과목명 : 기계학습

과제명 : 7, 8, 9장 예제 프로그램 작성

학 과 : 소프트웨어학과

제 출 일 자 : 2021.11.26

학 번 : 1726088

이 름 : 최민수

7장 예제

[문제] 예제 데이터(날씨에 따른 운동여부 데이터)를 전처리과정시키기

[코드]

```
from sklearn.preprocessing import LabelEncoder

weather = ['Sunny','Sunny','Overcast','Rainy','Rainy',
          'Rainy','Overcast','Sunny','Sunny','Rainy',
          'Sunny','Overcast','Overcast','Rainy']
play = ['No','No','Yes','Yes','Yes',
       'No','Yes','No','Yes','Yes',
       'Yes','Yes','Yes','No']

le= LabelEncoder()
X=le.fit_transform(weather) #부호화 시키는 과정
y=le.fit_transform(play)
print(X)
print(y)
```

[실행결과]

```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
[0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

[문제] 예제 데이터(날씨에 따른 운동여부 데이터)에 가우시안분포를 사용하여 운동하러 갈 확률 구하기

[코드]

```
from sklearn.naive_bayes import GaussianNB

model=GaussianNB()
model.fit(X.reshape(-1,1),y.reshape(-1,1))

predicted=model.predict([[0]]) #Overcast일 때 운동하러 갈 확률
print("Predicted Value:", predicted)
```

[실행결과]

```
Predicted Value: [1]
```

[문제] Perceptron을 Python을 통해 구현하기

[코드]

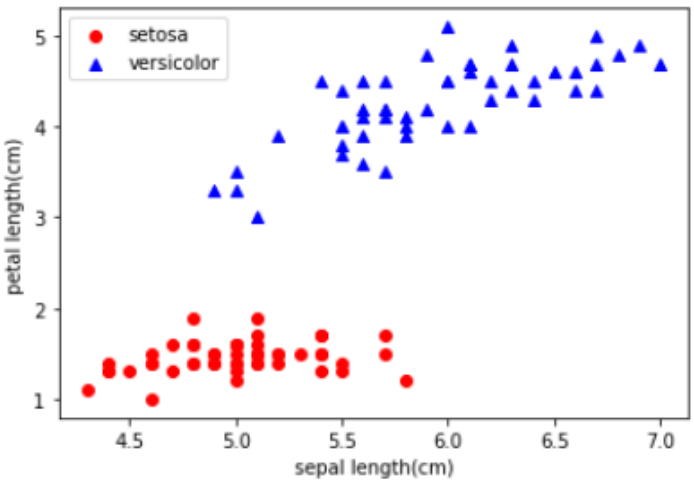
```
import numpy as np


class Perceptron():
    def __init__(self,eta=0.01,n_iter=50,random_state=1):
        self.eta=eta #수렴값을 나타내는 변수, 학습률
        self.n_iter=n_iter # 반복의 횟수
        self.random_state=random_state # 난수의 seed 값 고정시킴


    def predict(self, X): #테스트 함수
        hyperthesis=np.dot(X,self.w_[1:]) + self.w_[0]
        return np.where(hyperthesis>0.0,1,-1) #조건이 참이면1 거짓이면-1


    def fit(self,X,y):
        rgen=np.random.RandomState(self.random_state)
        self.w_=rgen.normal(loc=0.0,scale=0.01,size=X.shape[1]+1)
        self.errors_=[]
        for _ in range(self.n_iter):
            errors=0
            for features,target in zip(X,y):
                update=self.eta * (target - self.predict(features)) #경사하강법
                self.w_[1:]+=update * features
                self.w_[0]+=update
                errors+=int(update !=0.0) #분류가 잘못된 개수 count
            self.errors_.append(errors)
        return self
```

[문제] iris데이터를 가져오고 데이터 시각화하기

[코드]

```
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
iris=load_iris()
X=iris["data"][0:100,(0,2)] #100개의 데이터 중 꽃받침길이,꽃잎길이 2개의 특징사용
y=iris["target"][0:100] # 첫 50개 Iris-setosa, 다음 50개 Iris-versicolor
y=np.where(y==0,-1,1) # Iris-setosa면 -1, 아니면 1로 변경

plt.scatter(X[0:50,0],X[0:50,1],color="r",marker="o",label="setosa")
plt.scatter(X[50:100,0],X[50:100,1],color="b",marker="^",label="versicolor")
plt.xlabel('sepal length(cm)')
plt.ylabel('petal length(cm)')
plt.legend()
plt.show()
```

[실행결과]



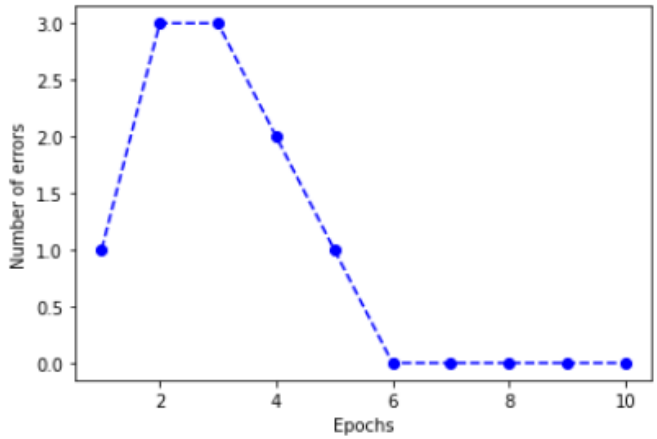
[문제] Perceptron으로 iris데이터를 학습한 결과를 시각화하기

[코드]

