Diabetes Prediction

STEP 1:- Import Dataset

In [10]:
```python
#import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```python
#import dataset
dataset = pd.read_csv("diabetes.csv")
dataset
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

STEP 2:- Data Inspection

In [3]:
```python
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
```

In [4]:
```python
dataset.isnull().sum()
```

Out[4]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [5]:
```python
dataset.describe()
```

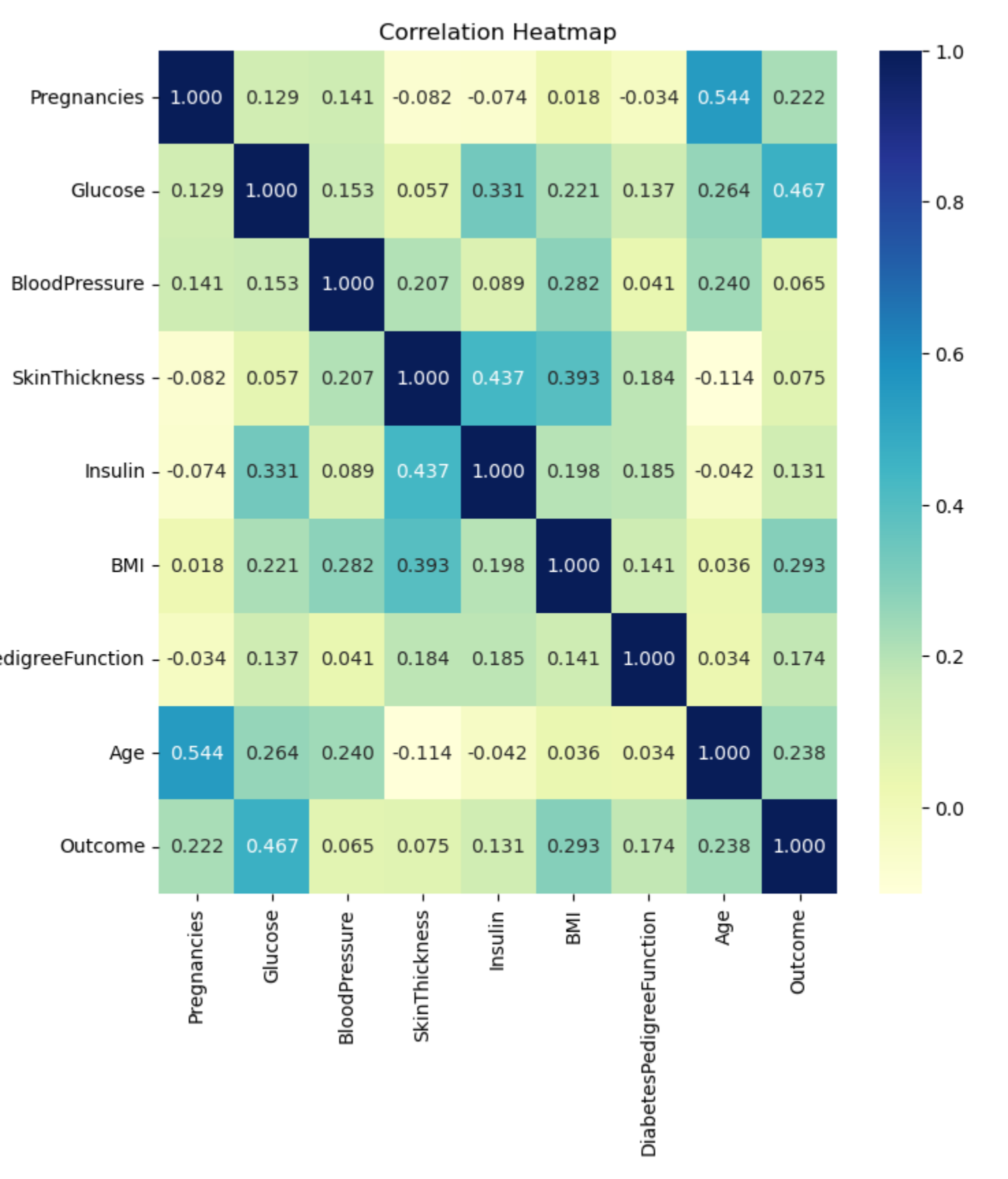
Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

STEP 3:- Data Visualization

In [13]:
```python
#plot of correlationship between independent variables
plt.figure(figsize=(8,8))
sns.heatmap(dataset.corr(),annot = True, fmt = ".3f", cmap = "YlGnBu")
