```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

```
dataset = pd.read_csv('dibeties.csv')
```

```
dataset.head()
```

	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

Next steps: Generate code with dataset | View recommended plots | New interactive sheet

```
dataset.shape
```

```
(768, 9)
```

```
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.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcom
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.00000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.34895
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.47695
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.00000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.00000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.00000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.00000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.00000

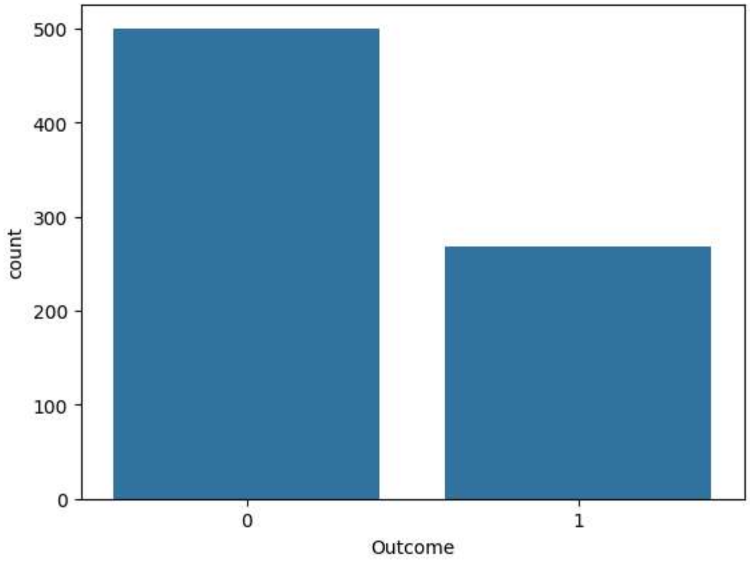
```
dataset.isnull().sum()
```

```
                          0
Pregnancies               0
Glucose                   0
BloodPressure             0
SkinThickness             0
Insulin                   0
BMI                       0
DiabetesPedigreeFunction  0
Age                       0
Outcome                   0

dtype: int64
```

DATA VISUALIZATION

```
sns.countplot(x = 'Outcome',data = dataset)
```

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



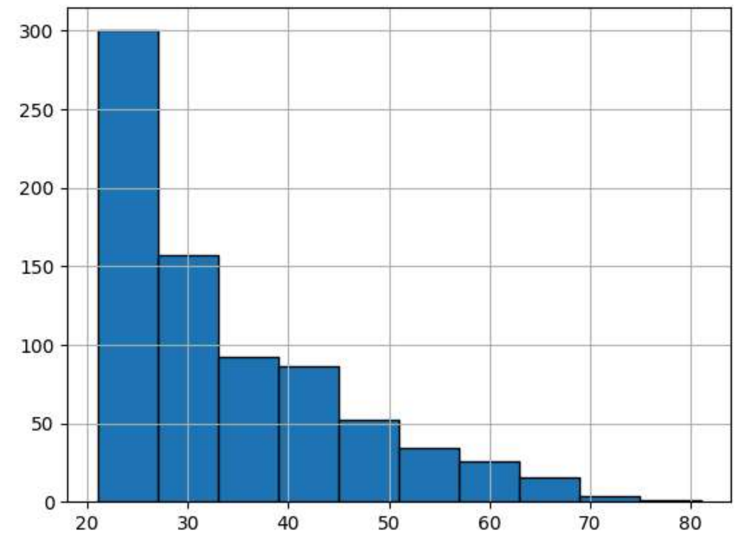
```
dataset["Outcome"].value_counts()*100/len(dataset)
```

```
              count
Outcome
   0      65.104167
   1      34.895833

dtype: float64
```

```
dataset.Outcome.value_counts()
```

```
          count
Outcome
   0        500
   1        268

dtype: int64
```

```
dataset["Age"].hist(edgecolor = "black");
```

