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:

True statements:

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

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

Part C:

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 with uniform 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%

According to the sheet, BNB preforms better with priors.

Question 2

Part A:

accuracy score for training dataset: 0.8564516129032258

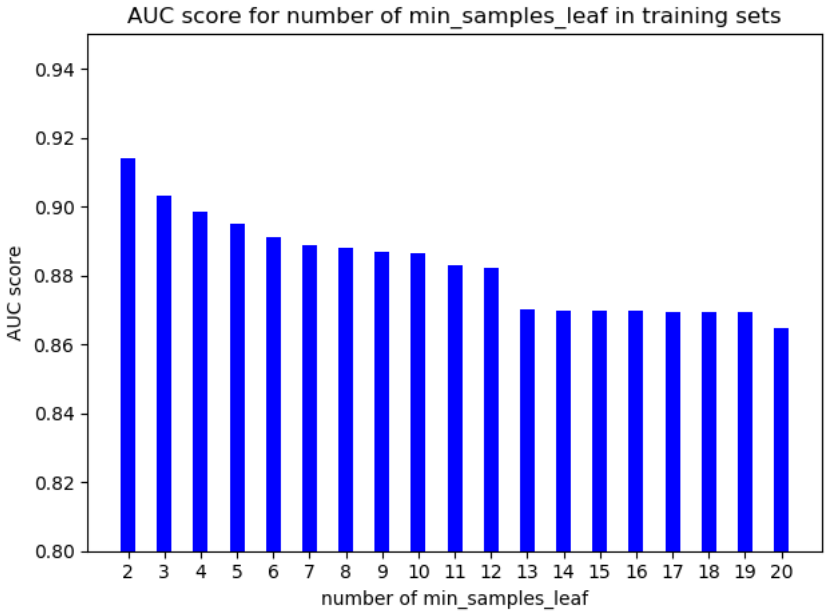
accuracy score for test dataset: 0.8277153558052435

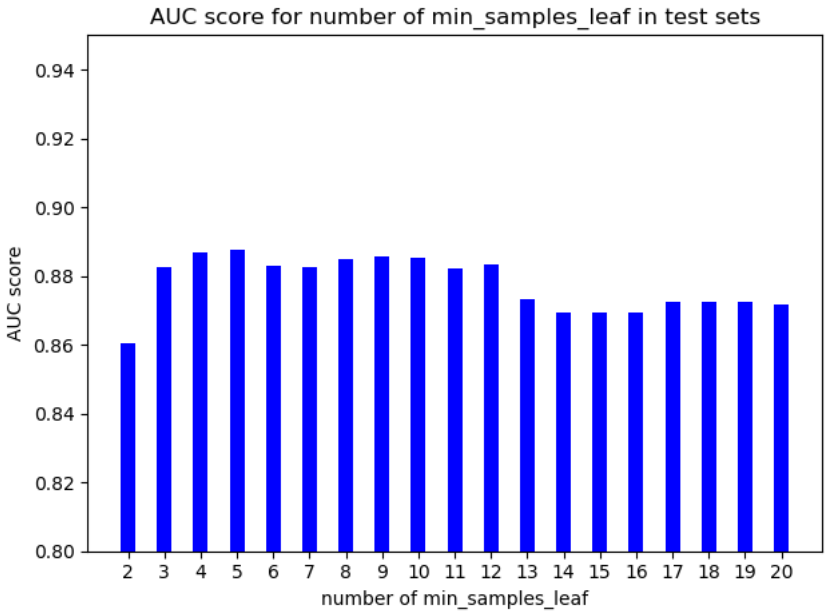
Part B:

The optimal number of min_samples_leaf is: 5

The AUC score is: 0.8877923976608187

Part C:





Part D:

P(S=true | G=female, C=1): 0.36885245901639346

The Code

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.metrics import roc_auc_score, roc_curve, auc
import matplotlib.pyplot as plt


def normalize(data):
    return (data - data.min()) / (data.max() - data.min())


def AUC_score(X, Y, model):
    _prob = model.predict_proba(X)[:,1]
    return roc_auc_score(Y, _prob)


def main():
    # Read data and creating test and training sets
    data = pd.read_csv("titanic.csv")
    data_normalized = normalize(data.iloc[:, :])
    training_set = data_normalized.iloc[:620, :]
    test_set = data_normalized.iloc[620:, :]
    training_set_x = training_set.iloc[:, :-1].values
    training_set_y = training_set.iloc[:, -1:].values
    test_set_x = test_set.iloc[:, :-1].values
    test_set_y = test_set.iloc[:, -1:].values

    # Part A
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(training_set_x, training_set_y)
    print('Part A(accuracy score for training dataset):', clf.score(training_set_x, training_set_y))
    print('Part A(accuracy score for test dataset):', clf.score(test_set_x, test_set_y))

    # Part B
    training_set_auc_score = []
    test_set_auc_score = []
    for i in range(2, 21):
        clf = tree.DecisionTreeClassifier(min_samples_leaf=i, random_state=1)
        clf.fit(training_set_x, training_set_y)

        training_set_auc_score.append(AUC_score(training_set_x, training_set_y, clf))
```

```
42.             test_set_auc_score.append(AUC_score(test_set_x, test_set_y, clf))
43.         print("₩nPart B,The optimal number of min_samples_leaf is: ", test_set_auc_score.index(max(test_set_
    auc_score)) + 2)
44.         print("Part B,The AUC score is: ", max(test_set_auc_score))
45.
46.         # Part C
47.         plt.bar(range(2, 21), training_set_auc_score, 0.4, color="blue")
48.         plt.ylim(0.8, 0.95)
49.         plt.xticks(range(2, 21))
50.         plt.xlabel("number of min_samples_leaf")
51.         plt.ylabel("AUC score")
52.         plt.title("AUC score for number of min_samples_leaf in training sets")
53.         plt.show()
54.         plt.bar(range(2, 21), test_set_auc_score, 0.4, color="blue")
55.         plt.ylim(0.8, 0.95)
56.         plt.xticks(range(2, 21))
57.         plt.xlabel("number of min_samples_leaf")
58.         plt.ylabel("AUC score")
59.         plt.title("AUC score for number of min_samples_leaf in test sets")
60.         plt.show()
61.
62.         # Part D
63.         survived, total = 0, 0
64.         for index, row in data.iterrows():
65.             if row['Pclass'] == 1 & row['Sex'] == 1:
66.                 total += 1
67.                 if row['Survived'] == 1:
68.                     survived += 1
69.         print("₩nPart D, P(S=true | G=female, C=1): ", survived / total)
70.
71.
72. if __name__ == '__main__':
73.     main()
```