This notebook aims to show result of multivariate classification dataset with solving naive-bayes theorem

Dataset contains 400 face images of size 64 pixels × 64 pixels I aim to detect test data is either male or female

import libraries ¶

In [1]:
```
import pandas as pd
import numpy as np
import math
from sklearn.metrics import confusion_matrix
```

Question 2 :

read files

In [2]:
```
imagesdf = pd.read_csv('hw01_images.csv',header=None)
labeldf = pd.read_csv('hw01_labels.csv',header=None)
```

Question 3 :

Divide data into 2 part which are test set and train set

In [3]:
```
train_x = imagesdf.iloc[0:200]
test_x = imagesdf.iloc[-200:]
train_y = labeldf.iloc[0:200]
test_y = labeldf.iloc[-200:]
```

Question 4 :

Get female and male data set from training data

In [4]:
```
female_df = train_x.iloc[train_y[train_y[0]==1].index,:]
male_df = train_x.iloc[train_y[train_y[0]==2].index,:]
```

Get prior probabilities for each class

In [5]:
```
priors = (len(train_y[train_y[0]==1]) / len(train_y),len(train_y[tr
ain_y[0]==2]) / len(train_y))
priors
```

Out[5]: (0.1, 0.9)

Estimate the mean parameters and standart deviation parameters for each pixels

In [6]:
```
means = [female_df.apply(lambda x : np.array(x).mean()), male_df.ap
ply(lambda x : np.array(x).mean())]
deviations = [female_df.apply(lambda x : np.array(x).std()), male_d
f.apply(lambda x : np.array(x).std())]
```

Question 5 :

Naive Bayes Classifier is one of the probabilistic classifier algorithm based on Bayes' Theorem

Naive Bayes with the “naive” assumption of conditional independence between each features which is given the value of the class variable

From Bayes' rule I wrote posterior probability, from there using discriminant function I scored for each class

In [7]:
```
def score_func(x,mu,sigma,prob):
    result = sum((-1/2*np.log(2*math.pi)) - (np.log(sigma)) - ((x-m
u)*(x-mu)/(2*sigma*sigma))) + np.log(prob)
    return result
```

To decide which class more probably I compared both class result and I returned like real value to show in confussion matrix

In [8]:
```
def decide_result(x):
    if score_func(x,means[0],deviations[0],0.1) > score_func(x,means[1],deviations[1],0.9):
        return 1
    else:
        return 2
```

train_y_hat shows prediction result of each image for train data

In [9]:
```
train_y_hat = train_x.apply(lambda x: decide_result(x), axis=1)
```

confussion matrix used to show accurracy of result for train data

In [10]:
```
confusion_matrix(train_y, train_y_hat)
```

Out[10]:
```
array([[ 18,   2],
       [ 24, 156]])
```

Question 6 :

test_y_hat shows prediction result of each image for train data

In [11]:
```
test_y_hat = test_x.apply(lambda x: decide_result(x), axis=1)
```

confussion matrix used to show accurracy of result for test data

In [12]:
```
confusion_matrix(test_y, test_y_hat)
```

Out[12]:
```
array([[ 15,   5],
       [ 19, 161]])
```