```
from sklearn.datasets import load_iris


iris=load_iris()
```

```
X=iris["data"][0:100,(0,2)] #100개의 데이터 중 꽃받침길이,꽃잎길이 2개의 특징사용
y=iris["target"][0:100] # 첫 50개 Iris-setosa, 다음 50개 Iris-versicolor
y=np.where(y==0,-1,1) # Iris-setosa면 -1, 아니면 1로 변경

p=Perceptron(eta=0.1,n_iter=10) #학습률=0.1,반복횟수 = 10회
p.fit(X,y)

plt.plot(range(1, len(p.errors_)+1), p.errors_,"bo--")
plt.xlabel('Epochs') # 반복 횟수
plt.ylabel('Number of errors') #에러 횟수
plt.show()
```

[실행결과]



처음엔 에러가 많이 발생하지만 반복 횟수가 늘어날수록 에러가 줄어드는 것을 확인

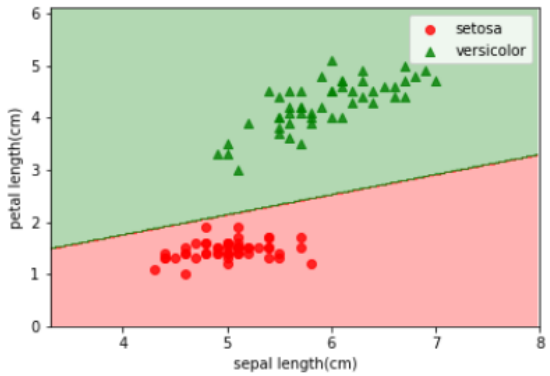
[문제] 결정영역을 그리는 함수 작성하고 시각화하기

[코드]

```
from sklearn.datasets import load_iris
from matplotlib.colors import ListedColormap

def plot_decision_region(X,y,classifier):
    markers=('o','^','x')
    colors=('red','green','blue')
    cmap=ListedColormap(colors[:len(np.unique(y))])
```

```
    x_min,x_max=X[:,0].min()-1,X[:,0].max()+1
    y_min,y_max=X[:,1].min()-1,X[:,1].max()+1
    xx,yy=np.meshgrid(np.arange(x_min,x_max,0.02),np.arange(y_min,y_max,0.02))
    Z=classifier.predict(np.array([xx.ravel(),yy.ravel()]).T)
    Z=Z.reshape(xx.shape)
    plt.contourf(xx,yy,Z,alpha=0.3,cmap=cmap)
    plt.axis([x_min,x_max,y_min,y_max])
    for idx,target in enumerate(np.unique(y)):
        plt.scatter(X[y==target,0],X[y==target,1],
        alpha=0.8,c=colors[idx],marker=markers[idx],
        label=np.where(target==-1,'setosa','versicolor'))

iris=load_iris()
X=iris["data"][0:100,(0,2)] #100개의 데이터 중 꽃받침길이,꽃잎길이 2개의 특징사용
y=iris["target"][0:100] # 첫 50개 Iris-setosa, 다음 50개 Iris-versicolor
y=np.where(y==0,-1,1) # Iris-setosa면 -1, 아니면 1로 변경

plot_decision_region(X,y,classifier=p)
plt.xlabel('sepal length(cm)')
plt.ylabel('petal length(cm)')
plt.legend()
plt.show()
```

[실행결과]



[문제] Python에서 제공하는 Perceptron모델 적용하기

[코드]

```
from sklearn.linear_model import Perceptron
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris=load_iris()
X=iris["data"][0:100,(0,2)] #100개의 데이터 중 꽃받침길이,꽃잎길이 2개의 특징사용
y=iris["target"][0:100] # 첫 50개 Iris-setosa, 다음 50개 Iris-versicolor
y=np.where(y==0,-1,1) # Iris-setosa면 -1, 아니면 1로 변경

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3) # 학습데이터와
테스트데이터를 나눌 때 사용
ppn=Perceptron(eta0=0.1,max_iter=30,random_state=1)
ppn.fit(X_train,y_train)
y_pred=ppn.predict(X_test)
print(f"accuracy: {accuracy_score(y_test,y_pred)}")
```

[실행결과]

```
accuracy: 1.0
```

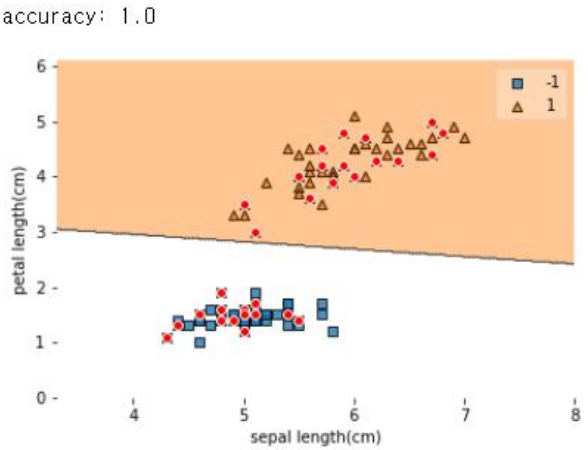
[문제] Binary Logistic Regression을 사용해 예측정확도 확인하기

[코드]

