داده کاوی

تمرین سری سوم

تمرین عملی

سوال ۱)

اول از همه کتاب خانه های مورد نیاز را import میکنیم.

```
import pandas as pd
from sklearn.svm import LinearSVC
from sklearn import svm
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
```

حال داده ها را میخوانیم و train و test را به نسبت ۴ به ۱(۰/۸ داده ها اموزش و ۰/۲ داده ها برای تست)

```
#reading data
data1 = pd.read_csv('data1.csv')
data2 = pd.read_csv('data2.csv')
#seperating train and test (partition is 90 to 10)
#data1
data1_train,data1_test = train_test_split(data1,train_size = 0.8,test_size=0.2)
#data2
data2_train,data2_test = train_test_split(data2,train_size = 0.8,test_size=0.2)
```

حال دسته بند های خطی را میسازیم برای داده های یک و دو و آن هارا آموزش میدهیم.

```
#creating our linear classifier
#data1
linear_classifier1 = LinearSVC()
linear_classifier1.fit(data1_train[['X','Y']],data1_train['Class'])
#data2
linear_classifier2 = LinearSVC()
linear_classifier2.fit(data2_train[['X','Y']],data2_train['Class'])
```

تابع زیر نیز برای محاسبه دقت بر حسب داده و مدل میباشد

```
def accuracy(testData,classifier):
    acc = 0
    for i in range(len(testData)):
        predicted_lable = classifier.predict([testData[['X','Y']].iloc[i]])[0]
        if predicted_lable == testData['Class'].iloc[i]:
            acc += 1
    return acc / len(testData)
```

دقت دسته بند برای داده ۱ و ۲ را در حالت خطی مشاهده میکنیم

```
#seeing the accuracy of linear data
print('data1 linear accuracy: ',accuracy(data1_test, linear_classifier1))
print('data2 linear accuracy: ',accuracy(data2_test, linear_classifier2))
```

```
data1 linear accuracy:  1.0
data2 linear accuracy:  0.4
```

حال برای داده دوم یک دسته بند غیر خطی تعریف میکنیم و دقت آن را محاسبه میکنیم

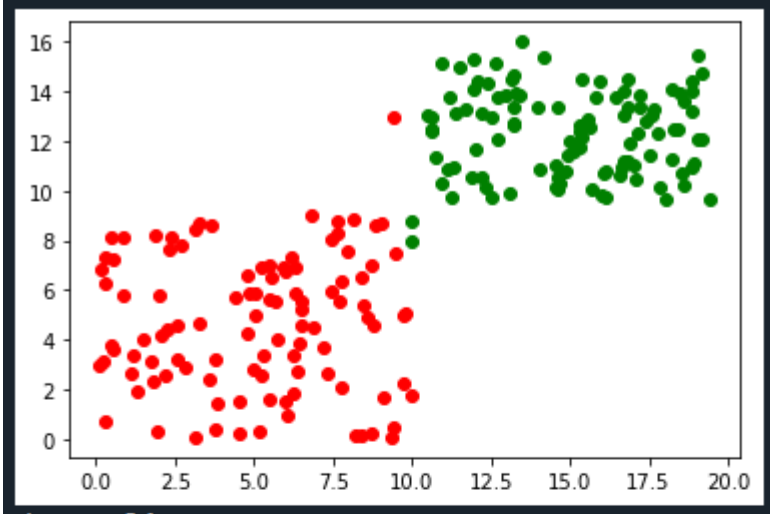
```
#non linear svc for data2
nonlinear_classifier2 = svm.NuSVC()
nonlinear_classifier2.fit(data2_train[['X','Y']],data2_train['Class'])
print('data2 non linear accuracy:',accuracy(data2_test,nonlinear_classifier2))
```

```
data2 non linear accuracy: 0.9833333333333333
```

