cheolsu = LogisticRegression()
        cheolsu.fit(self.train_X, self.train_y)
        prediction = cheolsu.predict(self.test_X)

        rate2 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate2))


    def run_DT(self):
        youngja = DecisionTreeClassifier()
        youngja.fit(self.train_X, self.train_y)
        prediction = youngja.predict(self.test_X)

        rate3 = metrics.accuracy_score(prediction, self.test_y) * 100
        print('인식률: {0:.1f}'.format(rate3))

    def run_NN(self):
        minsu = KNeighborsClassifier(n_neighbors=3)  # this examines 3
```

```
neighbours for putting the new data into a class
            minsu.fit(self.train_X, self.train_y)
            prediction = minsu.predict(self.test_X)

            rate4 = metrics.accuracy_score(prediction, self.test_y) * 100
            print('인식률: {0:.1f}'.format(rate4))
```

```
from abc import *
class MachineLearning:
    df = MyDataFrame()
    model = MyModel()

    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    @abstractmethod
```

```
    def prepare_data(self):
        pass

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()
```

자 그러면 다시 처음부터 새로운 프로그램을 작성해보겠습니다. 새로운 파이썬 파일을 만들고 라이브러리를 사용할 수 있도록 임포트 합니다. 그리고 다음과 같이 가장 간단한 프로그램 코드를 작성합니다.

```
from myai import *
```

```
ml = MachineLearning()
ml.run()
```

이 프로그램을 실행하면 어떻게 될까요? 아래와 같이 오류가 발생합니다. 사실 머신러닝 객체 ml을 생성한 후 csv 파일을 로드해야 하는데 그것을 하지 않아서 생기는 문제입니다.

```
Traceback (most recent call last):
  File "C:\Users\USER\PycharmProjects\pythonProject3\my04.py", line 4, in
<module>
    ml.run()
  File "C:\aiLib\myai.py", line 183, in run
    self.show_info()
  File "C:\aiLib\myai.py", line 161, in show_info
    self.df.show_file_info()
  File "C:\aiLib\myai.py", line 25, in show_file_info
    self.data_frame.info()
AttributeError: 'int' object has no attribute 'info'
```

MachineLearning 클래스는 load_csv, prepare_data 추상 함수를 가지고 있어서 반드시 재정의해야 하는데 재정의하지 않고 있습니다. load_csv 함수를 재정의하여 csv 파일을 로드해야 하고, prepare_data 함수에서 학습과 테스트시 어떤 컬럼 데이터를 이용할지 지정해 주어야 합니다. 그렇지 않아서 생기는 문제입니다.

추상 함수를 재정의하지 않으면 무조건 오류가 발생하도록 해서 이러한 오류를 사전에 막을 수 있는데요, 바로 MachineLearning 클래스를 추상

클래스로 선언하는 것입니다.

```
from abc import *
class MachineLearning(metaclass = ABCMeta):
    df = MyDataFrame()
    model = MyModel()

    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    @abstractmethod
    def prepare_data(self):
        pass

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
```

```
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()
        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()
```

그러면 다시 컴파일을 수행해 봅니다. MachineLearning 클래스를 추상 클래스로 선언하였기 때문에 아래와 같이 추상 함수를 재정의하도록 오류를 냅니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my04.py
Traceback (most recent call last):
  File "C:\Users\USER\PycharmProjects\pythonProject3\my04.py", line 3, in
<module>
    ml = MachineLearning()
TypeError: Can't instantiate abstract class MachineLearning with abstract
methods load_csv, prepare_data
```

```
Process finished with exit code 1
```

따라서 아래와 같이 두 개의 가상함수를 재정의하도록 코드를 변경해야 합니다.

```
from myai import *

class IrisMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv('Iris.csv')

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm'], 'Species', 0.4)
        self.model.set_data(a, b, c, d)


ml = IrisMachineLearning()
ml.run()
```

Step1 1년 후에 이 코드를 다시 작성한다면?

실행하면 결과가 잘 출력됩니다. 그런데 한 가지 아쉬운 점은 추상 함수 prepare_data 함수를 재정의할 때 그 내부에 있는 코드를 쉽게 암기하여 작성할 수 없다는 것입니다. 6개월, 1년 후 다시 작성하려면 아마 그 내용이 거의 생각이 나지 않을 것 같습니다. 그래서 이 부분을 잊어버려도 손쉽게 작성할 수 있도록 코드를 수정하면 좋을 듯 합니다.

