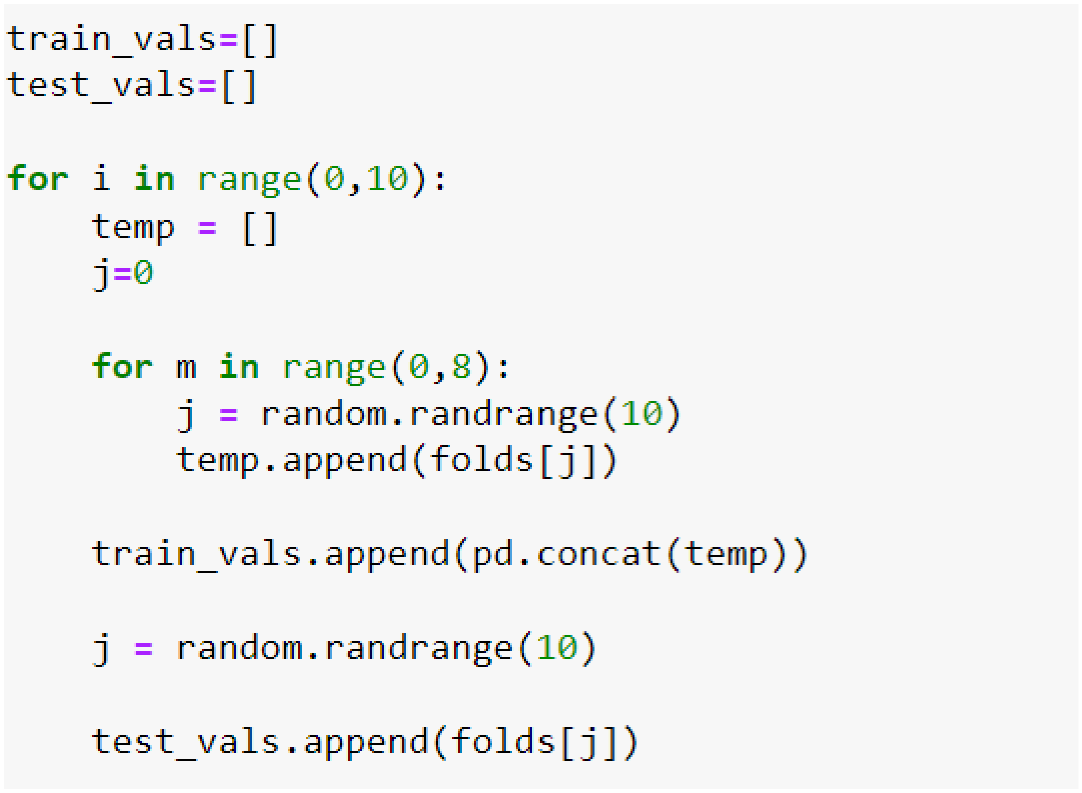
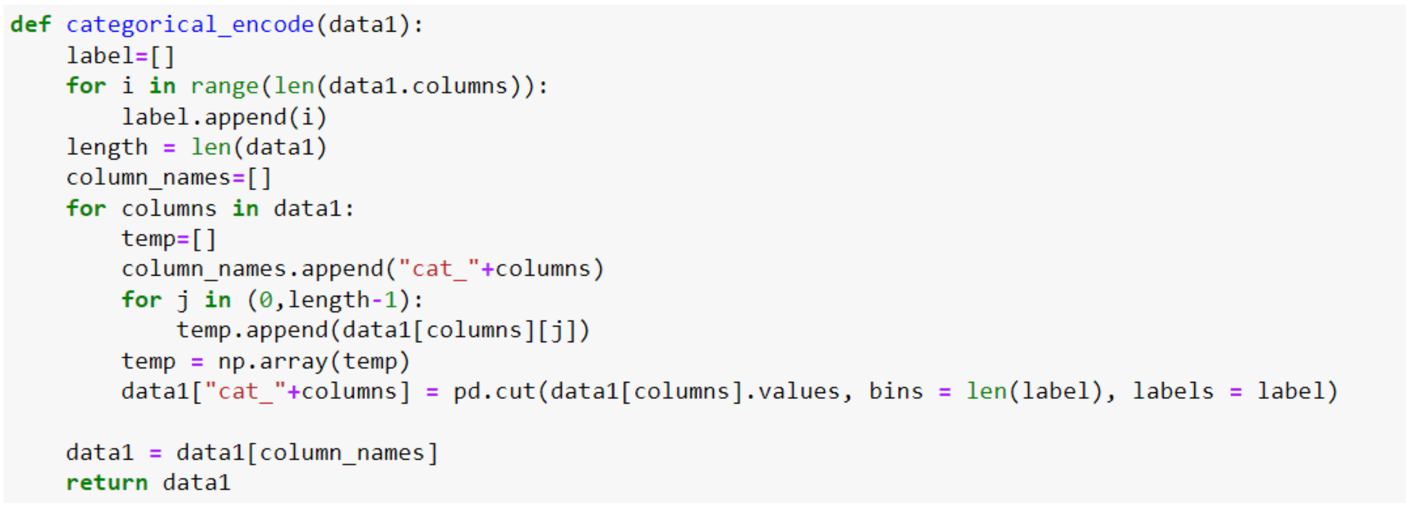
Report of Results of Question 2: Bayesian (Naïve Bayes) Classifier

