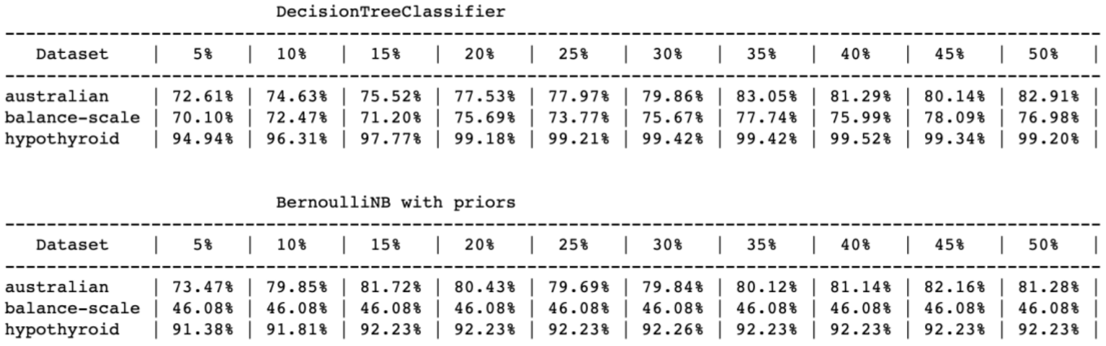
Rongtao Shen
z5178114

Homework 2

Question 1

Part A



```
                          DecisionTreeClassifier
-------------------------------------------------------------------------------------------------------
   Dataset     |   5%   |  10%   |  15%   |  20%   |  25%   |  30%   |  35%   |  40%   |  45%   |  50%   |
-------------------------------------------------------------------------------------------------------
australian     | 72.61% | 74.63% | 75.52% | 77.53% | 77.97% | 79.86% | 83.05% | 81.29% | 80.14% | 82.91% |
balance-scale  | 70.10% | 72.47% | 71.20% | 75.69% | 73.77% | 75.67% | 77.74% | 75.99% | 78.09% | 76.98% |
hypothyroid    | 94.94% | 96.31% | 97.77% | 99.18% | 99.21% | 99.42% | 99.42% | 99.52% | 99.34% | 99.20% |

                          BernoulliNB with priors
-------------------------------------------------------------------------------------------------------
   Dataset     |   5%   |  10%   |  15%   |  20%   |  25%   |  30%   |  35%   |  40%   |  45%   |  50%   |
-------------------------------------------------------------------------------------------------------
australian     | 73.47% | 79.85% | 81.72% | 80.43% | 79.69% | 79.84% | 80.12% | 81.14% | 82.16% | 81.28% |
balance-scale  | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% |
hypothyroid    | 91.38% | 91.81% | 92.23% | 92.23% | 92.23% | 92.26% | 92.23% | 92.23% | 92.23% | 92.23% |
```

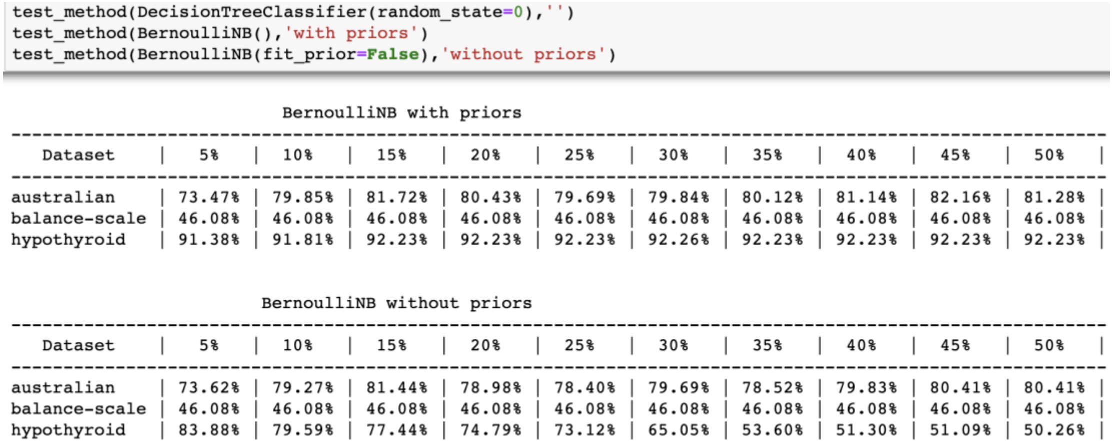
Part B

(3) most of the 6 models show a learning curve

(4) All 3 Decision Tree models are generally better than Bernoulli Naïve Bayes models