여기에서는 단순히 입력 컬럼과 타겟 컬럼을 설정만 하고, 그걸 가지고 데이터를 만드는 코드는 라이브러리 안에 숨겨 버리는 것입니다. 어떤가요, 이해가 되었나요?

```
from myai import *

class IrisMachineLearning (MachineLearning):
    def load_csv(self):
        self.df.load_csv('Iris.csv')

    def set_input_cols(self):
        self.input_cols = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm']

    def set_target_col(self):
        self.target_col = 'Species'
```

```
ml = IrisMachineLearning()
ml.run()
```

앞서 있었던 prepare_data 함수를 추상 함수가 아닌 일반 함수로 바꾸고, 여기에 있는 set_input_cols와 set_target_col을 추상 함수로 작성하는 것이지요. 재정의하지 않으면 오류가 나도록요.

MachineLearning 클래스에 가서 새로운 추상 함수 2개를 만들고, 원래 우리가 재정의했었던 process_data 함수로 MachineLearning 클래스로 옮긴 후 일반 함수로 수정합니다.

```
from abc import *
class MachineLearning(metaclass = ABCMeta):
    df = MyDataFrame()
    model = MyModel()

    input_cols = 0
    target_col = 0

    @abstractmethod
    def load_csv(self):
        pass

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()
```

```
    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    @abstractmethod
    def set_input_cols(self):
        pass

    @abstractmethod
    def set_target_col(self):
        pass

    def prepare_data(self):
        a, b, c, d = self.df.prepare_data(self.input_cols, self.target_col,
0.4)
        self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        # (1) 데이터 로드
        self.load_csv()
```

```
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()

        self.set_input_cols()
        self.set_target_col()

        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()
```

자, 이제 6개월 혹은 1년 후에 이 코드를 기억이 나지 않은 상태에서 작성한다고 가정하고 코드를 작성해보세요. 새로운 파이썬 파일을 작성하고 MachineLearning 객체를 생성한 후 run 멤버 함수를 호출하겠지요.

```
from myai import *

ml = MachineLearning()
ml.run()
```

코드를 실행하면 추상 클래스여서 객체를 만들 수 없다는 오류 메시지와 함께 재정의해야 할 추상화 함수로 아래에 load_csv, set_input_cols, set_target_col 추상 멤버 함수가 있다는 것도 보여줍니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
```

```
C:\Users\USER\PycharmProjects\pythonProject3\my05.py
Traceback (most recent call last):
  File "C:\Users\USER\PycharmProjects\pythonProject3\my05.py", line 3, in <module>
    ml = MachineLearning()
TypeError: Can't instantiate abstract class MachineLearning with abstract methods load_csv, set_input_cols, set_target_col

Process finished with exit code 1
```

그래서 아래와 같이 추상 함수 재정의를 하겠지요. 일단 재빠르게 다음과 같이 작성하겠습니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def load_csv(self):
        pass

    def set_input_cols(self):
        pass

    def set_target_col(self):
        pass


ml = MaleFemaleMachineLearning()
```

```
ml.run()
```

로드 할 데이터로 male_female.csv 파일을 로드하도록 코드를 수정 하도록 하겠습니다. 아래와 같습니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def load_csv(self):
        self.df.load_csv('male_female.csv')

    def set_input_cols(self):
        pass

    def set_target_col(self):
        pass


ml = MaleFemaleMachineLearning()
ml.run()
```

다행인 것은 일단 실행해 보면 오류가 나긴 나지만 아래 강조한 부분과 같이 컬럼 이름들이 출력되도록 라이브러리가 만들어져 있습니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my05.py
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 43 entries, 0 to 42
Data columns (total 6 columns):
 #    Column      Non-Null Count   Dtype
---   ------      --------------   -----
 0    Id          43 non-null      int64
 1    Height      43 non-null      int64
 2    Weight      43 non-null      int64
 3    FeetSize    43 non-null      int64
 4    Year        43 non-null      int64
 5    Sex         43 non-null      int64
dtypes: int64(6)
memory usage: 2.1 KB
Index(['Id', 'Height', 'Weight', 'FeetSize', 'Year', 'Sex'], dtype='object')
(25, 6)
(18, 6)
Traceback (most recent call last):
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\indexes\base.py", line 3790, in get_loc
    return self._engine.get_loc(casted_key)
  File "index.pyx", line 152, in pandas._libs.index.IndexEngine.get_loc
  File "index.pyx", line 181, in pandas._libs.index.IndexEngine.get_loc
  File "pandas\_libs\hashtable_class_helper.pxi", line 7080, in
pandas._libs.hashtable.PyObjectHashTable.get_item
  File "pandas\_libs\hashtable_class_helper.pxi", line 7088, in
pandas._libs.hashtable.PyObjectHashTable.get_item
KeyError: 0
```

```
The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "C:\Users\USER\PycharmProjects\pythonProject3\my05.py", line 15, in <module>
    ml.run()
  File "C:\aiLib\myai.py", line 202, in run
    self.prepare_data()
  File "C:\aiLib\myai.py", line 181, in prepare_data
    a, b, c, d = self.df.prepare_data(self.input_cols, self.target_col, 0.4)
  File "C:\aiLib\myai.py", line 85, in prepare_data
    train_X = train[input_cols]  # 키와 발크기만 선택
  File "C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\pandas\core\frame.py", line 3896, in __getitem__
    indexer = self.columns.get_loc(key)
  File "C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\pandas\core\indexes\base.py", line 3797, in get_loc
    raise KeyError(key) from err
KeyError: 0

Process finished with exit code 1
```