```
from sklearn.linear_model import LogisticRegression
from mlxtend.plotting import plot_decision_regions #통계분석 기능 지원 라이브러리

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) #테스트데이터 30%
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(f"accuracy: {accuracy_score(y_test, y_pred)}")
plot_decision_regions(X, y, clf=model) #모델의 평면을 결정하는 그래프
plt.scatter(X_test[:,0], X_test[:,1],
marker="o", color="r", edgecolor="w")
plt.xlabel('sepal length(cm)')
```

```
plt.ylabel('petal length(cm)')
plt.show()
```

[실행결과]



[문제] Multinomial Logistic Regression을 사용하기 위해 iris dataset 내용 확인과 데이터 시각화하기

[코드]

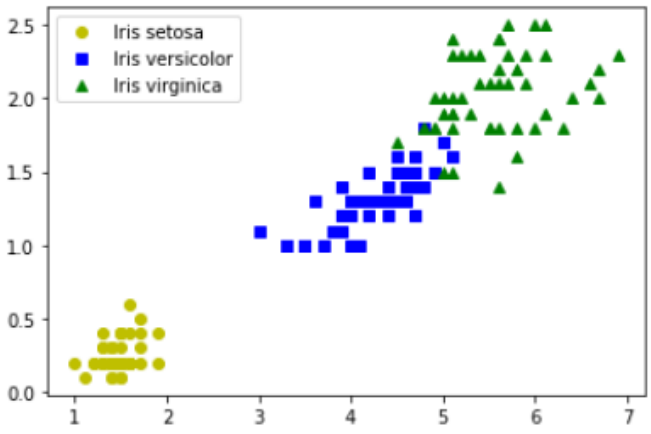
```
import pandas as pd
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
df
```

```
iris = load_iris()
X = iris["data"][:,(2,3)] # 꽃잎의 길이,너비만 사용
y = iris["target"]
plt.plot(X[y==0,0], X[y==0,1], "yo", label="Iris setosa")
plt.plot(X[y==1, 0], X[y==1,1], "bs", label="Iris versicolor")
plt.plot(X[y==2,0], X[y==2, 1], "g^", label="Iris virginica")
plt.legend()
plt.show()
```

[실행결과]

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns



[문제] Multinomial Logistic Regression을 사용하기

[코드]

```
from sklearn import metrics
from sklearn import model_selection
iris = load_iris()
X= iris["data"][:, (2,3)] # 꽃잎의 길이,너비만 사용
y= iris["target"]
X_train,X_test, y_train,y_test = model_selection.train_test_split(X, y, test_size=0.3)
# 테스트데이터 30%만 사용
softmax_reg = LogisticRegression(multi_class="multinomial", C=1, penalty='l2') # 클래스가 N개이므로 multinomial사용, C=정규화 강도의 역수
# L2규제는 l2를 적용
softmax_reg.fit(X_train, y_train) #학습
```

```
y_predict = softmax_reg.predict(X_test) #softmax함수를 사용해 제일 높은 확률구함
score = metrics.r2_score(y_test, y_predict)
print(f"accuracy = {score}")
```

[실행결과]

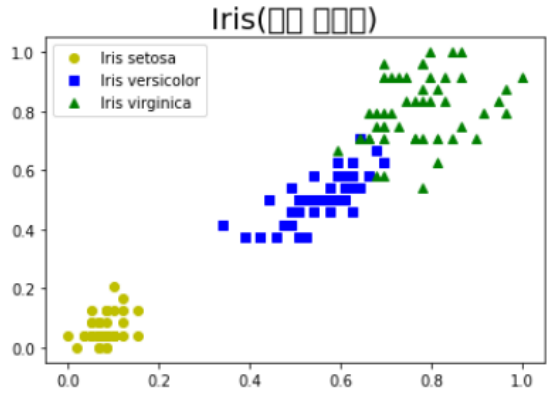
```
accuracy = 0.9129593810444874
```

[문제] MinMaxScaler()를 사용해 정규화하고 데이터 시각화하기

[코드]

```
from sklearn.preprocessing import MinMaxScaler
iris = load_iris()
x_data = iris["data"][:, (2,3)] # 꽃잎의 길이와 너비만 사용
scaler = MinMaxScaler() #minmaxscaler()를 사용해 0~1사이로 정규화
scaler.fit(x_data)
X = scaler.transform(x_data)
y = iris["target"]
plt.title("Iris(붓꽃 데이터)", fontsize=20)
plt.plot(X[y==0, 0], X[y==0, 1], "yo", label="Iris setosa")
plt.plot(X[y==1, 0], X[y==1, 1], "bs", label="Iris versicolor")
plt.plot(X[y==2, 0], X[y==2, 1], "g^", label="Iris virginica")
plt.legend()
plt.show()
```

[실행결과]

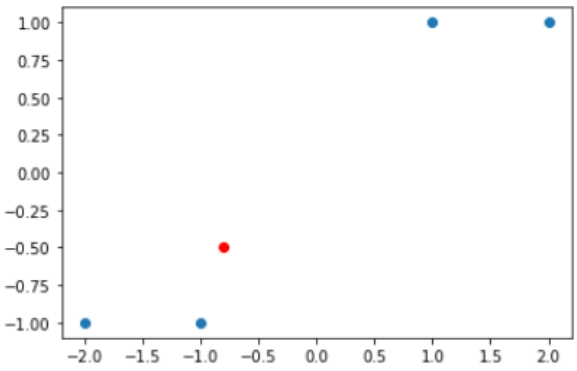


[문제] 경사하강법 알고리즘을 사용해 임의의 2개의 클래스로 나눈 후 첫번째 데이터 클래스 확인하기

[코드]