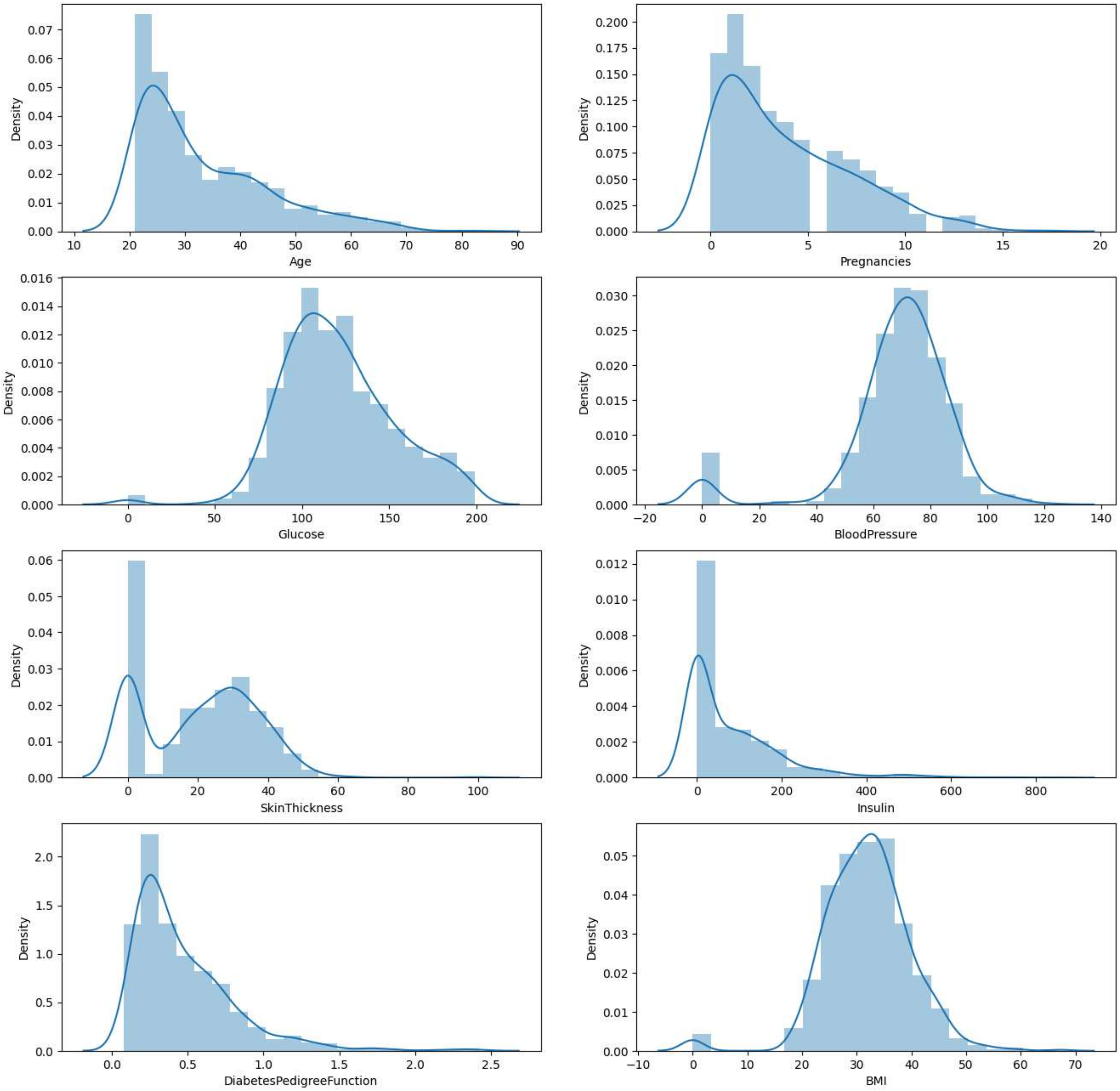


```
print("Max Age: " + str(dataset["Age"].max()) + " Min Age: " + str(dataset["Age"].min()))
```