따라서 이를 참고하여 입력 컬럼과 타겟 컬럼을 아래와 같이 설정합니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def load_csv(self):
        self.df.load_csv('male_female.csv')

    def set_input_cols(self):
        self.input_cols = 'Height', 'Weight', 'FeetSize'

    def set_target_col(self):
        self.target_col = 'Sex'


ml = MaleFemaleMachineLearning()
ml.run()
```

자, 어떤가요.? 이제 한참 시간이 지난 후에도 이런 코드를 작성하는 것이 가능할까요?

세 개의 추상 함수를 재정의하면서 조금 더 개선할 점도 있을 수 있습니다. 데이터 로드 함수 load_csv도 아래 set_input_cols, set_target_col 추상 함수처럼 file_name 이라는 멤버 변수에 로드할 csv 파일 이름을 지정하고, 실제 파일을 로드하는 코드는 라이브러리에서 하도록 하는 것입니다. 말이 나온 김에 그렇게 한 번 해보도록 하겠습니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def set_file(self):
        self.file_name = 'male_female.csv'

    def set_input_cols(self):
        self.input_cols = 'Height', 'Weight', 'FeetSize'

    def set_target_col(self):
        self.target_col = 'Sex'


ml = MaleFemaleMachineLearning()
ml.run()
```

load_csv 파일 대신에 set_file을 호출하도록 합니다. 이를 위하여 MachineLearning 모듈을 다음과 같이 수정합니다.

```
from abc import *
class MachineLearning(metaclass = ABCMeta):
    df = MyDataFrame()
    model = MyModel()

    input_cols = 0
    target_col = 0
    file_name = 0
```

```
        @abstractmethod
        def set_file(self):
            pass

        def load_csv(self):
            self.df.load_csv(self.file_name)

        def show_info(self):
            self.df.show_file_info()
            self.df.show_cols()

        def visualize(self):
            # self.df.show_hist()
            # self.df.show_boxplot('Species', 'SepalWidthCm')
            pass

        @abstractmethod
        def set_input_cols(self):
            pass

        @abstractmethod
        def set_target_col(self):
            pass

        def prepare_data(self):
            a, b, c, d = self.df.prepare_data(self.input_cols, self.target_col,
0.4)
```

```
        self.model.set_data(a, b, c, d)

    def run_models(self):
        self.model.run_LR()
        self.model.run_DT()
        self.model.run_NN()
        self.model.run_SVM()

    def run(self):
        self.set_file()

        # (1) 데이터 로드
        self.load_csv()
        # (2) 데이터 정보 표시
        self.show_info()
        # (3) 데이터 시각화
        self.visualize()

        self.set_input_cols()
        self.set_target_col()

        # (4) 학습/테스트 데이터 준비하기
        self.prepare_data()
        # (5) 머신러닝 모델 실행하기
        self.run_models()
```

자 이제 마지막으로 몇 가지 코드를 정리해보겠습니다. 현재 레벨 0에 있는 두 줄의 코드를 run이라는 함수로 추상화를 해보겠습니다. 아래와 같습니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def set_file(self):
        self.file_name = 'male_female.csv'

    def set_input_cols(self):
        self.input_cols = 'Height', 'Weight', 'FeetSize'

    def set_target_col(self):
        self.target_col = 'Sex'


def run():
    ml = MaleFemaleMachineLearning()
    ml.run()

run()
```

그리고 run 함수를 Application 이라는 이름의 클래스의 멤버 함수로 작성을 하겠습니다.

