Diabetes Paitents

March 31, 2024

1 Diabetes Patients

[3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[5]:
```python
data = pd.read_csv("E:\diabetes.csv")
data
```

[5]:
```
     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]
```

[6]:
```
data.shape
```

[6]:
```
(768, 9)
```

[7]:
```
data.head()
```

[7]:
```
   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

   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
```

[8]:
```
data.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
```

[9]:
```
#checking null values
pd.isnull(data).sum()
```

[9]:
```
Pregnancies                 0
Glucose                     0
```

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

[10]:
```
data.columns
```

[10]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[11]:
```
data.describe()
```

[11]:

	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

1.0.1 Insights

The features "Glucose", "Blood Pressure" ,"Skin Thickness", "Insulin" and "BMI" all have a minimum value of 0. This is illogical because these values can't be zero. Therefore, in our circumstance, this can be safely referred to as • missing data". The 0-valued rows must either be remove or replaced with the mean or median value for that feature.

[12]:
```
df_cp = data.copy()
df_cp['Glucose'] = data['Glucose'].replace(0, data['Glucose'].median())
df_cp['BloodPressure'] = data['BloodPressure'].replace(0, data['BloodPressure'].
 ↪median())
```

```
df_cp['SkinThickness'] = data['SkinThickness'].replace(0, data['SkinThickness'].
 ↪median())
df_cp['Insulin'] = data['Insulin'].replace(0, data['Insulin'].median())
df_cp['BMI'] = data['BMI'].replace(0, data['BMI'].median())
```

[13]:

```
df_cp.describe()
```

[13]:

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  121.656250      72.386719      27.334635   94.652344
std       3.369578   30.438286      12.096642       9.229014  105.547598
min       0.000000   44.000000      24.000000       7.000000   14.000000
25%       1.000000   99.750000      64.000000      23.000000   30.500000
50%       3.000000  117.000000      72.000000      23.000000   31.250000
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    32.450911                  0.471876   33.240885    0.348958
std      6.875366                  0.331329   11.760232    0.476951
min     18.200000                  0.078000   21.000000    0.000000
25%     27.500000                  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
```

[14]:

```
df_cp.isnull().sum()
```

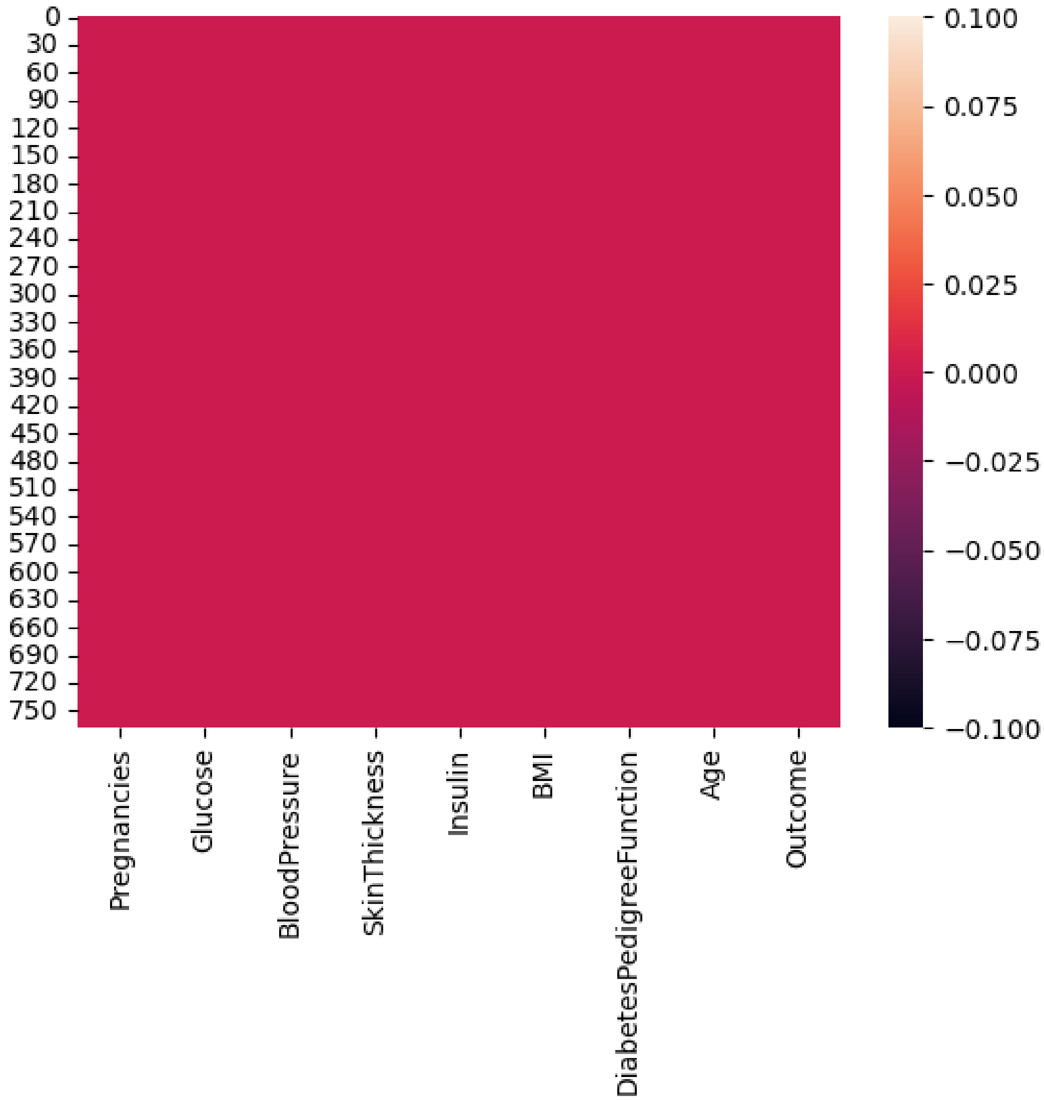
[14]:

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

[15]:

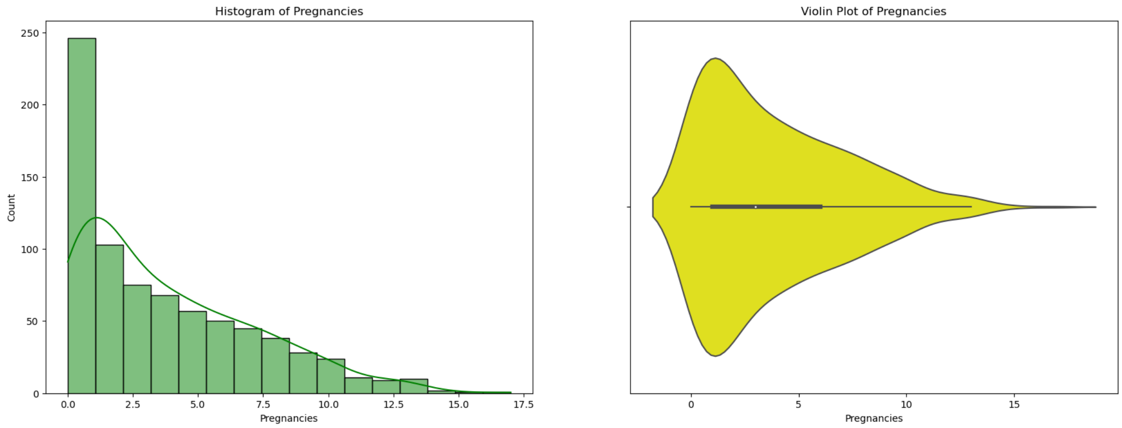
```
sns.heatmap(df_cp.isnull())
```

[15]: <Axes: >



1.1 Univariate Analysis

[17]:
```python
#Pregnancies
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="Pregnancies", kde=True, ax=ax1[0], color='green')
ax1[0].set_title('Histogram of Pregnancies')
sns.violinplot(data=df_cp, x="Pregnancies", ax=ax1[1], color='yellow')
ax1[1].set_title('Violin Plot of Pregnancies')
plt.show()
```



[18]:
```
df_cp["Pregnancies"].value_counts()
```

[18]:
```
Pregnancies
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
Name: count, dtype: int64
```

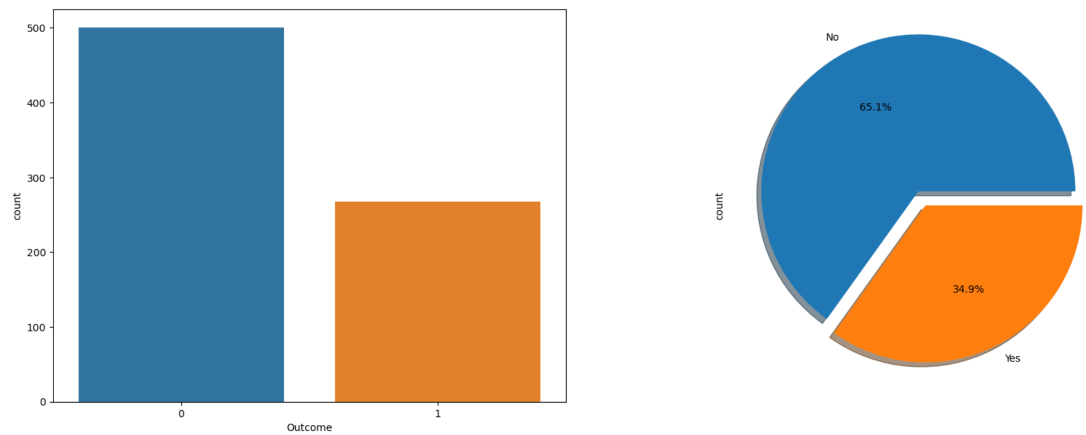
From the above analysis we observe that: • Most patients had O, 1 and 2 pregnancies. • Also. patients had upto 17 pregnancies!

[19]:
```
print("Median of Pregnancies: ", df_cp["Pregnancies"].median())
print("Maximum of Pregnancies: ", df_cp["Pregnancies"].max())
print("Mean of Pregnancies: ", df_cp["Pregnancies"].mean())
```

