```
#import libraries
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	DiabetesPedigreeI
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	

768 rows × 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()
```

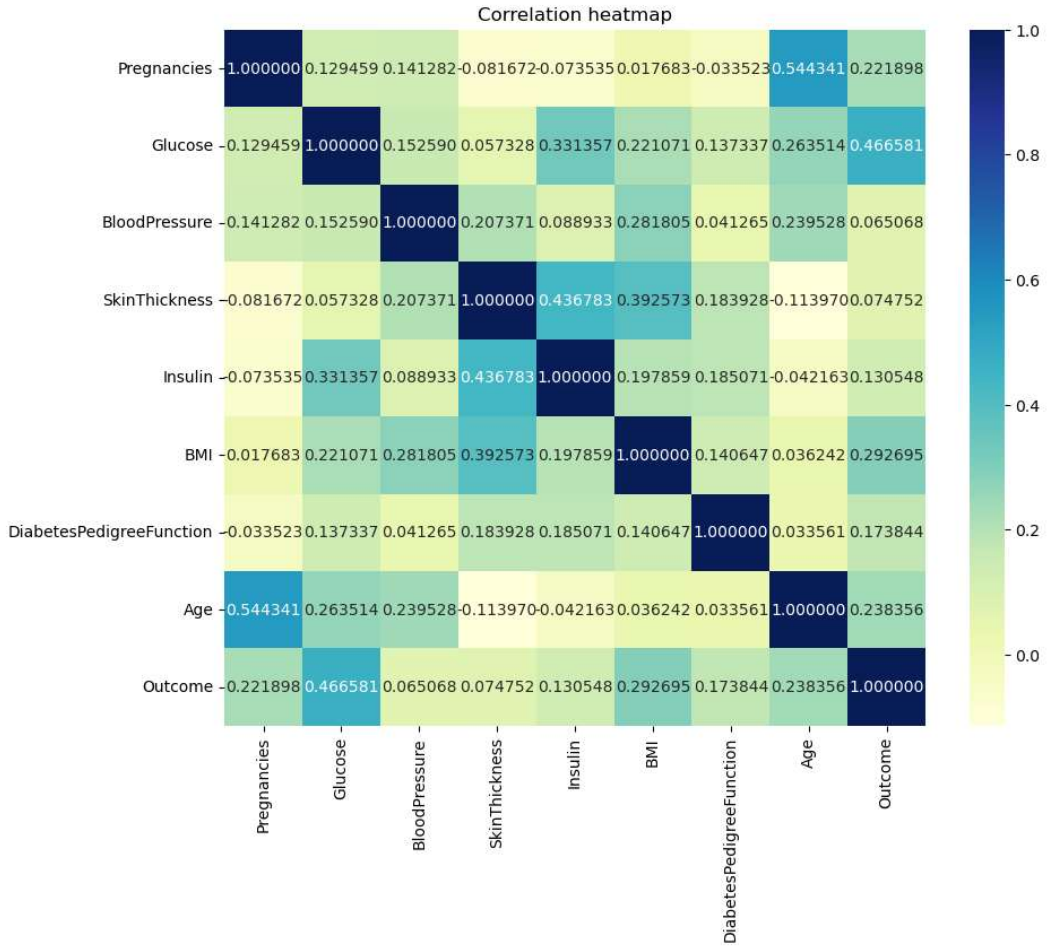
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.

```
#correlation plo of independemt variables
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')
```




```
#exploring pregnency and target variables
plt.figure(figsize=(10,8))
#plotting density function graph of the pregnancies and target variables
kde=sns.kdeplot(dataset["Pregnancies"][dataset["Outcome"]==1],color="Red",shade = True)
kde=sns.kdeplot(dataset["Pregnancies"][dataset["Outcome"]==0],color="Blue",shade = True)
kde.set_xlabel("Pregnancies")
kde.set_ylabel("Density")
kde.legend(["Postive","Negative"])
```

