

EDA, FE and Logistic Regression (Classification) Models (Diabetes Dataset)

Aryan Shriva

Linkedin: <https://www.linkedin.com/in/aryan-shriva-0811/> (<https://www.linkedin.com/in/aryan-shriva-0811/>).

GitHub <https://github.com/AryanShriva/Machine-Learning> (<https://github.com/AryanShriva/Machine-Learning>).

1. EDA and FE

1. Data Profiling
2. Statistical analysis
3. Graphical Analysis
4. Data Cleaning
5. Data Scaling
6. Outlier Trimming

2. Logistic Regression (Classification) Models

1. Logistic Regression
2. Performance metrics for above models

Dataset: <https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv> (<https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv>)

1.0 Importing Dataset

```
In [2]: 1 import pandas as pd
2 import numpy as np
3
4 ### Visualisation libraries
5 import seaborn as sns
6 import matplotlib.pyplot as plt
7 %matplotlib inline
8
9 ### To ignore warnings
10 import warnings
11 warnings.filterwarnings('ignore')
12
13 ### Machine Learning Libraries
14 from sklearn.model_selection import train_test_split
15 from sklearn.preprocessing import StandardScaler
16 from sklearn.linear_model import LogisticRegression
17 from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
18
19 ### To be able to see maximum columns on screen
20 pd.set_option('display.max_columns', 500)
```

```
In [3]: 1 dataset=pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv")
```

```
In [4]: 1 ### exporting file to csv for future use
2 dataset.to_csv("diabetes.csv")
```

1.1 Stastical Analysis

```
In [5]: 1 dataset.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6		0.627
1	1	85	66	29	0	26.6		0.351
2	8	183	64	0	0	23.3		0.672
3	1	89	66	23	94	28.1		0.167
4	0	137	40	35	168	43.1		2.288

```
In [6]: 1 dataset.columns
```

```
Out[6]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
       dtype='object')
```

In [7]: 1 dataset.describe()

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [8]: 1 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 [9]: 1 dataset.shape

Out[9]: (768, 9)

1.2 Checking Missing values

```
In [10]: 1 dataset.isnull().sum()
```

```
Out[10]: Pregnancies      0  
Glucose          0  
BloodPressure    0  
SkinThickness    0  
Insulin          0  
BMI              0  
DiabetesPedigreeFunction 0  
Age              0  
Outcome          0  
dtype: int64
```

```
In [11]: 1 for feature in dataset.columns:  
2     print("{} has {} no of unique categories".format(feature, dataset[featur
```

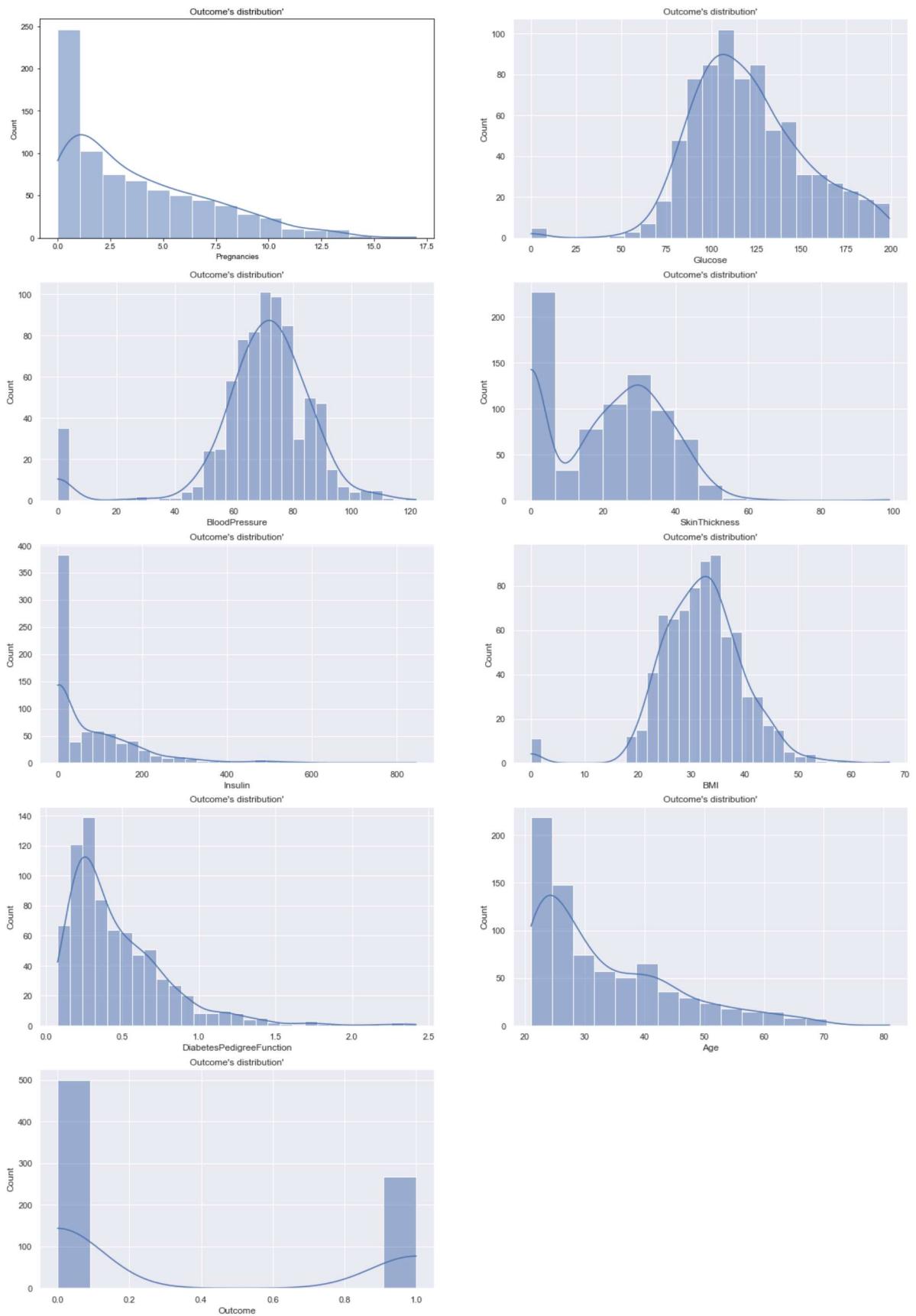
```
Pregnancies has 17 no of unique categories  
Glucose has 136 no of unique categories  
BloodPressure has 47 no of unique categories  
SkinThickness has 51 no of unique categories  
Insulin has 186 no of unique categories  
BMI has 248 no of unique categories  
DiabetesPedigreeFunction has 517 no of unique categories  
Age has 52 no of unique categories  
Outcome has 2 no of unique categories
```

2.0 Graphical Analysis

2.1 Checking Distribution of features

In [12]:

```
1 plt.figure(figsize=(20,30))
2 for i in enumerate(dataset.columns):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(7,5)})
5     sns.histplot(data=dataset, x=i[1], kde=True)
6     plt.title("{}'s distribution".format(feature))
```



Observations

1. Pregnancies has right skewed distribution, this indicates this feature has outliers towards right side of distribution.

2. Glucose has outliers towards left side of distribution.
 3. BloodPressure has outliers towards left side of distribution.
 4. Insulin has right skewed distribution, this indicates this feature has outliers towards right side of distribution.
 5. BMI has outliers towards left side of distribution.
 6. DiabetesPedigreeFunction has outliers towards left side of distribution.
 7. Age has outliers towards left side of distribution.
8. Features like Glucose, BloodPressure, SkinThickness, Insulin, BMI have lot of zero values so replace these values with its mean.

2.2 Replacing zero values with mean and rechecking Distribution of features

Note: In place of mean we can also use median, mode or any random value.

