COMP9417- Machine Learning & Data Mining

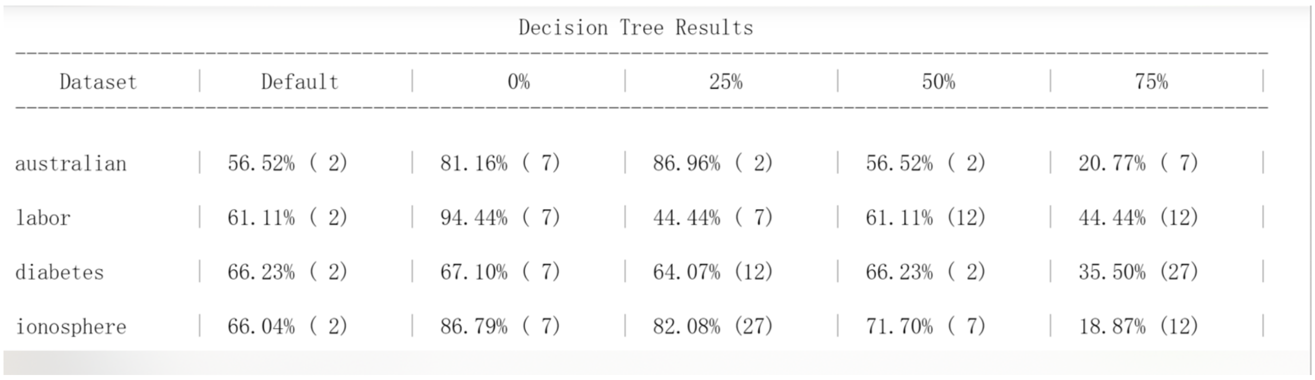
Homework 2

Name: Chongshi Wang

Question 1 :

Part A:

Answer:



```
                                          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          |  61.11% ( 2)    |  94.44% ( 7)    |  44.44% ( 7)    |  61.11% (12)    |  44.44% (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)    |
```

Part B:

By increasing the value of the value of “max_depth” parameter we can expect this to:
--- (1) overfitting not changed by decreasing max_depth of the decision tree
--- (2) decrease overfitting by increasing max_depth of the decision tree
--- (3) increase overfitting by decreasing max_depth of the decision tree
--- (4) increase overfitting by increasing max_depth of the decision tree

Answer: (4)

Part C：

--- (1) no
--- (2) yes, for 1/4 of the datasets
--- (3) yes, for 2/4 of the datasets
--- (4) yes, for 3/4 of the datasets
--- (5) yes, for 4/4 of the datasets

Answer: (2)

Question 2

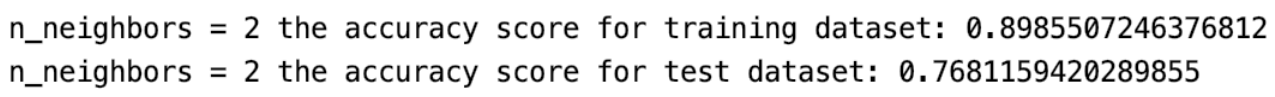
Part A:

Implement a kNN classifier for Australian credit risk prediction using sklearn library. You should set the n_neighbors =2 for training the model. What is your accuracy score for training and test dataset?

Answer:

The accuracy score for training dataset = 0.8985507246376812

The accuracy score for test dataset = 0.7681159420289855



```
n_neighbors = 2 the accuracy score for training dataset: 0.8985507246376812
n_neighbors = 2 the accuracy score for test dataset: 0.7681159420289855
```

Part B:

Find optimal number of neighbors by developing a search algorithm to find the optimal value of k. You should find the optimal number of k in a range between 1 to 30 and finding optimal value for number of k. please use AUC score to find the optimal number of neighbors.

Answer:

The optimal number of neighbors = 5

AUC score of the optimal number of neighbors = 0.8990299823633157

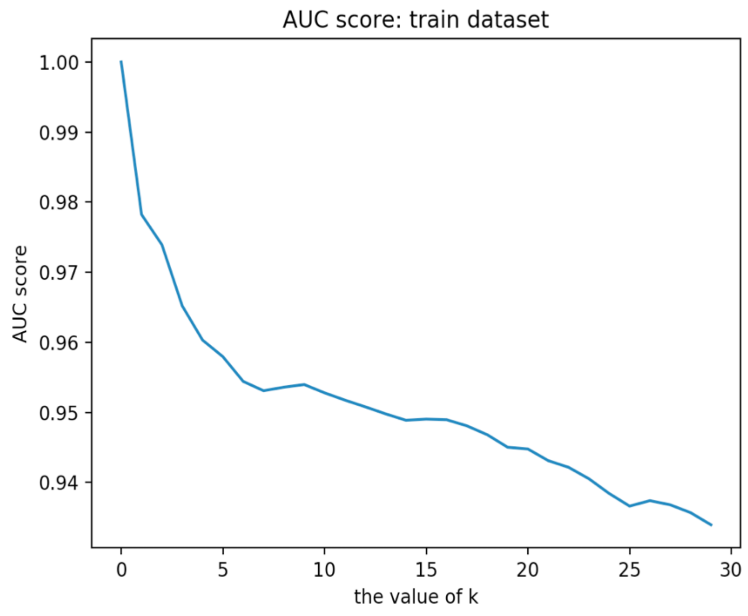
```
AUC score of the optimal number of neighbours is: 0.8990299823633157
the optimal number of neighbours is: 5
```

Part C:

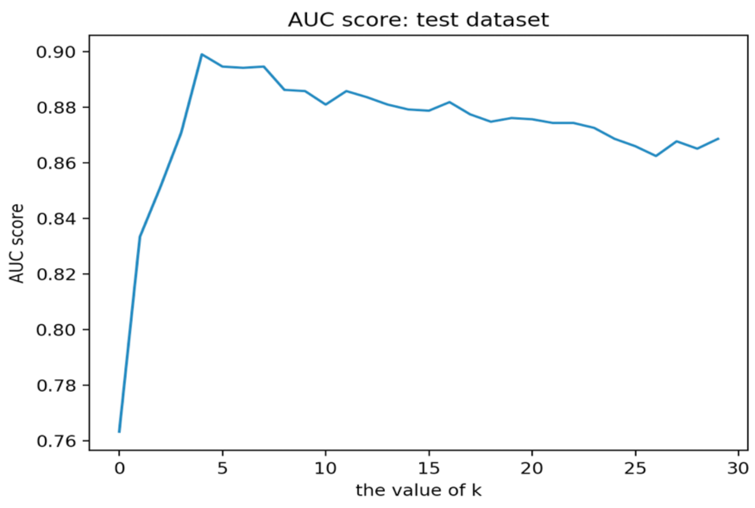
Plot the AUC score for all iterations (k: 1,...,30) in training and test sets. (one plot for training, and one for test set).

Answer:

The plot for training:



The plot for test:



Part D:

Compute precision and recall evaluation metrics for your kNN model with optimal number of neighbors and another model that you have built in part A. Compare these metrics for these two models.

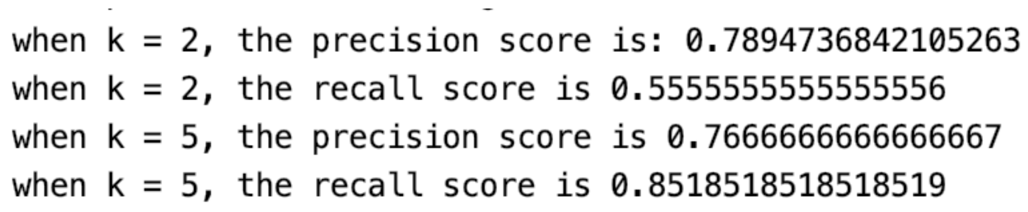
Answer:

When k = 2
the precision score = 0.7894736842105263
the recall score = 0.5555555555555556
When k = 5
the precision score = 0.7666666666666667
the recall score = 0.8518518518518519

Explanation:

An ideal system with high precision and high recall will return many results, with all results labeled correctly.