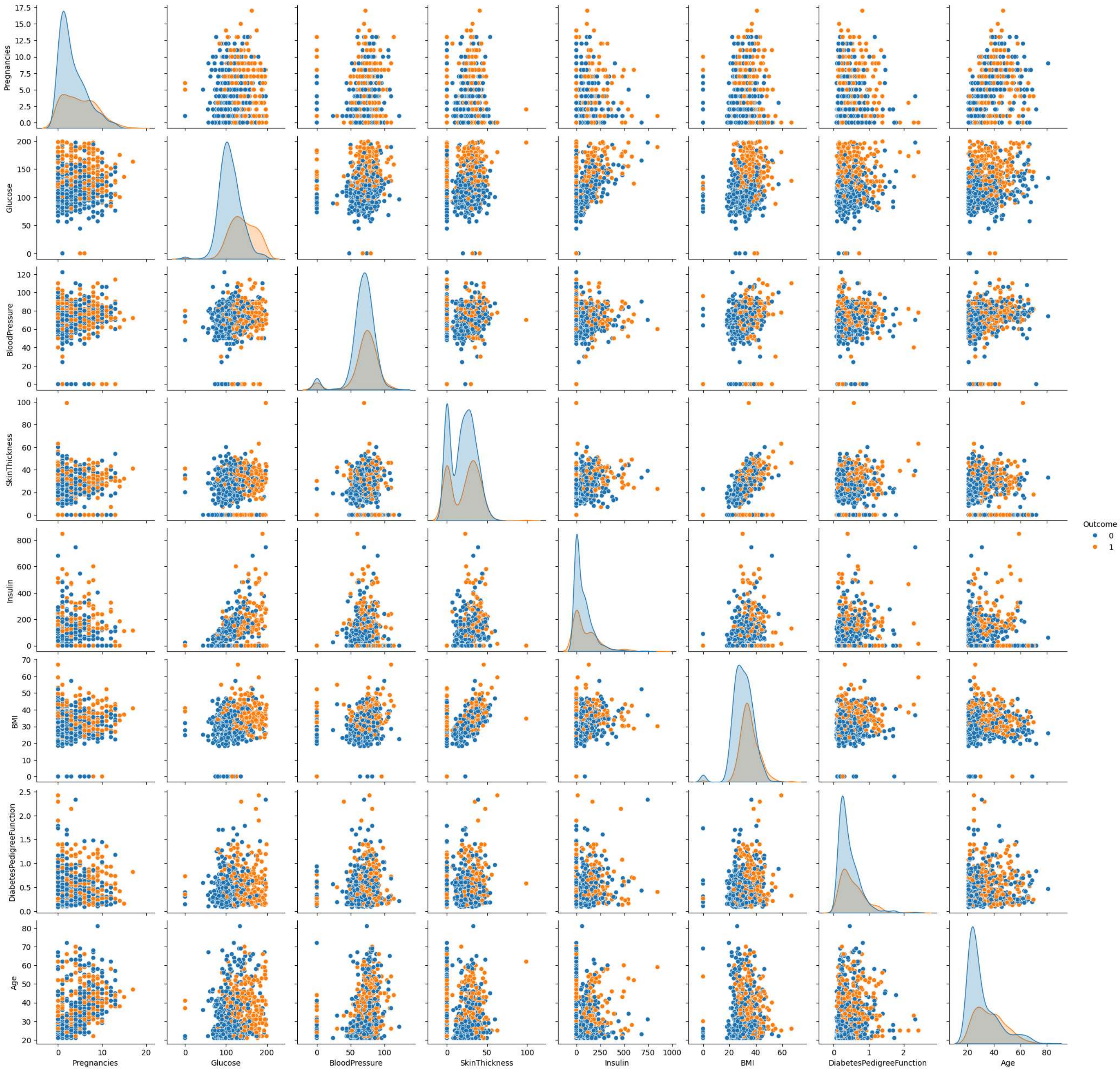
```
Max Age: 81 Min Age: 21
```

```
fig, ax = plt.subplots(4,2, figsize=(16,16))
sns.distplot(dataset.Age, bins = 20, ax=ax[0,0])
sns.distplot(dataset.Pregnancies, bins = 20, ax=ax[0,1])
sns.distplot(dataset.Glucose, bins = 20, ax=ax[1,0])
sns.distplot(dataset.BloodPressure, bins = 20, ax=ax[1,1])
sns.distplot(dataset.SkinThickness, bins = 20, ax=ax[2,0])
sns.distplot(dataset.Insulin, bins = 20, ax=ax[2,1])
sns.distplot(dataset.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])
sns.distplot(dataset.BMI, bins = 20, ax=ax[3,1])
```