```
Median of Pregnancies:  3.0
Maximum of Pregnancies:  17
Mean of Pregnancies:  3.8450520833333335
```

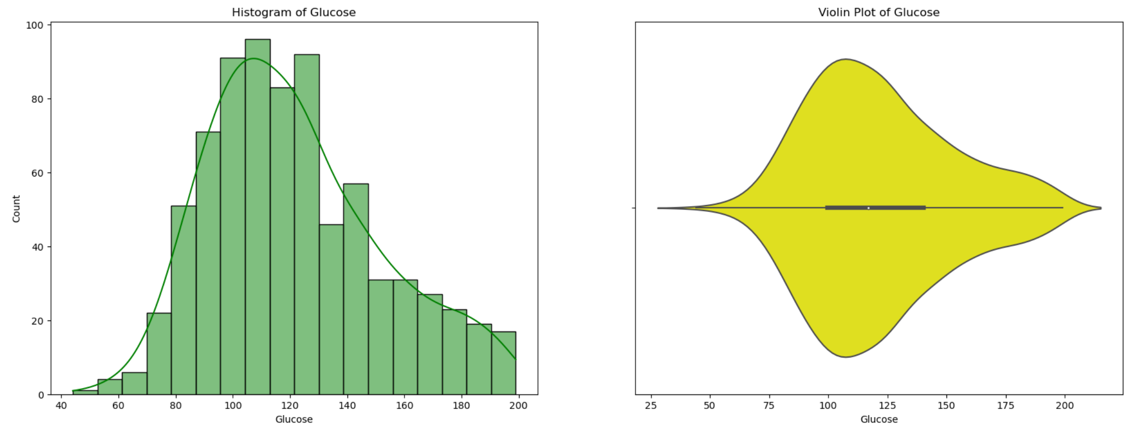
From the above analysis we observe that: • Median value of Pregnancies is 3. • Maximum value of Pregnancies is 17

[20]:
```
#Analysis of Outcome
fig, ax = plt.subplots(1, 2, figsize=(20, 7))
sns.countplot(data=df_cp, x="Outcome", ax=ax[0])
data["Outcome"].value_counts().plot.pie(explode=[0.1, 0],autopct="%1.1f%%",␣
↪labels=["No", "Yes"], shadow=True, ax=ax[1])
plt.show()
```



We observe from the above plot that: • 65.1% patients in the dataset do NOT have diabetes. • 34.9% patients in the dataset has diabetes.

[21]:
```
#Glucose
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="Glucose", kde=True, ax=ax1[0], color='green')
ax1[0].set_title('Histogram of Glucose')
sns.violinplot(data=df_cp, x="Glucose", ax=ax1[1], color='yellow')
ax1[1].set_title('Violin Plot of Glucose')
plt.show()
```



[22]:
```
print("Median of Glucose: ", df_cp["Glucose"].median())
print("Maximum of Glucose: ", df_cp["Glucose"].max())
print("Mean of Glucose: ", df_cp["Glucose"].mean())
```

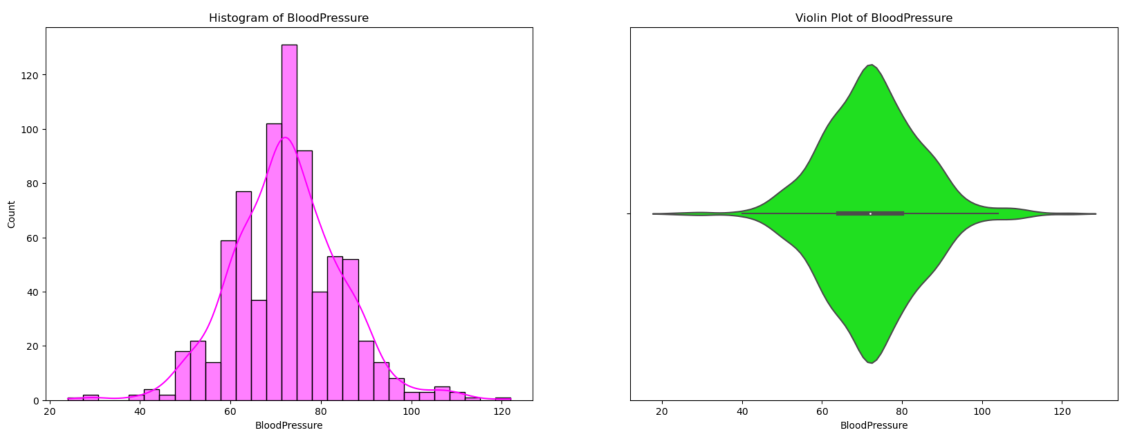
```
Median of Glucose:  117.0
Maximum of Glucose:  199
Mean of Glucose:  121.65625
```

We observe that: • Median (117.0) and mean (121.65625) of Glucose lie very close to each other i.e. the distribution is more or less symmetric and uniform.

[23]:
```
print("Rows with Glucose value of 0: ", df_cp[df_cp["Glucose"] == 0].shape[0])
```

