Q1.

Part A.

As the picture shows below:

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.

The correct answer is (4).

Part C.

The correct answer is (2).

Q2.

Part A.

Accuracy score for training dataset is **0.8969404186795491**

Accuracy score for test dataset is **0.7681159420289855**

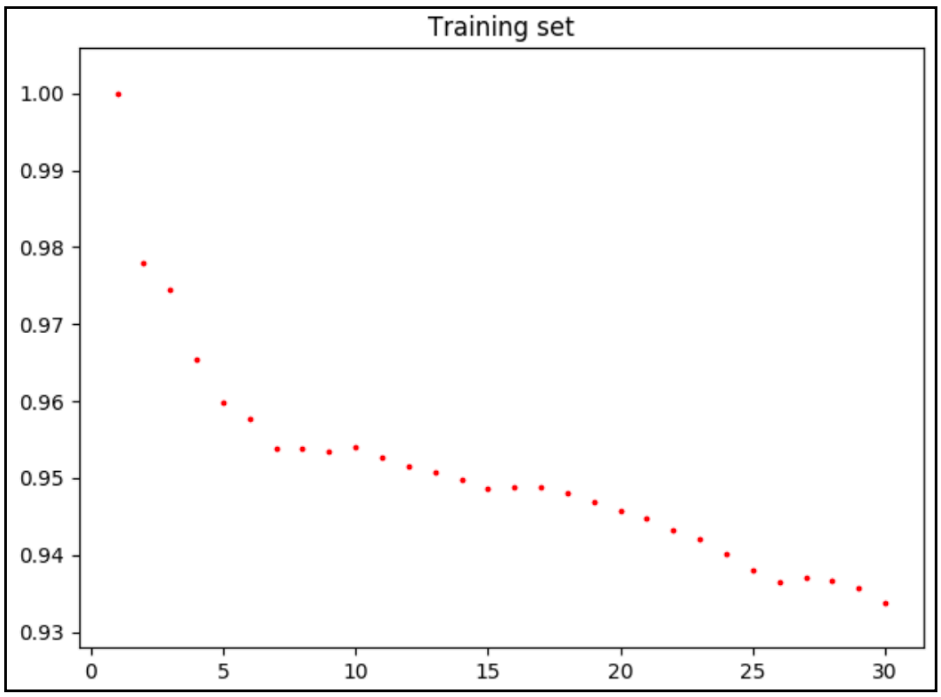
Part B.

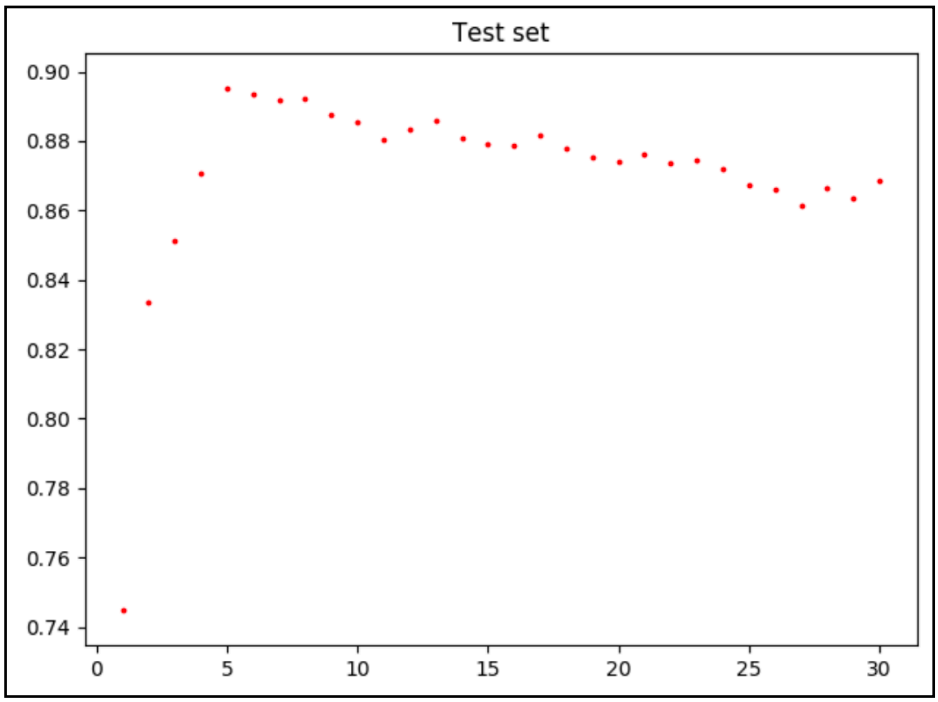
The optimal number of k is **5** and in that case the AUC score is **0.8950617283950617**