1) The Train_D_Bayesian dataset has been randomly divided into 80% for training and 20% for testing in the following code snippet.



```
train_vals=[]
test_vals=[]

for i in range(0,10):
    temp = []
    j=0

    for m in range(0,8):
        j = random.randrange(10)
        temp.append(folds[j])

    train_vals.append(pd.concat(temp))

    j = random.randrange(10)

    test_vals.append(folds[j])
```

The categorical variables have been encoded using appropriate encoding here.



```
def categorical_encode(data1):
    label=[]
    for i in range(len(data1.columns)):
        label.append(i)
    length = len(data1)
    column_names=[]
    for columns in data1:
        temp=[]
        column_names.append("cat_"+columns)
        for j in (0,length-1):
            temp.append(data1[columns][j])
        temp = np.array(temp)
        data1["cat_"+columns] = pd.cut(data1[columns].values, bins = len(label), labels = label)

    data1 = data1[column_names]
    return data1
```

2) A feature value is considered as an outlier if its value is greater than standard deviation (mean + 3 × standard deviation). Those rows having 50% or more values of attributes in the outlier region are discarded.

```
means=[]
stds=[]
vars=[]
def remove_outliers(data1):
    for column in data1:
        means.append(mean(data1[column]))
        stds.append(std(data1[column]))
        vars.append(var(data1[column]))
#     print("means ",means)
#     print("std ",stds)

    for i in range(len(data1)):
        count=0
        for j in range(len(data1.columns)-1):

            if(data1.iloc[i,j] >(means[j]+3*stds[j])):
                count+=1
        if(count>len(data1.columns)):
            data1.drop(data1.index[i])
    return data1
```

Among 768 tuples present initially in the dataset,none were removed as outliers.

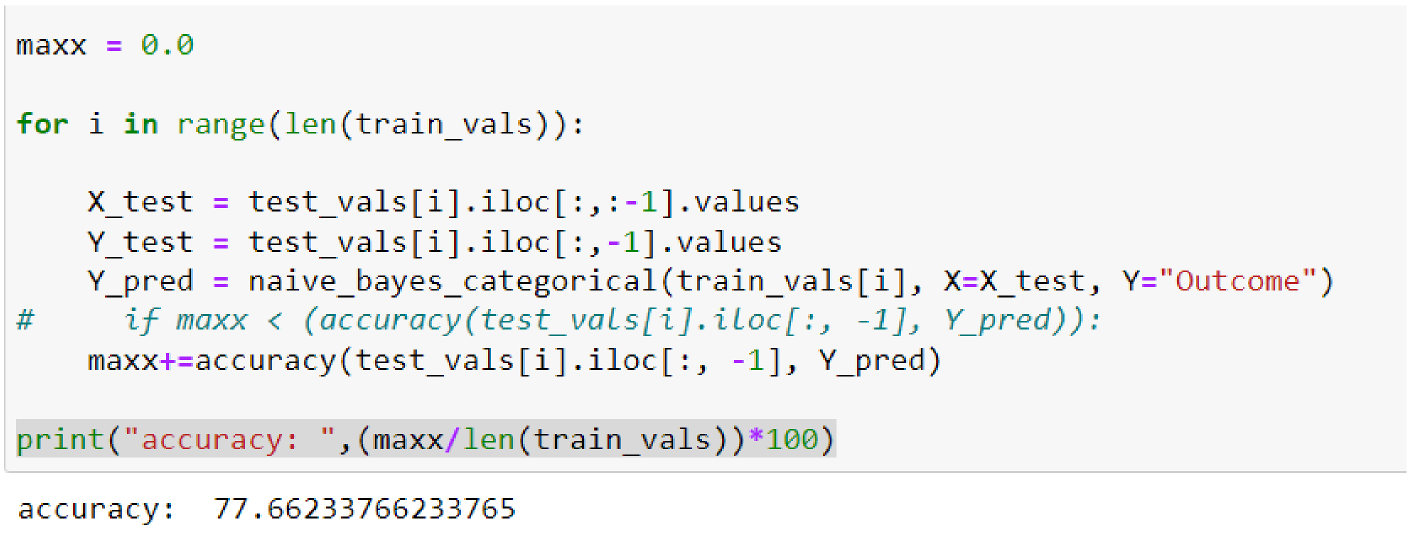
3) Making 10 folds for 10-fold cross validation.

```
def kFold(data)->list:
#     print(floor(len(data)/10))
    k  = floor(len(data)/10)
    folds=[]
    for i in range(0,len(data)//k):
        folds.append(data.loc[i*k:(i+1)*k])
#     print(len(folds))
    return folds
```

