머신러닝

20135174 정태이

-전체 코드

import os

import urllib.request

from sklearn.datasets import fetch\_mldata

#데이터가져오기

def download\_mnist\_data(url, path):

path = os.path.join(path, "mldata")

if not os.path.isdir(path):

os.makedirs(path)

filename = os.path.basename(url)

filepath = os.path.join(path,filename)

if not os.path.isfile(filepath):

urllib.request.urlretrieve(url,filepath)

if \_\_name\_\_ == "\_\_main\_\_":

url = "http://210.115.230.193/data/mnist-original.mat" #url

data\_home = "datasets"

download\_mnist\_data(url,data\_home)

mnist = fetch\_mldata('MNIST original', data\_home=data\_home)

print(mnist)

X,y = mnist["data"],mnist["target"]

print(X.shape, y.shape) #이미지는 7만개, 각 이짐에는 784개의 특성 존재

import matplotlib

import matplotlib.pyplot as plt

# 숫자 그림과 실제 레이블 확인

some\_digit = X[36000]

some\_digit\_image = some\_digit.reshape(28,28)

plt.imshow(some\_digit\_image, cmap=matplotlib.cm.binary, interpolation="nearest")

plt.axis("off")

plt.show()

print(y[36000])

#테스트 세트를 만들고 따로 떼어놓기

X\_train, X\_test, y\_train, y\_test = X[:60000], X[60000:], y[:60000],y[60000:]

import numpy as np

#훈련 세트를 섞어서 모든 교차 검증 폴드가 비슷해지도록 만듦

shuffle\_index = np.random.permutation(60000)

X\_train, y\_train = X\_train[shuffle\_index], y\_train[shuffle\_index]

#5만 True, 나머지 숫자는 False

y\_train\_5 = (y\_train == 5)

y\_test\_5 = (y\_test == 5)

from sklearn.linear\_model import SGDClassifier

#SGDClassifier 모델을 만들고 전체 훈련 세트를 사용해서 훈련시킴

sgd\_clf = SGDClassifier(max\_iter=5, random\_state=42)

sgd\_clf.fit(X\_train, y\_train\_5)

#5의 이미지 감지

sgd\_clf.predict([X[36000]])

#cross\_val\_score()함수와 거의 같은 작업을 수행하는 코드

from sklearn.model\_selection import StratifiedKFold

from sklearn.base import clone

skfolds = StratifiedKFold(n\_splits=3, random\_state=42) #계층적 샘플링 수행

for train\_index, test\_index in skfolds.split(X\_train, y\_train\_5):

clone\_clf = clone(sgd\_clf)

X\_train\_folds = X\_train[train\_index]

y\_train\_folds = y\_train\_5[train\_index]

X\_test\_fold = X\_train[test\_index]

y\_test\_fold = y\_train\_5[test\_index]

clone\_clf.fit(X\_train\_folds, y\_train\_folds)

y\_pred = clone\_clf.predict(X\_test\_fold)

n\_correct = sum(y\_pred == y\_test\_fold)

print(n\_correct / len(y\_pred))

from sklearn.model\_selection import cross\_val\_score

#SGDClassifier 모델 평가

cross\_val\_score(sgd\_clf, X\_train,y\_train\_5, cv=3, scoring="accuracy")

from sklearn.base import BaseEstimator

#더미 분류기 만들어서 비교

class Never5Classifier(BaseEstimator):

def fit(self, X, y=None):

pass

def predict(self, X):

return np.zeros((len(X),1),dtype=bool)

never\_5\_clf = Never5Classifier()

print(cross\_val\_score(never\_5\_clf, X\_train, y\_train\_5,cv=3,scoring="accuracy"))

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import confusion\_matrix

#함수 사용해서 오차 행렬 만들기

y\_train\_pred = cross\_val\_predict(sgd\_clf, X\_train, y\_train\_5, cv=3)

y\_train\_pred\_no\_cv = sgd\_clf.predict(X\_train)

print(confusion\_matrix(y\_train\_5, y\_train\_pred))

print(confusion\_matrix(y\_train\_5, y\_train\_pred\_no\_cv))

from sklearn.metrics import precision\_score, recall\_score

#정확도 확인

print(precision\_score(y\_train\_5, y\_train\_pred))

print(recall\_score(y\_train\_5,y\_train\_pred))

from sklearn.metrics import f1\_score

print(f1\_score(y\_train\_5, y\_train\_pred)) #F1점수 계산

#각 샘플의 점수를 얻음

y\_scores = sgd\_clf.decision\_function([some\_digit])

print(y\_scores)