```
In [13]: 1  ### creating copy of dataset for further analysis so that we can also perform
2
3 data=dataset.copy()
4 data.head()
```

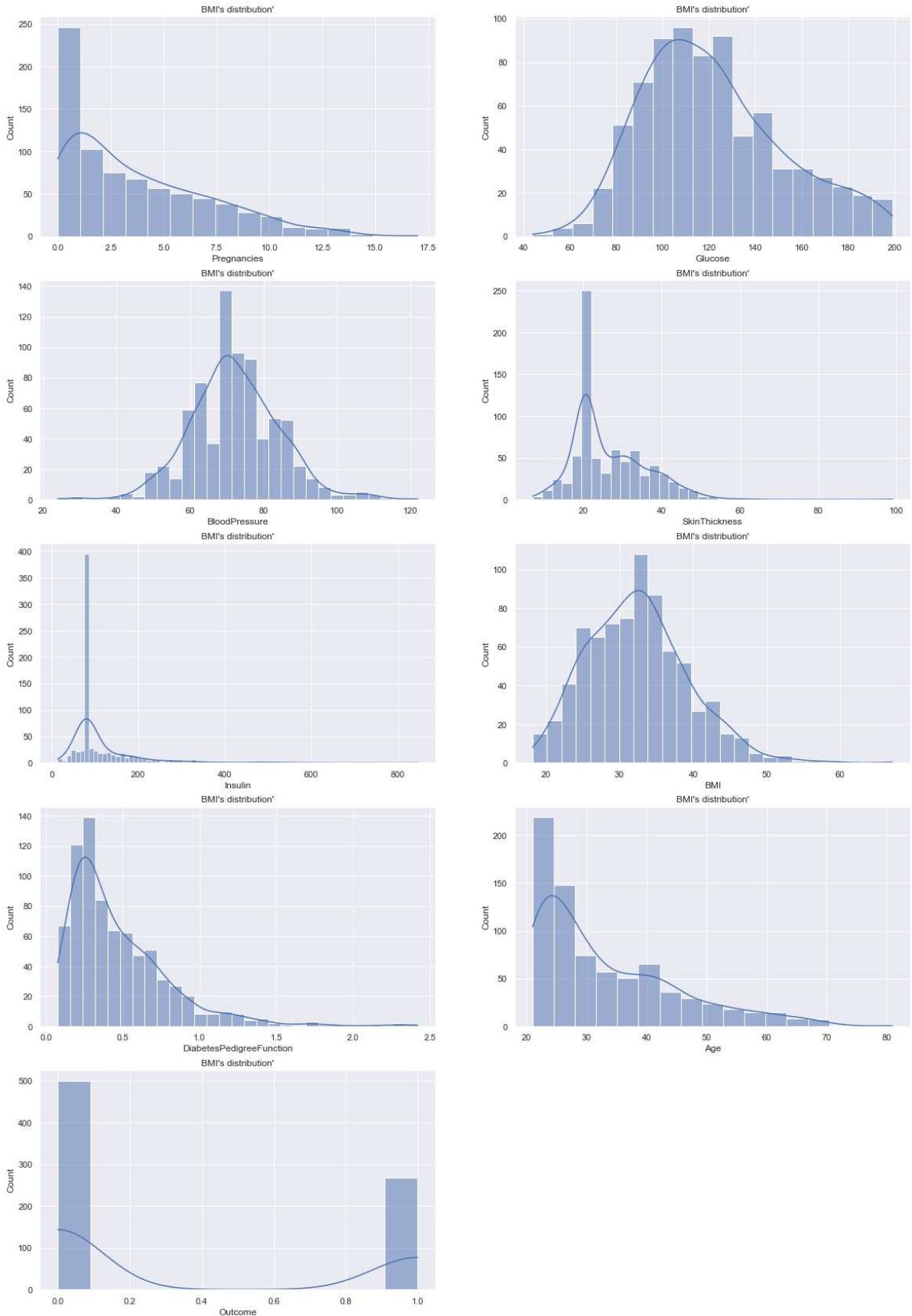
Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6		0.627
1	1	85	66	29	0	26.6		0.351
2	8	183	64	0	0	23.3		0.672
3	1	89	66	23	94	28.1		0.167
4	0	137	40	35	168	43.1		2.288

```
In [14]: 1  ### Replacing zero values in feature with mean values of that feature
2
3 for feature in ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
4     data[feature]=data[feature].replace(0,data[feature].mean())
```

In [15]:

```
1 plt.figure(figsize=(20,30))
2 for i in enumerate(data.columns):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(7,5)})
5     sns.histplot(data=data, x=i[1], kde=True)
6     plt.title("{}'s distribution".format(feature))
```



Observations

- After replacing zero values with means of these features Glucose, BloodPressure, SkinThickness, Insulin, BMI the distribution skewness managed a little bit.

2.3 Checking Outliers in independent features

```
In [16]: 1 ### Getting independent features
2 independent_features=[feature for feature in data.columns if feature not in
3 print(independent_features)

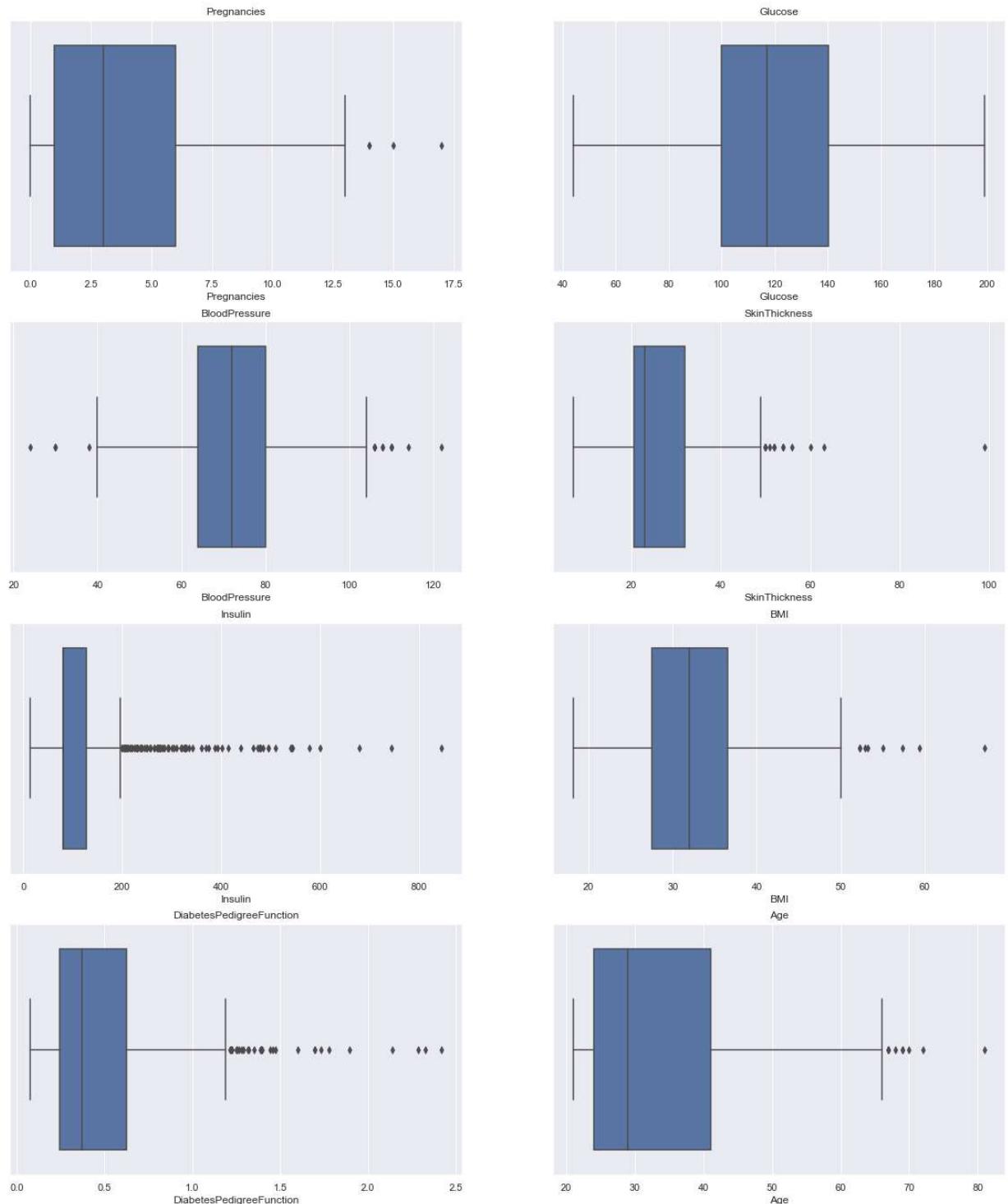
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age']
```

In [17]:

```

1 plt.figure(figsize=(20,30))
2 for i in enumerate(independent_features):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(7,5)})
5     sns.boxplot(data=data, x=i[1])
6     plt.title("{}".format(i[1]))

```



Observations

Note: Some outliers are already handled when we replaced the zero values with mean.

1. Glucose has zero outliers.
2. Pregnancies has some outliers on upper boundary side.
3. BloodPressure has outliers on both sides of boundary.
4. SkinThickness, BMI and Age have outliers on upper boundary side.
5. Insulin and DiabetesPedigreeFunction has large no of outliers on upper boundary side.

2.4 Trimming outliers

```
In [18]: 1 def outlier_trimmer_upper(data_set, feature, trimming_value):
2     """This function takes dataset, feature to be trimmed and the value after
3     and returns the dataset after trimming outliers in input feature.
4     """
5     threshold=data_set[feature].quantile(trimming_value/100)
6     data_set=data_set[data_set[feature]<threshold]
7     return data_set
8
9 def outlier_trimmer_lower(data_set, feature, trimming_value):
10    """This function takes dataset, feature to be trimmed and the value after
11    and returns the dataset after trimming outliers in input feature.
12    """
13    threshold=data_set[feature].quantile(trimming_value/100)
14    data_set=data_set[data_set[feature]>threshold]
15    return data_set
```

```
In [19]: 1 ### shape of data before trimming
2 data.shape
```

Out[19]: (768, 9)

```
In [20]: 1 ### removing 1 percent outliers in BloodPressure, SkinThickness, BMI and Age
2
3 for feature in ['BloodPressure', 'SkinThickness', 'BMI', 'Age']:
4     data=outlier_trimmer_upper(data, feature, 99)
```

```
In [21]: 1 ### shape of data after trimming
2 data.shape
```

Out[21]: (733, 9)

```
In [22]: 1 ### removing 2 percent outliers in Insulin and DiabetesPedigreeFunction as they have large no of outliers
2
3 for feature in ['Insulin', 'DiabetesPedigreeFunction']:
4     data=outlier_trimmer_upper(data, feature, 98)
```

```
In [23]: 1 data.shape
```

Out[23]: (703, 9)

```
In [24]: 1   ### removing 0.5 percent outliers in BMI, Glucose, BloodPressure on Lower si  
2  
3   for feature in ['Glucose', 'BloodPressure', 'BMI']:  
4       data=outlier_trimmer_lower(data, feature, 0.5)
```

```
In [25]: 1   data.shape
```

```
Out[25]: (688, 9)
```

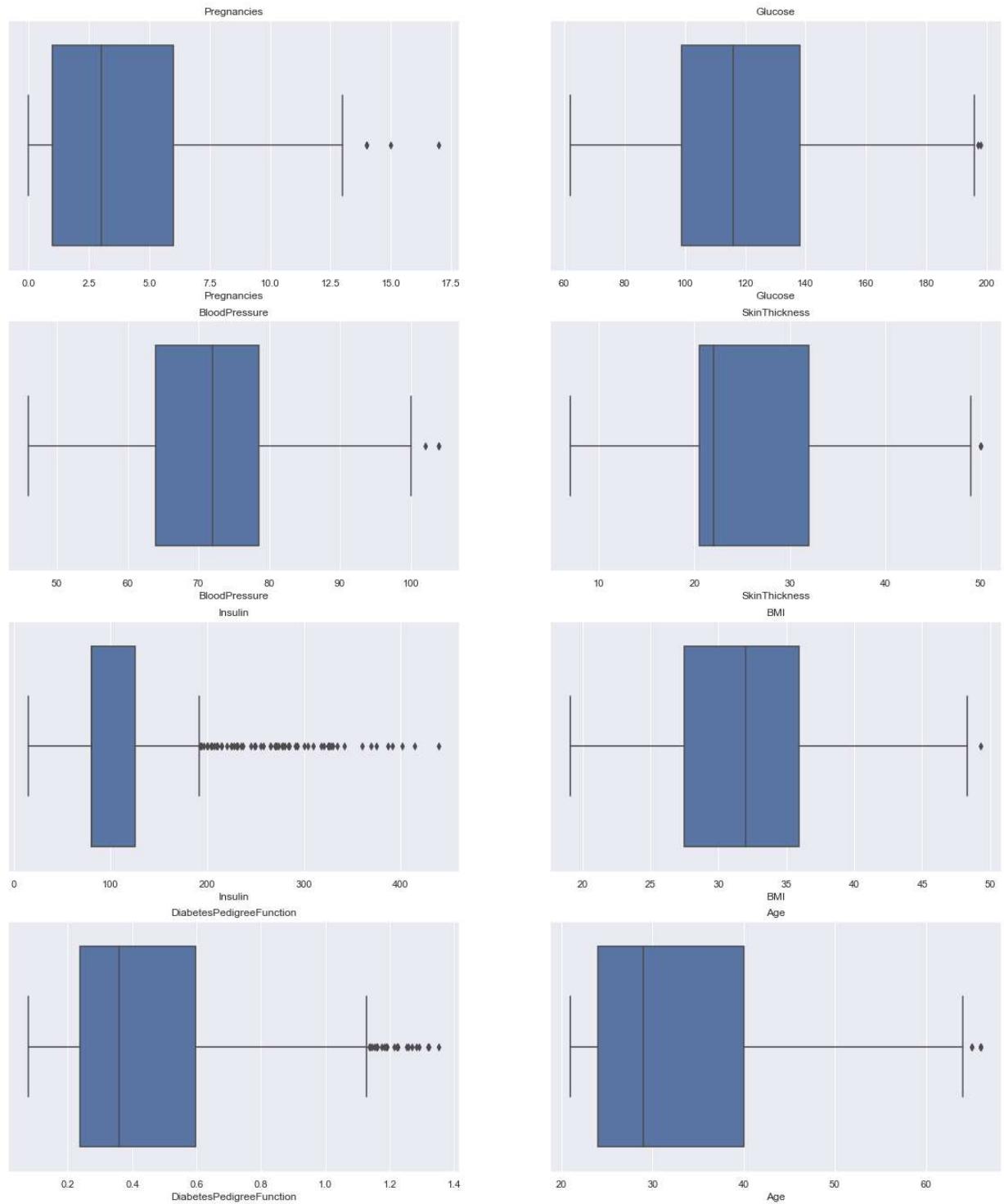
2.5 Re-checking outliers after trimming outliers in independent features

In [26]:

```

1 plt.figure(figsize=(20,30))
2 for i in enumerate(independent_features):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(7,5)})
5     sns.boxplot(data=data, x=i[1])
6     plt.title("{}".format(i[1]))

```



Observations

After Trimming Outliers

1. BloodPressure has very few outliers on left sides of boundary now.
2. SkinThickness has no outliers now.
3. BMI and Age have very few outliers on upper boundary side now.
4. No of outliers is reduced for Insulin and DiabetesPedigreeFunction.
5. Now no feature has outliers on lower side.

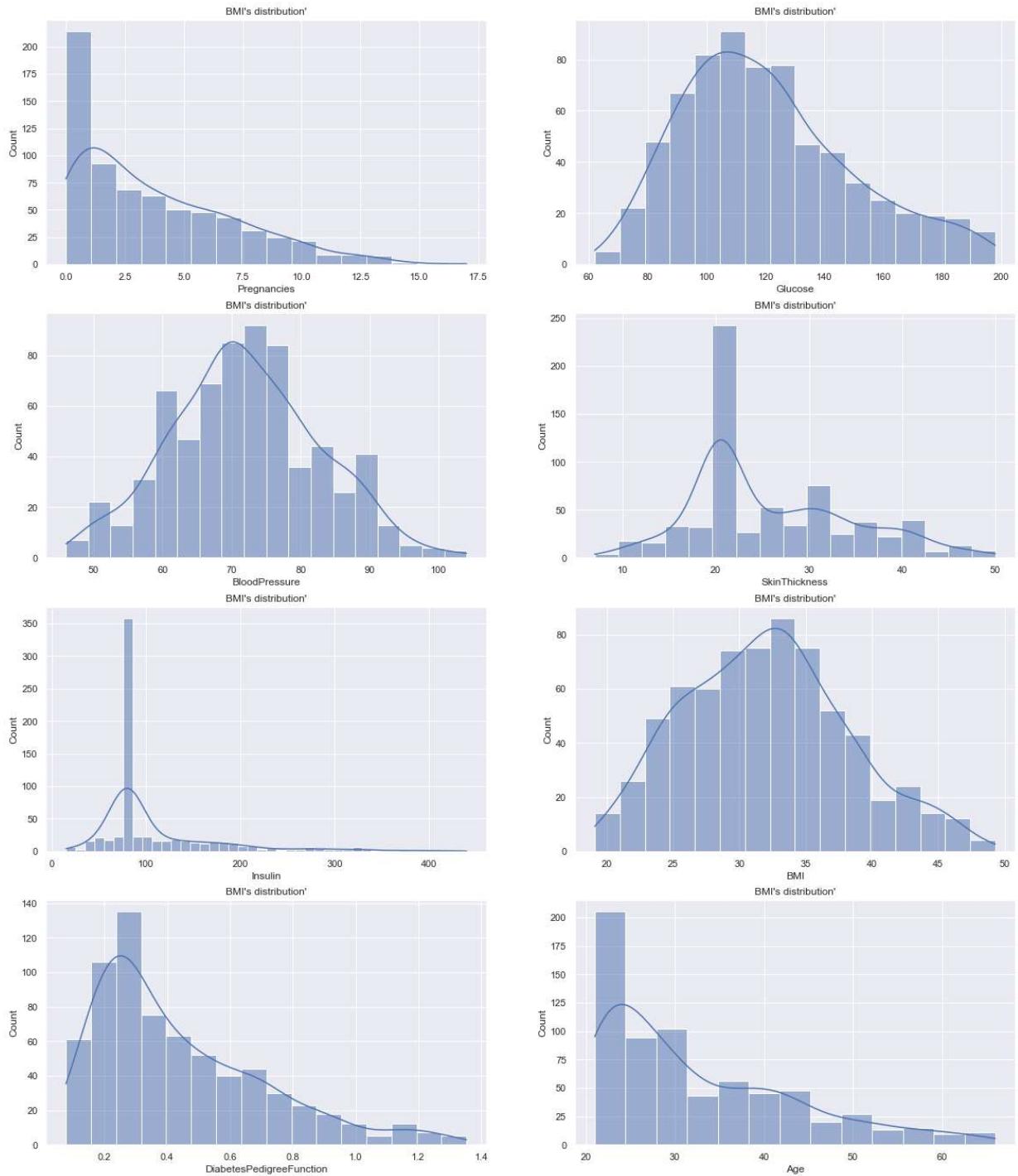
2.6 Rechecking distribution of independent feature after trimming outliers

In [27]:

```

1 plt.figure(figsize=(20,30))
2 for i in enumerate(independent_features):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(7,5)})
5     sns.histplot(data=data, x=i[1], kde=True)
6     plt.title("{}'s distribution".format(feature))

```

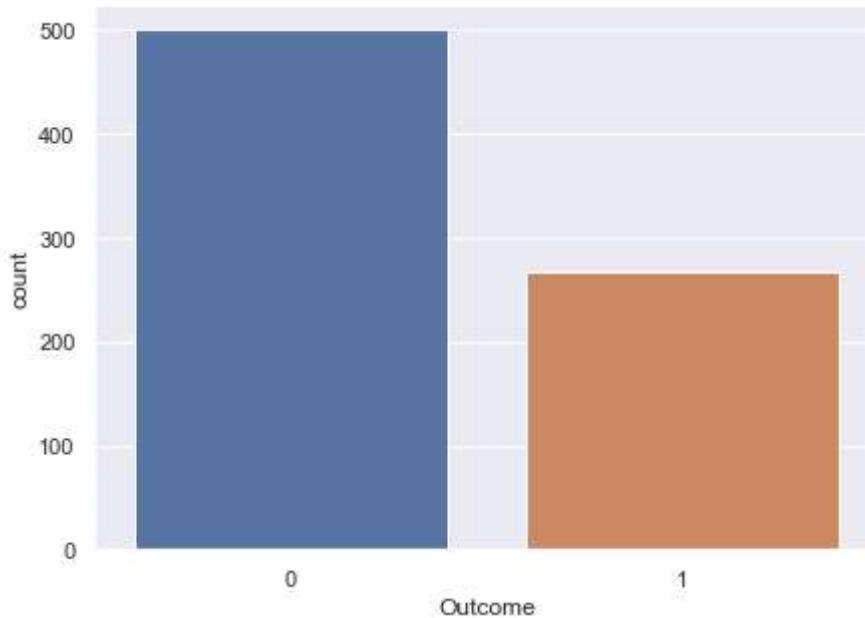


2.7 Checking imbalance of data before and after trimming outliers

2.7.1 Before trimming outliers