```
from sklearn.linear_model import SGDClassifier


X=np.array([[-1,-1],[-2,-1],[1,1],[2,1]]) #평면을 짓는 2개의 클래스로 나눔
y=np.array([1,1,2,2])
clf=SGDClassifier(max_iter=1000,tol=1e-3) # 반복횟수 1000회, 정밀도 =1e-3
clf.fit(X,y)


print(clf.predict([[-0.8,-0.5]]))
plt.scatter(X[:,0],X[:,1])
plt.scatter(-0.8,-0.5,marker="o",color="r")
plt.show()
```

[실행결과]



[문제] 경사하강법 알고리즘을 사용해 임의의 2개의 클래스로 나눈 후 첫번째 데이터 클래스 확인하기

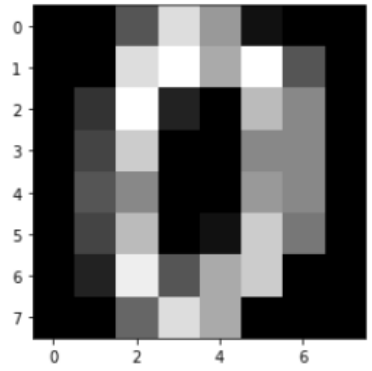
[코드]

```
import numpy as np
from sklearn import datasets # digits datasets 가져오기


digits=datasets.load_digits()
X,y=digits.data,digits.target
```

```
print(X.shape, y.shape)


first_digit=X[0]
first_digit_image=first_digit.reshape(8,8) #첫번째 데이터 사이즈 조정
plt.imshow(first_digit_image,cmap="gray")
plt.show()
```

[실행결과]

```
(1797, 64) (1797,)
```



[문제] 경사하강법 알고리즘을 사용해 임의의 2개의 클래스로 나눈 후 첫번째 데이터 클래스 확인하기

[코드]

```
from sklearn.metrics import accuracy_score
from sklearn.linear_model import SGDClassifier
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3) # 테스트데이터 30%


model=SGDClassifier() #경사하강법 알고리즘 사용
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
print(f"accuracy:{accuracy_score(y_test,y_pred)}")
```

[실행결과]

```
accuracy:0.9462962962962963
```

[문제] MNIST데이터를 가져와 경사하강법 알고리즘 적용하기

[코드]

```
In [19]: from sklearn.datasets import fetch_openml
         mnist=fetch_openml('mnist_784',version=1,return_X_y=True)
         mnist.keys()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\datasets\_openml.py:57: Run
timeWarning: Invalid cache, redownloading file
  warn("Invalid cache, redownloading file", RuntimeWarning)
```

```
---------------------------------------------------------------------------
-----
EOFError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\datasets\_openml.py
in wrapper(*args, **kw)
     52             try:
---> 53                 return f(*args, **kw)
     54             except HTTPError:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\datasets\_openml.py
in _load_arff_response(url, data_home, return_type, encode_nominal, pars
e_arff)
    465                             encode_nominal=encode_nominal)
--> 466         return parse_arff(arff)
    467
```

[실행결과]

Import warning, fetch, version, cache값을 건드려봤지만 실행이 되지 않았습니다. 내용은 숙지했습니다.

8장 예제

[문제] numpy를 사용해 SVM클래스 구현하기

[코드]

```
class SVM:
    def __init__(self,X,y,epochs,lr,C):
        self.X=X;
        self.y=y;
        self.epochs=epochs
        self.lr=lr; #학습률
        self.C=C #정규화 역수
        self.X=np.column_stack((np.ones(len(X)),X))
        self.w=np.ones(len(self.X[0]))

    def distances(self, w, with_lagrange=True):
        distances=self.y * (np.dot(self.X,w))-1
        if with_lagrange:
            distances[distances>0]=0
        return distances

    def get_cost_grads(self,X,w,y):
        distances=self.distances(w)
        L=1/2 * np.dot(w,w) -self.C*np.sum(distances)
        dw=np.zeros(len(w))
        for ind, d in enumerate(distances):
            if d==0:
                di=w
            else:
                di=w-(self.C*y[ind]*X[ind])
            dw+=di
        return L, dw/len(X)

    def fit(self):
        for i in range(self.epochs):
            L, dw=self.get_cost_grads(self.X,self.w,self.y)
            self.w=self.w-self.lr*dw #미분된 가중치만큼 기울기 변경(경사하강법)
```