```
Rows with Glucose value of 0:  0
```

[21]:
```
#BloodPressure
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="BloodPressure", kde=True, ax=ax1[0],
 ↪color='magenta')
ax1[0].set_title('Histogram of BloodPressure')
sns.violinplot(data=df_cp, x="BloodPressure", ax=ax1[1], color='lime')
ax1[1].set_title('Violin Plot of BloodPressure')
plt.show()
```



[22]:
```
print("Median of Blood Pressure: ", df_cp["BloodPressure"].median())
print("Maximum of Blood Pressure: ", df_cp["BloodPressure"].max())
print("Mean of Pressure: ", df_cp["BloodPressure"].mean())
```

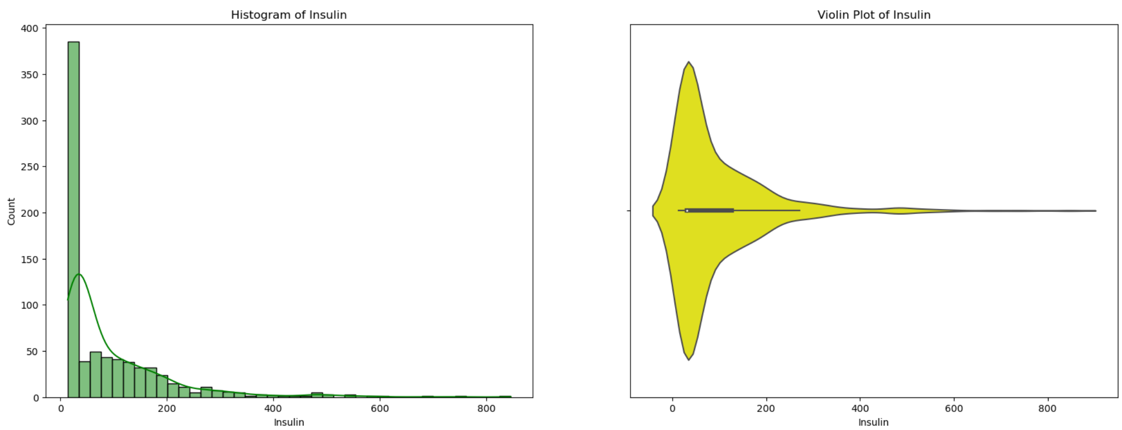
```
Median of Blood Pressure:  72.0
Maximum of Blood Pressure:  122
Mean of Pressure:  72.38671875
```

[23]:
```
print("Rows with BloodPressure value of 0: ", df_cp[df_cp["BloodPressure"] ==␣
↪0].shape[0])
```

```
Rows with BloodPressure value of 0:  0
```

From above we can observe that mean of blood pressure is 72.3867, Median is 72 and maximum blood pressure is 122.

[24]:
```
#Insulin
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="Insulin", kde=True, ax=ax1[0], color='green')
ax1[0].set_title('Histogram of Insulin')
sns.violinplot(data=df_cp, x="Insulin", ax=ax1[1], color='yellow')
ax1[1].set_title('Violin Plot of Insulin')
plt.show()
```

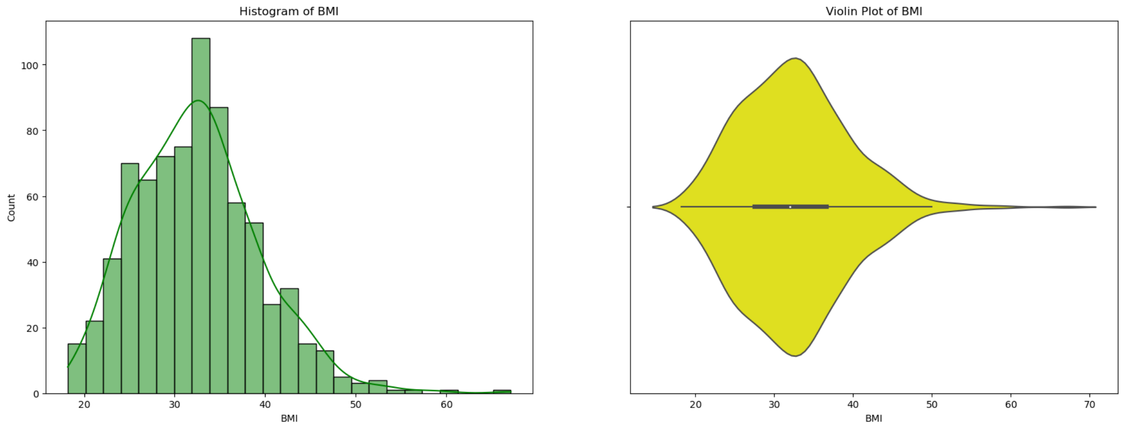


[25]:
```
print("Median of Insulin: ", df_cp["Insulin"].median())
print("Maximum of Insulin: ", df_cp["Insulin"].max())
print("Mean of Insulin: ", df_cp["Insulin"].mean())
```

