1. The probability that it is Monday and that a student is absent is 4%. Since there are 5 College days in a week, the probability that it is Monday is 25%. What is the probability that a student is absent given that today is Monday? Apply Baye's rule in python to get the result.

In [1]:
```
Probability_of_Monday = 0.25
Probability_of_Monday_and_Student_absent = 0.04

Probability_of_Student_absent_given_Monday =Probability_of_Monday_and_Student_absent/
print("Probability of Student Absent given Monday = " ,Probability_of_Student_absent_
```

```
Probability of Student Absent given Monday =  0.16
```

1. Given the following statistics, what is the probability that a woman has cancer if she has a positive mammogram result?

• One percent of women over 50 have breast cancer.

• Ninety percent of women who have breast cancer test positive on mammograms.

• Eight percent of women will have false positives

In [2]:
```
Prob_M_given_C = 0.01
Prob_C = 0.9
Prob_M_given_notC = 0.08
Prob_notC = 0.99
Prob_C_given_M = (Prob_M_given_C*Prob_C)/((Prob_M_given_C*Prob_C)+ (Prob_M_given_notC
Prob_C_given_M
```

Out[2]:
```
0.10204081632653063
```

- Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering a few test data sets.

In [3]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
df=pd.read_csv("C:\\Users\\Lenovo\\OneDrive\\Desktop\\diabetes.csv")
df
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
0	6	148	72	35	0	33.6	0.627	5
1	1	85	66	29	0	26.6	0.351	3
2	8	183	64	0	0	23.3	0.672	3
3	1	89	66	23	94	28.1	0.167	2

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
4	0	137	40	35	168	43.1	2.288	3
...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	6
764	2	122	70	27	0	36.8	0.340	2
765	5	121	72	23	112	26.2	0.245	3
766	1	126	60	0	0	30.1	0.349	4
767	1	93	70	31	0	30.4	0.315	2

In [5]:
```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [7]:
```
import sklearn
from sklearn.model_selection import train_test_split
```

In [9]:
```
x,y=df.iloc[:,:-1],df.iloc[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

In [10]:
```
x_train
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
603	7	150	78	29	126	35.2	0.692	5
118	4	97	60	23	0	28.2	0.443	2
247	0	165	90	33	680	52.3	0.427	2
157	1	109	56	21	135	25.2	0.833	2
468	8	120	0	0	0	30.0	0.183	3
...	...	...	...	...	...	...	...	

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
763	10	101	76	48	180	32.9	0.171	6
192	7	159	66	0	0	30.4	0.383	3
629	4	94	65	22	0	24.7	0.148	2
559	11	85	74	0	0	30.1	0.300	3
684	5	136	82	0	0	0.0	0.640	6

In [11]:
```
x_test
```

Out[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
661	1	199	76	43	0	42.9	1.394	2
122	2	107	74	30	100	33.6	0.404	2
113	4	76	62	0	0	34.0	0.391	2
14	5	166	72	19	175	25.8	0.587	5
529	0	111	65	0	0	24.6	0.660	3
...	...	...	...	...	...	...	...	
476	2	105	80	45	191	33.7	0.711	2
482	4	85	58	22	49	27.8	0.306	2
230	4	142	86	0	0	44.0	0.645	2
527	3	116	74	15	105	26.3	0.107	2
380	1	107	72	30	82	30.8	0.821	2

154 rows × 8 columns

In [12]:
```
y_train
```

Out[12]:
```
603    1
118    0
247    0
157    0
468    1
      ..
763    0
192    1
629    0
559    0
684    0
Name: Outcome, Length: 614, dtype: int64
```

In [13]:
```
y_test
```

Out[13]:
```
661    1
```

```
122    0
113    0
14     1
529    0
      ..
476    1
482    0
230    1
527    0
380    0
```

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [17]:
```
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [18]:
```
x_train
```

Out[18]:
```
array([[ 0.90832902,  0.91569367,  0.44912368, ...,  0.37852648,
         0.67740401,  1.69955804],
       [ 0.03644676, -0.75182191, -0.47230103, ..., -0.50667229,
        -0.07049698, -0.96569189],
       [-1.12606292,  1.38763205,  1.06340683, ...,  2.54094063,
        -0.11855487, -0.88240283],
       ...,
       [ 0.03644676, -0.84620959, -0.21634972, ..., -0.94927168,
        -0.95656442, -1.04898095],
       [ 2.0708387 , -1.12937261,  0.24436264, ..., -0.26640405,
        -0.50001442,  0.11706589],
       [ 0.32707418,  0.47521786,  0.65388473, ..., -4.07275877,
         0.52121586,  2.94889395]])
```

In [19]:
```
x_test
```

Out[19]:
```
array([[-0.8354355 ,  2.45735903,  0.34674316, ...,  1.35224513,
         2.78594417, -0.96569189],
       [-0.54480808, -0.43719633,  0.24436264, ...,  0.17619533,
        -0.1876381 , -0.88240283],
       [ 0.03644676, -1.41253563, -0.36992051, ...,  0.22677812,
        -0.22668514, -0.71582471],
       ...,
       [ 0.03644676,  0.66399321,  0.85864578, ...,  1.4913478 ,
         0.53623395, -0.96569189],
       [-0.25418066, -0.15403331,  0.24436264, ..., -0.74694053,
        -1.07971278, -0.79911377],
       [-0.8354355 , -0.43719633,  0.14198211, ..., -0.17788417,
         1.06487079, -0.79911377]])
```

In [20]:
```
from sklearn.naive_bayes import GaussianNB
```

In [21]:
```
classifier=GaussianNB()
classifier.fit(x_train,y_train)
```

Out[21]:
```
GaussianNB()
```

In [23]:
```
### prediction

y_pred=classifier.predict(x_test)
y_pred
```

Out[23]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [26]:
```
### accuracy

from sklearn.metrics import confusion_matrix,accuracy_score
```

In [30]:
```
confusion_matrix=confusion_matrix(y_test,y_pred)
confusion_matrix
```

Out[30]:
```
array([[93, 14],
       [18, 29]], dtype=int64)
```

In [29]:
```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Out[29]:
```
0.7922077922077922
```

In [ ]:
```

```