```
    def predict(self,X):
        X=np.column_stack((np.ones(len(X)),X))
        return np.sign(X @ self.w)
```

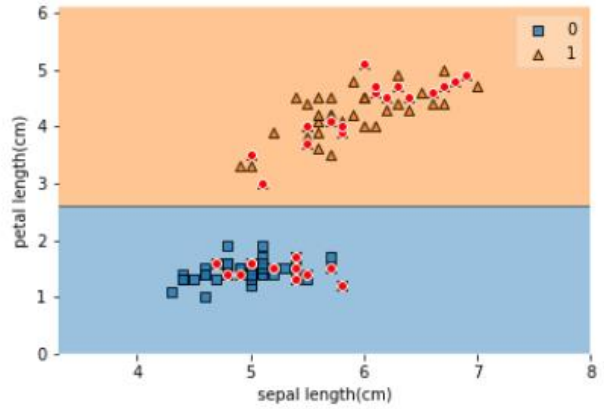
[문제] iris dataset에 SVM적용하기

[코드]

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from mlxtend.plotting import plot_decision_regions
import matplotlib.pyplot as plt
import numpy as np
iris = load_iris()
X=iris["data"][0:100,(0,2)] # 100개의 데이터 중 꽃받침 길이,꽃잎 길이만 사용
y=iris["target"][0:100] # 처음 50개는 Iris-setosa, 다음 50개는 Iris-versicolor
y=np.where(y==0,0,1) # Iris-setosa면 0, 아니면 1로 변경
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
#테스트데이터 30%
clf=SVC(kernel='linear',C=0.5) #선형분류기사용
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print(f"accuracy: {accuracy_score(y_test, y_pred)}")
plot_decision_regions(X,y,clf=clf)
plt.scatter(X_test[:,0],X_test[:,1],marker="o",color="r",edgecolor="w")
plt.xlabel('sepal length(cm)')
plt.ylabel('petal length(cm)')
plt.show()
```

[실행결과]

```
accuracy: 1.0
```



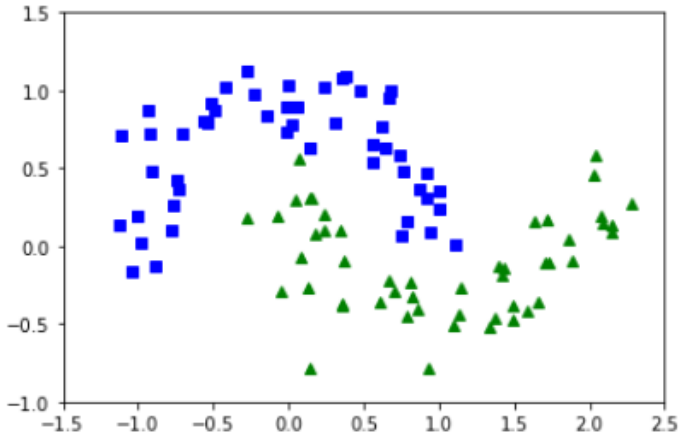
로지스틱 회귀보다 우수한 성능을 가진 모델이라는 것을 확인가능

[문제] 비선형 데이터인 make_moons 가져온 후 시각화하기

[코드]

```
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
X,y=make_moons(n_samples=100, noise=0.15) #선형분류가 쉽지 않은 datasets

plt.plot(X[:,0][y==0],X[:,1][y==0],"bs")
plt.plot(X[:,0][y==1],X[:,1][y==1],"g^")
plt.axis([-1.5,2.5,-1.0,1.5])
plt.show()
```

[실행결과]

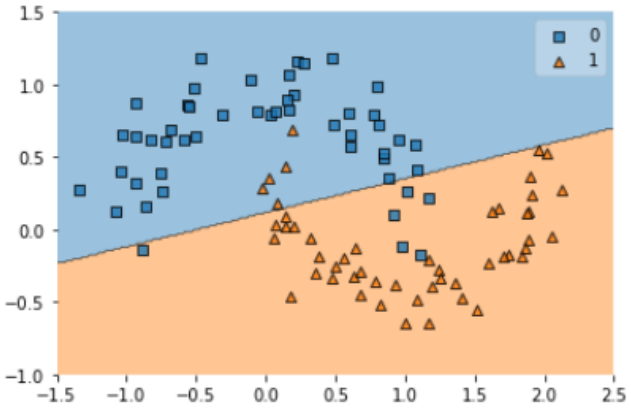


[문제] 비선형 데이터인 선형분류 후 시각화하기

[코드]

```
from sklearn.datasets import make_moons
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from mlxtend.plotting import plot_decision_regions
import matplotlib.pyplot as plt

X,y=make_moons(n_samples=100,noise=0.15)
svm_clf=make_pipeline(StandardScaler(), SVC(kernel='linear',C=10))

svm_clf.fit(X,y)

plot_decision_regions(X,y,clf=svm_clf)
plt.axis([-1.5,2.5,-1.0,1.5])
plt.show()
```

[실행결과]



비선형데이터이기 때문에 선형분류를 아무리 잘해도 이상치가 생기는 것을 확인

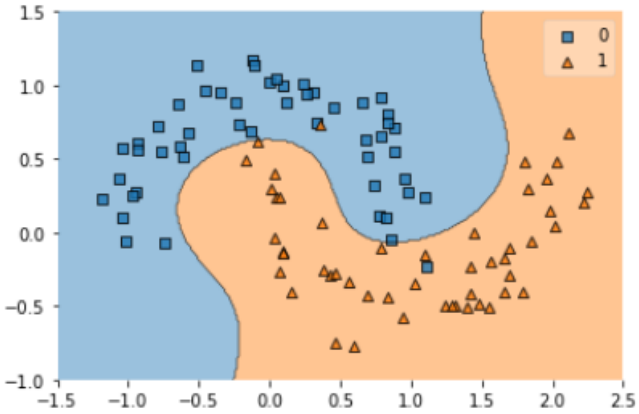
[문제] 비선형 데이터인 비선형분류 후 시각화하기

[코드]

```
from sklearn.preprocessing import PolynomialFeatures

X,y=make_moons(n_samples=100,noise=0.15)
```

```
svm_clf=make_pipeline(PolynomialFeatures(degree=3), #3차원의 다항분포 활용
                      StandardScaler(), SVC(kernel='linear',C=10))

svm_clf.fit(X,y)

plot_decision_regions(X,y,clf=svm_clf)
plt.axis([-1.5,2.5,-1.0,1.5])
plt.show()
```