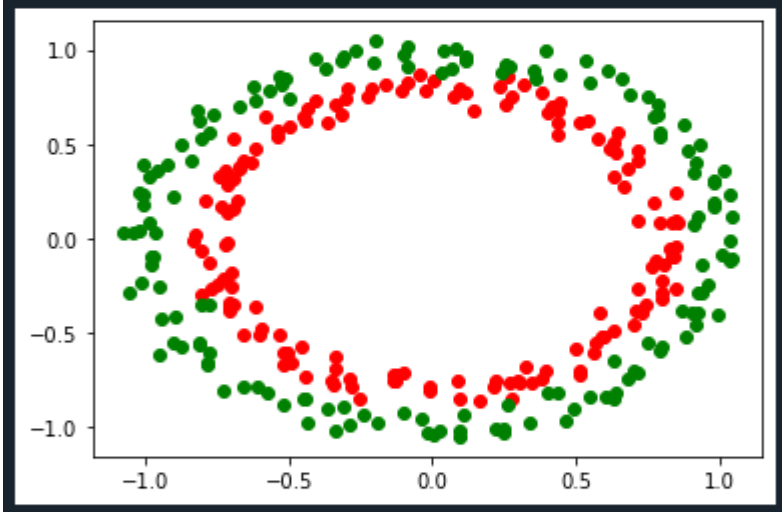
کد زیر نیز برای plot کردن داده هاست

```
#plotting for data1 linear form
print('data1 linear')
plt.scatter(data1['X'][(data1['Class'] == 1)],data1['Y'][(data1['Class'] == 1)],color = 'red')
plt.scatter(data1['X'][(data1['Class'] == 0)],data1['Y'][(data1['Class'] == 0)],color = 'green')
plt.show()
#plotting for data2 linear form
print('data2 linear')
plt.scatter(data2['X'][(data2['Class'] == 1)],data2['Y'][(data2['Class'] == 1)],color = 'red')
plt.scatter(data2['X'][(data2['Class'] == 0)],data2['Y'][(data2['Class'] == 0)],color = 'green')
plt.show()
#plotting for data2 non linear form
print('data2 non linear')
plt.scatter(data2['X'][(data2['Class'] == 1)],data2['Y'][(data2['Class'] == 1)],color = 'red')
plt.scatter(data2['X'][(data2['Class'] == 0)],data2['Y'][(data2['Class'] == 0)],color = 'green')
plt.show()
```

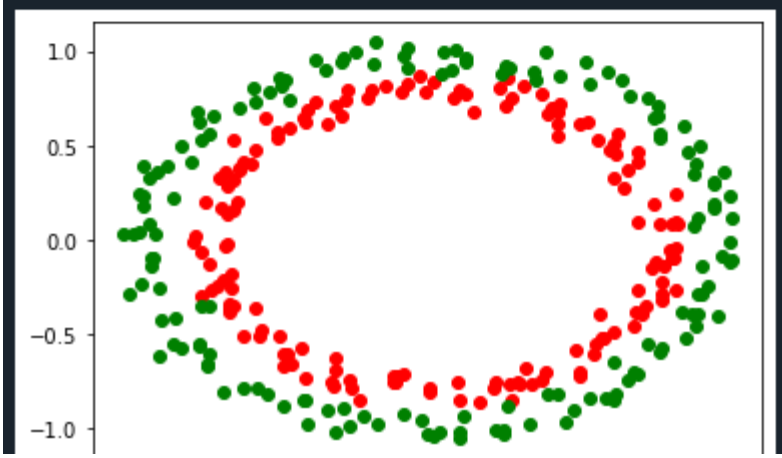
data1 linear



data2 linear



data2 non linear



سوال ۲)

در ابتدا کتاب خانه های مورد نیاز را import میکنیم

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
```

داده را میخوانیم و اسم ویژگی های آن را در آرایه نگهداری میکنیم و بعد از آن پیش پردازش داده را انجام میدهیم و داده های آموزش و تست را جدا میکنیم

```
data = pd.read_csv('data3.csv')
features = ['sepal_length','sepal_width','petal_length','petal_width']
#preprocessing data
lb = LabelEncoder()
for i in data:
    data[i] = lb.fit_transform(data[i])
#splitting train and test data
dataTrain,dataTest = train_test_split(data,train_size = 0.8,test_size = 0.2)
```

حال جنگل ها را با تعداد درخت متفاوت میسازیم

```
#creating our random forest classifiers
#with 10 trees
randomForest1 = RandomForestClassifier(n_estimators = 10)
randomForest1.fit(dataTrain[features],dataTrain['species'])
#with 30 tress
randomForest2 = RandomForestClassifier(n_estimators = 30)
randomForest2.fit(dataTrain[features],dataTrain['species'])
#with 50 trees
randomForest3 = RandomForestClassifier(n_estimators = 50)
randomForest3.fit(dataTrain[features],dataTrain['species'])
#with 70 trees
randomForest4 = RandomForestClassifier(n_estimators = 70)
randomForest4.fit(dataTrain[features],dataTrain['species'])
#with 90 trees
randomForest5 = RandomForestClassifier(n_estimators = 90)
randomForest5.fit(dataTrain[features],dataTrain['species'])
#with 120 trees
randomForest6 = RandomForestClassifier(n_estimators = 120)
randomForest6.fit(dataTrain[features],dataTrain['species'])
#function below if for calculation accuracy
```

تابع زیر نیز برای محاسبه دقت میباشد

```
def accuracy(test,model):
    acc = 0
    for i in range(len(test)):
        predictedLable = model.predict(test[features])[0]
        if predictedLable == test['species'].iloc[i]:
            acc += 1
    return acc/len(test)
```

و دقت را برای تمامی جنگل ها محاسبه میکنیم و مشاهده میکنیم به ازای هر تعداد درخت دقت یکسانی میدهد.

