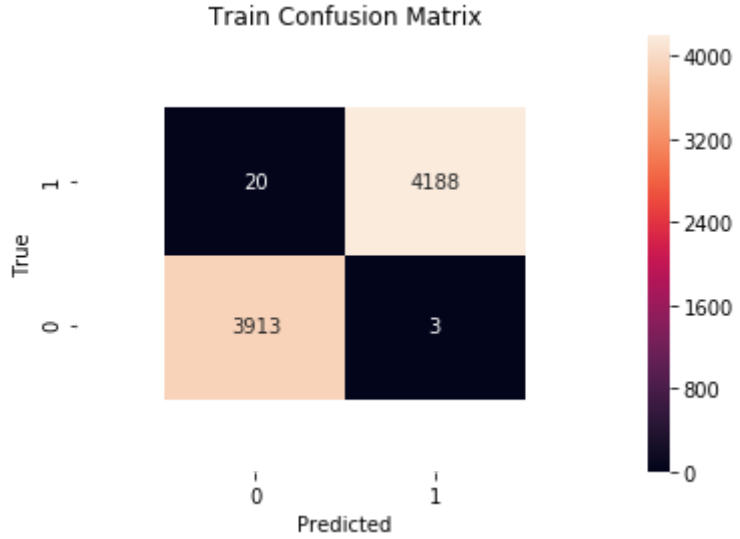
```
#Question 2 script, takes a while to finish
#link to dataset: http://archive.ics.uci.edu/ml/datasets/Mushroom

import pandas as pd
import os
import sklearn.metrics
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn

#MiniProject1 = os.path.abspath('MiniProject1')


#from google.colab import files
#uploaded = files.upload()

!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)


link = 'https://drive.google.com/open?id=1_1_DdYhZ8C2niw9FTtaM-SenVoHPr3Jm'
fluff, id = link.split('=')
print (id) # Verify that you have everything after '='
downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('agaricus-lepiota.data')


myfile = open('agaricus-lepiota.data')
lines = myfile.readlines()
```

```
1_1_DdYhZ8C2niw9FTtaM-SenVoHPr3Jm
```

```
myData = []

for line in lines:
    line = line.strip()
    line = line.split(',')
    myData.append(line)

#Storing data in Dataframe and shuffling rows
myFrame = pd.DataFrame(myData)
np.random.shuffle(myFrame.values)
print('Done storing data, now we train')
y = myFrame[0].copy()
K = 2 #2 classes
py = np.zeros(2,float) #prior vector
classes = ['p','e']




#Training error from training on all data
print("Training on all training data and getting training error")
py[0] = y[y=='p'].count()/y.count() #class 0 is poisonous
py[1] = y[y=='e'].count()/y.count() #class 1 is edible
```

```
yhat = np.zeros((myFrame.shape[0],K),float) #predictions array
alpha = 0 #smoothing term, laplace if 1

for i in range(myFrame.shape[0]):
    for k in range(K):
        px_y = np.zeros(myFrame.shape[1] - 1, float)
        for j in range(px_y.shape[0]):
            px_y[j] = (myFrame[(myFrame[j+1]==myFrame.iloc[i,j+1]) & (myFrame[0]==classes[k]
                       (y[y==classes[k]].count()+((myFrame.shape[1]-1)*alpha))
        yhat[i,k] = np.prod(px_y)*py[k]
        #print(yhat[i,:])
        #print('Sample',i,'is done')
print('Done training, now predicting')
pred = np.argmax(yhat, axis=1)
actuals = y.copy()
actuals[actuals==classes[0]]=0
actuals[actuals==classes[1]]=1
actuals = actuals.to_numpy(dtype=int)

print(pred)

training_error = 1 - np.mean(actuals==pred)
print('Training Error is:',training_error)
cm = sklearn.metrics.confusion_matrix(actuals,pred)
plt.figure()
ax = sn.heatmap(cm, annot=True, fmt='g')
ax.set_xlabel('Predicted')
ax.set_ylabel('True')
ax.set_title('Train Confusion Matrix')
plt.xlim([-0.5,2.5])
plt.ylim([-0.5,2.5])
plt.show()
```

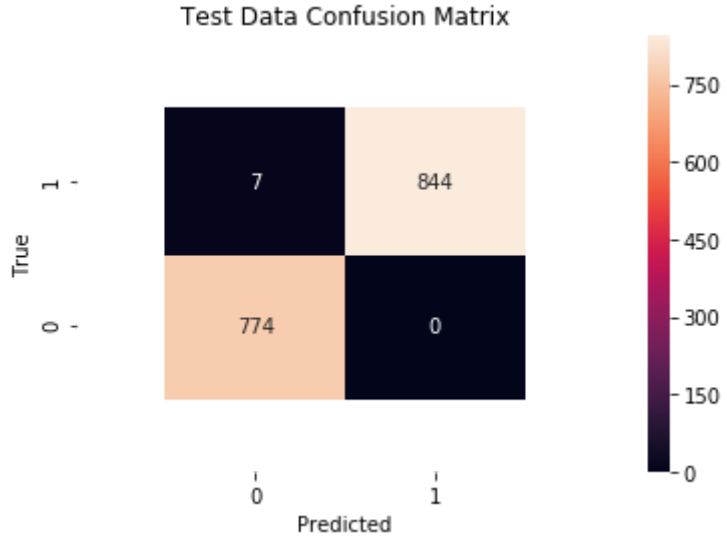
```
Done storing data, now we train
Training on all training data and getting training error
Done training, now predicting
[1 0 0 ... 0 0 0]
Training Error is: 0.0028311176760216217
```




```
#Test split block
print("Training on 80% of data and getting test set error")