#임계값이 0

threshold = 0

y\_some\_digit\_pred = (y\_scores > threshold)

print(y\_some\_digit\_pred)

threshlod = 200000 #임계값= 200000

y\_some\_digit\_pred = (y\_scores > threshold)

print(y\_some\_digit\_pred)

from sklearn.metrics import precision\_recall\_curve

#결정 점수를 반환

y\_scores = cross\_val\_predict(sgd\_clf, X\_train,y\_train\_5, cv = 3, method="decision\_function")

precisions, recalls, thresholds = precision\_recall\_curve(y\_train\_5, y\_scores)

#임계값의 함수로 정밀도와 재현율을 그림

def plot\_precision\_recall\_vs\_threshold(precisions,recalls,thresholds):

plt.plot(thresholds, precisions[:-1],"b--", label="precision")

plt.plot(thresholds, recalls[:-1],"g--", label="recall")

plt.xlabel("threshold")

plt.legend(loc="center left")

plt.xlim([-600000, 600000])

plt.ylim([0,1])

plot\_precision\_recall\_vs\_threshold(precisions, recalls, thresholds)

plt.show()

y\_train\_pred\_90 = (y\_scores > 70000)

print(precision\_score(y\_train\_5, y\_train\_pred\_90))

print(recall\_score(y\_train\_5, y\_train\_pred\_90))

from sklearn.metrics import roc\_curve

#TPR에 대한 FPR 곡선 그리기

def plot\_roc\_curve(fpr, tpr, label=None):

plt.plot(fpr, tpr, linewidth=2, label=label)

plt.plot([0,1],[0,1],'k--')

plt.axis([0,1,0,1])

plt.xlabel('FPR')

plt.ylabel('TPR')

fpr, tpr, thresholds = roc\_curve(y\_train\_5, y\_scores)

plot\_roc\_curve(fpr,tpr)

plt.show()

from sklearn.metrics import roc\_auc\_score

print(roc\_auc\_score(y\_train\_5, y\_scores)) # ROC의 AUC를 계산하는 함수

from sklearn.ensemble import RandomForestClassifier

# 훈련 세트의 샘플에 대한 점수 얻기

forest\_clf = RandomForestClassifier(random\_state = 42)

y\_probas\_forest = cross\_val\_predict(forest\_clf, X\_train, y\_train\_5, cv=3, method="predict\_proba")

y\_scores\_forest = y\_probas\_forest[:,1] #양성 클래스에 대한 확률을 점수로 사용

fpr\_forest, tpr\_forest, thresholds\_forest = roc\_curve(y\_train\_5, y\_scores\_forest)

# ROC곡선을 함께 그려서 비교

plt.plot(fpr, tpr, "b:",label="SGD")

plot\_roc\_curve(fpr\_forest,tpr\_forest,"RandimForest")

plt.legend(loc="lower right")

plt.show()

roc\_auc\_score(y\_train\_5, y\_scores\_forest)

sgd\_clf.fit(X\_train, y\_train) #훈련시키기

sgd\_clf.predict([some\_digit])

some\_digit\_scores = sgd\_clf.decision\_function([some\_digit])

print(some\_digit\_scores)

print(np.argmax(some\_digit\_scores))

print(sgd\_clf.classes\_)

print(sgd\_clf.classes\_[5])

from sklearn.multiclass import OneVsOneClassifier

# OvO 전략을 사용하는 다중 분류기

ovo\_clf = OneVsOneClassifier(SGDClassifier(max\_iter=5, random\_state=42))

ovo\_clf.fit(X\_train, y\_train)

print(ovo\_clf.predict([some\_digit]))

print(len(ovo\_clf.estimators\_))

forest\_clf.fit(X\_train, y\_train) # 훈련시키기

print(forest\_clf.predict([some\_digit]))

print(forest\_clf.predict\_proba([some\_digit]))

#정확도 평가해보기

print(cross\_val\_score(sgd\_clf, X\_train, y\_train, cv=3, scoring="accuracy"))

