In [ ]:

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [ ]:

```
df = pd.read_csv('diabetes.csv')
df.head()
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [ ]:

```
temp = df.groupby("Outcome").size()
temp
```

Out[ ]:

```
Outcome
0    500
1    268
dtype: int64
```

In [ ]:

```
y = df['Outcome']

# drop the col 'outcome'
x = df.drop(['Outcome'],axis=1)
```

In [ ]:

```
# Split data into train & test
x_train, x_test, y_train, y_test = train_test_split(x, y, stratify=y, random_state=42)
```

In [ ]:

```
x_train.describe()
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	576.000000	576.000000	576.000000	576.000000	576.000000	576.000000	576.000000	576.00000C
mean	3.831597	120.767361	69.170139	20.723958	77.899306	32.064583	0.480200	33.53645ε
std	3.312864	31.771380	18.699887	15.877307	107.415003	7.861032	0.333188	11.87875ε
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.084000	21.00000C
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.600000	0.245750	24.00000C
50%	3.000000	116.500000	72.000000	23.000000	40.000000	32.400000	0.384000	30.00000C
75%	6.000000	141.000000	80.000000	32.000000	129.250000	36.525000	0.646250	41.00000C

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
max	17.000000	199.000000	122.000000	99.000000	744.000000	67.100000	2.329000	81.000000

In [ ]:

```
x_test.describe()
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	192.000000	192.000000	192.000000	192.000000	192.000000	192.000000	192.000000	192.000000
mean	3.885417	121.276042	68.911458	19.973958	85.500000	31.776562	0.446906	32.354167
std	3.542915	32.650006	21.253333	16.203689	136.216758	7.969892	0.325265	11.381513
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	64.000000	0.000000	0.000000	26.075000	0.237000	23.000000
50%	3.000000	119.500000	72.000000	22.000000	0.000000	31.600000	0.343000	28.000000
75%	6.000000	140.000000	80.000000	32.000000	120.000000	36.600000	0.563500	39.000000
max	15.000000	197.000000	106.000000	63.000000	846.000000	59.400000	2.420000	68.000000

In [ ]:

```
train_mean_pos = x_train[y_train==1].mean()
train_std_pos = x_train[y_train==1].std()
train_mean_neg = x_train[y_train==0].mean()
train_std_neg = x_train[y_train==0].std()
```

In [ ]:

```
from math import sqrt
from math import pi
from math import exp

# formula of Gausian NB
def cond_probability(x, mean, std):
    exponent = exp(-((x - mean)**2/(2*std**2)))
    return (1 / (sqrt(2*pi)*std)) * exponent
```

In [ ]:

```
def predict(row):
    prob_pos = len(x_train[y_train==1]) / len(x_train)

    for i in range(0,len(row)):
        prob_pos = prob_pos * cond_probability(row[i],train_mean_pos[i],train_std_pos[i]
)

    prob_neg = len(x_train[y_train==0]) / len(x_train)

    for i in range(0,len(row)):
        prob_neg = prob_neg * cond_probability(row[i],train_mean_neg[i],train_std_neg[i]
)

    return [prob_pos,prob_neg]
```

In [ ]:

```
predictions_raw = []

for row in x_test.values.tolist():
    predictions_raw.append(predict(row))
```

In [ ]:

```
predictions_raw[0]
```

```
predictions_raw[0]
```

Out[ ]:

```
[1.6299028206157718e-14, 1.0044068228290291e-14]
```

In [ ]:

```
predictions_raw
```

Out[ ]:

```
[[1.6299028206157718e-14, 1.0044068228290291e-14],
 [4.1416091656376674e-13, 1.3042909855968897e-12],
 [1.1228918162921918e-13, 3.698652043170301e-12],
 [4.27008488096741e-13, 7.034723756517606e-13],
 [7.913590093643907e-14, 1.281328360925972e-15],
 [6.03509035365965e-14, 1.390772897035608e-12],
 [1.6545496022765486e-13, 8.00346760473016e-13],
 [3.1403211509634367e-17, 4.768433856559617e-16],
 [1.270299259926624e-13, 1.1016259499091278e-13],
 [2.1748571686748456e-14, 2.1739905042451717e-12],
 [4.971297069213275e-13, 4.668622177374758e-12],
 [4.5137523313178456e-14, 2.6278537816854723e-15],
 [5.544956140320912e-14, 4.4363696298950895e-13],
 [8.398499050734631e-14, 2.789658783732221e-12],
 [2.3296901261242477e-14, 1.430621412993164e-14],
 [1.5705424247395808e-16, 2.391268492201095e-18],
 [1.3372550935227474e-16, 8.091703856113028e-17],
 [3.7850981610406474e-14, 4.424312795466002e-13],
 [5.1879097868291456e-18, 8.258115874425753e-17],
 [1.2095736900610271e-13, 4.805126942734823e-13],
 [9.658216711483473e-15, 6.459669635048539e-13],
 [1.183122395213394e-14, 9.596716117647106e-14],
 [6.764511366588288e-14, 3.615779409361753e-12],
 [1.755577077926644e-13, 3.001997377913335e-14],
 [9.397709330573138e-15, 9.325669412062575e-17],
 [1.118137006075826e-14, 3.341977046168377e-16],
 [7.687344900125402e-14, 1.5324955999168888e-12],
 [4.6537294420847004e-14, 2.1821689009236796e-12],
 [2.8420836238274412e-14, 1.3687460385561877e-16],
 [1.0336716098079303e-14, 5.724349865277855e-16],
 [3.064263619029307e-14, 1.834317232515579e-14],
 [7.740869466236125e-22, 8.576082902615116e-20],
 [3.2799869031042725e-14, 1.434624604023498e-12],
 [3.3583316981093224e-15, 8.074554543102154e-16],
 [1.096277615483422e-15, 1.6209337376166807e-17],
 [1.265061920949801e-13, 2.7700385695182056e-12],
 [6.292688660660292e-13, 3.2138378418203097e-12],
 [3.9629147975623396e-14, 2.645364830925672e-13],
 [1.1577540063741118e-13, 1.990808675200153e-12],
 [6.142034970748263e-14, 3.067051385463074e-12],
 [8.4583798534465e-14, 2.4626376484433194e-14],
 [2.4508208415499863e-15, 2.479525060007909e-13],
 [7.711713709363677e-14, 1.7602849122722366e-12],
 [8.110012598919184e-14, 3.946218483160915e-14],
 [2.2521228861482883e-16, 3.6750051465924380e-17],
 [1.5899868755432368e-14, 7.612226772925874e-15],
 [2.792993372970847e-13, 2.3526306593919896e-14],
 [2.827946983429931e-14, 2.0880790640945582e-13],
 [1.2995520917319937e-14, 1.5868715069105584e-15],
 [5.669107734189506e-23, 9.600589771404165e-28],
 [8.328226357474412e-14, 3.3605345306489253e-13],
 [2.556944251939017e-15, 7.104161234074214e-16],
 [3.63981659884254e-13, 7.957141827022834e-13],
 [2.6488598289024356e-16, 6.2139946709797650e-16],
 [1.101020286316147e-14, 3.8489365673927420e-15],
 [3.4525035812643664e-14, 1.9768802759399186e-14],
 [5.981916266506276e-14, 6.785146920457135e-15],
 [1.6399845908843516e-13, 2.0510188018021236e-12],
 [2.8091747715051665e-13, 6.30795582310155e-13],
 [5.368485633424055e-14, 1.335632029669294e-13],
 [8.429351643682075e-15, 3.1119544413721364e-17],
 [3.2354573296143167e-14, 2.7383234285842393e-14]
```

