Homework2

Full name: Rong Zhen zID: z5225226

Q1: PartA

Decision Tree Results

Dataset	Default	0%	25%	50%	75%
australian	56.52% (2)	81.16% (7)	86.96% (2)	56.52% (2)	20.77% (7)
labor	66.67% (2)	94.44% (7)	44.44% (7)	66.67% (7)	50.00% (12)
diabetes	66.23% (2)	67.10% (7)	64.07% (12)	66.23% (2)	35.50% (27)
ionosphere	66.04% (2)	86.79% (7)	82.08% (27)	71.70% (7)	18.87% (12)

PartB

---4

PartC

--2

Q2:

Part A:

Answer: Accuracy for training set: 0.8969404186795491

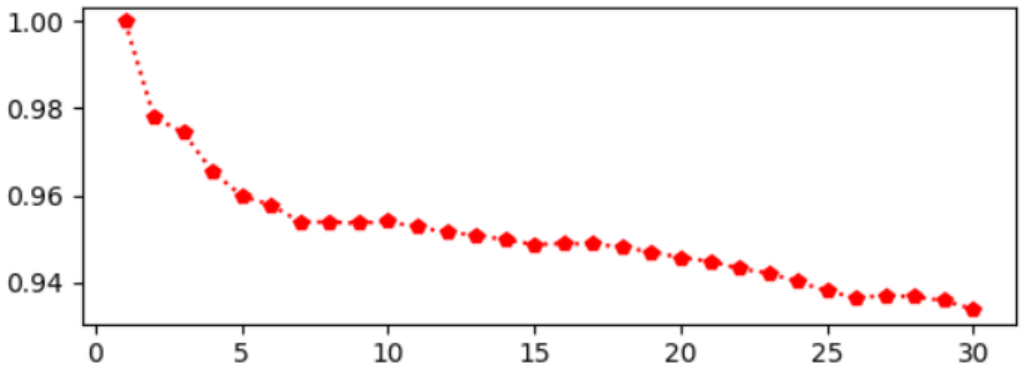
Accuracy for test set: 0.7681159420289855

Part B:

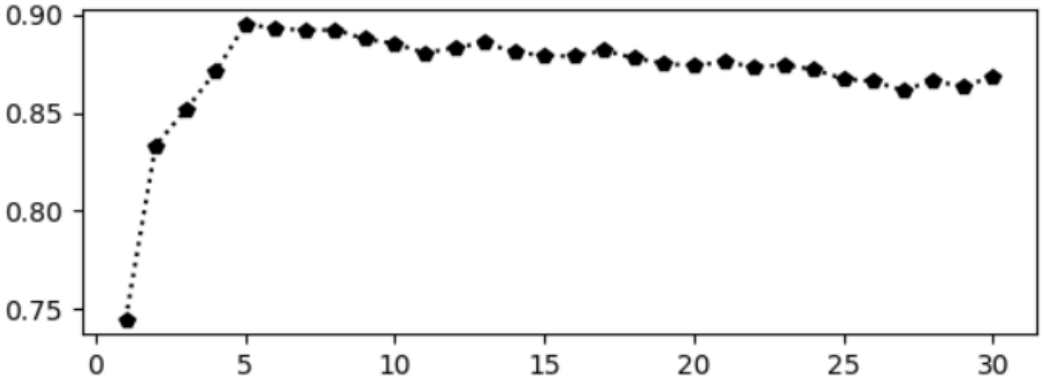
Answer: The optimal number of k is 5.

Part C:

Plot for training set:



Plot for test set:



Part D:

Precision and recall for k=5 is:(0.7666666666666667, 0.8518518518518519)

precision and recall for k=2 is:(0.7894736842105263, 0.555555555555556)

Compare: We can see the precision scores of these two models are almost the same. But when k=5, the recall score is much larger. So the model with k=5 is better in general.

```
import csv
import math
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.metrics import recall_score,precision_score

x = [[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]]
y = []
with open('CreditCards.csv','r') as csvfile:
```

```
    r = csv.reader(csvfile)
    for i,rows in enumerate(r):
        if i==0:
            name = rows
with open('CreditCards.csv','r') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        for i in range(0,14):
            x[i].append(float(row[name[i]]))
        y.append(float(row['Y']))

#pre-processing
def normalisation(xlist):
     xmin = min(xlist)
     xmax = max(xlist)
     for i in range(len(xlist)):
         xlist[i] = (xlist[i] - xmin)/(xmax - xmin)
         #print(xlist)
     return xlist

list1 = []
for i in range(len(x)):
    list1.append(normalisation(x[i]))
list2 = [[row[i] for row in list1] for i in range(690)]

#creating test and training sets
x_training = np.array(list2[0:621])
y_training = np.array(y[0:621])
x_test = np.array(list2[621:690])
y_test = np.array(y[621:690])

#Part A get two accuracy
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 2)
knn.fit(x_training,y_training)
print("Accuracy for training set: ",knn.score(x_training,y_training))
print("Accuracy2 for test set: ",knn.score(x_test,y_test))
#Accuracy:  0.8969404186795491
#Accuracy2:  0.7681159420289855

#Part B AUC score for training and test sets
#Part C plot them
neighbors = np.arange(1,31)

```

```
def auclist(rangelist,xlist,ylist,xtest,ytest):
    auclist_train = np.empty(len(rangelist))
    auclist_test = np.empty(len(rangelist))
    for i,k in enumerate(rangelist):
        knn_n = KNeighborsClassifier(n_neighbors=k)
        knn_n.fit(xlist,ylist)
        y_pred = knn_n.predict_proba(xlist)
        y_pred2 = knn_n.predict_proba(xtest)
        auclist_train[i] = roc_auc_score(ylist,y_pred[:,1])
        auclist_test[i] = roc_auc_score(ytest,y_pred2[:,1])
    return auclist_train,auclist_test
auclist_train,auclist_test = auclist(neighbors,x_training,y_training,x_test,y_test)
auclist_testlist = auclist_test.tolist()

print("the optimal value is: ",auclist_testlist.index(max(auclist_testlist))+1)


fig = plt.figure()
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)

ax1.plot(range(1,31), auclist_train, "p:",label="training", color='r')
ax2.plot(range(1,31), auclist_test, "p:",label="test", color='k')
plt.show()



#Part D precision and recall for k=5 and k=2
def precision_and_recall(k,xlist,ylist,xtest,ytest):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(xlist,ylist)
    y_pred = knn.predict(xtest)
    return precision_score(ytest,y_pred),recall_score(ytest,y_pred)

print("precision and recall for k = 5 is: ",precision_and_recall(5,x_training,y_training,x_test,y_test))
print("precision and recall for k = 2 is: ",precision_and_recall(2,x_training,y_training,x_test,y_test))
```