The precision score of k = 2 slightly > that of k =5, but the recall score of k = 2 extremely < that of k = 5. Thus, the kNN model with optimal number of neighbors has better performance.



```
when k = 2, the precision score is: 0.7894736842105263
when k = 2, the recall score is 0.5555555555555556
when k = 5, the precision score is 0.7666666666666667
when k = 5, the recall score is 0.8518518518518519
```

The coding:

```python
import numpy
import csv
import math
import matplotlib.pyplot as py
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

```
file_path = r'/Users/edwin/downloads/CreditCards.csv'

# read csv file
def read_function(file):
    data = []
    with open(file) as f:
        read_file = csv.reader(f)
        for line in read_file:
            data.append(line)
    final_data = numpy.array(data[1:]).astype(float)
    return final_data

# nomalization: dataset
def nomalization(data):
    length = len(data)
    for i in range(14):
        max_number = data[:,i].max()
        min_number = data[:,i].min()
        difference = max_number - min_number
        for j in range(length):
            data[j,i] = (data[j,i] - min_number)/difference
    return data

data = read_function(file_path)
nomalization_data = nomalization(data)
train_data = nomalization_data[:621,:]
test_data = nomalization_data[621:,:]
train_x = train_data[:,:13]
train_y = train_data[:,14]
test_x = test_data[:,:13]
test_y = test_data[:,14]

###################### Part A ########################

knn_classifier1 = KNeighborsClassifier(n_neighbors=2)
knn_classifier1.fit(train_x,train_y)
pre1 = knn_classifier1.predict(train_x)
acc_score1 = accuracy_score(train_y,pre1)
print("n_neighbors = 2 the accuracy score for training dataset:",acc_score1)
pre2 = knn_classifier1.predict(test_x)
acc_score2 = accuracy_score(test_y,pre2)
print("n_neighbors = 2 the accuracy score for test dataset:",acc_score2)
```

```
###################### Part B & Part C##########################

auc_index = []
auc_train = []
auc_test = []
max = 0


for i in range(1,31):
   knn_best = KNeighborsClassifier(n_neighbors= i)
   knn_best.fit(train_x,train_y)
   prediction_train = knn_best.predict_proba(train_x)
   prediction_test = knn_best.predict_proba(test_x)
   auc_t1 = roc_auc_score(y_true = train_y, y_score= prediction_train[:,1])
   auc_t2 = roc_auc_score(y_true= test_y, y_score= prediction_test[:,1])
   auc_train.append(auc_t1)
   auc_test.append(auc_t2)
   if auc_t2 > max:
      max = auc_t2
      auc_index.append(i)

print("AUC score of the optimal number of neighbours is:",max)
print("the optimal number of neighbours is:",auc_index[-1])

py.plot(auc_train)
py.xlabel("the value of k")
py.ylabel("AUC score")
py.title("AUC score: train dataset")
py.show()

py.plot(auc_test)
py.xlabel("the value of k")
py.ylabel("AUC score")
py.title("AUC score: test dataset")
py.show()

###################### Part D##########################

# K = 2
precision_k0 = precision_score(y_true= test_y, y_pred= pre2)
recall_k0 = recall_score(y_true= test_y, y_pred= pre2)
print("when k = 2, the precision score is:",precision_k0)
print("when k = 2, the recall score is",recall_k0)
```

```
# K = 5
knn_5 = KNeighborsClassifier(n_neighbors=5)
knn_5.fit(train_x, train_y)
pre5 = knn_5.predict(test_x)
precision_k5 = precision_score(y_true= test_y, y_pred= pre5)
recall_k5 = recall_score(y_true= test_y, y_pred= pre5)
print("when k = 5, the precision score is",precision_k5)
print("when k = 5, the recall score is",recall_k5)
```