plt.title("Correlation Heatmap")
```

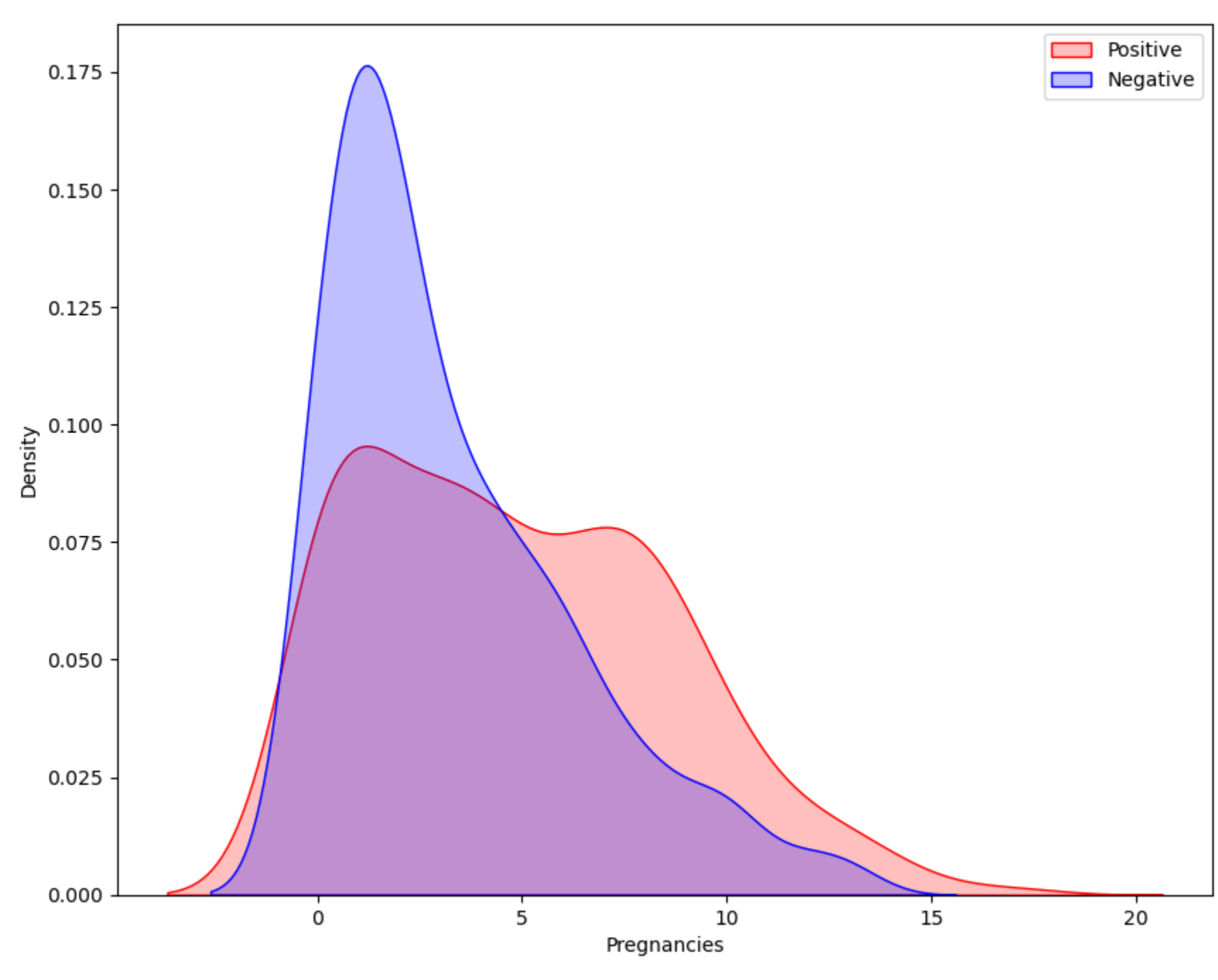
Out[13]:
```
Text(0.5, 1.0, 'Correlation Heatmap')
```



In [14]:
```python
#exploring pregnancy 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(["Positive","Negative"])
```

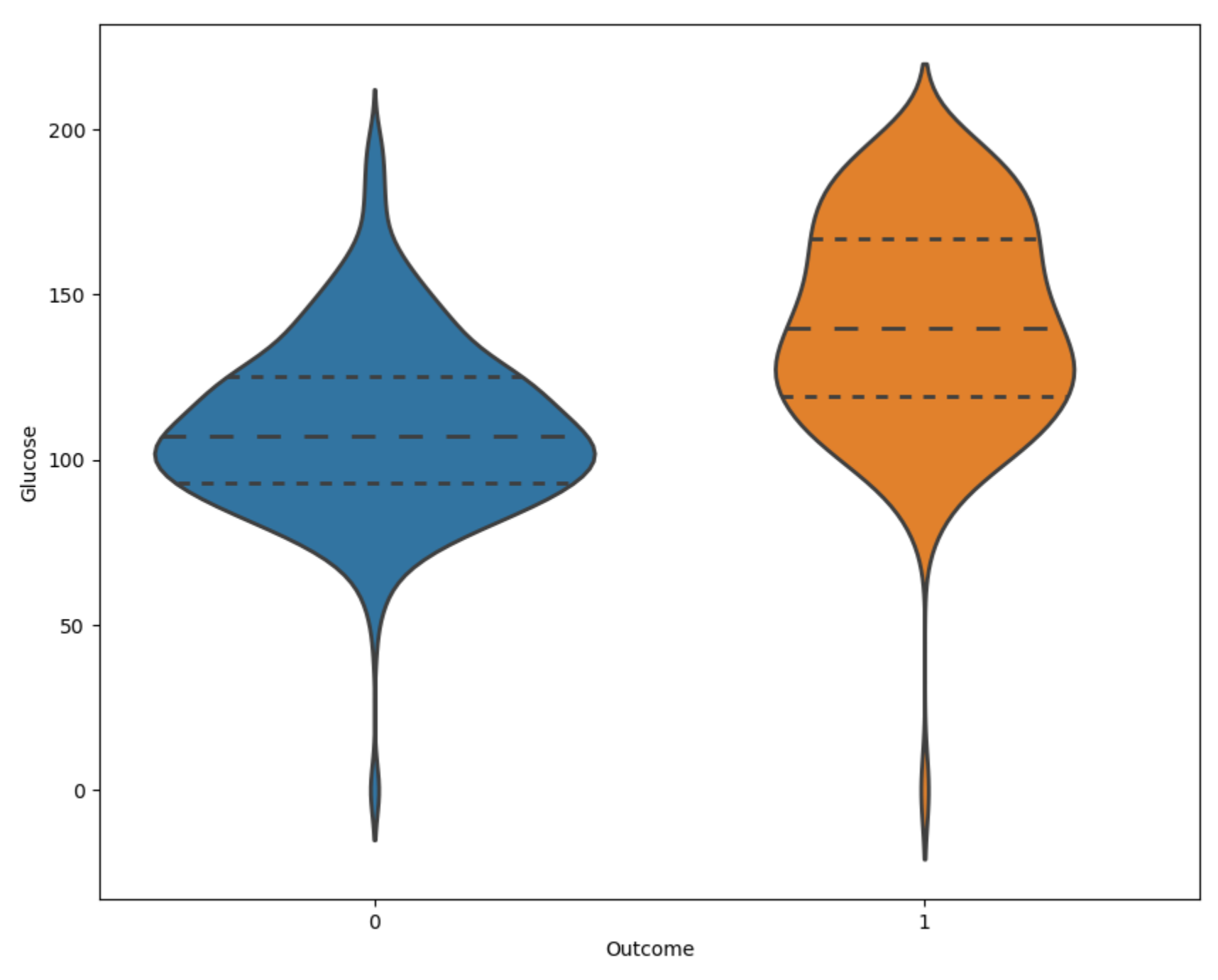
```
C:\Users\armst\AppData\Local\Temp\ipykernel_15176\185164329.py:4: FutureWarning:

`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\armst\AppData\Local\Temp\ipykernel_15176\185164329.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)
```

Out[14]:
```
<matplotlib.legend.Legend at 0x1ed978ad1d0>
```



In [16]:
```python
#exploring glucose and target variables
plt.figure(figsize =(10,8))
sns.violinplot(data=dataset,x="Outcome", y="Glucose", split= True, linewidth=2, inner = "quart")
```

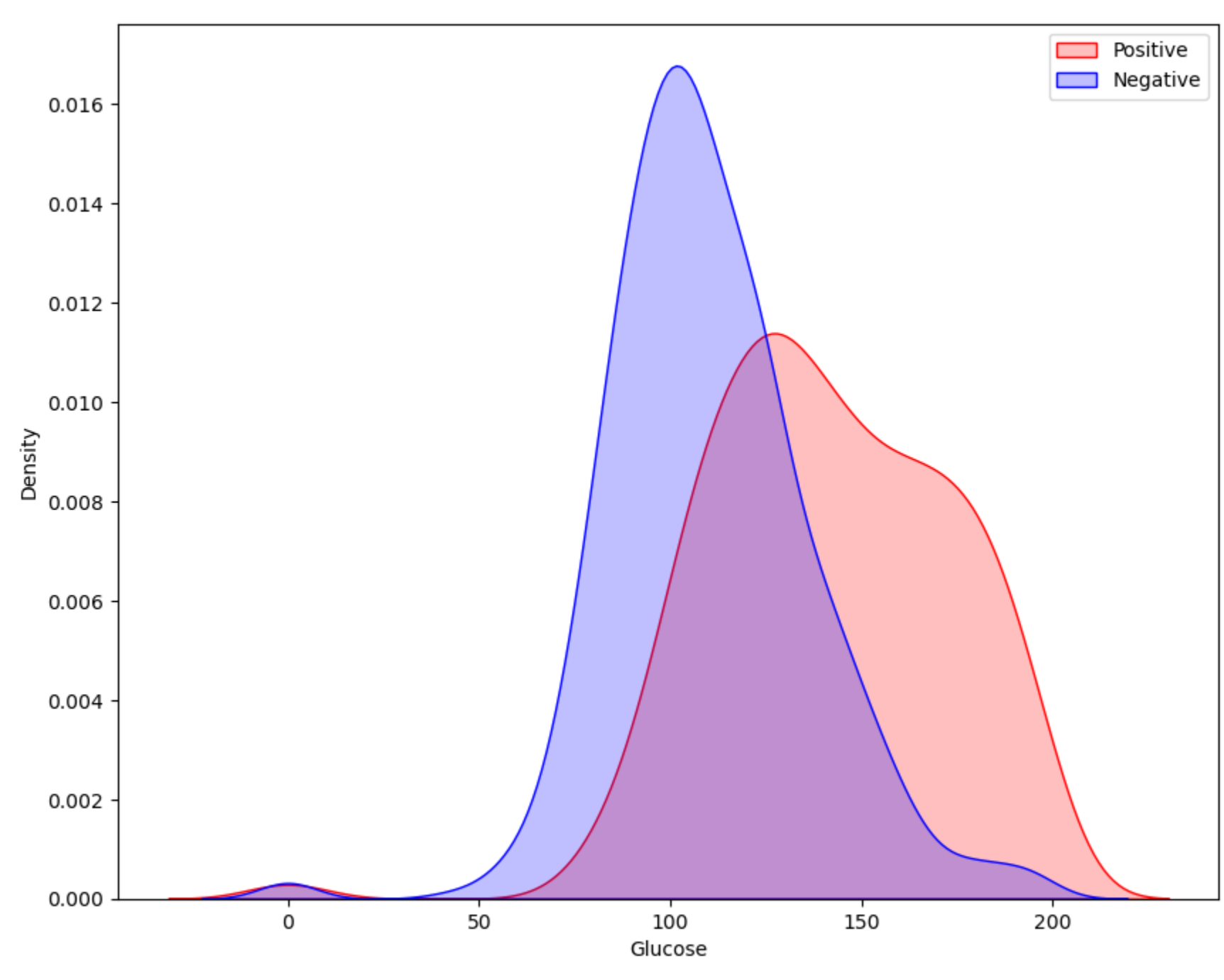
Out[16]:
```
<Axes: xlabel='Outcome', ylabel='Glucose'>
```



In [17]:
```python
#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(["Positive","Negative"])
```

