9417 assignment2

Zheng Qiwen z5240149

Question 1:

Part A

This is screenshot of the result table.

图片包含 收据, 文字

描述已自动生成

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:

图片包含 文字

描述已自动生成

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

地图的截图

描述已自动生成

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)