```
Median of Insulin:  31.25
Maximum of Insulin:  846.0
Mean of Insulin:  94.65234375
```

From above we can observe that mean of insulin is 94.65, median is 31.25 and maximum insulin is 846

[26]:
```
#Analysis of BMI
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="BMI", kde=True, ax=ax1[0], color='green')
ax1[0].set_title('Histogram of BMI')
sns.violinplot(data=df_cp, x="BMI", ax=ax1[1], color='yellow')
ax1[1].set_title('Violin Plot of BMI')
plt.show()
```



[27]:
```
print("Median of BMI: ", df_cp["BMI"].median())
print("Maximum of BMI: ", df_cp["BMI"].max())
print("Mean of BMI: ", df_cp["BMI"].mean())
```

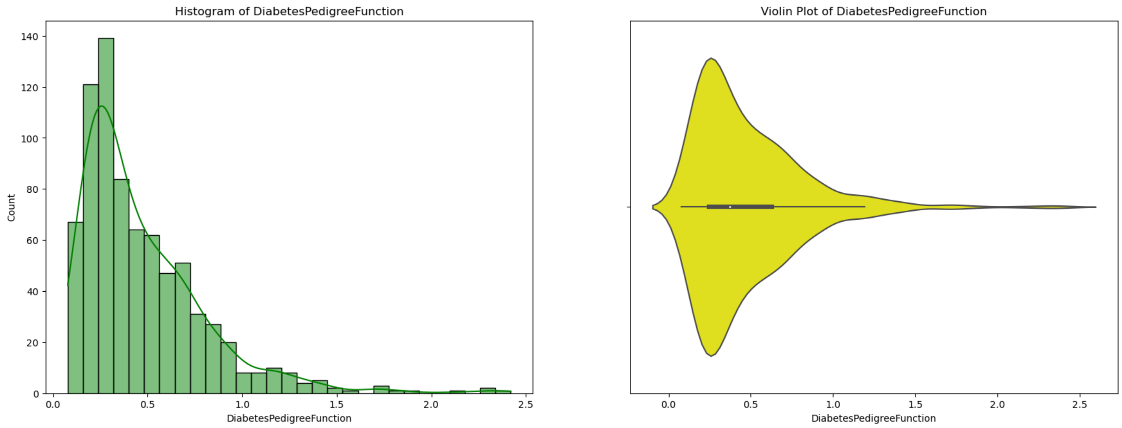
```
Median of BMI:  32.0
Maximum of BMI:  67.1
Mean of BMI:  32.45091145833333
```

[28]:
```
print("Rows with BMI value of 0: ", df_cp[df_cp["BMI"] == 0].shape[0])
```

```
Rows with BMI value of 0:  0
```

We observe that: • Median (32.0) and Mean (32.4509) of BMI are very close to each other. Thus, the distribution is more or less symmetric and uniform • Maximum BMI is 67.1.

[29]:
```
#Analysis of Diabetes Pedigree Function
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="DiabetesPedigreeFunction", kde=True, ax=ax1[0],␣
 ↪color='green')
ax1[0].set_title('Histogram of DiabetesPedigreeFunction')
sns.violinplot(data=df_cp, x="DiabetesPedigreeFunction", ax=ax1[1],␣
 ↪color='yellow')
ax1[1].set_title('Violin Plot of DiabetesPedigreeFunction')
plt.show()
```

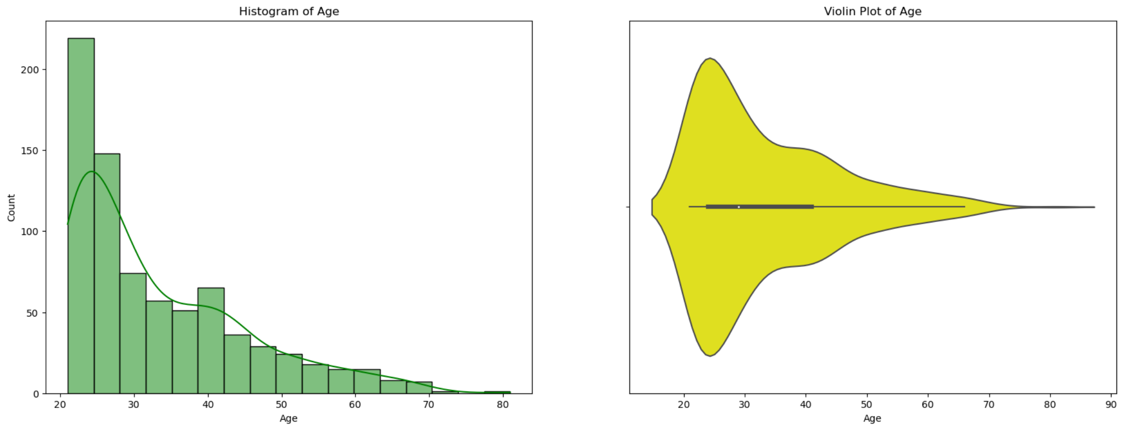


[30]:
```
print("Median of DiabetesPedigreeFunction: ", df_cp["DiabetesPedigreeFunction"].
  ↪median())
print("Maximum of DiabetesPedigreeFunction: ",␣
  ↪df_cp["DiabetesPedigreeFunction"].max())
print("Mean of DiabetesPedigreeFunction: ", df_cp["DiabetesPedigreeFunction"].
  ↪mean())
```

