9417 assignment2

Zheng Qiwen z5240149

Question 1:

Part A

This is screenshot of the result table.

DecisionTreeClassifier

Dataset	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
australian	72.61%	74.35%	75.36%	77.39%	77.83%	79.71%	83.77%	81.16%	80.72%	83.48%
balance-scale	69.92%	75.04%	69.12%	74.24%	74.40%	75.52%	78.08%	75.68%	77.92%	76.64%
hypothyroid	94.94%	96.31%	97.77%	99.18%	99.20%	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	73.48%	79.86%	81.45%	80.43%	79.71%	79.86%	79.86%	81.16%	82.17%	81.88%
balance-scale	46.08%	46.08%	46.08%	46.08%	46.24%	46.08%	46.08%	46.24%	46.24%	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)(5)

Part C

After adding one line of code at the end(test_method(BernoulliNB(fit_prior=False), without priors)),we get the following table:

BernoulliNB without priors

Dataset	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
australian	73.62%	79.42%	81.45%	78.99%	78.12%	78.55%	78.70%	80.00%	80.43%	80.58%
balance-scale	46.08%	46.08%	46.08%	46.08%	46.24%	46.08%	46.08%	46.24%	46.24%	46.08%
hypothyroid	83.51%	79.51%	77.44%	74.82%	68.48%	64.54%	54.03%	51.25%	51.06%	50.82%

Comparing two tables, we can find BNB preforms better with priors, so the answer would be (1)

Question 2:

Part A

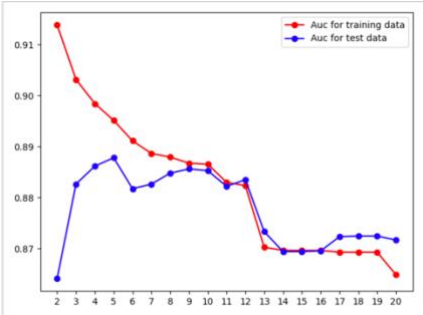
Accuracy score for training set: 0.8564516129032258.

However I got different accuracy score for test set from time to time, mostly it would be :0.8277153558052435, but there are small chance that I got 0.8314606741573034.It is because some randomness of building decision tree.

Part B

Optimal number of min samples leaf is : 5

Part C



Part D

The probability of this question is 0.36885245901639346.

The code is shown as following:

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

#read the data from dataset
df = pd.read_csv('titanic.csv')
target_name = "Survived"
target = df[target_name].values.reshape(-1,1)
all_features = df[0:].values[:,0:5]

#normalization
all_features_nom = (all_features-np.min(all_features,axis=0))/(np.max(all_features,axis=0)-np.min(all_features,axis=0))

#for partA to train the model and get the accuracy scores
cls1 = DecisionTreeClassifier()
cls1.fit(all_features_nom[0:620],target[0:620])

print('Accuracy score for training set: ',cls1.score(all_features_nom[0:620],target[0:620]))
print('Accuracy score for test set: ',cls1.score(all_features_nom[620:],target[620:]))

#preparation of getting min_samples_leaf and the plot
min_samples_leaf = 2
optimal_auc_test = 0
plt.xticks(range(2,21))
auc_trainset = []
auc_testset = []
for i in range(2,21):
    cls2 = DecisionTreeClassifier(min_samples_leaf = i)
    cls2.fit(all_features_nom[0:620],target[0:620])
    auc_train = roc_auc_score(target[0:620],cls2.predict_proba(all_features_nom[0:620])[:620,1])
    auc_test =    roc_auc_score(target[620:],cls2.predict_proba(all_features_nom[620:])[:620,1])
    auc_trainset.append(auc_train)
    auc_testset.append(auc_test)
    if auc_test > optimal_auc_test:
        optimal_auc_test = auc_test
        min_samples_leaf = i

#show the answers
print("optimal number of min_samples_leaf is : ",min_samples_leaf)
plt.plot(range(2,21),auc_trainset,'-o',c='red',label ='Auc for training data' )
```

```
plt.plot(range(2,21),auc_testset,'-o',c='blue',label='Auc for test data')
plt.legend()
plt.show()

#probability for P(S=true|G=female,C=1)
#count for people having that condition and count for people surviving under that condition
count_for_condition = 0
count_for_s = 0
for p in range(len(all_features)):
    if all_features[p][0]==1 and all_features[p][1] == 1:
        count_for_condition += 1
        if target[p][0] == 1:
            count_for_s += 1

pro = count_for_s/count_for_condition
print ('The possibility is:',pro)
```