```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

dataset = pd.read_csv("diabetes.csv")
dataset

     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0              6      148             72             35        0  33.6

1              1       85             66             29        0  26.6

2              8      183             64              0        0  23.3

3              1       89             66             23       94  28.1

4              0      137             40             35      168  43.1

..           ...      ...            ...            ...      ...   ...

763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1

767            1       93             70             31        0  30.4


     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
dataset.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

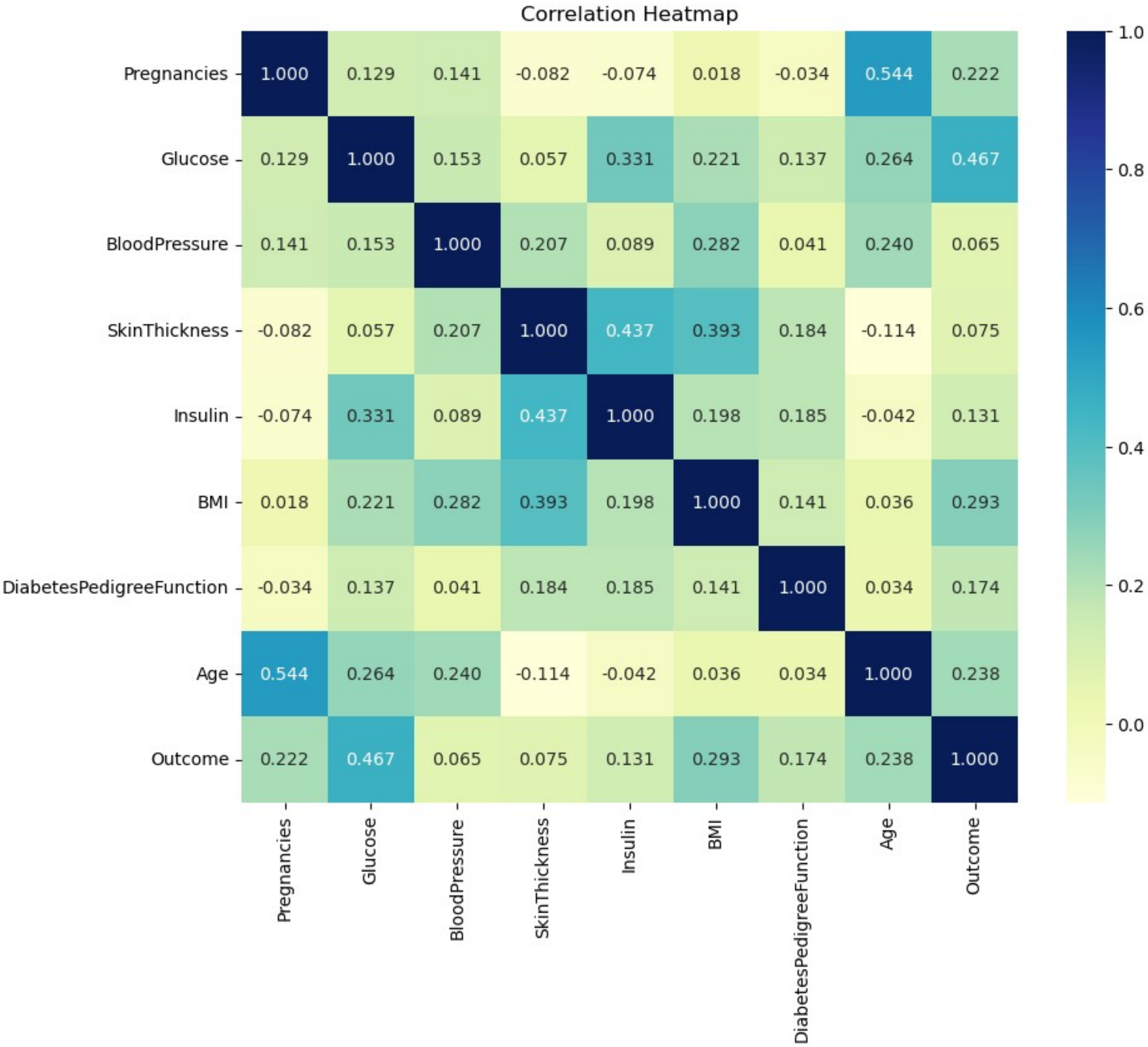
```
dataset.describe()
```

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

```
              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

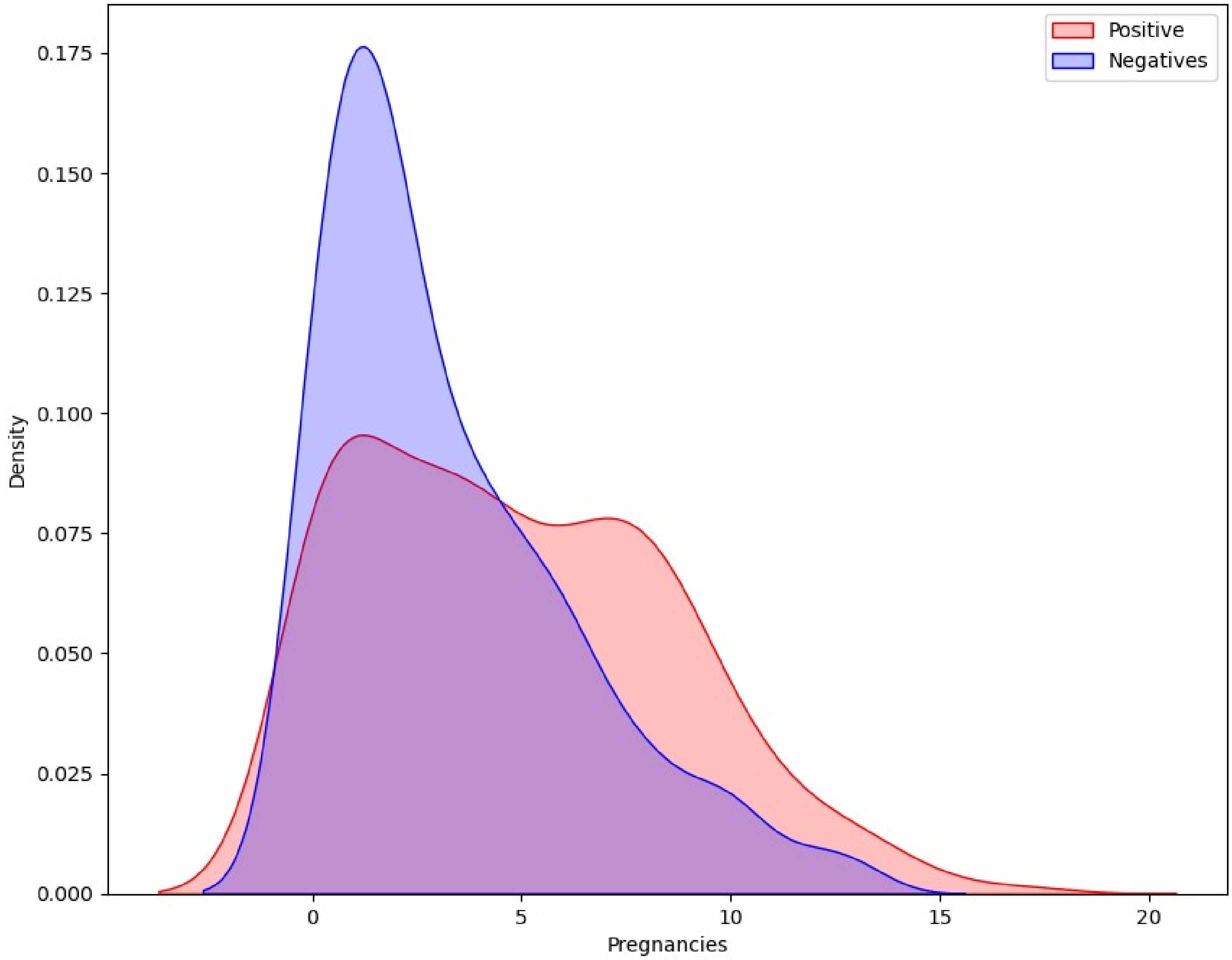
```
plt.figure(figsize=(10,8))
sns.heatmap(dataset.corr(),annot = True, fmt=".3f",cmap="YlGnBu")
plt.title("Correlation Heatmap")
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



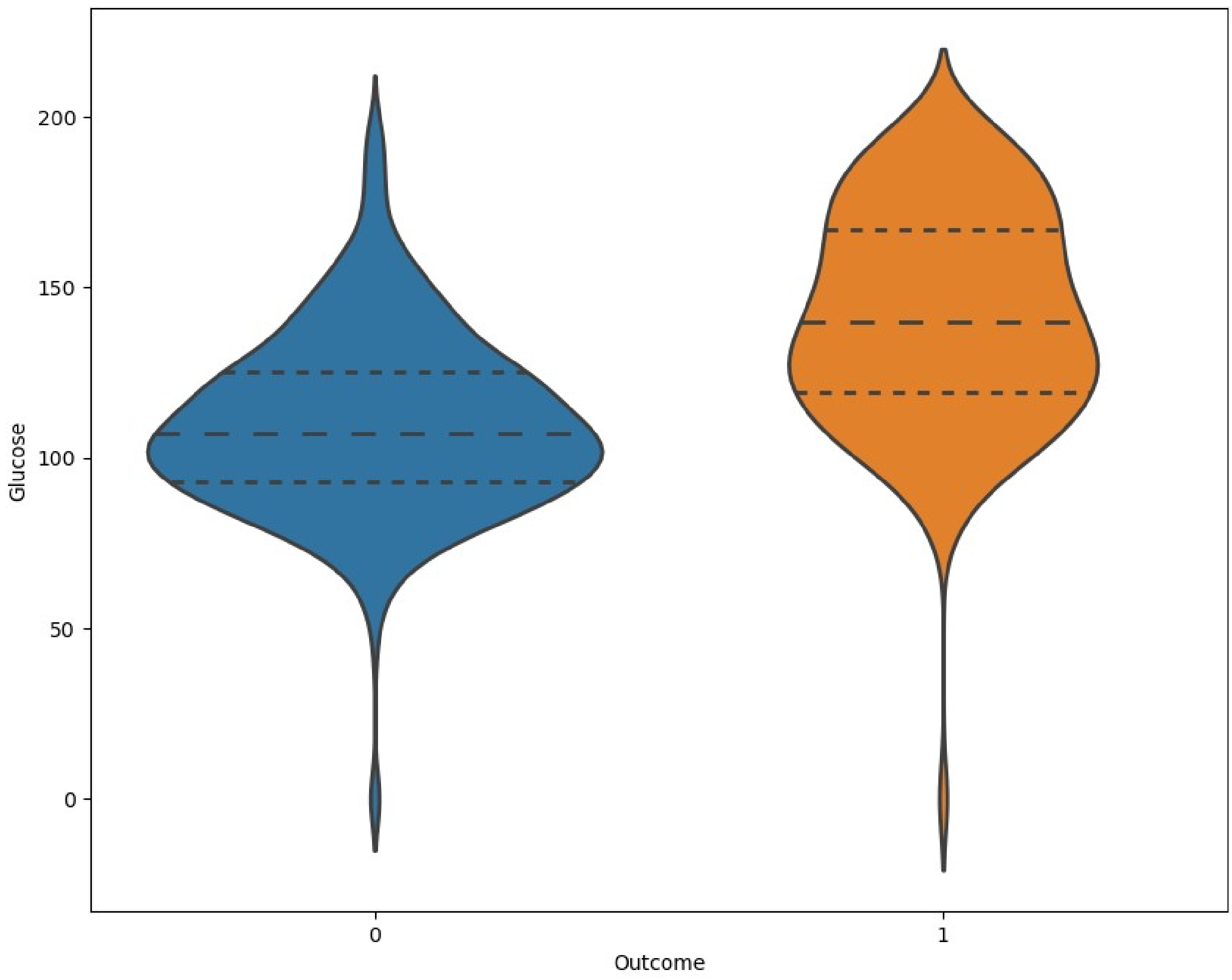
```
plt.figure(figsize=(10,8))
kde = sns.kdeplot(dataset["Pregnancies"]
[dataset["Outcome"]==1],color="red",fill=True)
kde = sns.kdeplot(dataset["Pregnancies"]
[dataset["Outcome"]==0],color="blue",fill=True)
kde.set_xlabel("Pregnancies")
kde.set_ylabel("Density")
kde.legend(["Positive","Negatives"])
```