[실행결과]



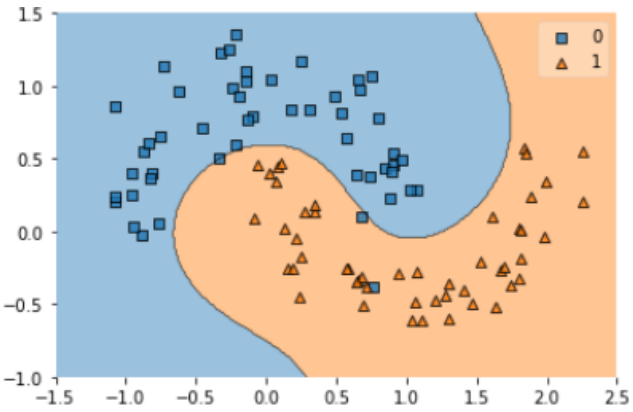
비선형분류기인 polynomial 을 활용해 비선형 데이터 이진 분류하는 것을 확인, 그러나 아주 완벽하게 분류되지는 않음

[문제] 비선형 데이터인 비선형분류 후(kernel 값 적용) 시각화하기

[코드]

```
X,y=make_moons(n_samples=100,noise=0.15)
svm_clf=make_pipeline(StandardScaler(),SVC(kernel='poly',degree=3,coef0=1,C=5))
# 커널을 polynomial로 지정
svm_clf.fit(X,y)

plot_decision_regions(X,y,clf=svm_clf)
plt.axis([-1.5,2.5,-1.0,1.5])
plt.show()
```

[실행결과]



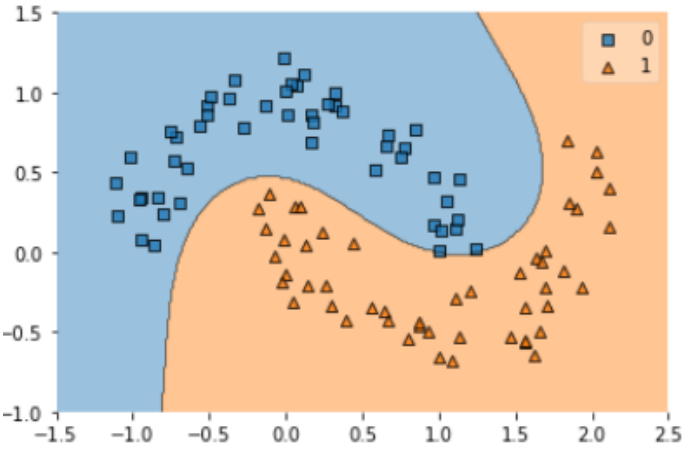
이전 문제에선 kernel을 다루지 않아 완벽하게 이진분류가 되지 않았지만 kernel='poly'를 사용해 이진분류를 조금이나마 완벽하게 다룸

[문제] 비선형 데이터인 비선형분류 후(r 값 적용) 시각화하기

[코드]

```
X,y=make_moons(n_samples=100,noise=0.15)
svm_clf=make_pipeline(StandardScaler(),SVC(kernel='rbf',gamma=0.1,C=100))
# 커널을 rbf로 지정
svm_clf.fit(X,y)

plot_decision_regions(X,y,clf=svm_clf)
plt.axis([-1.5,2.5,-1.0,1.5])
plt.show()
```

[실행결과]



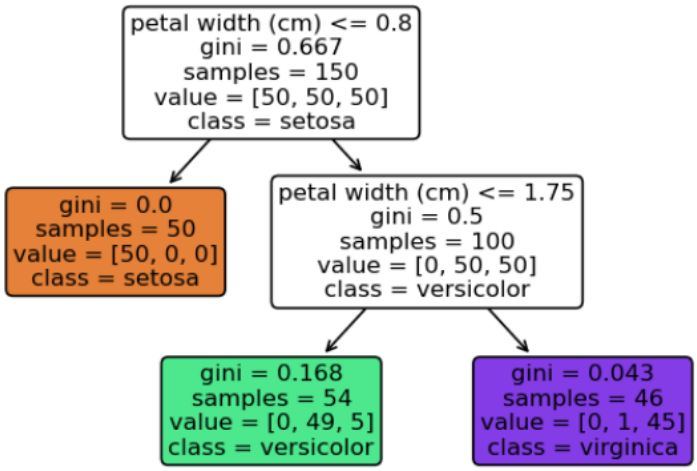
RBF커널을 사용해 완벽하게 이진분류를 완료

[문제] 결정트리를 사용해 iris dataset에 적용하기

[코드]

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

iris = load_iris()
X = iris.data[:, 2:] # 꽃잎의 길이와 너비만 사용
y = iris.target

clf = DecisionTreeClassifier(max_depth=2) #차수는 2까지 허용하여 gini계수가 0이 아니어도 계산하지 않음
clf.fit(X, y)

plt.figure(dpi=120)
plot_tree(clf,feature_names=iris.feature_names[2:],
          class_names=iris.target_names,
          filled=True,rounded=True) #배경색과 모서리 각도조절
plt.show()
```

[실행결과]



9장 예제

[문제] Voting방식을 python으로 적용하기

[코드]

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

models=[
    ('knn',KNeighborsClassifier()),
    ('svc',SVC(probability=True)),
    ('dtc',DecisionTreeClassifier())
]

hard_vote=VotingClassifier(models,voting='hard')
hard_vote.fit(X_train,y_train)

soft_vote=VotingClassifier(models,voting='soft')
soft_vote.fit(X_train,y_train)
```