Part C

(1) BNB performs better with priors



```
test_method(DecisionTreeClassifier(random_state=0),'')
test_method(BernoulliNB(),'with priors')
test_method(BernoulliNB(fit_prior=False),'without priors')
```

```
                          BernoulliNB with priors
-------------------------------------------------------------------------------------------------------
   Dataset     |   5%   |  10%   |  15%   |  20%   |  25%   |  30%   |  35%   |  40%   |  45%   |  50%   |
-------------------------------------------------------------------------------------------------------
australian     | 73.47% | 79.85% | 81.72% | 80.43% | 79.69% | 79.84% | 80.12% | 81.14% | 82.16% | 81.28% |
balance-scale  | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% |
hypothyroid    | 91.38% | 91.81% | 92.23% | 92.23% | 92.23% | 92.26% | 92.23% | 92.23% | 92.23% | 92.23% |

                        BernoulliNB without priors
-------------------------------------------------------------------------------------------------------
   Dataset     |   5%   |  10%   |  15%   |  20%   |  25%   |  30%   |  35%   |  40%   |  45%   |  50%   |
-------------------------------------------------------------------------------------------------------
australian     | 73.62% | 79.27% | 81.44% | 78.98% | 78.40% | 79.69% | 78.52% | 79.83% | 80.41% | 80.41% |
balance-scale  | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% | 46.08% |
hypothyroid    | 83.88% | 79.59% | 77.44% | 74.79% | 73.12% | 65.05% | 53.60% | 51.30% | 51.09% | 50.26% |
```

Question 2

Part A

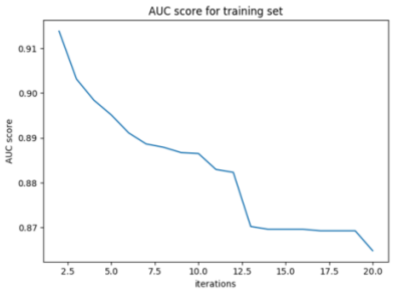
Accuracy score for training dataset: 0.8564516129032258

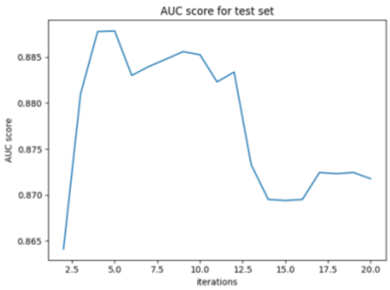
Accuracy score for test dataset: 0.8277153558052435

Part B

An optimal number of min_samples_leaf is 5

Part C

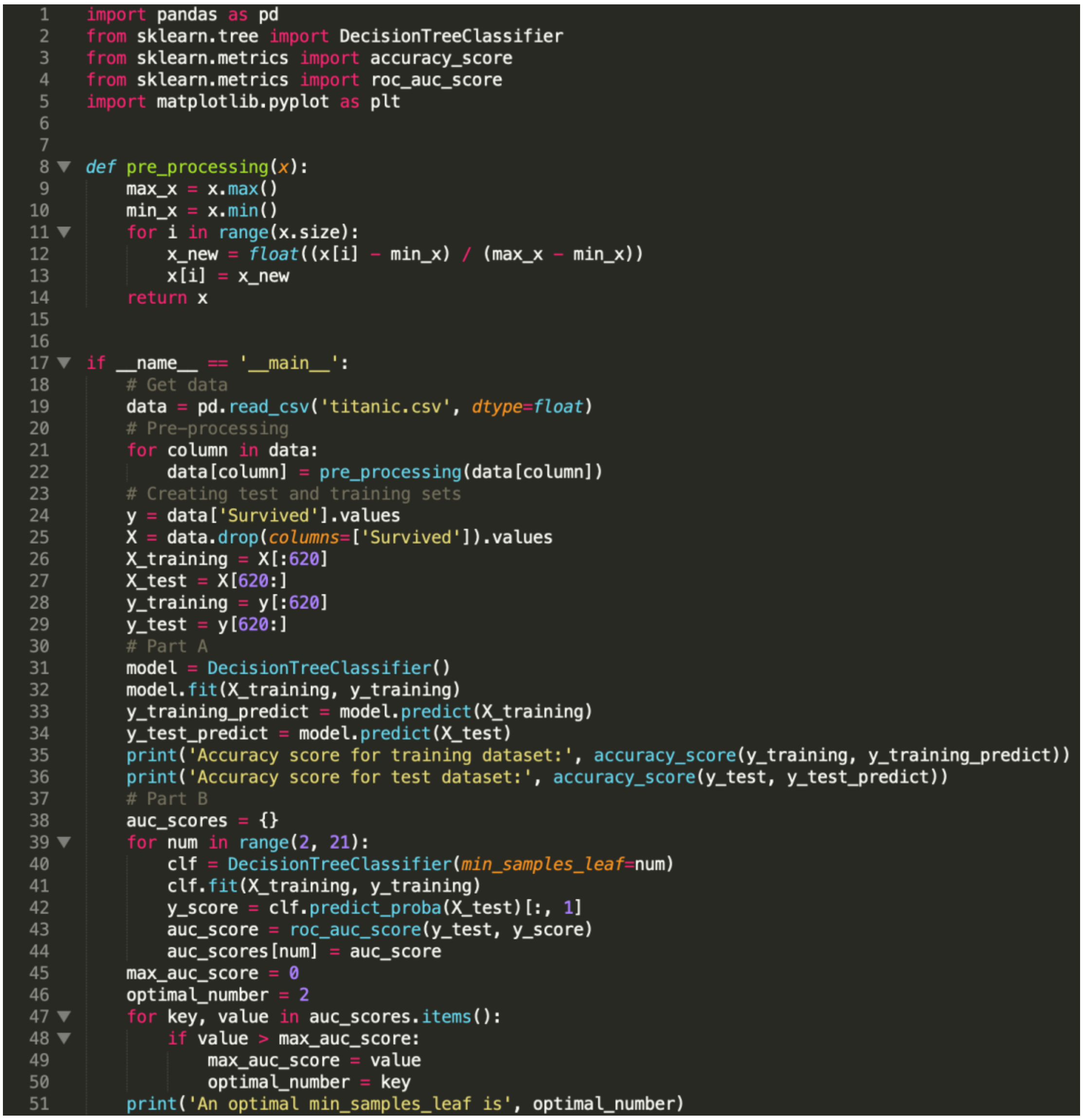




Part D

The probability is 0.36885245901639346

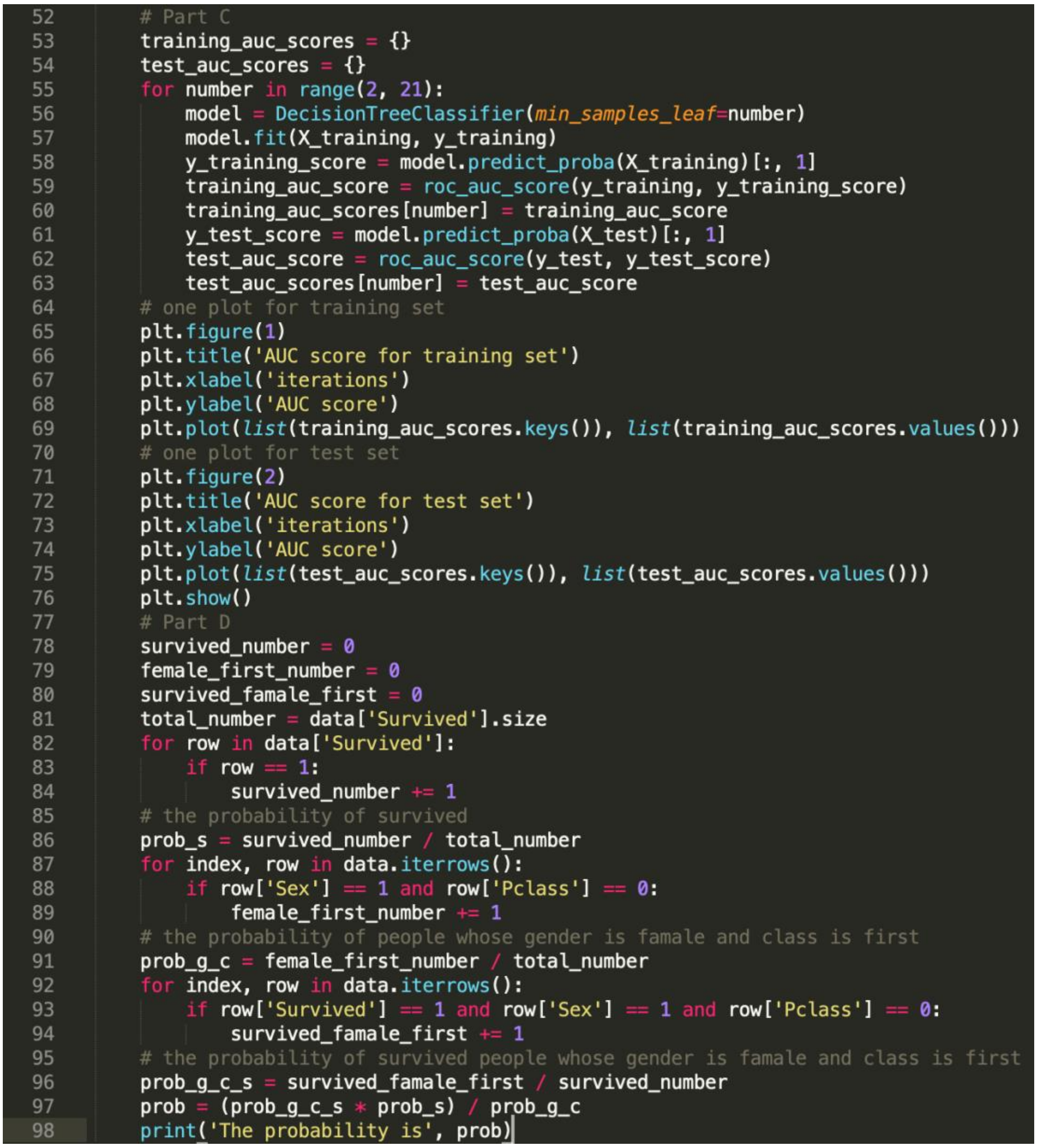
Code



```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt


def pre_processing(x):
    max_x = x.max()
    min_x = x.min()
    for i in range(x.size):
        x_new = float((x[i] - min_x) / (max_x - min_x))
        x[i] = x_new
    return x


if __name__ == '__main__':
    # Get data
    data = pd.read_csv('titanic.csv', dtype=float)
    # Pre-processing
    for column in data:
        data[column] = pre_processing(data[column])
    # Creating test and training sets
    y = data['Survived'].values
    X = data.drop(columns=['Survived']).values
    X_training = X[:620]
    X_test = X[620:]
    y_training = y[:620]
    y_test = y[620:]
    # Part A
    model = DecisionTreeClassifier()
    model.fit(X_training, y_training)
    y_training_predict = model.predict(X_training)
    y_test_predict = model.predict(X_test)
    print('Accuracy score for training dataset:', accuracy_score(y_training, y_training_predict))
    print('Accuracy score for test dataset:', accuracy_score(y_test, y_test_predict))
    # Part B
    auc_scores = {}
    for num in range(2, 21):
        clf = DecisionTreeClassifier(min_samples_leaf=num)
        clf.fit(X_training, y_training)
        y_score = clf.predict_proba(X_test)[:, 1]
        auc_score = roc_auc_score(y_test, y_score)
        auc_scores[num] = auc_score
    max_auc_score = 0
    optimal_number = 2
    for key, value in auc_scores.items():
        if value > max_auc_score:
            max_auc_score = value
            optimal_number = key
    print('An optimal min_samples_leaf is', optimal_number)
```



```
    # Part C
    training_auc_scores = {}
    test_auc_scores = {}
    for number in range(2, 21):