```
Median of DiabetesPedigreeFunction:  0.3725
Maximum of DiabetesPedigreeFunction:  2.42
Mean of DiabetesPedigreeFunction:  0.47187630208333325
```

We observe that: • Violin plot distribution is dense in the interval 0.0 - 1.0 • The histogram is higly skewed on the left side.

[31]:
```
#Analysis of Age
fig1, ax1 = plt.subplots(1, 2, figsize=(20, 7))
sns.histplot(data=df_cp, x="Age", kde=True, ax=ax1[0], color='green')
ax1[0].set_title('Histogram of Age')
sns.violinplot(data=df_cp, x="Age", ax=ax1[1], color='yellow')
ax1[1].set_title('Violin Plot of Age')
plt.show()
```



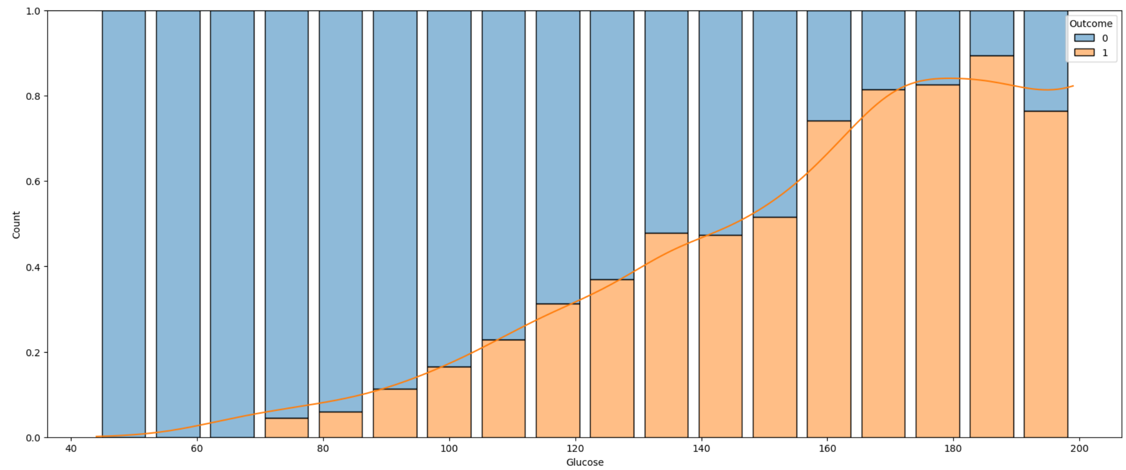
[32]:
```
print("Median of Age: ", df_cp["Age"].median())
print("Maximum of Age: ", df_cp["Age"].max())
print("Mean of Age: ", df_cp["Age"].mean())
```

```
Median of Age:  29.0
Maximum of Age:  81
Mean of Age:  33.240885416666664
```

From above we can observe that Mean of age is 33.24, Median is 29, and maximum age is 81.

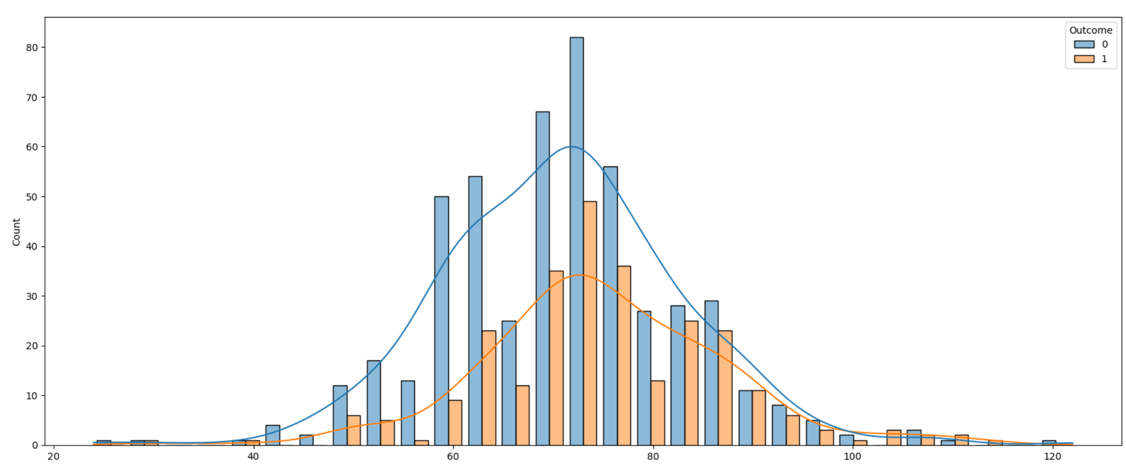
1.2 Multivariate Analysis

[33]:
```
#Analysis of Glucose and Outcome
fig15, ax15 = plt.subplots(figsize=(20, 8))

sns.histplot(data=df_cp, x="Glucose", hue="Outcome", shrink=0.8,␣
 ↪multiple="fill", kde=True, ax=ax15)
plt.show()
```



From the above plot, we see a positive linear correlation. • As the value of Glucose increases, the count of patients having diabetes increases i.e. value of Outcome as 1, increases. • Also, after the Glucose value of 125, there is a steady increase in the number of patients having Outcome of 1.

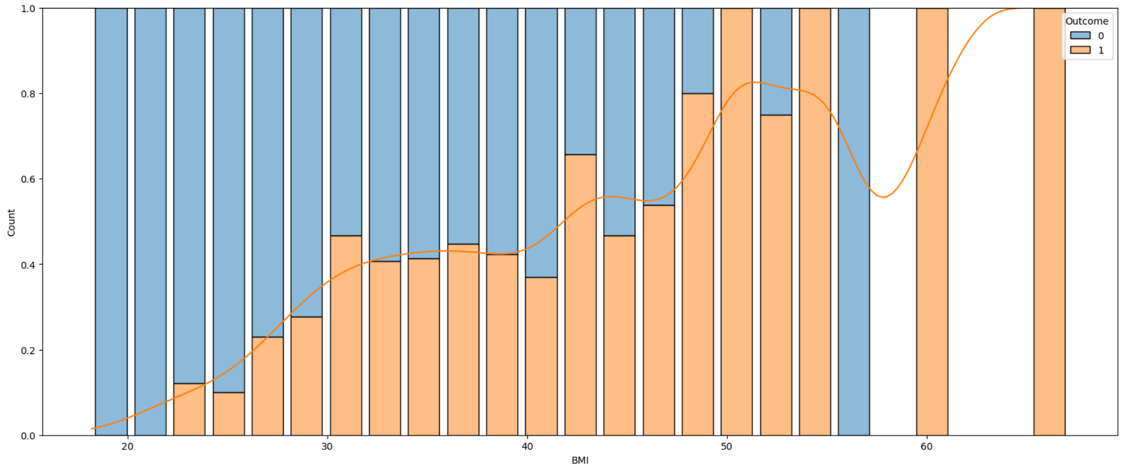
[34]:
```
#Analysis of BloodPressure and Outcome
fig16, ax16 = plt.subplots(figsize=(20, 8))

