Homework01: Naïve Bayes’ Classifier

Buğra Sipahioğlu

October 12, 2019

Importing Libraries

In [502]:

```
# Import packages
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

Importing Data

In [503]:

```
# Load the data
X = pd.read_csv("hw01_images.csv")
y = pd.read_csv("hw01_labels.csv")
# Substract 1 from each element in y for binary classification
y = y.apply(lambda x : x - 1)
# Split the data for training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, shuffle
=False, stratify=None)
# Transform pandas dataframes to numpy arrays for easy iteration
X_train = X_train.to_numpy()
X_test = X_test.to_numpy()
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()
# Get the number of samples and training data
N = y.shape[0]
num_train_data = y_train.shape[0]
```

Splitting Data

In [504]:

```
# Split the training data with respect to gender (0:female, 1:male).
# Purpose of these division is to execute corresonding R code: sample_means <- s
apply(X = 1:K, FUN = function(c) {mean(x[y == c])})
gender_dict = {} # [0]: holds all female data, [1] holds male
pred_0 = 0
pred_1 = 0
for label in range(num_train_data):
    if y_train[label] == 1:
        if pred_1 == 0:
            gender_dict[1]=X_train[label,:].reshape(X_train[label,:].shape[0],1)
            pred_1 += 1
        else:
            gender_dict[1]= np.append(gender_dict[1],X_train[label,:].reshape(X_
train[label,:].shape[0],1),axis=1)
    else:
        if pred_0 == 0:
            gender_dict[0]=X_train[label,:].reshape(X_train[label,:].shape[0],1)
            pred_0 += 1
        else:
            gender_dict[0]= np.append(gender_dict[0],X_train[label,:].reshape(X_
train[label,:].shape[0],1),axis=1)
```

Parameter Estimation

In [512]:

```
# Calculate means
mean_0 = gender_dict[0].mean(axis=1)
mean_1 = gender_dict[1].mean(axis=1)
print("Mean_0: ", mean_0)
print("Mean_1: ", mean_1)
# Calculate standart deviations
std_0 = gender_dict[0].std(axis=1)
std_1 = gender_dict[1].std(axis=1)
print("Std_0: ", std_0)
print("Std_1: ", std_0)
# Calculate prior probabilities.(1000 is for normalization.)
prior_0 = np.sum(gender_dict[0]) / (1000 * N)
prior_1 = np.sum(gender_dict[1]) / (1000 * N)
print("Prior_0: ", prior_0)
print("Prior_1: ", prior_1)
```

```
Mean_0:  [0.37960784 0.39823529 0.41196078 ... 0.24529412 0.25039216
0.25666667]
Mean_1:  [0.39075474 0.39480776 0.40129258 ... 0.27429072 0.28079746
0.27913244]
Std_0:  [0.14428282 0.14884652 0.16024047 ... 0.18140583 0.18077655
0.17962957]
Std_1:  [0.14428282 0.14884652 0.16024047 ... 0.18140583 0.18077655
0.17962957]
Prior_0:  0.10907705538355693
Prior_1:  0.9371953314659195
```

Parametric Classification

In [514]:

```
# Calculate the likelihood of the features
likelihood_0 = np.log((1 / (np.sqrt(2 * np.pi * np.square(std_0)))) * np.exp(-np.square(X_test - mean_0) / (2 * np.square(std_0))))
likelihood_1 = np.log((1 / (np.sqrt(2 * np.pi * np.square(std_1)))) * np.exp(-np.square(X_test - mean_1) / (2 * np.square(std_1))))
# Calculate the posterior probabilities
joint_prob_0 = np.prod(likelihood_0,axis=1) * (gender_dict[0].shape[0]/X_train.shape[0])
joint_prob_1 = np.prod(likelihood_1,axis=1) * (gender_dict[1].shape[0]/X_train.shape[0])
y_pred=1*(joint_prob_0>joint_prob_1)
```

Confusion Matrix

In [515]:

```
#Calculating the confusion matrix for test
confusion_matrix_test=np.array([[len([i for i in range(0,y_test.shape[0]) if y_test[i]==0 and y_pred[i]==0]),len([i for i in range(0,y_test.shape[0]) if y_test[i]==0 and y_pred[i]==1])]
                                 ,[len([i for i in range(0,y_test.shape[0]) if y_test[i]==1 and y_pred[i]==1]),len([i for i in range(0,y_test.shape[0]) if y_test[i]==1 and y_pred[i]==0])]])
#Calculating the confusion matrix for train
confusion_matrix_train=np.array([[len([i for i in range(0,y_train.shape[0]) if y_train[i]==0 and y_pred[i]==0]),len([i for i in range(0,y_train.shape[0]) if y_train[i]==0 and y_pred[i]==1])]
                                 ,[len([i for i in range(0,y_train.shape[0]) if y_train[i]==1 and y_pred[i]==1]),len([i for i in range(0,y_train.shape[0]) if y_train[i]==1 and y_pred[i]==0])]])
```

In [516]:

```
print(confusion_matrix_test)
```

```
[[ 20   0]
 [ 15 165]]
```

In [517]:

```
print(confusion_matrix_train)
```

```
[[ 20   0]
 [ 15 164]]
```