[실행결과]

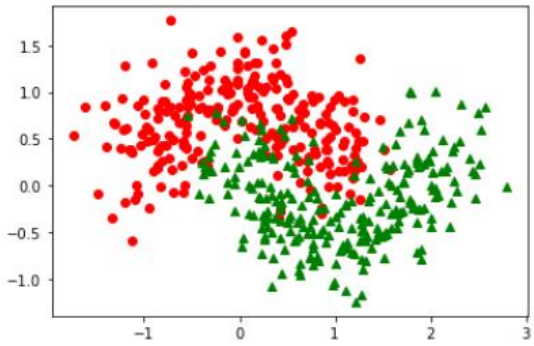
Voting방식에 hard voting과 soft voting방식을 구현하는 방식을 알게 됨

[문제] 비선형데이터셋 (make_moons) 시각화하기

[코드]

```
X,y=make_moons(n_samples=500,noise=0.30,random_state=42) #샘플 개수 500개, 노이즈 30%,
random-state= seed와 같은역할
plt.plot(X[:,0][y==0],X[:,1][y==0],"ro")
plt.plot(X[:,0][y==1],X[:,1][y==1],"g^")
plt.show()
```

[실행결과]



[문제] 비선형데이터셋 (make_moons) hard voting으로 예측하기

[코드]

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score

log_clf=LogisticRegression()
rnd_clf=RandomForestClassifier(n_estimators=10)
svm_clf=SVC()

models=[('lr',log_clf),('rf',rnd_clf),('svm',svm_clf)]
voting_clf=VotingClassifier(estimators=models, voting='hard') # hard voting 사용
voting_clf.fit(X_train, y_train)

for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred=clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test,y_pred))
```

[실행결과]

```
LogisticRegression 0.864
RandomForestClassifier 0.888
SVC 0.896
VotingClassifier 0.912
```

보팅방식이 다른 모델들보다 좀 더 높은 정확도를 가지는 것을 확인

[문제] 비선형데이터셋 (make_moons) soft voting으로 예측하기

[코드]

```
log_clf=LogisticRegression()
rnd_clf=RandomForestClassifier(n_estimators=10)
svm_clf=SVC(probability=True)

models=[('lr',log_clf),('rf',rnd_clf),('svm',svm_clf)]
voting_clf=VotingClassifier(estimators=models, voting='soft') # soft voting 사용
voting_clf.fit(X_train, y_train)

for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred=clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test,y_pred))
```

[실행결과]

```
LogisticRegression 0.864
RandomForestClassifier 0.872
SVC 0.896
VotingClassifier 0.912
```

다른 모델들보다 조금 높은 정확도를 가지는 것을 확인

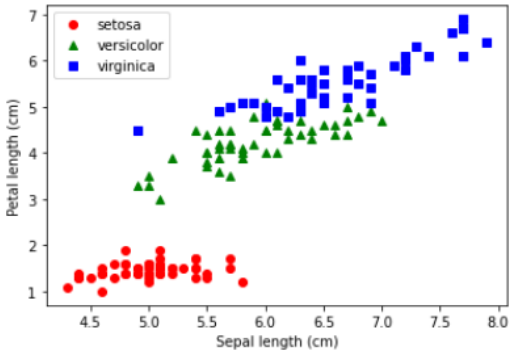
[문제] iris dataset 가져오고 시각화하기

[코드]

```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data[:, [0, 2]] # 꽃받침의 길이와 꽃잎의 길이
y = iris.target
plt.plot(X[:,0][y==0], X[:,1][y==0], "ro", label='setosa')
plt.plot(X[:,0][y==1], X[:,1][y==1], "g^", label='versicolor')
```

```
plt.plot(X[:,0][y==2], X[:,1][y==2], "bs", label='virginica')
plt.xlabel('Sepal length (cm)')
plt.ylabel('Petal length (cm)')
plt.legend()
plt.show()
```

[실행결과]



[문제] iris dataset에 soft voting 적용하고 정확률 확인

[코드]

```
from sklearn.metrics import accuracy_score

X_train,X_test,y_train,y_test = train_test_split(X, y)
dt_clf=DecisionTreeClassifier(max_depth=4)
knn_clf=KNeighborsClassifier(n_neighbors=7)
svm_clf=SVC(gamma=0.1, probability=True)

models=[('dt',dt_clf),('knn',knn_clf),('svm',svm_clf)]
voting_clf=VotingClassifier(models,voting='soft',weights=[2, 1, 2]) # soft voting 방식
voting_clf.fit(X_train,y_train)
for clf in (dt_clf,knn_clf,svm_clf,voting_clf):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    print(f"{clf.__class__.__name__}: {accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
DecisionTreeClassifier: 0.9474
KNeighborsClassifier: 0.9211
SVC: 0.9211
VotingClassifier: 0.9474
```

다른 모델들과 유사한 정확도를 보이는 것으로 미루어 볼 때 Voting방식은 단순한 구조로 이루어져 있어 약학습기를 합친다고 무조건 좋은 모델이 아닌 것을 확인

[문제] iris dataset에 Bagging 적용하고 정확률 확인

[코드]

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

dt_clf=DecisionTreeClassifier() #결정트리 사용
bag_clf=BaggingClassifier(dt_clf, n_estimators=500) #기본 분류기 결정트리,분류기 개수 500개
bag_clf.fit(X_train,y_train)

for clf in (dt_clf,bag_clf):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    print(f"{clf.__class__.__name__}:{accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
DecisionTreeClassifier:0.9211
BaggingClassifier:0.9211
```