```
[3.2334373296143167e-14, 2.7385234285842393e-14],
[1.3851705509777416e-14, 1.2440865602702788e-15],
[4.259537871652103e-14, 2.4548275594602237e-12],
[6.416732316788837e-14, 8.051830948116969e-13],
[3.3867607083093192e-21, 4.042711648928659e-32],
[4.212503091543637e-15, 1.4265475587949826e-13],
[6.831144165539682e-17, 4.177396424184603e-16],
[5.5024319084909555e-14, 1.5695840120665226e-12],
[4.026641243811283e-14, 8.21414521521943e-13],
[1.91645218659486e-15, 2.3260188708681967e-15],
[6.061736344523169e-13, 6.387338940412256e-13],
[2.2391253502371933e-16, 1.9693100065003145e-17],
[1.8740931578745582e-14, 1.3250160396486317e-14],
[2.3384439998284034e-18, 5.351514617548195e-17],
[4.70011178198929e-14, 2.701230907525572e-12],
[2.5345681491359323e-15, 1.4730937036178274e-14],
[4.166858069566878e-15, 5.3041085159926355e-16],
[7.127570554914743e-15, 8.156113978743444e-13],
[6.035614453053104e-14, 2.5778742702429068e-12],
[3.207658029431876e-13, 1.2645180647592308e-12],
[1.2743108376163112e-13, 2.5662425369989746e-14],
[4.2118255584078704e-14, 2.951301978498283e-12],
[2.0684342542150025e-15, 3.6968891418535217e-13],
[6.489200461348366e-15, 6.160490118091317e-17],
[1.5317198654004198e-13, 6.583891387462631e-13],
[2.0928058204564963e-13, 1.104865321174777e-12],
[5.4432103974679863e-14, 6.34114241891859e-13],
[9.339533252696141e-14, 9.830603130590594e-13],
[6.888052527212437e-14, 1.0312676269910467e-12],
[1.5331635145391462e-14, 1.6481988392036621e-13],
[1.6732883964420244e-13, 9.887143658291459e-15],
[1.6393388287184174e-14, 1.2883900743149129e-12],
[2.584684115004588e-14, 2.143065992988467e-12],
[1.0779608081465417e-13, 7.142773159162612e-13],
[5.817892856603071e-15, 1.3043233256823392e-13],
[3.66167014782506e-14, 2.102071965130574e-13],
[1.0983249036126792e-13, 3.141850779971701e-12],
[1.1729105618454695e-14, 1.4201890196806946e-12],
[1.0646331475145844e-13, 3.4153895581833534e-13],
[2.395145311031316e-13, 2.1983375746935341e-13],
[5.776648970792973e-14, 9.338583135225629e-14],
[1.9405916243374873e-14, 1.2342043499851908e-14],
[7.932190905637278e-14, 5.020571547242019e-13],
[5.002001134379249e-14, 4.0880118828508585e-15],
[1.153052716698971e-13, 7.151429954958447e-13],
[3.0831355112386724e-13, 2.8586548355828566e-13],
[1.3286647876063532e-14, 7.754483346050318e-15],
[4.866401302135483e-14, 2.792212844778245e-13],
[3.217868366597603e-13, 1.3515794921474683e-12],
[5.1497953781472836e-14, 4.722152947682009e-13],
[1.2848649459286725e-14, 4.345183504437752e-14],
[2.3469049330730658e-14, 2.4232359331867946e-13],
[2.5308287439148835e-15, 1.9052472885201613e-18],
[1.3771313539266821e-14, 4.497986803260288e-16],
[1.9642222577851796e-14, 1.1547842413253717e-12],
[7.785651009225734e-15, 2.231721551168261e-15],
[1.990630396369431e-14, 1.750857485336168e-14],
[8.053600503650073e-14, 7.725260716755345e-13],
[7.875636054088754e-14, 1.1293812717363344e-14],
[6.143303512732002e-15, 6.116136481000171e-16],
[3.0680291611991428e-15, 1.7483623807312887e-17],
[5.105038035489349e-14, 1.0518021583407042e-13],
[2.9847301503831337e-14, 5.154485725297729e-13],
[3.6415467160476134e-22, 4.5554260827002255e-20],
[2.7166847705976433e-17, 2.7219009923958426e-16],
[1.9981845337489943e-17, 2.376341447232758e-16],
[1.508735078893333e-13, 6.157755920578688e-14],
[4.895752240954972e-15, 4.237978225225542e-15],
[3.7888456048999625e-14, 8.364517556326407e-13],
[9.926633846323988e-14, 3.65382232001222e-14],
[1.797639334658097e-14, 2.5504754101837593e-13],
[1.241874026419547e-14, 5.503768041317292e-15],
[1.1980756283450628e-13, 9.64479641790301e-13],
```