Part C.

As the pictures show below:

COMP9417 ZID: z5230310 Name: Tian Liu





Part D.

kNN model with optimal number of neighbours **2**:

The precision score is **0.7894736842105263**

The recall score is **0.5555555555555556**

kNN model with optimal number of neighbours **5**:

The precision score is **0.7666666666666667**

The recall score is **0.8518518518518519**

CODE:

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score, accuracy_score, recall_score, precision_score
def pre_processing(data):
    x = data.transpose()
    for i in x:
        max_x = max(i)
        min_x = min(i)
        for j in range(len(i)):
            i[j] = (i[j] - min_x) / (max_x - min_x)
    return x.transpose()
data_frame = pd.read_csv('CreditCards.csv')
data = data_frame.values[:]
norm_set = pre_processing(data[:, :-1])
train_set_x = norm_set[:621, :]
train_set_y = data[:621, -1]
test_set_x = norm_set[621:, :]
test_set_y = data[621:, -1]
# PartA
kNN = KNeighborsClassifier(2)
kNN.fit(train_set_x, train_set_y)
y_predict_test = kNN.predict(test_set_x)
y_predict_train = kNN.predict(train_set_x)
scores_test = accuracy_score(test_set_y, y_predict_test)
scores_train = accuracy_score(train_set_y, y_predict_train)
print('accurcy score for test set:', scores_test)
print('accurcy score for training set:', scores_train)
```

```
# PartB find the optimal number of neighbours
train_score = []
test_score = []
for k in range(1, 31):
    kNN = KNeighborsClassifier(k)
    kNN.fit(train_set_x, train_set_y)
    y_predict_test = kNN.predict_proba(test_set_x)
    y_predict_train = kNN.predict_proba(train_set_x)
    auc_score_test = roc_auc_score(test_set_y, y_predict_test[:, 1])
    auc_score_train = roc_auc_score(train_set_y, y_predict_train[:, 1])
    train_score.append(auc_score_train)
    test_score.append(auc_score_test)
index, max_accurancy = max(enumerate(test_score), key=lambda item: item[1])
max_k = index + 1
axis_x = [i for i in range(1, 31)]
print(max_k, max_accurancy)
# PartC plot the AUC score
plt.scatter(axis_x, train_score, color="r", s=3)
plt.title('Training set')
plt.show()
plt.scatter(axis_x, test_score, color="r", s=3)
plt.title('Test set')
plt.show()
# PartD kNN - model when k = 2
kNN = KNeighborsClassifier(2)
kNN.fit(train_set_x, train_set_y)
y_predict_test = kNN.predict(test_set_x)
precision_s = precision_score(test_set_y, y_predict_test)
recall_s = recall_score(test_set_y, y_predict_test)
```

```
print('precision score: ', precision_s)

print('recall score: ', recall_s)

# kNN - model when k = 5

kNN = KNeighborsClassifier(5)

kNN.fit(train_set_x, train_set_y)

y_predict_test = kNN.predict(test_set_x)

precision_s = precision_score(test_set_y, y_predict_test)

recall_s = recall_score(test_set_y, y_predict_test)

print('precision score: ', precision_s)

print('recall score: ', recall_s)
```