```
from myai import *
```

```
class MaleFemaleMachineLearning(MachineLearning):
    def set_file(self):
        self.file_name = 'male_female.csv'

    def set_input_cols(self):
        self.input_cols = 'Height', 'Weight', 'FeetSize'

    def set_target_col(self):
        self.target_col = 'Sex'


class Application:
    def run():
        ml = MaleFemaleMachineLearning()
        ml.run()

run()
```

그리고 run 멤버 함수를 정적 멤버 함수로 선언하겠습니다. 정적 멤버 함수는 개체를 만들지 않아도 바로 호출할 수 있는 함수입니다. 다음과 같습니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def set_file(self):
        self.file_name = 'male_female.csv'
```

```
    def set_input_cols(self):
        self.input_cols = 'Height', 'Weight', 'FeetSize'

    def set_target_col(self):
        self.target_col = 'Sex'


class Application:
    @staticmethod
    def run():
        ml = MaleFemaleMachineLearning()
        ml.run()

Application.run()
```

Application 클래스를 라이브러리로 사용하려면 무엇이 문제일까요. 다름이 아니라 라이브러리로 사용할 수 없는 클래스가 있기 때문입니다. MaleFemaleMachineLearning 클래스 안에는 데이터에 따라 달라지는 코드가 들어 있습니다. 따라서 MaleFemaleMachineLearning 클래스를 Application 클래스 밖으로 꺼내면 됩니다. 다음과 같이 run 함수 호출할 때 파라미터로 넘겨주면 됩니다.

```
from myai import *

class MaleFemaleMachineLearning(MachineLearning):
    def set_file(self):
        self.file_name = 'male_female.csv'
```

```
    def set_input_cols(self):
        self.input_cols = 'Height', 'Weight', 'FeetSize'

    def set_target_col(self):
        self.target_col = 'Sex'


class Application:
    @staticmethod
    def run(ml):
        ml.run()
```

```
Application.run(MaleFemaleMachineLearning())
```

이로써 응용 프레임워크를 만드는 과정이 어느 정도 완성 되었습니다. 그러면 이 코드를 이용해서 다시 처음부터 프로그램을 작성 해 보겠습니다. 먼저 가장 간단한 프로그램을 작성해보겠습니다.

```
from myai import *

Application.run(MachineLearning())
```

그러면 다음과 같은 오류 메시지가 표시됩니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my06.py
Traceback (most recent call last):
```

```
  File "C:\Users\USER\PycharmProjects\pythonProject3\my06.py", line 3, in <module>
    Application.run(MachineLearning())
TypeError: Can't instantiate abstract class MachineLearning with abstract methods set_file, set_input_cols, set_target_col

Process finished with exit code 1
```

MachineLearning 클래스는 추상 클래스이므로 인스턴스를 만들 수 없다는 오류 메시지와 함께 재정의해야 하는 추상 함수 3개를 보여줍니다. 따라서 이 3개의 멤버 함수를 재정의 해야 겠지요.

```
from myai import *

class IrisML (MachineLearning):
    def set_file(self):
        self.file_name = 'Iris.csv'

    def set_input_cols(self):
        pass

    def set_target_col(self):
        pass


Application.run(IrisML())
```

오류 메시지를 보면 위쪽에 데이터 파일에 있는 컬럼 이름들을 볼 수 있습니다. 이를 입력 컬럼과 타겟 컬럼에 적절히 복사하여 설정합니다.

```
from myai import *

class IrisML (MachineLearning):
    def set_file(self):
        self.file_name = 'Iris.csv'

    def set_input_cols(self):
        self.input_cols = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm']

    def set_target_col(self):
        self.target_col = 'Species'

Application.run(IrisML())
```

컴파일한 후 실행하면 결과가 잘 표시되는 것을 볼 수 있습니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my06.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #    Column            Non-Null Count   Dtype
```

```
---   ------            --------------   -----
 0    Id                150 non-null     int64
 1    SepalLengthCm     150 non-null     float64
 2    SepalWidthCm      150 non-null     float64
 3    PetalLengthCm     150 non-null     float64
 4    PetalWidthCm      150 non-null     float64
 5    Species           150 non-null     object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm', 'Species'], dtype='object')
(90, 6)
(60, 6)
인식률: 91.7
인식률: 91.7
인식률: 90.0
인식률: 90.0

Process finished with exit code 0
```