```
print('10 trees accuracy:',accuracy(dataTest, randomForest1))
print('30 trees accuracy:',accuracy(dataTest, randomForest2))
print('50 trees accuracy:',accuracy(dataTest, randomForest3))
print('70 trees accuracy:',accuracy(dataTest, randomForest4))
print('90 trees accuracy:',accuracy(dataTest, randomForest5))
print('120 trees accuracy:',accuracy(dataTest, randomForest6))
```

```
10 trees accuracy: 0.5
30 trees accuracy: 0.5
50 trees accuracy: 0.5
70 trees accuracy: 0.5
90 trees accuracy: 0.5
120 trees accuracy: 0.5
```

سوال ۳)

اول کتاب خانه های مورد نظر را import میکنیم

```
import pandas as pd
import random
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
import math
```

داده را میخوانیم و پیش پردازش های لازم را انجام میدهیم(برای مدیریت داده های missing سطر هایی که در آن missing value میباشد را حذف میکنیم.)

```
#preprocessing our data
#elimination of instances with missing value
data = data.dropna()
data = data.reset_index()
```

```
#covertion of pclass column
for i in range(len(data)):
    if data['Pclass'][i] == 1:
        data['Pclass'][i] = 'one'
    elif data['Pclass'][i] == 2:
        data['Pclass'][i] = 'two'
    else:
        data['Pclass'][i] = 'three'
#convertion of target class
for i in range(len(data)):
    if data['Target_class'][i] == 1:
        data['Target_class'][i] = 'one'
    else:
        data['Target_class'][i] = 'zero'
#covertion of Age class
for i in range(len(data)):
    if data['Age'][i] < 10:
        data['Age'][i] = 'A'
    elif 10 <= data['Age'][i] < 20:
        data['Age'][i] = 'B'
    elif 20 <= data['Age'][i] < 30:
        data['Age'][i] = 'C'
    elif 30 <= data['Age'][i] < 40:
        data['Age'][i] = 'D'
    elif 40 <= data['Age'][i] < 50:
        data['Age'][i] = 'E'
    else:
        data['Age'][i] = 'F'
lb = LabelEncoder()
for i in data:
    data[i] = lb.fit_transform(data[i])
```

داده هارا به ترتیب در آرایه ذخیره میکنیم.

```
#saving our data in array
dataArray = []
for i in range(len(data)):
    dataArray.append(data.loc[i])
```

وزن های هر داده و وضعیت آنکه آیا درست پیش بینی شده اند یا خیر را در یک آرایه قرار میدهیم

```
#initializing weights
weightsList = [1/len(data)]*len(data)
#list below is for correct predic staus
statusList = [False]*len(data)
#algorithm below is for implementation of ada boost
#using naive bayes as classifier
gnb = GaussianNB()
```

برای دسته بندی داده ها را در هر بخش نیز از بیض ساده استفاده میکنیم

```
gnb = GaussianNB()
```

الگوریتم زیر نیز پیاده سازی adaboost میباشد که در هر مرحله از آن confusion matrix را نمایش میدهیم

```
k = 10
#numbers of samples boost
samples_number = math.floor(len(data)*0.8)
for i in range(k):
    dataSamples = random.choices(dataArray,weights = weightsList,k = samples_number)
    dataFrameSamples = pd.DataFrame(dataSamples)
    #learning based on selected samples
    gnb.fit(dataFrameSamples[['Pclass','Sex','Age']],dataFrameSamples['Target_class'])
    error = 0
    for i in range(len(data)):
        predictedLable = gnb.predict([data[['Pclass','Sex','Age']].iloc[i]])[0]
        if predictedLable == data['Target_class'][i]:
            statusList[i] = True
        else:
            statusList[i] = False
            error += weightsList[i]
    error /= len(data)
    print('err:',error)
    alfa = math.log((1-error)/error,math.e)/2
    for i in range(len(data)):
        if statusList[i] == True:
            weightsList[i] *= math.e**(-1*alfa)
        else:
            weightsList[i] *= math.e**(alfa)
    for i in range(len(weightsList)):
        weightsList[i] /= sum(weightsList)
    #creating confusion matrix
    tp = 1
    fn = 1
    fp = 1
    tn = 1
    for i in range(len(data)):
        predictedLable = gnb.predict([data[['Pclass','Sex','Age']].iloc[i]])[0]
        if predictedLable == 1 and data['Target_class'][i] == 1:
            tp += 1
        elif predictedLable == 1 and data['Target_class'][i] == 0:
            fn += 1
        elif predictedLable == 0 and data['Target_class'][i] == 1:
            fp += 1
        else:
            tn += 1
```

```
print('true positive:',tp-1)
print('false negative:',fn-1)
print('false positive:',fp-1)
print('true negative:',tn-1)
print('precision:',(tp)/(fp+tp))
print('recall:',(tp)/(fn+tp))
print('--------------------------------')
```

در شکل زیر نیز یکی از confusion matrix ها اورده شده است.

```
err: 1.9843934360417912e-06
true positive: 360
false negative: 93
false positive: 64
true negative: 197
precision: 0.8474178403755869
recall: 0.7934065934065934
```