```
In [28]: 1 sns.countplot(data=dataset, x='Outcome')
```

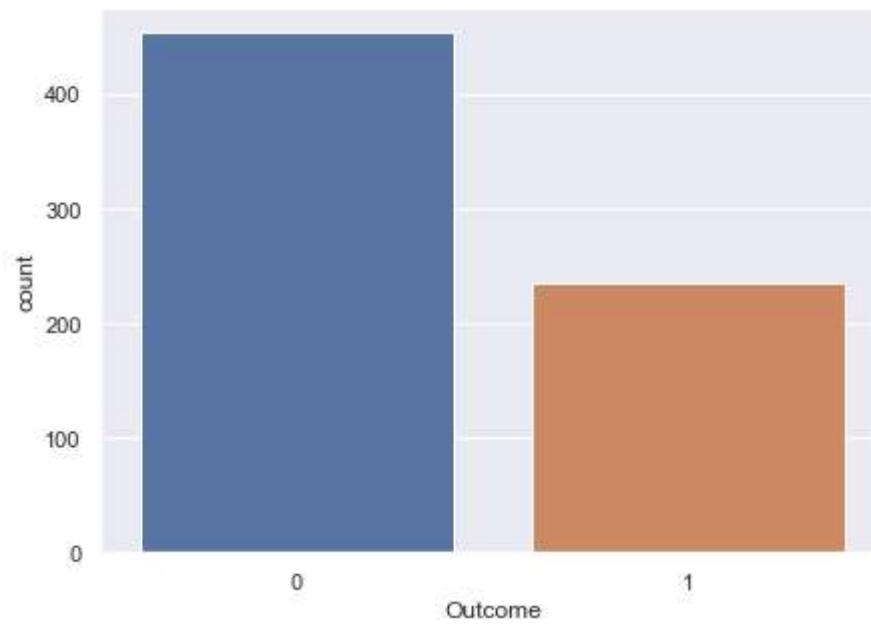
```
Out[28]: <AxesSubplot:xlabel='Outcome', ylabel='count'>
```



2.7.2 After trimming outliers

```
In [29]: 1 sns.countplot(data=data, x='Outcome')
```

```
Out[29]: <AxesSubplot:xlabel='Outcome', ylabel='count'>
```



Observations

1. We have imbalance in our dataset, lets not handle this imbalance and check accuracy, precision and recall
2. Then handle this imbalance and check accuracy, precision and recall

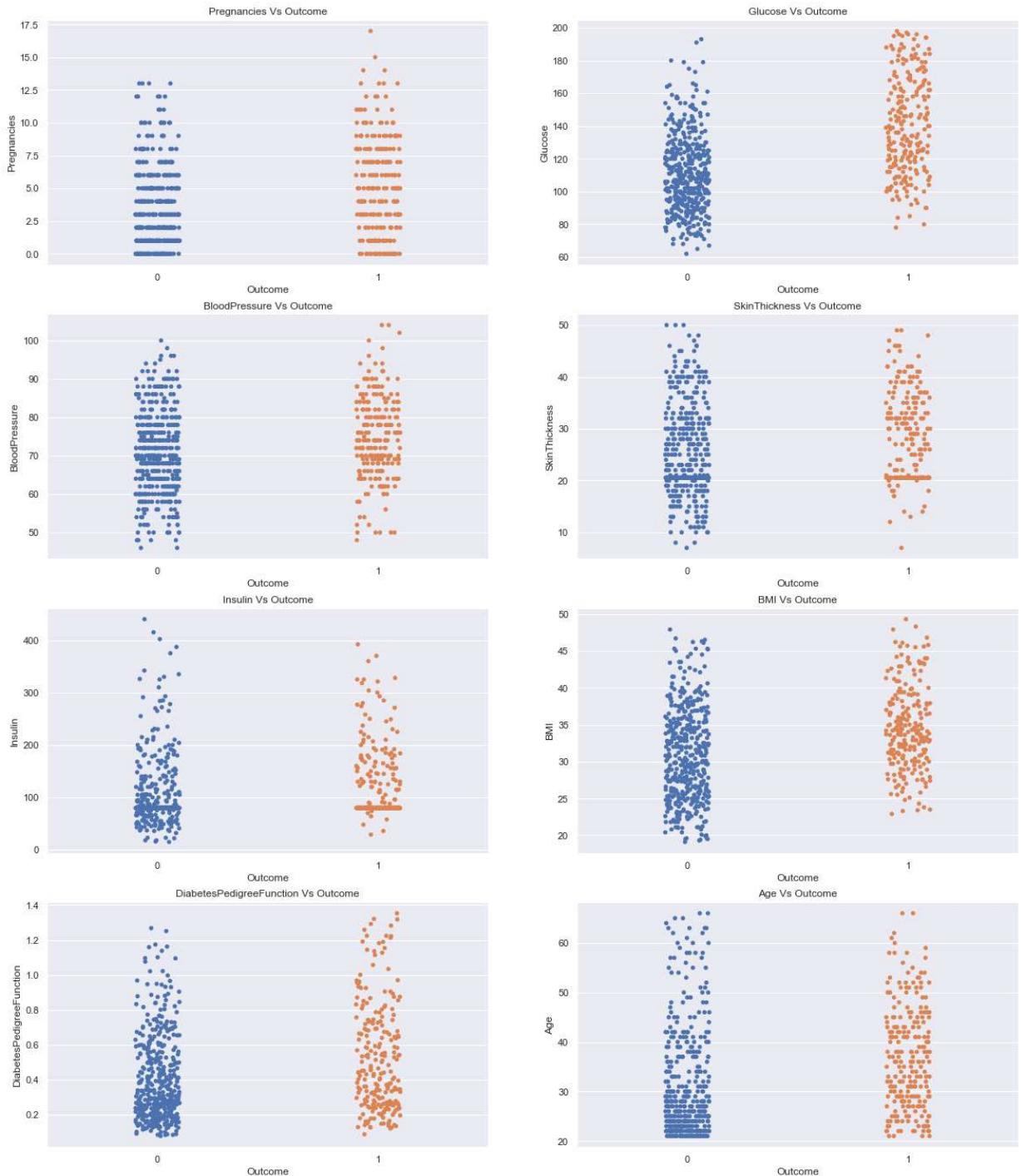
2.8 Relationship between independent features and dependent feature

In [30]:

