In [464]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [465]:
```
#Importing the data
pima=pd.read_csv("diabetes.csv")
```

In [466]:
```
pima
```

Out[466]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

In [467]:
```
#Taking required features for training into a list
features=["Glucose","BloodPressure","SkinThickness","Outcome"]
#Creating a DataFrame with the list features
X_temp=pima[features]
#Spliting the Datasets into 2 parts i.e Training Set and Test Set
X_train,X_test = train_test_split(X_temp, test_size=0.5)
#defining two DataFrames for two Classes A("Outcome"==0) and B("Outcome"==1)
X_trainA=X_train[X_train["Outcome"]==0]
X_trainB=X_train[X_train["Outcome"]==1]
```

In [468]:
```
#calculating the prior probability of the classes
prior_prob_A=X_trainA.shape[0]/(X_trainA.shape[0]+X_trainB.shape[0])
prior_prob_B=X_trainB.shape[0]/(X_trainA.shape[0]+X_trainB.shape[0])
```

In [469]:
```
y_trainA=X_trainA[["Outcome"]]
y_trainB=X_trainB[["Outcome"]]

#Removing Outcome Column Feature from Both DataFrames
X_trainA=X_trainA[["Glucose","BloodPressure","SkinThickness"]]
X_trainB=X_trainB[["Glucose","BloodPressure","SkinThickness"]]
```

In [470]:
```
#function for calculating the mean
def mean(x):
    return sum(x)/x.shape[0]
```

In [471]:
```
#mean of the features of classA
meanVarA1=mean(X_trainA.iloc[:,0])
meanVarA2=mean(X_trainA.iloc[:,1])
meanVarA3=mean(X_trainA.iloc[:,2])
```

In [472]:
```
#mean of the features of classB
meanVarB1=mean(X_trainB.iloc[:,0])
meanVarB2=mean(X_trainB.iloc[:,1])
meanVarB3=mean(X_trainB.iloc[:,2])
```

In [473]:
```
#function which takes "mean" and "covariance" as the parameters and returns th
e likelihood of the Feature Vector
def likelihood(x,mu,co):
    #inverse of the covariance matrix
    inv=np.linalg.inv(co)
    p1=1/(np.sqrt(((2*np.pi)**3)*np.linalg.det(co)))
    p2=np.exp(-0.5*np.dot(np.dot((x-mu).T,inv),(x-mu)))
    p=p1*p2
    return p
```

In [474]:
```
#defining the meanVector which stacked all the means of the taken Features
MeanVectorA=np.array([[meanVarA1,meanVarA2,meanVarA3]])
MeanVectorB=np.array([[meanVarB1,meanVarB2,meanVarB3]])
```

In [475]:
```
#calculating the covariance matrix for both classes(A and B)
covA=np.cov(X_trainA.T)
covB=np.cov(X_trainB.T)
```

In [494]:
```
##Testing one test sample on the classsifier
a=X_test.iloc[381,0:3].to_numpy().reshape(X_test.shape[1]-1,1)
postA=likelihood(a,MeanVectorA.T,covA)*prior_prob_A
postB=likelihood(a,MeanVectorB.T,covB)*prior_prob_B
if(postA<postB):
    print("Class B")
else:
    print("Class A")
```

```
Class A
```

In [477]:
```
#Accuracy of the Total Test Set
predicted_outcome=[]
for k in range(X_test.shape[0]):
    #Changing the test sample from Pandas Series to numpy Array and reshaping
 it
    a=X_test.iloc[k,0:3].to_numpy().reshape(X_test.shape[1]-1,1)
    #Calculating the posterior probabilities
    postA=likelihood(a,MeanVectorA.T,covA)*prior_prob_A
    postB=likelihood(a,MeanVectorB.T,covB)*prior_prob_B
    #Whichever Posterior Probability is more, the test sample is labelled with
the class label
    if(postA<postB):
        predicted_outcome.append(1)
    else:
        predicted_outcome.append(0)
XTest=X_test.copy()
XTest["predictedOutcome"]=predicted_outcome
correct=0
wrong=0
#Checking how many did the classifier correctly labelled
for k in range(XTest.shape[0]):
    if(XTest.iloc[k,3]==XTest.iloc[k,4]):
        correct+=1
    else:
        wrong+=1

print("Total no of correctly predicted values:" +str(correct))
Accuracy=correct/X_test.shape[0]
print("Accuracy: "+str(ratio))
```

```
Total no of correctly predicted values:299
Accuracy: 0.78125
Total no of wrongly predicted values:85
Mean: 0.22135416666666666
```

In [478]:
```
#storing the accuracy of 10 iterations in a list
ListAccuracy=[]
```

In [479]:
```
ListAccuracy.append(Accuracy)
```

In [480]:
```
ListAccuracy
```

Out[480]:
```
[0.7473958333333334,
 0.7369791666666666,
 0.7161458333333334,
 0.7708333333333334,
 0.7369791666666666,
 0.7473958333333334,
 0.75,
 0.7083333333333334,
 0.7395833333333334,
 0.734375,
 0.7786458333333334]
```

In [481]:
```
#mean accuracy
Mean=sum(ListAccuracy)/len(ListAccuracy)
print(Mean)
```

```
0.7424242424242423
```

In [482]:
```
import math
std_dev=0
for p in range(len(ListAccuracy)):
    std_dev+=(ListAccuracy[p]-Mean)**2
std_d=std_dev/(len(ListAccuracy)-1)
std_deviation=math.sqrt(std_d)
```

In [483]:
```
#Standard Deviation of the accuracy
print(std_deviation)
```

```
0.020520256490892612
```