```
<matplotlib.legend.Legend at 0x162d705a5c0>
```

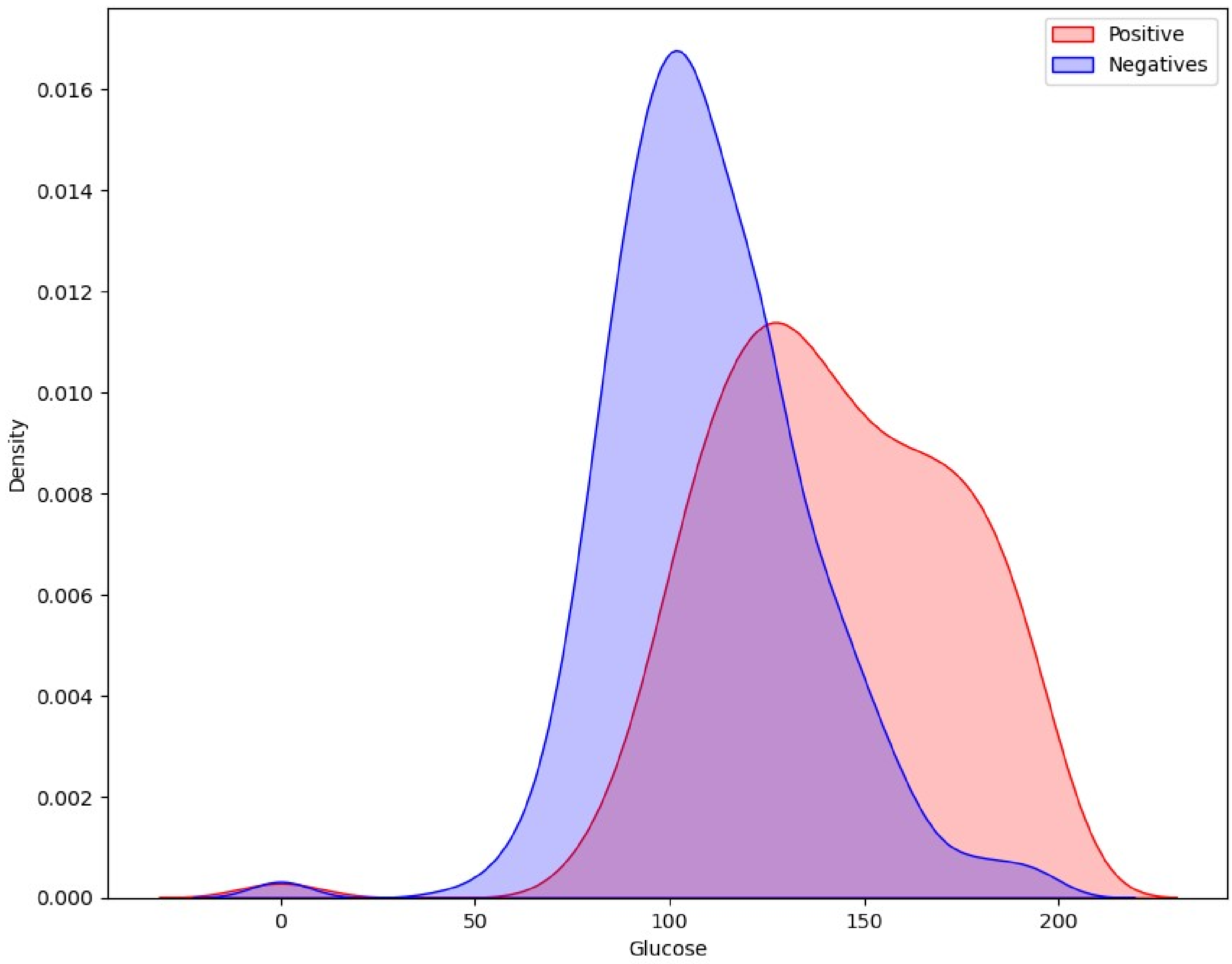


```
plt.figure(figsize=(10,8))
sns.violinplot(data=dataset,x="Outcome",y="Glucose",split=True,linewid
th=2,inner="quart")
```

```
<Axes: xlabel='Outcome', ylabel='Glucose'>
```



```
plt.figure(figsize=(10,8))
kde = sns.kdeplot(dataset["Glucose"]
[dataset["Outcome"]==1],color="red",fill=True)
kde = sns.kdeplot(dataset["Glucose"]
[dataset["Outcome"]==0],color="blue",fill=True)
kde.set_xlabel("Glucose")
kde.set_ylabel("Density")
kde.legend(["Positive","Negatives"])

<matplotlib.legend.Legend at 0x162d7200310>
```



```
dataset["Glucose"]=dataset["Glucose"].replace(0,dataset["Glucose"].med
ian())
dataset["BloodPressure"]=dataset["BloodPressure"].replace(0,dataset["B
loodPressure"].median())
dataset["BMI"]=dataset["BMI"].replace(0,dataset["BMI"].mean())
dataset["SkinThickness"]=dataset["SkinThickness"].replace(0,dataset["S
kinThickness"].mean())
dataset["Insulin"]=dataset["Insulin"].replace(0,dataset["Insulin"].mea
n())

dataset

     Pregnancies  Glucose  BloodPressure  SkinThickness    Insulin
BMI  \
0              6      148             72      35.000000  79.799479
33.6
1              1       85             66      29.000000  79.799479
26.6
2              8      183             64      20.536458  79.799479
23.3
```

```
3              1       89             66      23.000000   94.000000
28.1
4              0      137             40      35.000000  168.000000
43.1
..           ...      ...            ...            ...         ...
...
763           10      101             76      48.000000  180.000000
32.9
764            2      122             70      27.000000   79.799479
36.8
765            5      121             72      23.000000  112.000000
26.2
766            1      126             60      20.536458   79.799479
30.1
767            1       93             70      31.000000   79.799479
30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

```
X=dataset.drop(["Outcome"],axis=1)
Y=dataset["Outcome"]
```

```
X
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin
BMI  \
0              6      148             72      35.000000   79.799479
33.6
1              1       85             66      29.000000   79.799479
26.6
2              8      183             64      20.536458   79.799479
23.3
3              1       89             66      23.000000   94.000000
28.1
4              0      137             40      35.000000  168.000000
43.1
..           ...      ...            ...            ...         ...
```

```
...
763           10      101             76      48.000000  180.000000
32.9
764            2      122             70      27.000000   79.799479
36.8
765            5      121             72      23.000000  112.000000
26.2
766            1      126             60      20.536458   79.799479
30.1
767            1       93             70      31.000000   79.799479
30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

```
Y
```

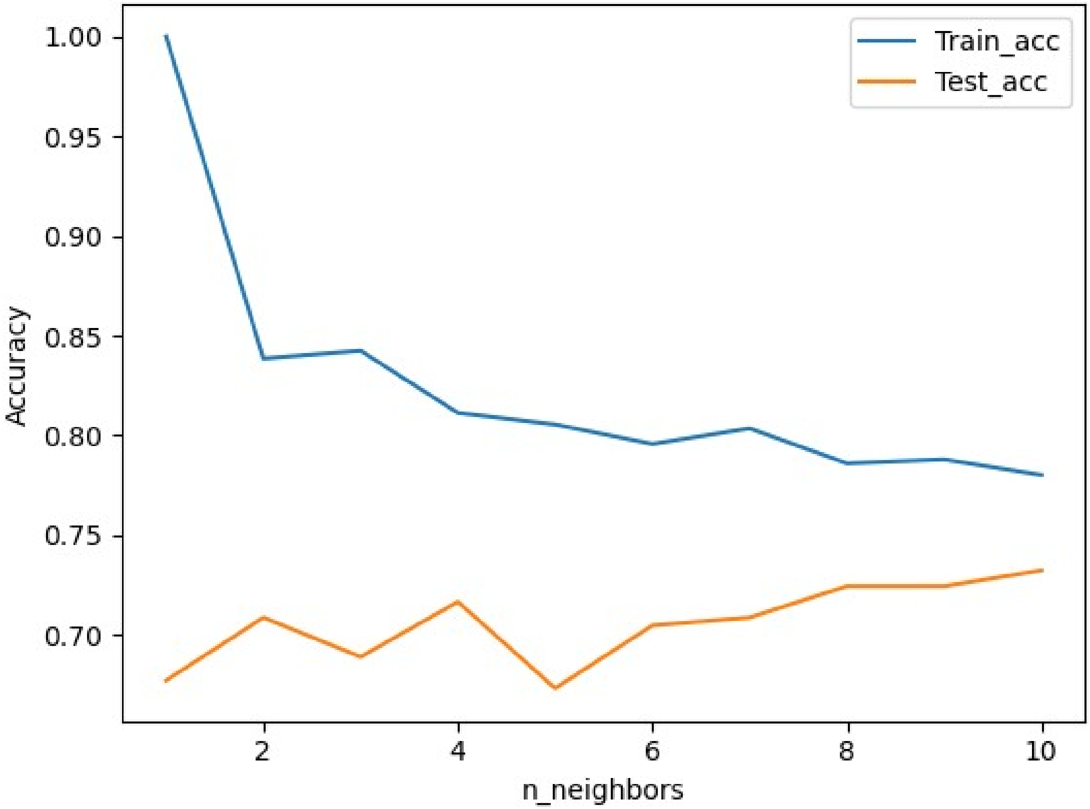
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

```python
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.33,random_state=42)

x_train
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin
BMI  \
464          10      115               98      20.536458   79.799479
24.0
223           7      142               60      33.000000  190.000000
28.8
393           4      116               72      12.000000   87.000000
22.1
766           1      126               60      20.536458   79.799479
30.1
570           3       78               70      20.536458   79.799479
32.5
..          ...      ...              ...            ...         ...
...
71            5      139               64      35.000000  140.000000
28.6
106           1       96              122      20.536458   79.799479
22.4
270          10      101               86      37.000000   79.799479
45.6
435           0      141               72      20.536458   79.799479
42.4
102           0      125               96      20.536458   79.799479
22.5

     DiabetesPedigreeFunction  Age
464                     1.022   34
223                     0.687   61
393                     0.463   37
766                     0.349   47
570                     0.270   39
..                        ...  ...
71                      0.411   26
106                     0.207   27
270                     1.136   38
435                     0.205   29
102                     0.262   21

[514 rows x 8 columns]
```

```python
from sklearn.neighbors import KNeighborsClassifier

train_acc=[]
test_acc=[]
for n_neighbors in range(1,11):
    knn=KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(x_train,y_train)
    train_acc.append(knn.score(x_train,y_train))
    test_acc.append(knn.score(x_test,y_test))

plt.plot(range(1,11),train_acc,label="Train_acc")
plt.plot(range(1,11),test_acc,label="Test_acc")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x162e0dcbac0>
```



```
knn=KNeighborsClassifier(n_neighbors=9)
knn.fit(x_train,y_train)
print(knn.score(x_train,y_train),": Training accuracy")
print(knn.score(x_test,y_test),": Test accuracy")
```

```
0.7879377431906615 : Training accuracy
0.7244094488188977 : Test accuracy
```

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(random_state=0)
dt.fit(x_train,y_train)
print(dt.score(x_train,y_train),": Training accuracy")
print(dt.score(x_test,y_test),": Test accuracy")
```

```
1.0 : Training accuracy
0.6811023622047244 : Test accuracy
```

```python
dt1=DecisionTreeClassifier(random_state=0,max_depth=3)
dt1.fit(x_train,y_train)
print(dt1.score(x_train,y_train),": Training accuracy")
print(dt1.score(x_test,y_test),": Test accuracy")
```

```
0.77431906614786 : Training accuracy
0.6929133858267716 : Test accuracy
```

```python
from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(random_state=42)
mlp.fit(x_train,y_train)
print(mlp.score(x_train,y_train),": Training accuracy")
print(mlp.score(x_test,y_test),": Test accuracy")
```

```
0.7509727626459144 : Training accuracy
0.6811023622047244 : Test accuracy
```

```python
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_sc=sc.fit_transform(x_train)
x_test_sc=sc.fit_transform(x_test)
```

```python
mlp1=MLPClassifier(random_state=42)
mlp1.fit(x_train_sc,y_train)
print(mlp1.score(x_train_sc,y_train),": Training accuracy")
print(mlp1.score(x_test_sc,y_test),": Test accuracy")
```

```
0.8346303501945526 : Training accuracy
0.7362204724409449 : Test accuracy
```

```
C:\Users\sujan\anaconda3\lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```