```

1 plt.figure(figsize=(20,30))
2 for i in enumerate(independent_features):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(7,5)})
5     sns.stripplot(data=data, y=i[1], x='Outcome')
6     plt.title("{} Vs Outcome".format(i[1]))

```



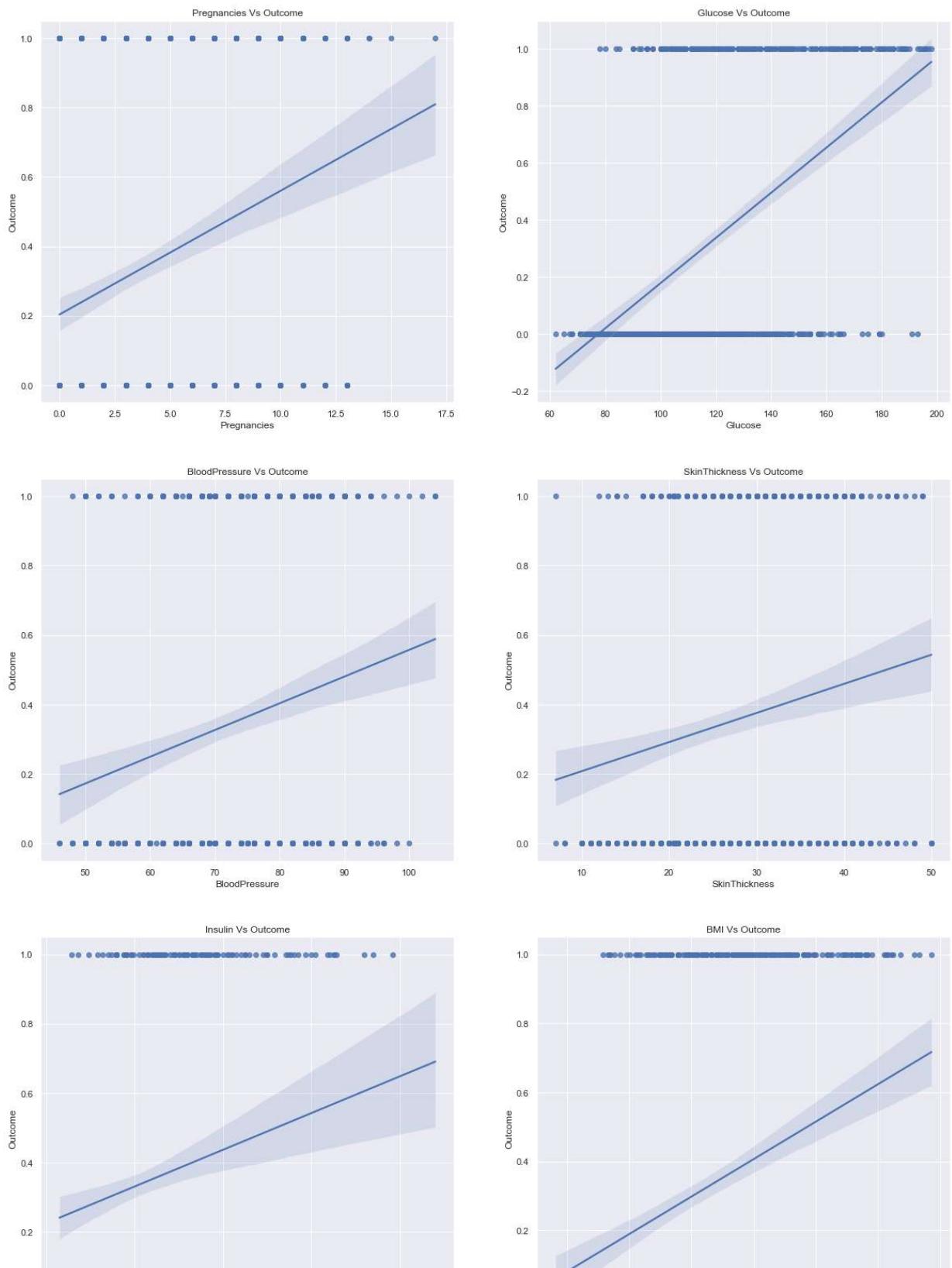
2.9 Checking the variation of slope between independent features and dependent feature

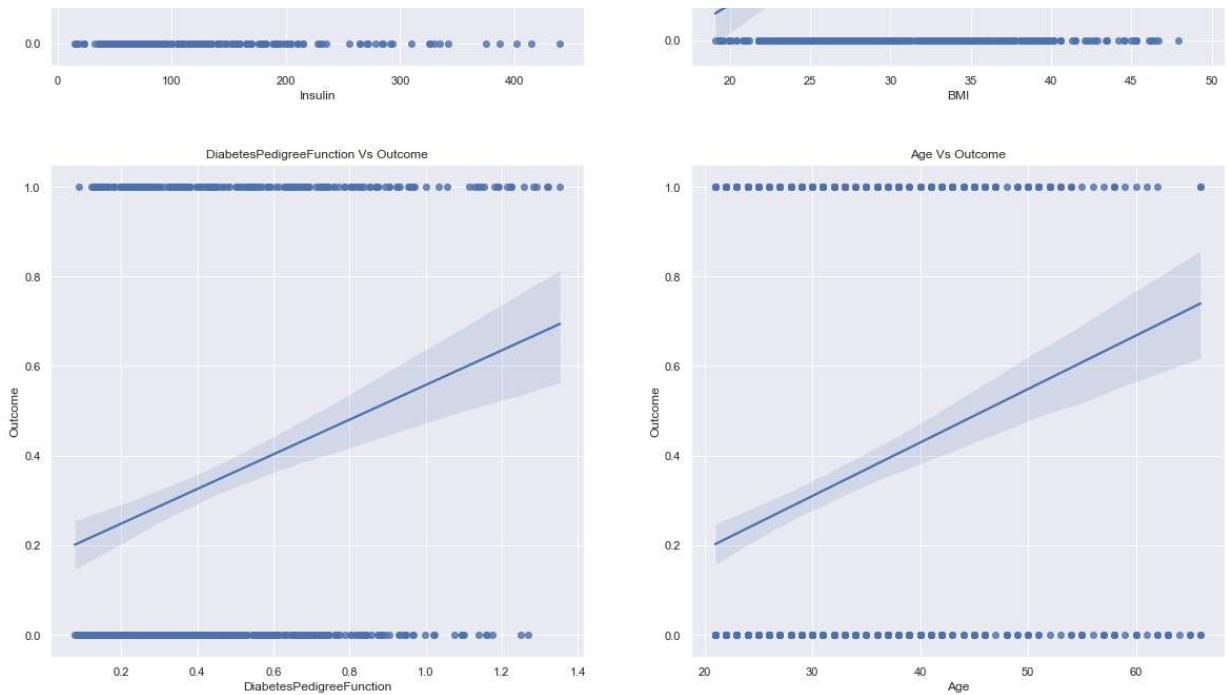
In [31]:

```

1 plt.figure(figsize=(20,50))
2 for i in enumerate(independent_features):
3     plt.subplot(5, 2, i[0]+1)
4     sns.set(rc={'figure.figsize':(8,10)})
5     sns.regplot(x=data[i[1]], y=data['Outcome'])
6     plt.xlabel(i[1])
7     plt.ylabel("Outcome")
8     plt.title("{} Vs Outcome".format(i[1]))

```





2.10 Checking correlation between independent features and dependent feature

```
In [32]: 1 corr=round(data[[feature for feature in data.columns]].corr(),2)
          2 corr
```

Out[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diak
Pregnancies	1.00	0.15	0.19	0.07	0.01	0.05	
Glucose	0.15	1.00	0.21	0.10	0.38	0.19	
BloodPressure	0.19	0.21	1.00	0.15	-0.00	-0.00	0.27
SkinThickness	0.07	0.10	0.15	1.00	0.26	0.26	0.53
Insulin	0.01	0.38	-0.00	0.26	1.00	0.20	0.20
BMI	0.05	0.19	0.27	0.53	0.20	1.00	
DiabetesPedigreeFunction	0.03	0.09	0.05	0.12	0.15	0.14	
Age	0.56	0.30	0.35	0.04	0.04	0.07	
Outcome	0.25	0.49	0.17	0.15	0.15	0.28	