[문제] make_moons 비선형 데이터에 Bagging 적용하고 정확률 확인

[코드]

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X,y=make_moons(n_samples=500,noise=0.30,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X,y)
dt_clf=DecisionTreeClassifier()
```

```
bag_clf=BaggingClassifier(dt_clf,n_estimators=500) #분류기 개수 500개, 결정트리 사용
bag_clf.fit(X_train, y_train)
for clf in (dt_clf,bag_clf):
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_test)
    print(f"{clf.__class__.__name__}:{accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
DecisionTreeClassifier:0.8880
BaggingClassifier:0.8960
```

결정트리보다는 조금 높지만 비선형데이터에서도 비슷한 정확률을 확인

[문제] Bagging에 대표적인 방식인 RandomForest방식을 비선형데이터에 적용시키기

[코드]

```
from sklearn.ensemble import RandomForestClassifier

X,y=make_moons(n_samples=500,noise=0.30,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X, y)
clf=RandomForestClassifier(n_estimators=500) #랜덤 포레스트는 기본 분류기 지정 x, 이미 결정트리 사용
clf.fit(X_train,y_train)
y_pred =clf.predict(X_test)
print(f"{clf.__class__.__name__}: {accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
RandomForestClassifier: 0.9040
```

이전 문제에서 다룬 선형데이터와 비선형데이터의 Bagging방식을 적용한 것과 정확률이 조금 높으나 비슷한 것을 확인

[문제] Boosting방식의 모듈 AdaBoost를 비선형데이터(make_moons)에 적용해 정확률확인하기

[코드]

```
from sklearn.ensemble import AdaBoostClassifier


clf = AdaBoostClassifier(n_estimators=200, learning_rate=0.1) #약분류기 결정트리사용,
약분류기 200개,학습률=1
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(f"{clf.__class__.__name__}:{accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
AdaBoostClassifier:0.9040
```

[문제] Boosting방식의 모듈 GBM(Gradient Boosting Machine)를 비선형데이터(make_moons)에 적용해 정확률 확인하기

[코드]

```
from sklearn.ensemble import GradientBoostingClassifier


X,y =make_moons(n_samples=500,noise=0.30,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X,y)
clf=GradientBoostingClassifier(n_estimators=200,learning_rate=0.1)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print(f"{clf.__class__.__name__}:{accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
GradientBoostingClassifier:0.8560
```

[문제] Boosting방식의 모듈 XGBoost를 비선형데이터(make_moons)에 적용해 정확률 확인하기

[코드]

```
import warnings
warnings.filterwarnings(action='ignore')
from xgboost import XGBClassifier
```

```
X,y=make_moons(n_samples=500,noise=0.30,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X,y)
clf=XGBClassifier(n_estimators=200,learning_rate=0.1)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print(f"{clf.__class__.__name__}: {accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
XGBClassifier: 0.9440
```

XGBoost 자체가 GBM의 과적합 문제를 규제로 해결했기 때문에 Adaboost와 GBM의 정확률보다 더 높은 정확률을 보이는 것을 확인

[문제] Boosting방식의 모듈 LightGBM을 비선형데이터(make_moons)에 적용해 정확률 확인하기

[코드]

```
from lightgbm import LGBMClassifier

X,y=make_moons(n_samples=500,noise=0.30,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X,y)
clf=LGBMClassifier(n_estimators=200,learning_rate=0.1)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print(f"{clf.__class__.__name__}:{accuracy_score(y_test,y_pred):.4f}")
```

[실행결과]

```
LGBMClassifier:0.9040
```

LightGBM 방식은 XGBoost의 느린 학습시간을 보완하기 위해 나온 모델이기 때문에 이렇게 작은 데이터셋에서는 다른 정확도를 보이진 않음

프로젝트 후기

7장의 분류문제 Perceptron, Logistic Regreesion, Multinomial, 8장의 Support Vetcor Machine, Decision Tree, 9장의 Ensemble 에 대한 대략적인 내용들을 배우고 예제 프로젝트를 하게 되었는데 단순했지만 흥미있게 진행하였습니다.

데이터들의 정확도가 중요한 것을 예전 강의에서 배워 알고는 있었지만 이번 프로젝트를 통해 데이터 하나하나 정확도를 높이는 것이 얼마나 중요하고 어려운 것인지를 깨닫게 되었습니다.

이번 프로젝트를 진행하면서 강의시간에 시간이 부족해 다루지 못했던 각종 모델,함수들의 옵션들을 살펴보면서 적용시켜보기도 해보았던 유익한 시간이기도 했으며 기본적인 Iris dataset과 Make_Moons같은 다루기 쉬운 선형,비선형 데이터밖에 못다뤄봐서 아쉬운 시간이기도 했습니다.

하나 배워가는 프로젝트 시간이었습니다. 감사합니다.