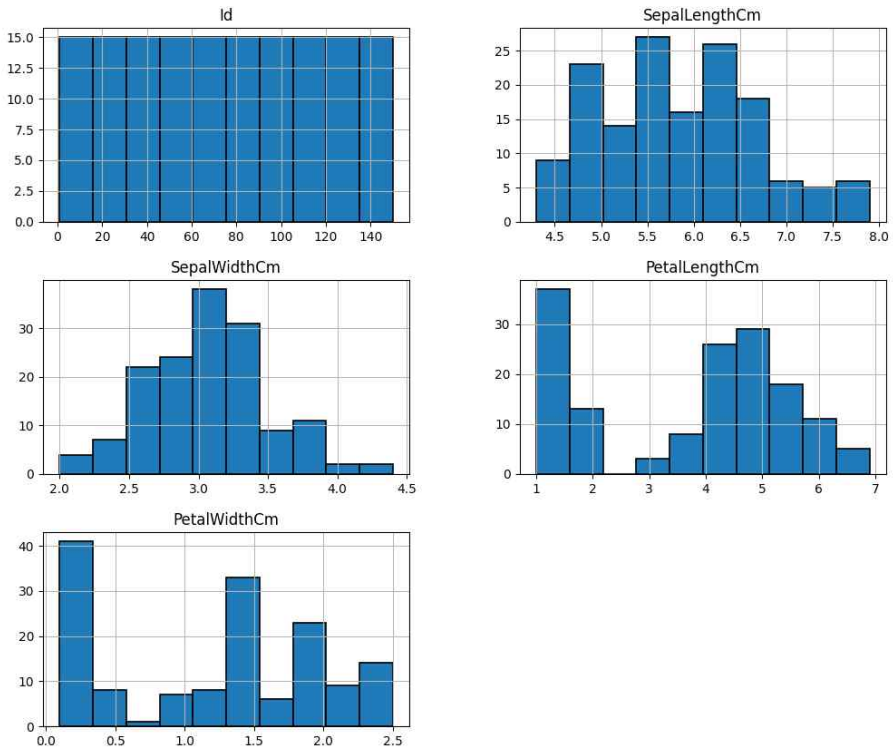
4개의 머신러닝 알고리즘을 이용해서 학습 후 테스트한 결과 인식 결과가 90% 이상 나오는 걸 볼 수 있습니다.

Step1 데이터 시각화와 전처리

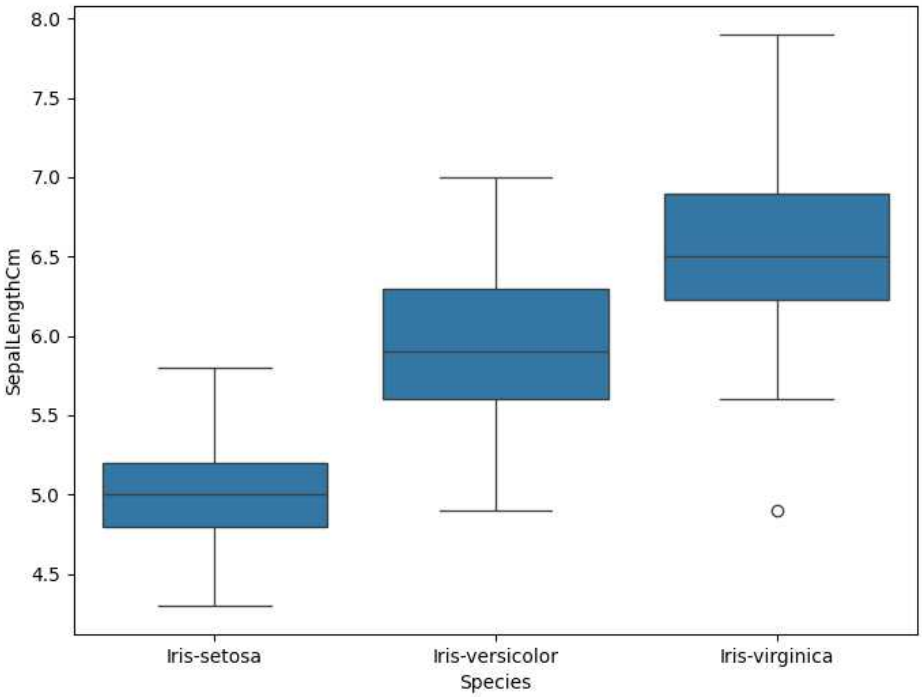
데이터 시각화를 하려면 다음과 같이 코드를 작성 할 수 있습니다. 히스토그램을 출력할 수도 있고, 박스 플랏을 출력할 수도 있습니다.

```
from myai import *

class IrisML (MachineLearning):
    def set_file(self):
        self.file_name = 'Iris.csv'

    def set_input_cols(self):
        self.input_cols = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm']

    def set_target_col(self):
        self.target_col = 'Species'

    def visualize(self):
        self.df.show_hist()
        self.df.show_boxplot('Species','SepalLengthCm')

Application.run(IrisML())
```

다음은 프로그램을 실행한 결과입니다.



붓꽃 유형별로 꽃 받침대 길이가 어떻게 분포되어 있는지 박스 형태로 시각화한 결과입니다.



