# Homework 2

Chang Wang

## Question 1

### Question part A

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) |

### Question part B

(4)

### Question part C

(2)

## Question 2

### Question part A

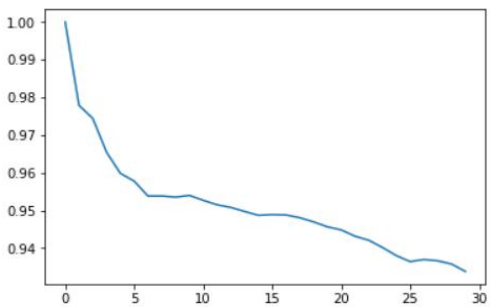
accuracy score for training dataset: 0.8969404186795491
accuracy score for test dataset: 0.7681159420289855

### Question part B

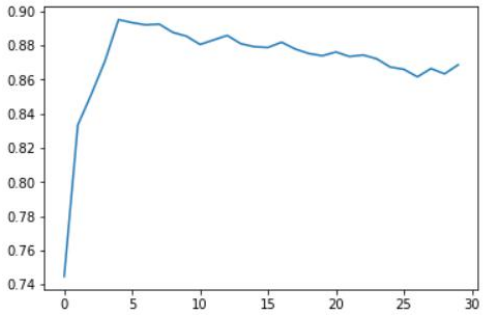
K = 5

### Question part C

Training dataset:




Test dataset:




Question part D

K = 2

Recall output = 0.5555555555555556

Precision output = 0.7894736842105263

K = 5

Recall output = 0.8518518518518519

Precision output = 0.7666666666666667

High precision means high accuracy for classified data. However, low recall means that many data are not successfully classified

Here is my codes:

```
import csv
from sklearn import neighbors
from sklearn.metrics import accuracy_score
from sklearn import metrics
from matplotlib import pyplot as plt


def get_data(file_name):
    data = []
    with open(file_name) as file:
        csv_file = csv.reader(file)
        for line in csv_file:
            data.append(line)
        return data[1:]


data_line = get_data('CreditCards.csv')


def transformation(data):
    for index in range(0, 14):
        max_number = 0
        min_number = 999999
        for row in range(len(data)):
            if float(data[row][index]) > max_number:
                max_number = float(data[row][index])
            if float(data[row][index]) < min_number:
```

```
                min_number = float(data[row][index])
        for row_a in range(len(data)):
            data[row_a][index] = (float(data[row_a][index]) - min_number) / (max_number - min_number)
    return data


data = transformation(data_line)
training = transformation(data[:621])
test = data[621:]


def X_train(data):
    x_train = []
    for x in range(len(data)):
        list_x = []
        for y in range(0, 14):
            list_x.append(float(data[x][y]))
        x_train.append(list_x)
    return x_train


def Y_train(data):
    y_train = []
    for i in range(len(data)):
        y_train.append(float(data[i][14]))
    return y_train


def accuracy_score_2(x_training, y_training, x_train, y_train):
    model = neighbors.KNeighborsClassifier(2)
    model.fit(x_training, y_training)
    score = accuracy_score(y_train, model.predict(x_train))
    return score


x_long = X_train(training)
y_long = Y_train(training)


x_short = X_train(test)
y_short = Y_train(test)


training_score = accuracy_score_2(x_long, y_long, x_long, y_long)
test_score = accuracy_score_2(x_long, y_long, x_short, y_short)


print("accuracy score for training dataset:", training_score)
print("accuracy score for test dataset:", test_score)


def Optimal_number(x_1, y_1, x_train, y_train):
    largest_AUC_score = 0
```

```
    optimal_number = 0
    AUC_list = []
    for index in range(1, 31):
        model = neighbors.KNeighborsClassifier(index)
        model.fit(x_1, y_1)
        AUC_score = metrics.roc_auc_score(y_train, model.predict_proba(x_train)[:, 1])
        if AUC_score > largest_AUC_score:
            largest_AUC_score = AUC_score
            optimal_number = index
        AUC_list.append(AUC_score)
    return optimal_number, AUC_list

training_optimal_number, training_AUC_list = Optimal_number(x_long, y_long, x_long, y_long)
test_optimal_number, test_AUC_list = Optimal_number(x_long, y_long, x_short, y_short)

print(training_optimal_number)
print(test_optimal_number)
plt.plot(training_AUC_list)
plt.show()
plt.plot(test_AUC_list)
plt.show()

def PartD(x_1, y_1, x_train, y_train, k):
    model = neighbors.KNeighborsClassifier(2)
    model.fit(x_1, y_1)
    recall_opt = metrics.recall_score(y_train, model.predict(x_train))
    prec_opt = metrics.precision_score(y_train, model.predict(x_train))
    model = neighbors.KNeighborsClassifier(k)
    model.fit(x_1, y_1)
    recall_opt_1 = metrics.recall_score(y_train, model.predict(x_train))
    prec_opt_1 = metrics.precision_score(y_train, model.predict(x_train))
    return recall_opt, prec_opt, recall_opt_1, prec_opt_1

recall_opt, prec_opt, recall_opt_1, prec_opt_1 = PartD(x_long, y_long, x_short, y_short, test_optimal_number)
print(recall_opt, prec_opt, recall_opt_1, prec_opt_1)
```