```
 [1.1980756283450628e-13, 9.644796417903501e-13],
 [1.7708084780294788e-13, 3.3583131059136166e-12],
 [2.115896108790813e-14, 1.7381568317891308e-12],
 [4.090349148102388e-14, 3.824553263211296e-13],
 [3.1968613975242464e-14, 1.1782188241079254e-12],
 [1.3183753554735355e-14, 1.1458436505899704e-16],
 [1.8357259734865713e-14, 1.148409890928165e-13],
 [9.988517629441808e-14, 2.0998733808711405e-12],
 [1.2330675504822156e-14, 5.019265296952992e-14],
 [3.598355357631848e-15, 2.1837912234269432e-15],
 [1.3761133140018997e-13, 1.3467851129196915e-12],
 [1.0351136739147127e-13, 2.345199115084973e-12],
 [4.985642063097899e-14, 3.5134565246213276e-12],
 [9.056787760625072e-14, 4.1024107951429705e-12],
 [2.582185638463316e-16, 5.770714610362788e-16],
 [2.8320147094694145e-15, 1.2379451152064554e-17],
 [8.258942940573513e-15, 2.5697926262755313e-14],
 [1.2006264133096996e-13, 5.342296863238633e-13],
 [1.6298538322600847e-15, 5.352983894002191e-18],
 [1.4932815553031663e-15, 1.1408908794266965e-18],
 [1.727207222322128e-14, 7.541970502364328e-14],
 [5.565908928026911e-14, 1.1380758527624086e-13],
 [1.890137373655673e-13, 1.6223194961993878e-12],
 [4.28006217462737e-14, 6.160576555442284e-13],
 [1.975258136505389e-13, 4.251931176338048e-12],
 [3.357711035556376e-15, 2.461367810571762e-13],
 [1.1353180726815441e-13, 2.0568692061149883e-12],
 [2.0519716422775245e-13, 2.963505901370027e-12],
 [5.747606307524705e-14, 3.6773514897839404e-14],
 [1.9466627023399043e-14, 1.5395093970105935e-12],
 [8.081746890569204e-14, 2.124316290544894e-14],
 [4.1079438863185537e-16, 5.119724139667534e-16],
 [3.911006361404436e-14, 1.5582184992074597e-12],
 [8.78665871943998e-14, 8.75790260109455e-14],
 [2.536897877683412e-14, 2.5952060266757175e-14],
 [3.550939507318185e-14, 1.6208594206357978e-15],
 [5.1289153409172587e-14, 9.88223547611495e-13],
 [4.5792180372365475e-15, 1.1173744960618269e-13],
 [4.6947369410023145e-14, 5.566506532480072e-15],
 [7.167068855510236e-15, 5.716957207774471e-13],
 [1.8783597154740783e-13, 9.079971302883305e-14],
 [2.2065082672291248e-14, 5.67396493751096e-14],
 [1.9870978425869938e-13, 2.7943928750798352e-14],
 [1.0405112268231946e-13, 1.6977366392150349e-12],
 [4.90024366364634e-19, 5.760234822850598e-26],
 [3.8046047838478014e-14, 4.210960751502484e-14],
 [1.1993802708430058e-13, 4.857283972222239e-15],
 [2.03392325968567e-14, 2.01101544323733e-12],
 [2.4620653963053718e-14, 1.898268343912023e-12],
 [2.0991591187204965e-13, 6.8108847408759e-13],
 [1.658863798328311e-16, 3.951576698867016e-21],
 [8.986182210760186e-14, 2.37017542363246e-14],
 [1.6602501172134693e-13, 3.0481234050189e-15],
 [3.4767024186716364e-14, 2.6471743226274595e-16],
 [1.9360205100997485e-15, 4.387900015455899e-14],
 [1.790043901215446e-13, 2.146384119355955e-12],
 [6.398193157862553e-14, 4.91843751683579e-13],
 [2.3383869028705516e-13, 1.1174062493286258e-12],
 [4.801342422000779e-13, 2.0040276229540157e-12]]
```

In [ ]:

```
predictions = []
for row in predictions_raw:
    if(row[0]>row[1]):
        predictions.append(1)
    else:
        predictions.append(0)
```

In [ ]:

```
# comparing our predictions and actual output for accuracy
```

```
accuracy_score(y_test.tolist(),predictions)
```

Out[ ]:

```
0.7135416666666666
```

In [ ]:

```
# plotting the output for comparison
confusion_matrix(y_test.tolist(),predictions)
```

Out[ ]:

```
array([[96, 29],
       [26, 41]])
```

In [ ]:

```
model = GaussianNB()   # Gaussian NB has been used becuase the data is continuous
model.fit(x_train,y_train)
```

Out[ ]:

```
GaussianNB()
```

In [ ]:

```
confusion_matrix(y_test,model.predict(x_test))
```

Out[ ]:

```
array([[96, 29],
       [26, 41]])
```