```
In [33]: 1 ### Plotting heatmap for visualising the correlation between features
2 sns.set(rc={'figure.figsize':(15,10)})
3 sns.heatmap(data=corr, annot=True, cmap="YlGnBu", vmin=-1, vmax=1)
```

Out[33]: <AxesSubplot:>



3.0 Model Building

3.1 Getting independent features in dataset(X) and dependent feature in series(y)

In [34]: 1 data.head()

Out[34]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167
5	5	116.0	74.0	20.536458	79.799479	25.6	0.201

In [35]: 1 X=data.iloc[:, :-1]

2 y=data.iloc[:, -1]

3

4 X.head()

Out[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167
5	5	116.0	74.0	20.536458	79.799479	25.6	0.201

In [36]: 1 y.head()

Out[36]: 0 1

1 0

2 1

3 0

5 0

Name: Outcome, dtype: int64

3.2 Splitting data into Training and Test data

In [37]: 1 *### random state train test split will be same with all people using random_*

2

3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, ra

In [38]: 1 X_train.head()

Out[38]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
746	1	147.0	94.0	41.000000	79.799479	49.3	0
264	4	123.0	62.0	20.536458	79.799479	32.0	0
340	1	130.0	70.0	13.000000	105.000000	25.9	0
322	0	124.0	70.0	20.000000	79.799479	27.4	0
565	2	95.0	54.0	14.000000	88.000000	26.1	0



In [39]: 1 y_train.head()

Out[39]:

746	1
264	1
340	0
322	1
565	0

Name: Outcome, dtype: int64

In [40]: 1 X_test.head()

Out[40]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
705	6	80.0	80.0	36.000000	79.799479	39.8	0.1
728	2	175.0	88.0	20.536458	79.799479	22.9	0.3
242	3	139.0	54.0	20.536458	79.799479	25.6	0.4
687	1	107.0	50.0	19.000000	79.799479	28.3	0.1
638	7	97.0	76.0	32.000000	91.000000	40.9	0.8



In [41]: 1 y_test.head()

Out[41]:

705	0
728	0
242	1
687	0
638	1

Name: Outcome, dtype: int64

In [42]: 1 *### both will have same shape*
2 X_train.shape, y_train.shape

Out[42]: ((584, 8), (584,))

```
In [43]: 1 ### both will have same shape
          2 X_test.shape, y_test.shape
```

Out[43]: ((104, 8), (104,))

3.3 Standardisation/ feature scaling the dataset

```
In [44]: 1 ### Crating a standard scaler object
          2 scaler=StandardScaler()
          3 scaler
```

Out[44]: StandardScaler()

```
In [45]: 1 ### using fit_transform to Standardize the train data
          2 X_train=scaler.fit_transform(X_train)
          3 X_train
```

Out[45]: array([[-0.86019009, 0.94854966, 2.01983918, ..., 2.89360233,
 -0.31591105, -0.51401632],
 [0.04169683, 0.1172319 , -0.91556073, ..., 0.016956 ,
 -0.79724224, 0.20241693],
 [-0.86019009, 0.35969958, -0.18171075, ..., -0.99735282,
 0.09978406, -0.9617871],
 ...,
 [0.3423258 , -0.47161818, 0.91906421, ..., 1.26405701,
 -0.57845534, 0.47107939],
 [1.84547065, -0.64480938, 1.2859892 , ..., 2.27836583,
 2.52102579, 0.47107939],
 [1.54484168, 1.74522918, 0.18521424, ..., 2.01231762,
 -0.15182088, 0.91885017]])

In [46]:

```

1 ### here using transform only to avoid data Leakage
2 ### (training mean and training std will be used for standardisation when we
3 X_test=scaler.transform(X_test)
4 X_test

```

Out[46]: array([[6.42954768e-01, -1.37221242e+00, 7.35601720e-01,
 1.18101174e+00, -4.38881751e-01, 1.31394105e+00,
 -9.75918211e-01, -4.24462162e-01],
 [-5.59561115e-01, 1.91842038e+00, 1.46945170e+00,
 -6.29808456e-01, -4.38881751e-01, -1.49619322e+00,
 -4.32597403e-01, -9.61787095e-01],
 [-2.58932144e-01, 6.71443739e-01, -1.64941070e+00,
 -6.29808456e-01, -4.38881751e-01, -1.04723686e+00,
 -1.55467326e-01, -9.61787095e-01],
 [-8.60190086e-01, -4.36979943e-01, -2.01633569e+00,
 -8.09731648e-01, -4.38881751e-01, -5.98280493e-01,
 -9.61332418e-01, -3.34908006e-01],
 [9.43583739e-01, -7.83362343e-01, 3.68676732e-01,
 7.12601531e-01, -2.66275971e-01, 1.49684920e+00,
 1.55471696e+00, -6.62455397e-02],
 [-1.16081906e+00, -6.44809383e-01, -7.32098232e-01,
 -1.04393675e+00, -4.38881751e-01, -1.81212547e+00,
 -7.02434583e-01, -1.05134125e+00],
 [1.24421271e+00, 1.08710262e+00, 5.52139226e-01,
 7.12601531e-01, 1.56757517e+00, 1.82940947e+00,
 2.60227790e-01, 2.91971082e-01],
 [2.74735756e+00, -4.71618183e-01, -1.81710750e-01,
 -6.29808456e-01, -4.38881751e-01, 3.82772301e-01,
 -7.06081031e-01, 1.72483757e+00],
 [4.16968264e-02, -1.02583002e+00, 1.46945170e+00,
 2.46913981e+00, -8.36464981e-01, 9.64752771e-01,
 -3.01325261e-01, -3.34908006e-01],
 [-8.60190086e-01, -1.23365946e+00, -7.32098232e-01,
 -3.41321439e-01, 1.03576360e-01, 8.31728664e-01,
 9.61376125e-02, -4.24462162e-01],
 [-2.58932144e-01, -1.25235782e-01, -2.63767217e-01,
 -6.29808456e-01, -4.38881751e-01, -1.39642514e+00,
 -9.39453727e-01, -8.72232940e-01],
 [1.54484168e+00, 1.51870138e-01, -1.81710750e-01,
 8.29704083e-01, 4.52639381e+00, 5.82308462e-01,
 -5.93041131e-01, 1.12862771e-01],
 [-5.59561115e-01, -8.87277063e-01, 3.68676732e-01,
 -9.26834200e-01, -6.51538815e-01, -4.95560490e-02,
 7.45205424e-01, -8.72232940e-01],
 [-5.59561115e-01, -2.63788742e-01, 1.28598920e+00,
 1.88362705e+00, 7.97049480e-01, 1.08114887e+00,
 -7.24313273e-01, -4.24462162e-01],
 [-5.59561115e-01, 8.44634939e-01, -1.28248571e+00,
 8.29704083e-01, 4.11786636e-01, -4.95560490e-02,
 -8.25383581e-02, -6.93124629e-01],
 [3.42325797e-01, 1.67595270e+00, -7.32098232e-01,
 -6.29808456e-01, -4.38881751e-01, 1.66608126e-01,
 -1.12906904e+00, 7.39741860e-01],
 [1.84547065e+00, 1.46812326e+00, 1.10252671e+00,
 -6.29808456e-01, -4.38881751e-01, -6.98048574e-01,
 -9.57685969e-01, 1.90394588e+00],
 [-5.59561115e-01, -5.06256423e-01, 7.35601720e-01,