test_split = 0.2
X_train = myFrame[:int((1-test_split)*myFrame.shape[0])]
X_test = myFrame[int((1-test_split)*myFrame.shape[0]):]
y_train = y.iloc[:int((1-test_split)*y.shape[0])]
y_test = y.iloc[int((1-test_split)*y.shape[0]):]
```

```
#Getting new priors
py[0] = y_train[y_train=='p'].count()/y_train.count() #class 0 is poisonous
py[1] = y_train[y_train=='e'].count()/y_train.count() #class 1 is edible

#New probabilities matrix
yhat = np.zeros((X_test.shape[0],K),float) #predictions array
alpha = 0 #smoothing term, laplace if 1


for i in range(X_test.shape[0]):
    px_y = np.zeros(X_train.shape[1]-1,float)
    for k in range(K):
        for j in range(px_y.shape[0]):
            px_y[j] = (X_train[(X_train[j+1]==X_test.iloc[i,j+1]) & (X_train[0]==classes[k])
                       (y_train[y_train==classes[k]].count()+((X_test.shape[1]-1)*alpha))
        yhat[i,k] = np.prod(px_y)*py[k]
print('Done training, now predicting')
pred = np.argmax(yhat, axis=1)
actuals = y_test.copy()
actuals[actuals==classes[0]]=0
actuals[actuals==classes[1]]=1

print(pred)
actuals = actuals.to_numpy(dtype=int)

test_error = 1 - np.mean(actuals==pred)
print('Test error is',test_error)

cm = sklearn.metrics.confusion_matrix(actuals,pred)
plt.figure()
ax = sn.heatmap(cm, annot=True, fmt='g')
ax.set_xlabel('Predicted')
ax.set_ylabel('True')
ax.set_title('Test Data Confusion Matrix')
plt.xlim([-0.5,2.5])
plt.ylim([-0.5,2.5])
plt.show()
```

```
Training on 80% of data and getting test set error
Done training, now predicting
[0 1 0 ... 0 0 0]
Test error is 0.004307692307692346
```




```
#Cross Validation Block
print("Training using k fold cross validation and getting error on validation set")
alpha = 0 #smoothing term, laplace if 1
```

```
from sklearn import model_selection
seed = 3421
np.random.seed(seed)
folds = 5
kfold = model_selection.KFold(n_splits=folds,shuffle=True,random_state=seed)
X = myFrame.iloc[:,list(myFrame.columns.values)[1]:].copy()
test_error = np.zeros(folds,dtype=float)
count = 0
preds = [] #probability mushroom is edible for every sample in every validation fold
actuals = [] #actual labels of every mushroom in every validation fold
hats = [] #predicted labels of every mushroom in every validation fold
for train,test in kfold.split(X,y):
    i = 0
    Xtrain = X.iloc[train]
    Xtest = X.iloc[test]
    ytrain = y.iloc[train]
    ytest = y.iloc[test]
    # Getting new priors
    py[0] = ytrain[ytrain == 'p'].count() / ytrain.count()  # class 0 is poisonous
    py[1] = ytrain[ytrain == 'e'].count() / ytrain.count()  # class 1 is edible
    print('Done Splitting, no we train')
    yhat = np.zeros((Xtest.shape[0], K), float)  # predictions array
    #Train
    for index, x in Xtest.iterrows():