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train.astype(np.float64))

print(cross\_val\_score(sgd\_clf, X\_train\_scaled, y\_train, cv=3, scoring = 'accuracy'))

#예측을 만듦

y\_train\_pred = cross\_val\_predict(sgd\_clf, X\_train\_scaled, y\_train, cv=3)

conf\_mx = confusion\_matrix(y\_train, y\_train\_pred)

print(conf\_mx)

plt.matshow(conf\_mx, cmap=plt.cm.gray)

plt.show()

row\_sums = conf\_mx.sum(axis=1, keepdims=True)

norm\_conf\_mx = conf\_mx / row\_sums

np.fill\_diagonal(norm\_conf\_mx, 0)

plt.matshow(norm\_conf\_mx, cmap=plt.cm.gray)

plt.show()

def plot\_digits(instances, images\_per\_row=10, \*\*options):

size = 28

images\_per\_row = min(len(instances), images\_per\_row)

images = [instances.reshape(size, size) for instances in instances]

n\_rows = (len(instances) - 1) // images\_per\_row + 1

row\_images = []

n\_empty = n\_rows \* images\_per\_row - len(instances)

images.append(np.zeros((size, size \* n\_empty)))

for row in range(n\_rows):

rimages = images[row \* images\_per\_row : (row + 1) \* images\_per\_row]

row\_images.append(np.concatenate(rimages, axis=1))

image = np.concatenate(row\_images, axis=0)

plt.imshow(image, cmap = matplotlib.cm.binary, \*\*options)

plt.axis("off")

cl\_a, cl\_b = 3,5

x\_aa = X\_train[(y\_train == cl\_a) & (y\_train\_pred == cl\_a)]

x\_ab = X\_train[(y\_train == cl\_a) & (y\_train\_pred == cl\_b)]

x\_ba = X\_train[(y\_train == cl\_b) & (y\_train\_pred == cl\_a)]

x\_bb = X\_train[(y\_train == cl\_b) & (y\_train\_pred == cl\_b)]

plt.figure(figsize=(8,8))

plt.subplot(221); plot\_digits(x\_aa[:25], images\_per\_row=5)

plt.subplot(222); plot\_digits(x\_ab[:25], images\_per\_row=5)

plt.subplot(223); plot\_digits(x\_ba[:25], images\_per\_row=5)

plt.subplot(224); plot\_digits(x\_bb[:25], images\_per\_row=5)

plt.show()

#각 숫자 이미지에 두 개의 타깃 레이블이 닮긴 배열을 만든다

from sklearn.neighbors import KNeighborsClassifier

y\_train\_large = (y\_train >= 7)

y\_train\_odd = (y\_train % 2 == 1)

y\_multilabel = np.c\_[y\_train\_large, y\_train\_odd]

knn\_clf = KNeighborsClassifier()

knn\_clf.fit(X\_train, y\_multilabel)

print(knn\_clf.predict([some\_digit])) #예측만들기

y\_train\_knn\_pred = cross\_val\_predict(knn\_clf, X\_train, y\_multilabel, cv=3, n\_jobs=-1)

print(f1\_score(y\_multilabel, y\_train\_knn\_pred, average="macro"))

#macro를 씀으로써 가중치를 두지 않음

# 훈련 세트와 테스트 세트에 randint함수를 사용해서 픽셀 강도에 노이즈 추가

noise = np.random.randint(0, 100, (len(X\_train), 784))

X\_train\_mod = X\_train + noise

noise = np.random.randint(0, 100, (len(X\_test), 784))

X\_test\_mod = X\_test + noise

y\_train\_mod = X\_train

y\_test\_mod = X\_test

def plot\_digit(data):

image = data.reshape(28,28)

plt.imshow(image, cmap=matplotlib.cm.binary, interpolation="nearest")

plt.axis("off")

some\_index = 5500

plt.subplot(121); plot\_digit(X\_test\_mod[some\_index])

plt.subplot(122); plot\_digit(y\_test\_mod[some\_index])

plt.show()

#깨끗하게 만들기

knn\_clf.fit(X\_train\_mod, y\_train\_mod)

clean\_digit = knn\_clf.predict([X\_test\_mod[some\_index]])

plot\_digit(clean\_digit)

plot\_digit(clean\_digit)

plt.show()

-결과화면









