sns.histplot(data=df_cp, x="BloodPressure", hue="Outcome", shrink=0.8,␣
 ↪multiple="dodge", kde=True, ax=ax16,)
plt.show()
```



BloodPressure values greater than 82, count of patients with Outcome as 1.

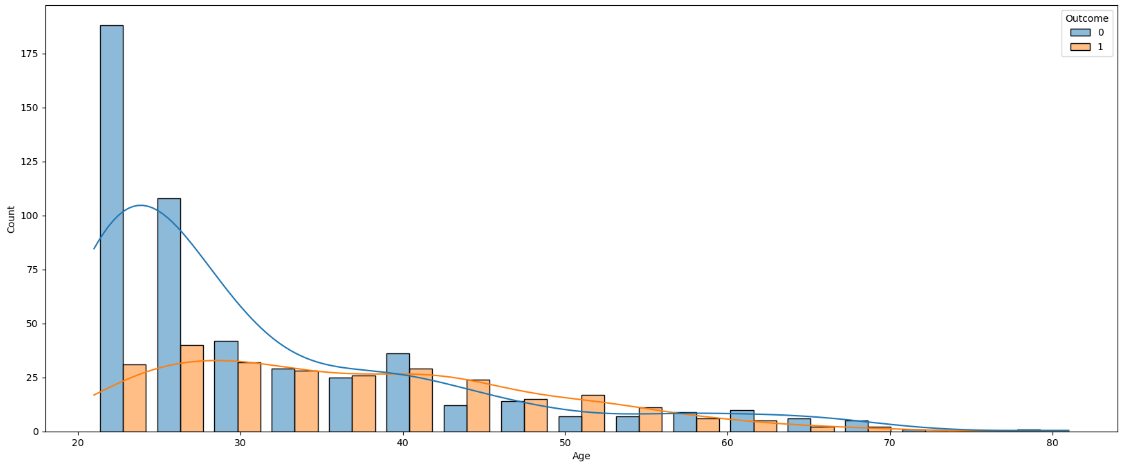
[35]:

```
#Analysis of BMI and Outcome
fig17, ax17 = plt.subplots(figsize=(20, 8))

sns.histplot(data=df_cp, x="BMI", hue="Outcome", shrink=0.8, multiple="fill",␣
 ↪kde=True, ax=ax17)
plt.show()
```



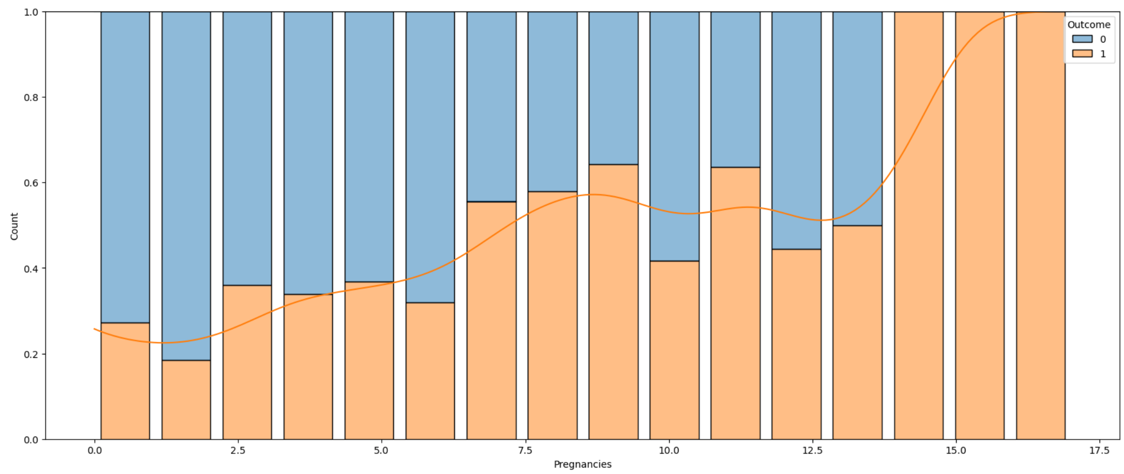
[36]:

```
#Analysis of Age and Outcome
fig18, ax18 = plt.subplots(figsize=(20, 8))

sns.histplot(data=df_cp, x="Age", hue="Outcome", shrink=0.8, multiple="dodge",␣
 ↪kde=True, ax=ax18)
plt.show()
```



The number of patients having diabetes is less than the number of people having diabetes.

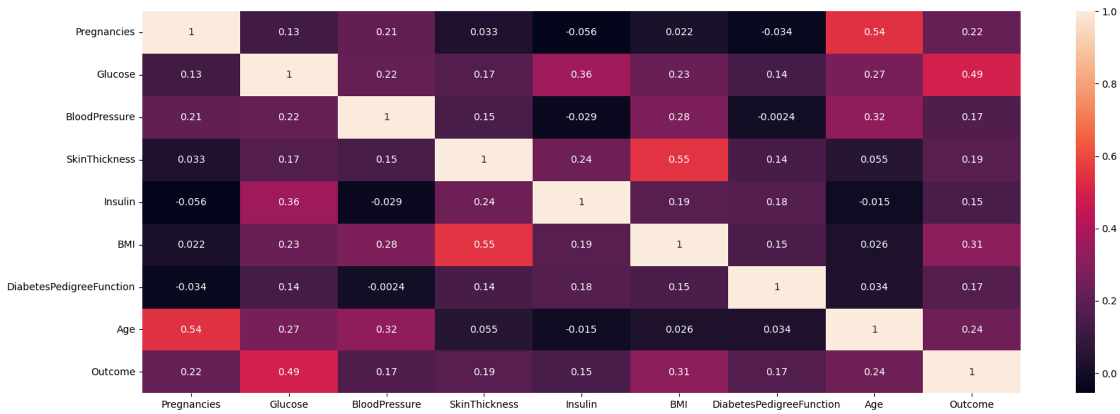
[37]:
```
#Analysis of Pregnancies and Outcome
fig19, ax19 = plt.subplots(figsize=(20, 8))

sns.histplot(data=df_cp, x="Pregnancies", hue="Outcome", shrink=0.8,␣
 ↪multiple="fill", kde=True, ax=ax19)
plt.show()
```



There is some positive linear correlation of Pregnancies with Outcome.

[39]:
```
#Analyzing Correlations
# The 2D correlation matrix
corr_matrix = df_cp.corr()
```

[40]:
```
# Plotting the heatmap of corr

fig20, ax20 = plt.subplots(figsize=(20, 7))
dataplot = sns.heatmap(data=corr_matrix, annot=True, ax=ax20)
plt.show()
```



[41]:
```
correlation=data.corr()
print(correlation)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528      -0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.073535  0.017683                 -0.033523
Glucose                   0.331357  0.221071                  0.137337
BloodPressure             0.088933  0.281805                  0.041265
SkinThickness             0.436783  0.392573                  0.183928
Insulin                   1.000000  0.197859                  0.185071
BMI                       0.197859  1.000000                  0.140647
DiabetesPedigreeFunction  0.185071  0.140647                  1.000000
Age                      -0.042163  0.036242                  0.033561
Outcome                   0.130548  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
BloodPressure             0.239528  0.065068
SkinThickness            -0.113970  0.074752
Insulin                  -0.042163  0.130548
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
```

```
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

[42]:
```
corr_matrix["Outcome"].sort_values(ascending=False)
```

[42]:
```
Outcome                     1.000000
Glucose                     0.492782
BMI                         0.312249
Age                         0.238356
Pregnancies                 0.221898
SkinThickness               0.189065
DiabetesPedigreeFunction    0.173844
BloodPressure               0.165723
Insulin                     0.148457
Name: Outcome, dtype: float64
```

We can observe that: • Glucose has the maximum positive linear correlation with Outcome. • Insulin has the lowest positive linear correlation with Outcome. • No feature has a negative linear correlation with Outcome .

[43]:
```
#TRAINING THE MODEL WITH THE HELP OF TRAIN TEST SPLIT
from sklearn.model_selection import train_test_split

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

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

[44]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LogisticRegression()
model.fit(X_train_scaled, Y_train)
```

[44]:
```
LogisticRegression()
```

[45]:
```
# Make predictions on the test data
predictions = model.predict(X_test)
predicted_probabilities = model.predict_proba(X_test)
from sklearn.metrics import accuracy_score, classification_report

# Calculate accuracy
accuracy = accuracy_score(Y_test, predictions)
print("Accuracy:", accuracy)
```

```python
# Generate a classification report
report = classification_report(Y_test, predictions, zero_division=0)

print("Classification Report:\n", report)
```

```
Accuracy: 0.36363636363636365
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00        98
           1       0.36      1.00      0.53        56

    accuracy                           0.36       154
   macro avg       0.18      0.50      0.27       154
weighted avg       0.13      0.36      0.19       154


C:\Users\rohit\anaconda3\Lib\site-packages\sklearn\base.py:457: UserWarning: X
has feature names, but LogisticRegression was fitted without feature names
  warnings.warn(
C:\Users\rohit\anaconda3\Lib\site-packages\sklearn\base.py:457: UserWarning: X
has feature names, but LogisticRegression was fitted without feature names
  warnings.warn(
```

[46]:

```python
from sklearn.metrics import precision_score, recall_score, f1_score
# Calculate precision
precision = precision_score(Y_test, predictions)
# Calculate recall
recall = recall_score(Y_test, predictions)
# Calculate F1-score
f1 = f1_score(Y_test, predictions)

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

```
Precision: 0.36363636363636365
Recall: 1.0
F1-score: 0.5333333333333333
```