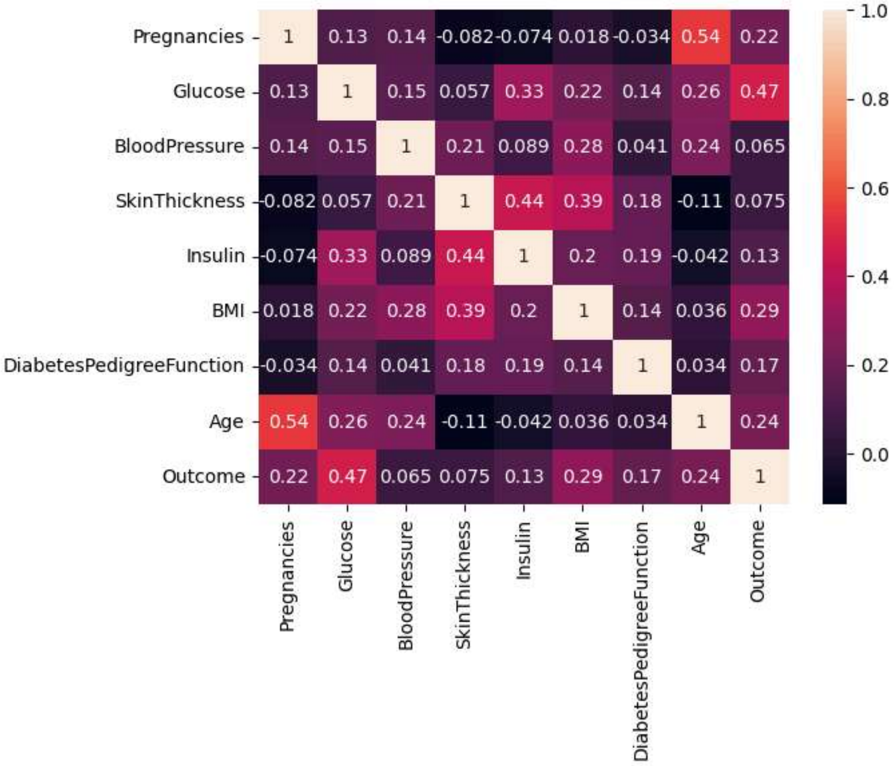
```
<Axes: xlabel='BMI', ylabel='Density'>
```



```
sns.pairplot(data = dataset, hue = 'Outcome')
plt.show()
```



```
sns.heatmap(dataset.corr(), annot = True)
plt.show()
```



```
dataset_new = dataset
```

```
dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dataset_new[["Glucose", "BloodPressure", "SkinThick
```

```
dataset_new.isnull().sum()
```

```
                            0
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0

dtype: int64
```

```
def median_target(var):
    temp = dataset_new[dataset_new[var].notnull()]
    temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    return temp
```

```
columns = dataset_new.columns
columns = columns.drop("Outcome")
for i in columns:
    median_target(i)
    dataset_new.loc[(dataset_new['Outcome'] == 0 ) & (dataset_new[i].isnull()), i] = median_target(i)[i][0]
    dataset_new.loc[(dataset_new['Outcome'] == 1 ) & (dataset_new[i].isnull()), i] = median_target(i)[i][1]
```

```
dataset_new.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

Next steps: Generate code with dataset_new View recommended plots New interactive sheet

```
dataset_new.isnull().sum()
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

```
dataset_new["Glucose"].fillna(dataset_new["Glucose"].mean(), inplace = True)
dataset_new["BloodPressure"].fillna(dataset_new["BloodPressure"].mean(), inplace = True)
dataset_new["SkinThickness"].fillna(dataset_new["SkinThickness"].mean(), inplace = True)
dataset_new["Insulin"].fillna(dataset_new["Insulin"].mean(), inplace = True)
dataset_new["BMI"].fillna(dataset_new["BMI"].mean(), inplace = True)
```

```
dataset_new.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	121.677083	30.464161	44.000	99.75000	117.0000	140.25000	199.00
BloodPressure	768.0	72.389323	12.106039	24.000	64.00000	72.0000	80.00000	122.00
SkinThickness	768.0	29.089844	8.890820	7.000	25.00000	28.0000	32.00000	99.00
Insulin	768.0	141.753906	89.100847	14.000	102.50000	102.5000	169.50000	846.00
BMI	768.0	32.434635	6.880498	18.200	27.50000	32.0500	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
dataset_scaled = sc.fit_transform(dataset_new)
```

```
dataset_scaled = pd.DataFrame(dataset_scaled)
```

```
X = dataset_scaled.iloc[:, [1, 4, 5, 7]].values
Y = dataset_scaled.iloc[:, 8].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 42, stratify = dataset_new['Outcome'] )
```

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", Y_train.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (614, 4)
X_test shape: (154, 4)
Y_train shape: (614,)
Y_test shape: (154,)
```

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state = 42)
logreg.fit(X_train, Y_train)
```

```
▾    LogisticRegression     i ?
LogisticRegression(random_state=42)
```

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, Y_train)
```

```
▾ GaussianNB  i ?
GaussianNB()
```

```
Y_pred_logreg = logreg.predict(X_test)
Y_pred_nb = nb.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_logreg = accuracy_score(Y_test, Y_pred_logreg)
accuracy_nb = accuracy_score(Y_test, Y_pred_nb)
```

```
print("Logistic Regression: " + str(accuracy_logreg * 100))
print("Naive Bayes: " + str(accuracy_nb * 100))
```

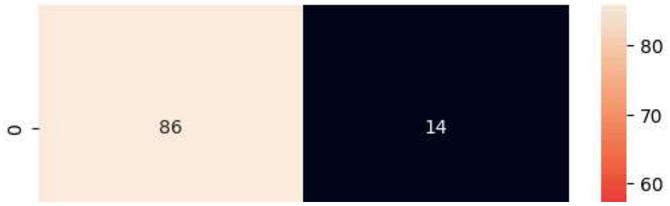
```
Logistic Regression: 74.67532467532467
Naive Bayes: 72.72727272727273
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred_logreg)
cm
```

```
array([[86, 14],
       [25, 29]])
```

```
sns.heatmap(pd.DataFrame(cm), annot=True)
```

```
<Axes: >
```


0
86
14
80
70
60