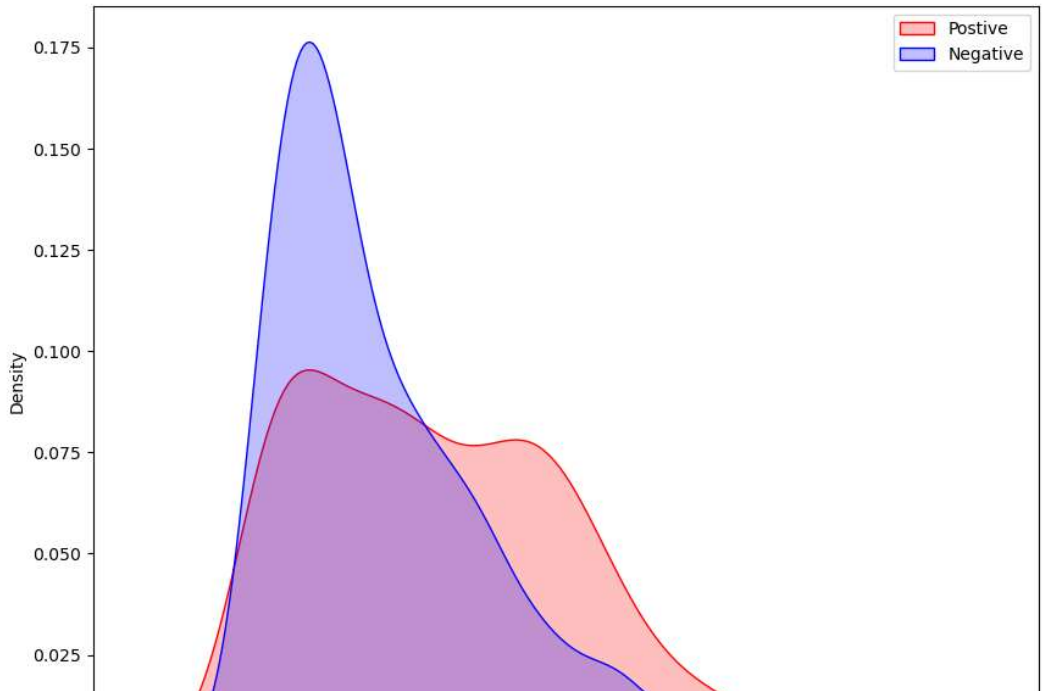
```
C:\Users\Dell\AppData\Local\Temp\ipykernel_3412\68954946.py:4: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kde=sns.kdeplot(dataset["Pregnancies"][dataset["Outcome"]==1],color="Red",shade = True)
C:\Users\Dell\AppData\Local\Temp\ipykernel_3412\68954946.py:5: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kde=sns.kdeplot(dataset["Pregnancies"][dataset["Outcome"]==0],color="Blue",shade = True)
<matplotlib.legend.Legend at 0x1c9e19b0190>
```




```
# exploring glucose and target variables
plt.figure(figsize=(10,8))
sns.violinplot(data=dataset,x="Outcome",y="Glucose",split=True,linewidth=2,inner="quart")
```

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

```
#exploring glucose and target variables
plt.figure(figsize=(10,8))
#plotting density function graph of the glucose and target variables
kde=sns.kdeplot(dataset["Glucose"][dataset["Outcome"]==1],color="Red",shade = True)
kde=sns.kdeplot(dataset["Glucose"][dataset["Outcome"]==0],color="Blue",shade = True)
kde.set_xlabel("Glucose")
kde.set_ylabel("Density")
kde.legend(["Postive","Negative"])
```