        model = DecisionTreeClassifier(min_samples_leaf=number)
        model.fit(X_training, y_training)
        y_training_score = model.predict_proba(X_training)[:, 1]
        training_auc_score = roc_auc_score(y_training, y_training_score)
        training_auc_scores[number] = training_auc_score
        y_test_score = model.predict_proba(X_test)[:, 1]
        test_auc_score = roc_auc_score(y_test, y_test_score)
        test_auc_scores[number] = test_auc_score
    # one plot for training set
    plt.figure(1)
    plt.title('AUC score for training set')
    plt.xlabel('iterations')
    plt.ylabel('AUC score')
    plt.plot(list(training_auc_scores.keys()), list(training_auc_scores.values()))
    # one plot for test set
    plt.figure(2)
    plt.title('AUC score for test set')
    plt.xlabel('iterations')
    plt.ylabel('AUC score')
    plt.plot(list(test_auc_scores.keys()), list(test_auc_scores.values()))
    plt.show()
    # Part D
    survived_number = 0
    female_first_number = 0
    survived_famale_first = 0
    total_number = data['Survived'].size
    for row in data['Survived']:
        if row == 1:
            survived_number += 1
    # the probability of survived
    prob_s = survived_number / total_number
    for index, row in data.iterrows():
        if row['Sex'] == 1 and row['Pclass'] == 0:
            female_first_number += 1
    # the probability of people whose gender is famale and class is first
    prob_g_c = female_first_number / total_number
    for index, row in data.iterrows():
        if row['Survived'] == 1 and row['Sex'] == 1 and row['Pclass'] == 0:
            survived_famale_first += 1
    # the probability of survived people whose gender is famale and class is first
    prob_g_c_s = survived_famale_first / survived_number
    prob = (prob_g_c_s * prob_s) / prob_g_c
    print('The probability is', prob)
```