```
C:\Users\armst\AppData\Local\Temp\ipykernel_15176\3390448368.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\armst\AppData\Local\Temp\ipykernel_15176\3390448368.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)
```

Out[17]:
```
<matplotlib.legend.Legend at 0x1ed99e21990>
```



In [18]:
```python
#replace 0 values with the mean/median of the respective feature
#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"].median())
#SkinThickness
dataset["SkinThickness"] = dataset["SkinThickness"].replace(0, dataset["SkinThickness"].median())
#Insulin
dataset["Insulin"] = dataset["Insulin"].replace(0, dataset["Insulin"].median())
```

STEP 4:- Data Preparation

In [21]:
```python
#split the feature and label
X = dataset.drop(["Outcome"], axis = 1)
y = dataset["Outcome"]
```

In [23]:
```python
X
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	30.5	33.6	0.627	50
1	1	85	66	29	30.5	26.6	0.351	31
2	8	183	64	23	30.5	23.3	0.672	32
3	1	89	66	23	94.0	28.1	0.167	21
4	0	137	40	35	168.0	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180.0	32.9	0.171	63
764	2	122	70	27	30.5	36.8	0.340	27
765	5	121	72	23	112.0	26.2	0.245	30
766	1	126	60	23	30.5	30.1	0.349	47
767	1	93	70	31	30.5	30.4	0.315	23