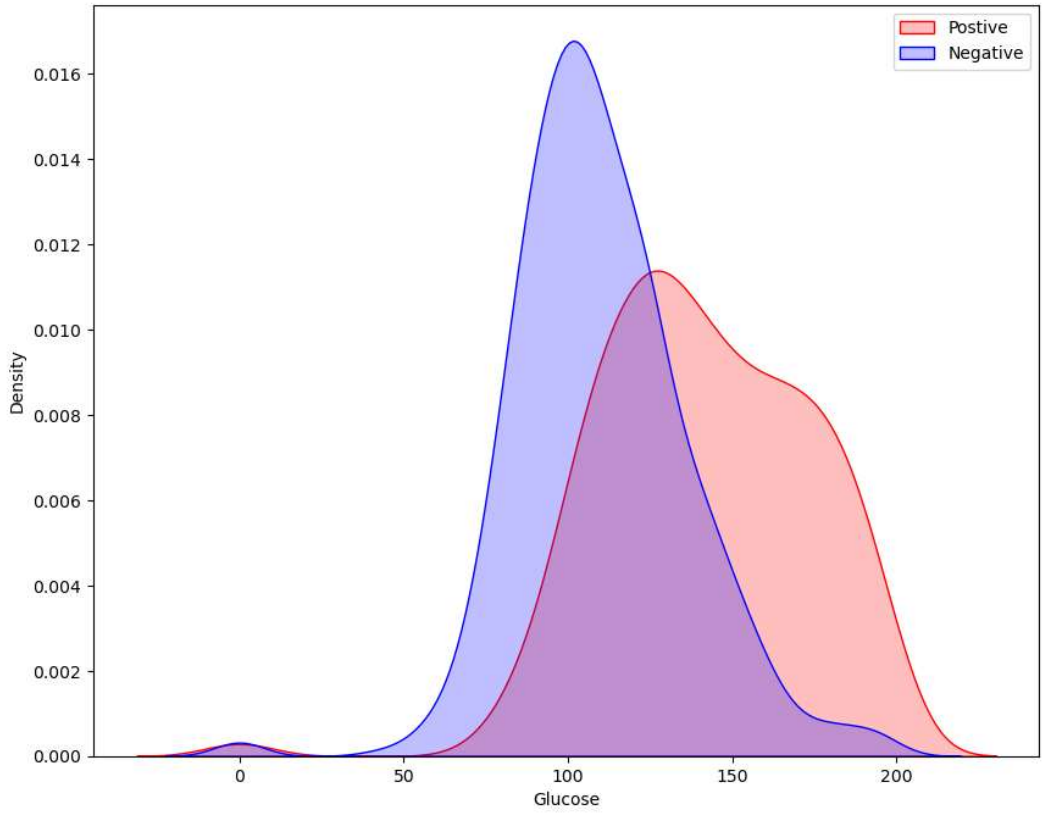
```
C:\Users\Dell\AppData\Local\Temp\ipykernel_3412\4242145369.py:4: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kde=sns.kdeplot(dataset["Glucose"][dataset["Outcome"]==1],color="Red",shade = True)
C:\Users\Dell\AppData\Local\Temp\ipykernel_3412\4242145369.py:5: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kde=sns.kdeplot(dataset["Glucose"][dataset["Outcome"]==0],color="Blue",shade = True)
<matplotlib.legend.Legend at 0x1c9e1977f10>
```




```
#replacing 0 values with mean and median of the resptive features
#glucose
dataset["Glucose"]=dataset["Glucose"].replace(0,dataset["Glucose"].median())
#bloodpressure
dataset["BloodPressure"]=dataset["BloodPressure"].replace(0,dataset["BloodPressure"].median())
#BMI
dataset["BMI"]=dataset["BMI"].replace(0,dataset["BMI"].mean())
#SkinThickness
dataset["SkinThickness"]=dataset["SkinThickness"].replace(0,dataset["SkinThickness"].mean())
#Insulin
dataset["Insulin"]=dataset["Insulin"].replace(0,dataset["Insulin"].mean())
```

```
dataset
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35.000000	79.799479	33.6	0.627	50
1	1	85	66	29.000000	79.799479	26.6	0.351	31
2	8	183	64	20.536458	79.799479	23.3	0.672	32
3	1	89	66	23.000000	94.000000	28.1	0.167	21
4	0	137	40	35.000000	168.000000	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48.000000	180.000000	32.9	0.171	63
764	2	122	70	27.000000	79.799479	36.8	0.340	27
765	5	121	72	23.000000	112.000000	26.2	0.245	30
766	1	126	60	20.536458	79.799479	30.1	0.349	47
767	1	93	70	31.000000	79.799479	30.4	0.315	23

768 rows × 9 columns