        px_y = np.zeros(Xtrain.shape[1], float)
        for k in range(K):
            for j in range(px_y.shape[0]):
                px_y[j] = (Xtrain[(Xtrain[j + 1] == x[j + 1]) & (ytrain == classes[k])][
                               1].count() + alpha) / (
                                     ytrain[ytrain == classes[k]].count() + ((Xtest.shape[1
            yhat[i, k] = np.prod(px_y) * py[k]
        i += 1
    preds.append(yhat[:,1]) #probabilities mushroom is edible
    hat = np.argmax(yhat, axis=1)
    hats.append(hat)
    actual = ytest.copy()
    actual[actual == classes[0]] = 0
    actual[actual == classes[1]] = 1
    actual = actual.to_numpy(dtype=int)
    actuals.append(actual)
    test_error[count] = (1 - np.mean(actual == hat))
    count += 1

#flattening lists out
myactuals = []
for sublist in actuals:
    for item in sublist:
        myactuals.append(item)

mypreds = []
for sublist in preds:
    for item in sublist:
        mypreds.append(item)

myhats = []
for sublist in hats:
    for item in sublist:
        myhats.append(item)
print('Done training, Predictions are: ')
print(myhats)

print('Test error for every fold is',test_error)
print('Test error for all folds is', np.mean(test_error))

#Reporting Summary statistics, Confusion Matrix, and ROC Curve
print("Cross-Validation Accuracy:")
print(sklearn.metrics.accuracy_score(myactuals, myhats))
print(sklearn.metrics.classification_report(myactuals, myhats))

cm = sklearn.metrics.confusion_matrix(myactuals, myhats)
```

```
plt.figure()
ax = sn.heatmap(cm, annot=True, fmt='g')
ax.set_xlabel('Predicted')
ax.set_ylabel('True')
ax.set_title('Cross Validation Confusion Matrix')
plt.xlim ([-0.5,2.5])
plt.ylim([-0.5,2.5])
plt.show()

fpr, tpr, threshold = sklearn.metrics.roc_curve(myactuals, mypreds)
roc_auc = sklearn.metrics.auc(fpr, tpr)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label='AUC = %0.2f' % roc_auc)
plt.legend(loc='lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
Training using k fold cross validation and getting error on validation set
Done Splitting, no we train
Done Splitting, no we train
Done Splitting, no we train
Done Splitting, no we train
Done Splitting, no we train
Done training, Predictions are:
[1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
Test error for every fold is [0.00553846 0.00369231 0.00246154 0.00246154 0.00246
Test error for all folds is 0.0033233800682076708
Cross-Validation Accuracy:
0.9966765140324964
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      3916
           1       1.00      0.99      1.00      4208

    accuracy                           1.00      8124
   macro avg       1.00      1.00      1.00      8124
weighted avg       1.00      1.00      1.00      8124
```