컬럼 간에 상관관계를 보여주는 히트 맵을 표시해 봅니다. 하지만 문제가 생겼습니다. 아래 메시지의 맨 마지막에 강조한 부분처럼, 히트 맵을 보여주려면 모든 데이터가 숫자로 되어 있어야 하는데, 컬럼 중 붓꽃 유형을 표현하는 Species 칼럼이 숫자가 아닌 문자열이 들어가 있어서 발생하는 오류입니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my06.py
<class  'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #    Column            Non-Null Count   Dtype
---   ------            --------------   -----
 0    Id                150 non-null     int64
 1    SepalLengthCm     150 non-null     float64
 2    SepalWidthCm      150 non-null     float64
 3    PetalLengthCm     150 non-null     float64
 4    PetalWidthCm      150 non-null     float64
 5    Species           150 non-null     object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm',
       'Species'],
      dtype='object')
Traceback (most recent call last):
  File "C:\Users\USER\PycharmProjects\pythonProject3\my06.py", line 19, in
<module>
    Application.run(IrisML())
  File "C:\aiLib\myai.py", line 216, in run
    ml.run()
  File "C:\aiLib\myai.py", line 202, in run
    self.visualize()
  File "C:\Users\USER\PycharmProjects\pythonProject3\my06.py", line 16, in
visualize
    self.df.show_heatmap()
```

```
  File "C:\aiLib\myai.py", line 75, in show_heatmap
    sns.heatmap(self.data_frame.corr(), annot=True, cmap='cubehelix_r')
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\frame.py", line 10707, in corr
    mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\frame.py", line 1892, in to_numpy
    result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\internals\managers.py", line 1656, in as_array
    arr = self._interleave(dtype=dtype, na_value=na_value)
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\internals\managers.py", line 1715, in _interleave
    result[rl.indexer] = arr
ValueError: could not convert string to float: 'Iris-setosa'

Process finished with exit code 1
```

따라서 Species 컬럼에 있는 문자열 값을 정수로 바꿔 줘야 합니다. 이와 같이 데이터를 전처리하기 위하여 preprocess라는 멤버 함수를 추가로 정의하면 좋을 듯 합니다.

```
from abc import *
class MachineLearning(metaclass = ABCMeta):
    df = MyDataFrame()
    model = MyModel()

    input_cols = 0
    target_col = 0
    file_name = 0

    @abstractmethod
    def set_file(self):
        pass

    def load_csv(self):
        self.df.load_csv(self.file_name)

    def show_info(self):
        self.df.show_file_info()
        self.df.show_cols()

    def visualize(self):
        # self.df.show_hist()
        # self.df.show_boxplot('Species', 'SepalWidthCm')
        pass

    @abstractmethod
    def set_input_cols(self):
```

```
            pass

        @abstractmethod
        def set_target_col(self):
            pass

        def prepare_data(self):
            a, b, c, d = self.df.prepare_data(self.input_cols, self.target_col,
0.4)
            self.model.set_data(a, b, c, d)

        def run_models(self):
            self.model.run_LR()
            self.model.run_DT()
            self.model.run_NN()
            self.model.run_SVM()

        def preprocess(self):
            pass

        def run(self):
            self.set_file()

            # (1) 데이터 로드
            self.load_csv()
            # (2) 데이터 정보 표시
            self.show_info()
```

```
            self.preprocess()

            # (3) 데이터 시각화
            self.visualize()

            self.set_input_cols()
            self.set_target_col()

            # (4) 학습/테스트 데이터 준비하기
            self.prepare_data()
            # (5) 머신러닝 모델 실행하기
            self.run_models()
```

위와 같이 라이브러리를 수정하였고, 이제 우리가 작성하는 코드에서 다음과 같이 재정의합니다.

```
from myai import *

class IrisML (MachineLearning):
    def set_file(self):
        self.file_name = 'Iris.csv'

    def set_input_cols(self):
        self.input_cols = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm']
```

```
    def set_target_col(self):
        self.target_col = 'Species'

    def visualize(self):
        #self.df.show_hist()
        #self.df.show_boxplot('Species','SepalLengthCm')
        self.df.show_heatmap()

    def preprocess(self):
        self.df.show_unique('Species')

Application.run(IrisML())
```

붓꽃 종류로 어떤 것이 있는지 먼저 알아보고자 위와 같이 코드를 작성한 것입니다. 아직 데이터를 전처리하지 않았기 때문에 실행할 경우 오류는 여전히 발생합니다. 붓꽃 유형을 확인한 결과 3가지 유형이 존재함을 알 수 있습니다.

```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my06.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              150 non-null    int64
 1   SepalLengthCm   150 non-null    float64
```

```
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm',
       'Species'],
      dtype='object')
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
Traceback (most recent call last):
  File "C:\Users\USER\PycharmProjects\pythonProject3\my06.py", line 21, in
<module>
    Application.run(IrisML())
  File "C:\aiLib\myai.py", line 223, in run
    ml.run()
  File "C:\aiLib\myai.py", line 209, in run
    self.visualize()
  File "C:\Users\USER\PycharmProjects\pythonProject3\my06.py", line 16, in
visualize
    self.df.show_heatmap()
  File "C:\aiLib\myai.py", line 75, in show_heatmap
    sns.heatmap(self.data_frame.corr(), annot=True, cmap='cubehelix_r')
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\frame.py", line 10707, in corr
```

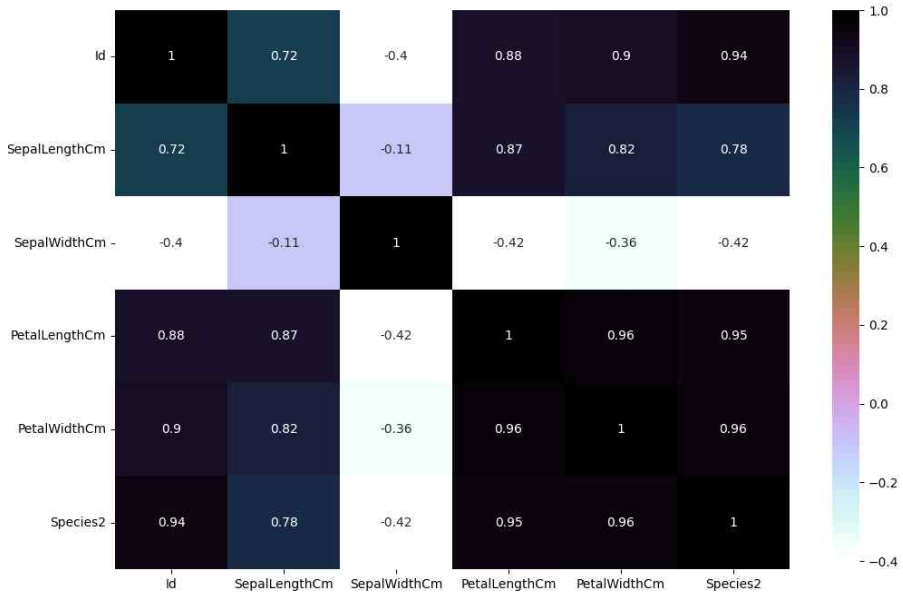
```
    mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\frame.py", line 1892, in to_numpy
    result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\internals\managers.py", line 1656, in as_array
    arr = self._interleave(dtype=dtype, na_value=na_value)
  File
"C:\Users\USER\PycharmProjects\pythonProject3\venv\lib\site-packages\panda
s\core\internals\managers.py", line 1715, in _interleave
    result[rl.indexer] = arr
ValueError: could not convert string to float: 'Iris-setosa'

Process finished with exit code 1
```

대치해야 할 꽃에 종류 문자열을 확인하고 각 유형에 대하여 어떤 숫자로 대치할지 설정합니다. 대치한 후 새롭게 생성할 컬럼 이름 Species2를 지정하고, 기존에 있는 붓꽃 컬럼 Species는 drop하여 삭제하도록 하였습니다.

```
from myai import *

class IrisML (MachineLearning):
    def set_file(self):
        self.file_name = 'Iris.csv'
```

```
    def set_input_cols(self):
        self.input_cols = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm']

    def set_target_col(self):
        self.target_col = 'Species'

    def visualize(self):
        #self.df.show_hist()
        #self.df.show_boxplot('Species','SepalLengthCm')
        self.df.show_heatmap()

    def preprocess(self):
        self.df.show_unique('Species')
        self.df.to_numeric('Species', {'Iris-setosa':0, 'Iris-versicolor':1,
'Iris-virginica':2}, 'Species2')
        self.df.drop('Species')


Application.run(IrisML())
```

프로그램을 실행 하였더니 특정 맵이 잘 표시 되고 있습니다.



이 중에서 가장 숫자를 큰 숫자를 찾아보면 96인데, 96 숫자가 가리키는 두 개의 컬럼 이름을 보면 꽃잎 길이와 꽃잎 너비입니다. 즉, 꽃잎 길이가 길수록 꽃님 너비도 길다라는 의미입니다. 그리고 95도 있는데, 이는 붓꽃의 종류와 꽃잎의 길이 상관관계입니다.

데이터를 성별 데이터(male_female.csv)로 바꾸기만 하면 최소한의 코딩으로 동일한 결과를 얻을 수 있습니다. 일단 다음 두 곳의 코드만 수정합니다.

```
from myai import *
```

```
class IrisML (MachineLearning):
    def set_file(self):
        self.file_name = 'male_female.csv'

    def set_input_cols(self):
        self.input_cols = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm']

    def set_target_col(self):
        self.target_col = 'Species'

    def visualize(self):
        #self.df.show_hist()
        #self.df.show_boxplot('Species','SepalLengthCm')
        self.df.show_heatmap()

    def preprocess(self):
        self.df.show_unique('Species')
        self.df.to_numeric('Species', {'Iris-setosa':0,
'Iris-versicolor':1,'Iris-virginica':2}, 'Species2')
        self.df.drop('Species')


Application.run(IrisML())
```

컴파일 후 실행하면 오류 메시지가 뜨지만, 다음과 같이 데이터 컬럼 이름들을 볼 수가 있습니다. 이를 이용하여 나머지 부분을 작성합니다.

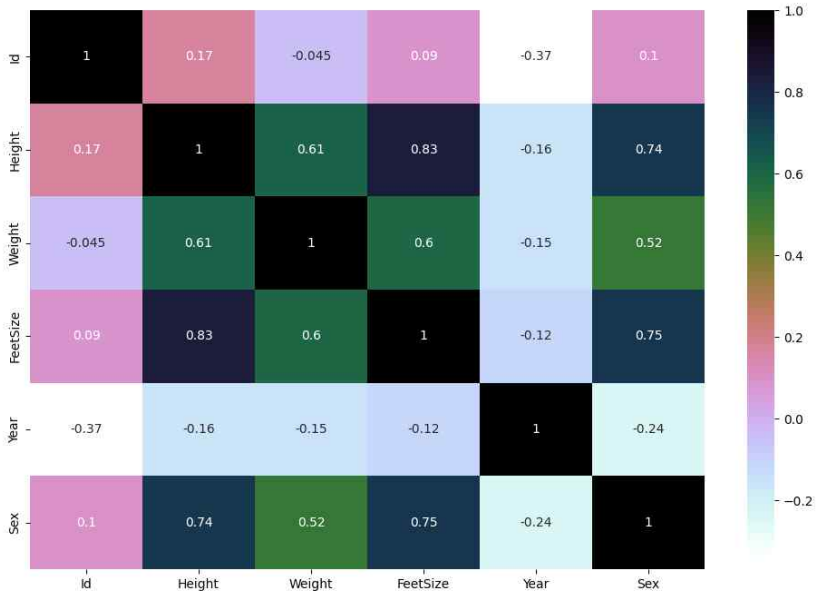
```
C:\Users\USER\PycharmProjects\pythonProject3\venv\Scripts\python.exe
C:\Users\USER\PycharmProjects\pythonProject3\my06.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43 entries, 0 to 42
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Id        43 non-null     int64
 1   Height    43 non-null     int64
 2   Weight    43 non-null     int64
 3   FeetSize  43 non-null     int64
 4   Year      43 non-null     int64
 5   Sex       43 non-null     int64
dtypes: int64(6)
memory usage: 2.1 KB
Index(['Id', 'Height', 'Weight', 'FeetSize', 'Year', 'Sex'], dtype='object')
(25, 6)
(18, 6)
```

입력 데이터로 키와 몸무게 그리고 발 사이즈를 설정하고, 타겟 데이터로 남녀 성별을 설정합니다. 그리고 데이터 전처리는 필요 없으며, 따라서 아래와 같이 pass 키워드만 작성합니다. 물론 preprocess 함수 자체를 삭제하여도 됩니다.

```
from myai import *

class IrisML (MachineLearning):
```

```
    def set_file(self):
        self.file_name = 'male_female.csv'

    def set_input_cols(self):
        self.input_cols = ['Height', 'Weight', 'FeetSize']

    def set_target_col(self):
        self.target_col = 'Sex'

    def visualize(self):
        #self.df.show_hist()
        #self.df.show_boxplot('Species','SepalLengthCm')
        self.df.show_heatmap()

    def preprocess(self):
        pass


Application.run(IrisML())
```

다음은 프로그램을 실행한 결과입니다. 특징 맵이 표시되었으며, 이로부터 키(Height)와 발크기(FeetSize)의 상관관계가 0.83으로 가장 큰 것을 알 수 있습니다.


Id
Height
Weight
FeetSize
Year
Sex
1
0.17
-0.045
0.09
-0.37
0.1
0.17
1
0.61
0.83
-0.16
0.74
-0.045
0.61
1
0.6
-0.15
0.52
0.09
0.83
0.6
1
-0.12
0.75
-0.37
-0.16
-0.15
-0.12
1
-0.24
0.1
0.74
0.52
0.75
-0.24
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2