This is the Naive Bayes class made from scratch which will be used to train the model.

```
def naive_bayes_categorical(df, X, Y):
    # get feature names
```

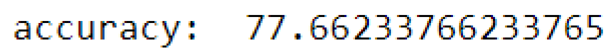
In the following code snippet, Naive Bayes Classifier has been trained & accuracy is printed.



```
maxx = 0.0

for i in range(len(train_vals)):

    X_test = test_vals[i].iloc[:,:-1].values
    Y_test = test_vals[i].iloc[:,-1].values
    Y_pred = naive_bayes_categorical(train_vals[i], X=X_test, Y="Outcome")
#     if maxx < (accuracy(test_vals[i].iloc[:, -1], Y_pred)):
    maxx+=accuracy(test_vals[i].iloc[:, -1], Y_pred)

print("accuracy: ",(maxx/len(train_vals))*100)
```

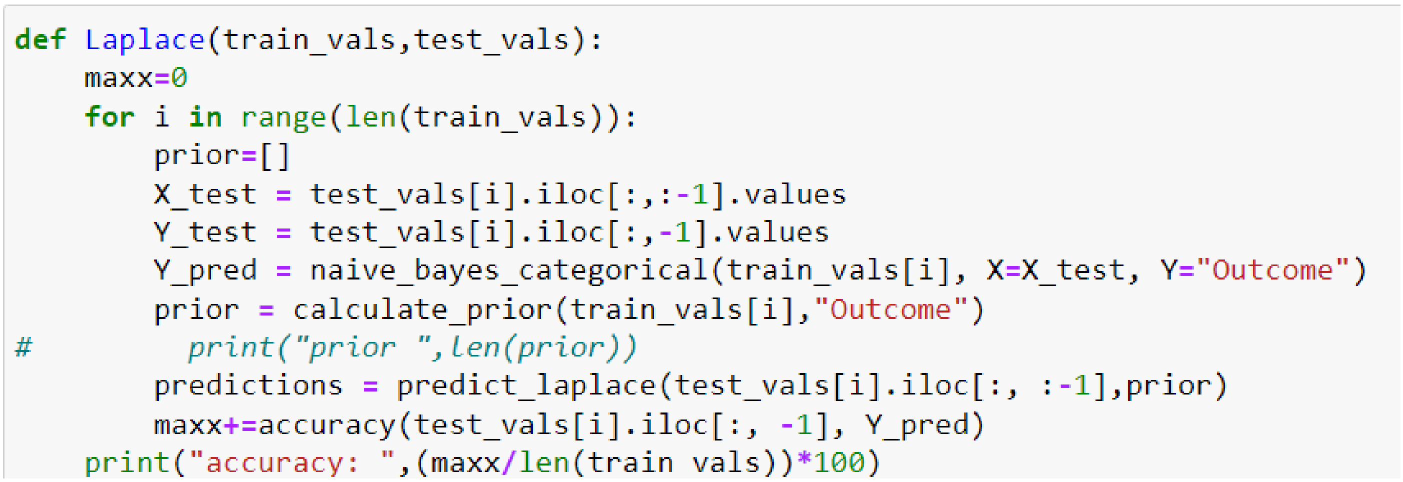
```
accuracy:  77.66233766233765
```

This is the overall accuracy on the test set.



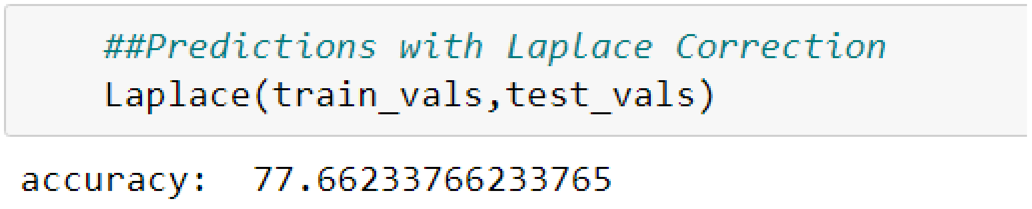
```
accuracy:  77.66233766233765
```

4) The Naïve Bayes Classifier using Laplace correction has been trained in the below code snippet.



```
def Laplace(train_vals,test_vals):
    maxx=0
    for i in range(len(train_vals)):
        prior=[]
        X_test = test_vals[i].iloc[:,:-1].values
        Y_test = test_vals[i].iloc[:,-1].values
        Y_pred = naive_bayes_categorical(train_vals[i], X=X_test, Y="Outcome")
        prior = calculate_prior(train_vals[i],"Outcome")
#         print("prior ",len(prior))
        predictions = predict_laplace(test_vals[i].iloc[:, :-1],prior)
        maxx+=accuracy(test_vals[i].iloc[:, -1], Y_pred)
    print("accuracy: ",(maxx/len(train_vals))*100)
```

This is the final accuracy printed.



```
##Predictions with Laplace Correction
Laplace(train_vals,test_vals)
```

```
accuracy:  77.66233766233765
```