```
#splitting the depandant variable and indepdent variable
x=dataset.drop(["Outcome"],axis=1)
y=dataset["Outcome"]
```

```
x
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35.000000	79.799479	33.6	0.627	50
1	1	85	66	29.000000	79.799479	26.6	0.351	31
2	8	183	64	20.536458	79.799479	23.3	0.672	32
3	1	89	66	23.000000	94.000000	28.1	0.167	21
4	0	137	40	35.000000	168.000000	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48.000000	180.000000	32.9	0.171	63
764	2	122	70	27.000000	79.799479	36.8	0.340	27
765	5	121	72	23.000000	112.000000	26.2	0.245	30
766	1	126	60	20.536458	79.799479	30.1	0.349	47
767	1	93	70	31.000000	79.799479	30.4	0.315	23

768 rows × 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
```

```
#splitting the dataset into training and testing dataset
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	DiabetesPedigreeFunction	Age
464	10	115	98	20.536458	79.799479	24.0	1.022	34
223	7	142	60	33.000000	190.000000	28.8	0.687	61
393	4	116	72	12.000000	87.000000	22.1	0.463	37
766	1	126	60	20.536458	79.799479	30.1	0.349	47
570	3	78	70	20.536458	79.799479	32.5	0.270	39
...	...	...	...	...	...	...	...	...
71	5	139	64	35.000000	140.000000	28.6	0.411	26
106	1	96	122	20.536458	79.799479	22.4	0.207	27
270	10	101	86	37.000000	79.799479	45.6	1.136	38
435	0	141	72	20.536458	79.799479	42.4	0.205	29
102	0	125	96	20.536458	79.799479	22.5	0.262	21

514 rows × 8 columns

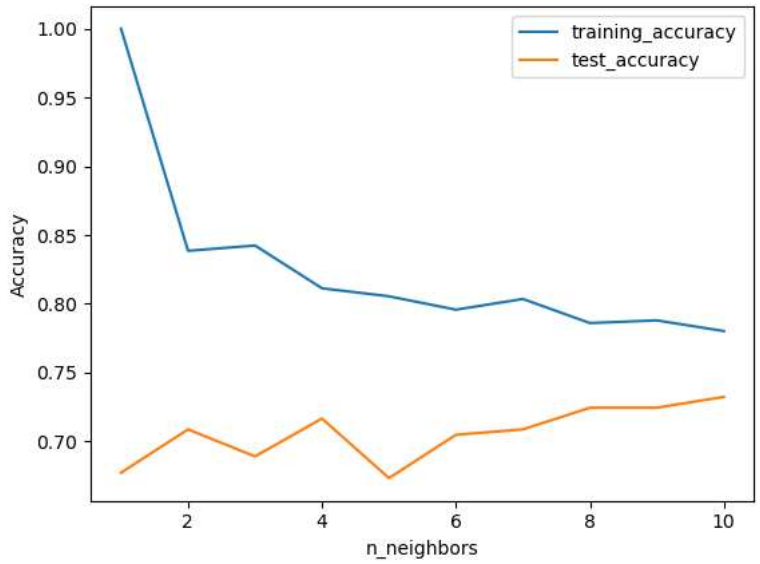
```
#knn
from sklearn.neighbors import KNeighborsClassifier


training_accuracy=[]
test_accuracy=[]
for n_neighbors in range(1,11):
    knn=KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(x_train,y_train)

    #check accuracy score
    training_accuracy.append(knn.score(x_train,y_train))
    test_accuracy.append(knn.score(x_test,y_test))


plt.plot(range(1,11),training_accuracy,label="training_accuracy")
plt.plot(range(1,11),test_accuracy,label="test_accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1c9e1b1da20>
```




```
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
```

```
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
```

```
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
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

```
mlp1=MLPClassifier(random_state=0)
mlp1.fit(x_train_scaled,y_train)
print(mlp1.score(x_train_scaled,y_train),":Training accuracy")
print(mlp1.score(x_test_scaled,y_test),":Test accuracy")
```

```
0.8326848249027238 :Training accuracy
0.7322834645669292 :Test accuracy
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimizer
  warnings.warn(
```