768 rows × 8 columns

In [24]:
```python
y
```

Out[24]:
```
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
```

In [25]:
```python
#split train and test dataset
!pip install scikit-learn
from sklearn.model_selection import train_test_split
```

```
Requirement already satisfied: scikit-learn in c:\users\armst\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\armst\anaconda3\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\armst\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\armst\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\armst\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
```

In [26]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [27]:
```python
X_train
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
464	10	115	98	23	30.5	24.0	1.022	34
223	7	142	60	33	190.0	28.8	0.687	61
393	4	116	72	12	87.0	22.1	0.463	37
766	1	126	60	23	30.5	30.1	0.349	47
570	3	78	70	23	30.5	32.5	0.270	39
...	...	...	...	...	...	...	...	...
71	5	139	64	35	140.0	28.6	0.411	26
106	1	96	122	23	30.5	22.4	0.207	27
270	10	101	86	37	30.5	45.6	1.136	38
435	0	141	72	23	30.5	42.4	0.205	29
102	0	125	96	23	30.5	22.5	0.262	21

514 rows × 8 columns

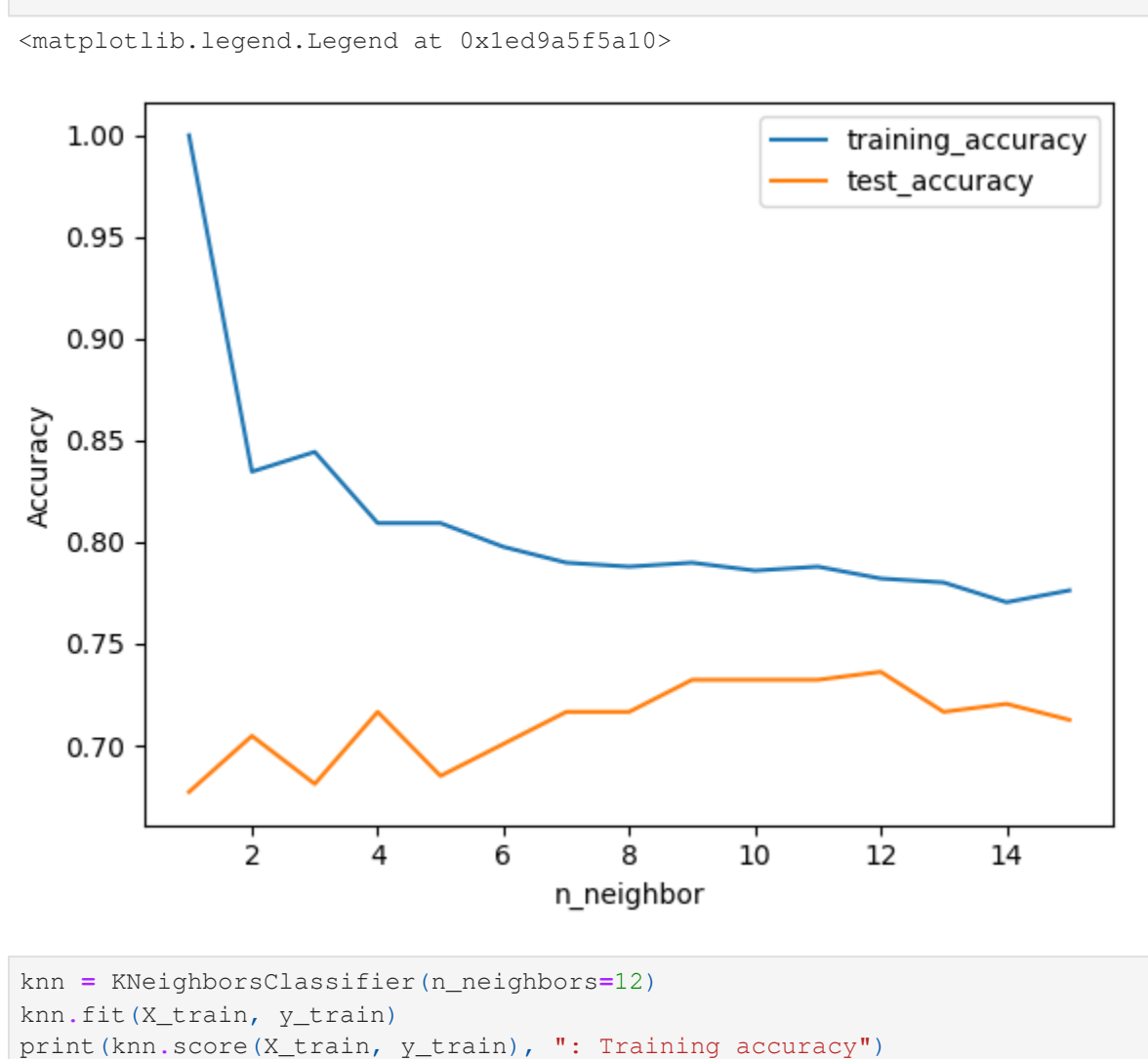
STEP 5:- Build ML Model

In [28]:
```python
#KNN
from sklearn.neighbors import KNeighborsClassifier
```

In [34]:
```python
training_accuracy = []
test_accuracy =[]
for n_neighbors in range(1,16):
    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))
```

In [35]:
```python
plt.plot(range(1,16), training_accuracy, label = "training_accuracy")
plt.plot(range(1,16), test_accuracy, label = "test_accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbor")
plt.legend()
```

Out[35]:
```
<matplotlib.legend.Legend at 0x1ed9a5f5a10>
```



In [36]:
```python
knn = KNeighborsClassifier(n_neighbors=12)
knn.fit(X_train, y_train)
print(knn.score(X_train, y_train), ": Training accuracy")
print(knn.score(X_test, y_test), ": Test accuracy")
```

```
0.7821011673151751 : Training accuracy
0.7362204724409449 : Test accuracy
```

In [37]:
```python
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=0, max_depth=4)
dt.fit(X_train, y_train)
print(dt.score(X_train, y_train), ": Training accuracy")
print(dt.score(X_test, y_test), ": Test accuracy")
```

```
0.8171206225680934 : Training accuracy
0.7244094488188977 : Test accuracy
```

In [ ]: