ENGR 421: Introduction to Machine Learning
Fall 2020 – Homework 3
Ege Yelken – 61742

The aim of this assignment is to implement a Naïve-Bayes classifier. I have followed the steps below:

1. Divided the data set into two by assigning the first 25 image of each class to the training set and remaining 14 images to the test set:

```
# splitting the dataset
train_x1 = imagesdf.iloc[0:25]
train_x2 = imagesdf.iloc[39:64]
train_x3 = imagesdf.iloc[78:103]
train_x4 = imagesdf.iloc[117:142]
train_x5 = imagesdf.iloc[156:181]
train_x = pd.concat([train_x1, train_x2, train_x3, train_x4, train_x5])

test_x1 = imagesdf.iloc[25:39]
test_x2 = imagesdf.iloc[64:78]
test_x3 = imagesdf.iloc[103:117]
test_x4 = imagesdf.iloc[142:156]
test_x5 = imagesdf.iloc[181:195]
test_x = pd.concat([test_x1, test_x2, test_x3, test_x4, test_x5])

train_y1 = labeldf.iloc[0:25]
train_y2 = labeldf.iloc[39:64]
train_y3 = labeldf.iloc[78:103]
train_y4 = labeldf.iloc[117:142]
train_y5 = labeldf.iloc[156:181]
train_y = pd.concat([train_y1, train_y2, train_y3, train_y4, train_y5])

test_y1 = labeldf.iloc[25:39]
test_y2 = labeldf.iloc[64:78]
test_y3 = labeldf.iloc[103:117]
test_y4 = labeldf.iloc[142:156]
test_y5 = labeldf.iloc[181:195]
test_y = pd.concat([test_y1, test_y2, test_y3, test_y4, test_y5])
```

2. Defined the Naïve-Bayes function and estimated the parameters for each class:

```
# Naive-Bayes
def bayes(X, mean, stddev, prior):
    return np.sum(np.log((1/(stddev * np.sqrt(2 * np.pi))) * np.exp(((mean - X)*(X - mean))/(2 * stddev * stddev)))) + np.log(prior)
```

```
# parameter estimation for a
a_means = (np.sum(train_x1, axis = 0)/len(train_y1))

a_stddev = np.sqrt((
        np.sum(train_x1 * train_x1, axis = 0)
        - 2 * np.sum(np.dot(train_x1, np.dot(a_means[:,np.newaxis], np.ones((1, len(a_means)))) * np.identity(len(a_means))), axis = 0)
        + (len(train_y1) * (a_means * a_means)))
    /len(train_y1))

a_prior = len(train_y1)/len(train_y)
```

… and so on for each class.

3. Encoded the letters as integers in order to calculate truth tables:

```
# encoding the letters as integers
train_y = np.where(train_y == 'A', 0, train_y)
train_y = np.where(train_y == 'B', 1, train_y)
train_y = np.where(train_y == 'C', 2, train_y)
train_y = np.where(train_y == 'D', 3, train_y)
train_y = np.where(train_y == 'E', 4, train_y)

test_y = np.where(test_y == 'A', 0, test_y)
test_y = np.where(test_y == 'B', 1, test_y)
test_y = np.where(test_y == 'C', 2, test_y)
test_y = np.where(test_y == 'D', 3, test_y)
test_y = np.where(test_y == 'E', 4, test_y)
```

```
print(confusion_matrix(train_y.tolist(), train_y_hat.tolist()))
```

```
[[25  0  0  0  0]
 [ 3 21  0  1  0]
 [ 0  0 24  0  1]
 [ 0  1  1 23  0]
 [ 1  0  3  0 21]]
```

```
print(confusion_matrix(test_y.tolist(), test_y_hat.tolist()))
```

```
[[13  1  0  0  0]
 [ 2 12  0  0  0]
 [ 0  0 12  0  2]
 [ 1  1  0 12  0]
 [ 0  0  2  0 12]]
```

4. Reshaped the dimensions of parameters to plot them with the **imshow** method:

```
import matplotlib.pyplot as plt

a = np.reshape(a_means.array, (16,20))
b = np.reshape(b_means.array, (16,20))
c = np.reshape(c_means.array, (16,20))
d = np.reshape(d_means.array, (16,20))
e = np.reshape(e_means.array, (16,20))

plt.imshow(a.T, cmap='gray')


plt.imshow(b.T, cmap='gray')


plt.imshow(c.T, cmap='gray')


plt.imshow(d.T, cmap='gray')


plt.imshow(e.T, cmap='gray')
```