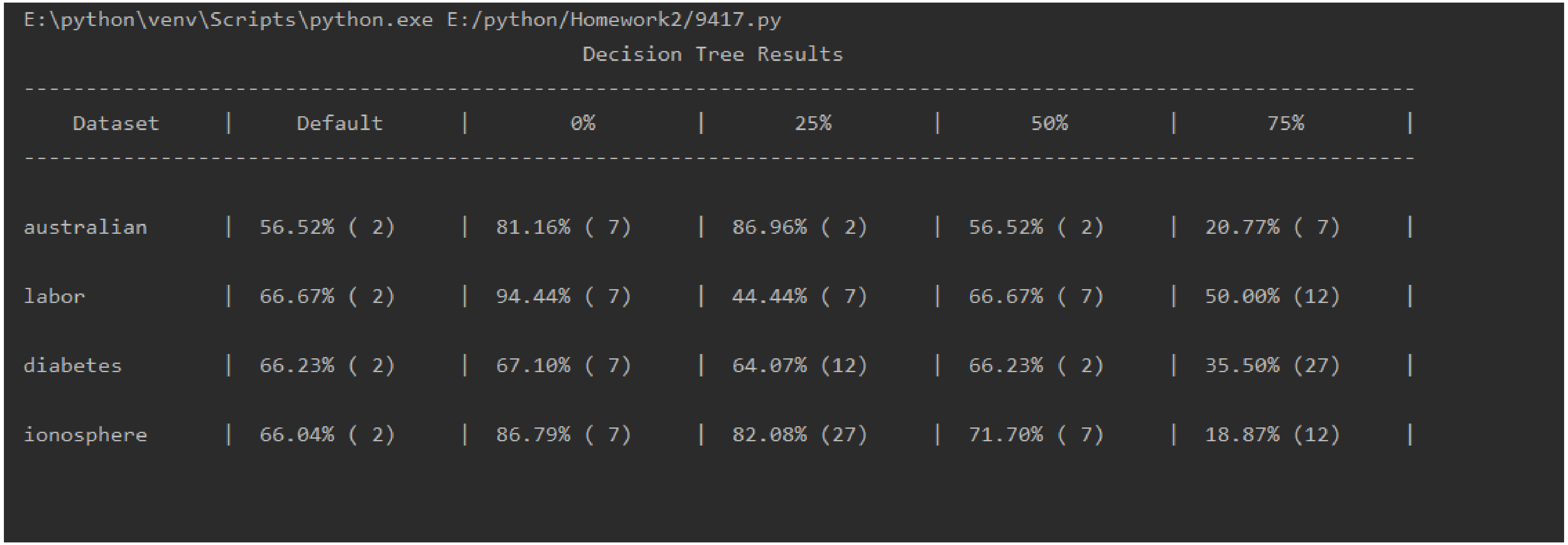
Homework 2

Z5048267 Lipeng TAO

Q1

Part A



```
E:\python\venv\Scripts\python.exe E:/python/Homework2/9417.py
                                                     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)     |
```

Part B answer is number(4)

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

Part C answer is number(2)

--- (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

Q2

Part A - [1 mark]

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?

Training set acc is 0.8969404186795491

Test set acc is 0.7681159420289855

Part B - [1 mark]

Find optimal number of neighbours 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 neighbours.

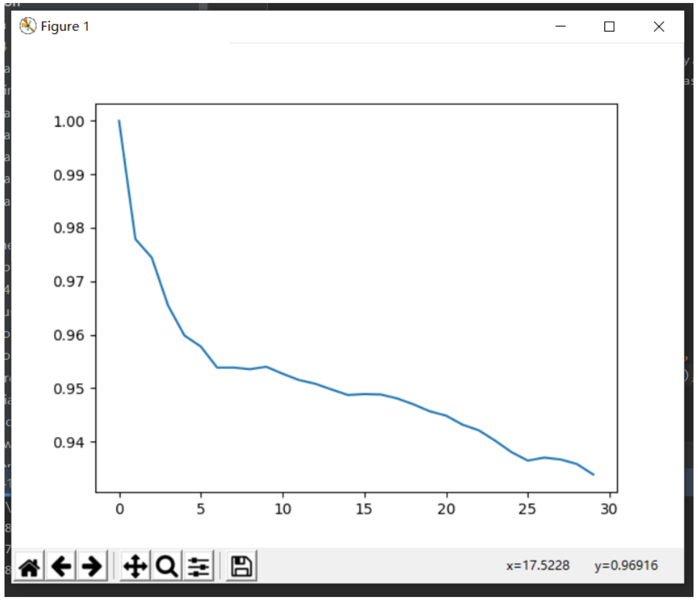
K is 5

K`s value is 0.8950617283950617

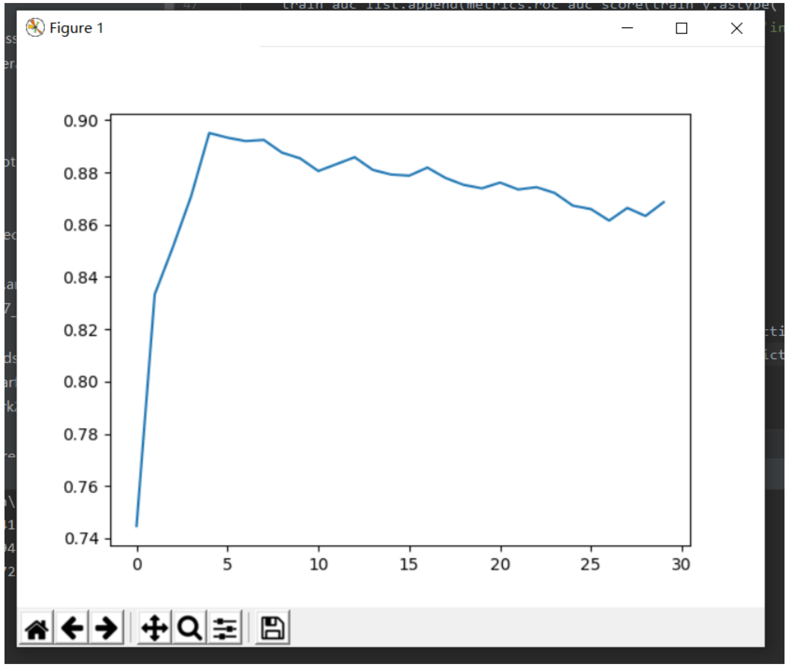
Part C - [0.5 mark]

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

Training set plot as below:



Test set plot as below:



Part D - [1 mark]

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

When K = 2:

```
Recall value is 0.5555555555555556          precision value is 0.7894736842105263
```

When K = 5

```
Recall value is 0.8518518518518519          precision value is 0.7666666666666667
```

Code as below:

```python
import numpy as np

import csv

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn import neighbors

from sklearn.metrics import accuracy_score

from sklearn import metrics
```

```
def csv_reader(file_name):

    data = []

    with open(file_name) as file:

        csv_file = csv.reader(file)

        for line in csv_file:

            data.append(line)

    return (np.array(data[1:])).astype(float)


raw_data = csv_reader('CreditCards.csv')

raw_X = raw_data[:, :-1]

raw_y = raw_data[:, -1]

min_max_scaler = preprocessing.MinMaxScaler()

processed_X = min_max_scaler.fit_transform(raw_X)
```

```
train_X = processed_X[:621, :]
test_X = processed_X[621:, :]
train_y = raw_y[:621]
test_y = raw_y[621:]

clf = neighbors.KNeighborsClassifier(2)
clf.fit(train_X, train_y.astype('int'))
prediction = clf.predict(train_X)
acc = accuracy_score(train_y.astype('int'), prediction)
print(acc)
prediction = clf.predict(test_X)
acc = accuracy_score(test_y.astype('int'), prediction)
```

```
print(acc)


recall_on = metrics.recall_score(test_y.astype('int'), prediction)

prec_on = metrics.precision_score(test_y.astype('int'), prediction)

train_auc_list = []

test_auc_list = []

for k in range(1, 31):

    clf = neighbors.KNeighborsClassifier(k)

    clf.fit(train_X, train_y.astype('int'))

    train_prediction = clf.predict_proba(train_X)

    test_prediction = clf.predict_proba(test_X)


    train_auc_list.append(metrics.roc_auc_score(train_y.astype('int'), np.array(train_prediction[:,-1])))
    test_auc_list.append(metrics.roc_auc_score(test_y.astype('int'), np.array(test_prediction[:,-1])))

opt_k = np.argmax(test_auc_list)+1
print(np.max(test_auc_list))
print(opt_k)
plt.plot(train_auc_list)
plt.show()
plt.plot(test_auc_list)
plt.show()

clf = neighbors.KNeighborsClassifier(opt_k)
clf.fit(train_X, train_y.astype('int'))
prediction = clf.predict(test_X)
```

```
recall_opt = metrics.recall_score(test_y.astype('int'), prediction)
prec_opt = metrics.precision_score(test_y.astype('int'), prediction)

print(recall_on, recall_opt)
print(prec_on, prec_opt)
```