```

2.23493471e+00, 1.27477541e+00, 2.99632233e-01,
9.71285224e-01, -3.34908006e-01],
[-2.58932144e-01, -2.98426982e-01, -9.15560726e-01,
-6.29808456e-01, -4.38881751e-01, -1.54607726e+00,
-1.10354390e+00, -1.05134125e+00],
[-2.58932144e-01, -7.14085863e-01, -1.64941070e+00,
-8.09731648e-01, -3.43328540e-01, -1.04723686e+00,
-1.05978652e+00, -7.82678784e-01],
[-1.16081906e+00, -3.67703462e-01, 1.46945170e+00,
4.78396426e-01, -4.38881751e-01, 1.00096072e-01,
1.49637379e+00, 4.71079394e-01],
[ 2.44672859e+00, -9.56553543e-01, -9.15560726e-01,
-2.21496227e+00, 2.30727983e+00, -7.14676587e-01,
1.75527163e+00, 1.00840433e+00],
[ 3.42325797e-01, 1.17231898e-01, 1.85214238e-01,
1.64942195e+00, -4.82023164e-01, 3.66144287e-01,
-6.40444960e-01, -4.24462162e-01],
[-1.16081906e+00, -2.98426982e-01, -6.40366985e-01,
-6.29808456e-01, -4.38881751e-01, -1.21351699e+00,
7.85316356e-01, -1.55799695e-01],
[-8.60190086e-01, 2.90423099e-01, 9.19064214e-01,
-1.04393675e+00, 1.15149130e+00, -7.31304600e-01,
-1.20199801e+00, -9.61787095e-01],
[ 3.42325797e-01, 2.90423099e-01, 7.35601720e-01,
-6.29808456e-01, -4.38881751e-01, 4.49284354e-01,
-1.09625101e+00, 1.09795848e+00],
[ 3.42325797e-01, 2.33407926e+00, 3.68676732e-01,
1.27088770e-01, 1.52134363e+00, 1.94580556e+00,
2.14908805e+00, 1.81439173e+00],
[-2.58932144e-01, -4.71618183e-01, -1.64941070e+00,
-5.75526543e-01, 7.66228453e-01, -1.65952143e-01,
-5.56576647e-01, -7.82678784e-01],
[ 3.04798653e+00, -6.79447623e-01, 5.52139226e-01,
-1.07116335e-01, 1.16690181e+00, 7.81844623e-01,
-1.19002842e-01, 1.18751264e+00],
[ 1.24421271e+00, 1.33171781e-02, 1.28598920e+00,
-6.29808456e-01, -4.38881751e-01, -5.81652479e-01,
-6.76909444e-01, -9.61787095e-01],
[-8.60190086e-01, -2.63788742e-01, 7.35601720e-01,
2.23493471e+00, 3.65555094e-01, 4.82540381e-01,
-8.30060276e-01, -7.82678784e-01],
[-2.58932144e-01, 4.28976059e-01, 7.35601720e-01,
-6.29808456e-01, -4.38881751e-01, 4.16028328e-01,
-1.55467326e-01, 1.00840433e+00],
[ 4.16968264e-02, -1.25235782e-01, 1.75174383e-03,
-1.62944951e+00, -3.27918026e-01, -1.62921733e+00,
6.69660255e-02, 3.81525238e-01],
[-8.60190086e-01, 2.09161158e+00, -2.63767217e-01,
-6.29808456e-01, -4.38881751e-01, 1.89592152e+00,
-5.93041131e-01, 7.39741860e-01],
[-8.60190086e-01, -4.36979943e-01, 1.75174383e-03,
4.78396426e-01, -4.04970595e-01, -1.82580157e-01,
1.37239455e+00, -7.82678784e-01],
[-8.60190086e-01, -2.13210619e-02, 1.28598920e+00,
1.53231940e+00, 1.72168031e+00, 2.27836583e+00,
1.32499072e+00, -3.34908006e-01],
[-5.59561115e-01, 4.79554182e-02, -1.81710750e-01,

```

```

7.12601531e-01, -2.04633916e-01, 1.19754496e+00,
1.60941369e+00, -8.72232940e-01],
[ 4.16968264e-02, -7.14085863e-01, 1.75174383e-03,
-1.04393675e+00, -4.38881751e-01, -1.04723686e+00,
-5.49283751e-01, -4.24462162e-01],
[-5.59561115e-01, -1.19902122e+00, -6.40366985e-01,
-6.29808456e-01, -4.38881751e-01, 1.28068503e+00,
1.76985742e+00, -5.14016317e-01],
[-8.60190086e-01, -1.25235782e-01, -1.81710750e-01,
2.44191322e-01, -4.38881751e-01, -7.47932614e-01,
-8.77464105e-01, -1.05134125e+00],
[-1.16081906e+00, 1.57203798e+00, 3.68676732e-01,
2.00072960e+00, 2.26104829e+00, 2.66081014e+00,
-6.76909444e-01, -6.03570473e-01],
[ 4.16968264e-02, 2.90423099e-01, -1.81710750e-01,
-6.29808456e-01, -4.38881751e-01, 3.99400314e-01,
-5.16465715e-01, -7.82678784e-01],
[-2.58932144e-01, 9.83187900e-01, -5.48635738e-01,
-1.07116335e-01, -4.38881751e-01, 1.00096072e-01,
-6.87848789e-01, -9.61787095e-01],
[ 3.42325797e-01, -1.25235782e-01, 1.85214238e-01,
-6.29808456e-01, -4.38881751e-01, -1.04723686e+00,
-8.88403450e-01, -2.45353851e-01],
[-1.16081906e+00, 2.09161158e+00, 1.65291419e+00,
9.98621757e-03, -2.81686485e-01, 7.65216610e-01,
-4.76354783e-01, 2.02416927e-01],
[ 1.24421271e+00, 2.68046166e+00, 1.85214238e-01,
-6.29808456e-01, -4.38881751e-01, -9.97352816e-01,
2.72158045e+00, 5.60633549e-01],
[-8.60190086e-01, -3.67703462e-01, -1.28248571e+00,
-9.26834200e-01, 1.18986874e-01, -5.65024466e-01,
-8.22767379e-01, -9.61787095e-01],
[ 4.16968264e-02, 3.25061339e-01, -1.09902322e+00,
-1.629444951e+00, 1.89119596e+00, -7.31304600e-01,
3.00338722e-01, -1.55799695e-01],
[ 1.24421271e+00, -8.52638823e-01, 1.75174383e-03,
-6.29808456e-01, -4.38881751e-01, 8.15100650e-01,
1.47187890e-01, 2.17260835e+00],
[-1.16081906e+00, 6.36805499e-01, -1.09902322e+00,
1.06390919e+00, 9.04923077e-01, 4.49284354e-01,
3.25863860e-01, -1.05134125e+00],
[ 1.54484168e+00, 1.57203798e+00, 1.46945170e+00,
-6.29808456e-01, -4.38881751e-01, -2.49092210e-01,
-5.20112164e-01, 1.45617510e+00],
[ 1.54484168e+00, 1.19101734e+00, 5.52139226e-01,
4.78396426e-01, -1.27581347e-01, -1.65952143e-01,
-1.02332204e+00, 1.09795848e+00],
[ 4.16968264e-02, 3.94337819e-01, -3.65173244e-01,
-5.75526543e-01, 8.89512563e-01, 1.99864153e-01,
-1.03790783e+00, -4.24462162e-01],
[-8.60190086e-01, 1.01782614e+00, -3.65173244e-01,
3.61293874e-01, 2.88502525e-01, -4.32000358e-01,
-3.48729090e-01, 8.29296016e-01],
[ 4.16968264e-02, -5.75532903e-01, -1.09902322e+00,
8.29704083e-01, 1.29018592e+00, -1.31328507e+00,
1.90112956e+00, 2.33086158e-02],
[ 4.16968264e-02, 7.40720219e-01, 1.85214238e-01,

```

```

-6.29808456e-01, -4.38881751e-01, -7.14676587e-01,
-7.31606170e-01, 6.50187705e-01],
[ 6.42954768e-01, 1.19101734e+00, 1.85214238e-01,
 7.12601531e-01, 1.30559643e+00, -4.32000358e-01,
 1.43803062e+00, 5.60633549e-01],
[-5.59561115e-01, 1.88378214e+00, 1.46945170e+00,
 1.29811429e+00, 1.80628929e-01, 2.09545769e+00,
 7.34266079e-01, -7.82678784e-01],
[-1.16081906e+00, 3.94337819e-01, 1.46945170e+00,
 -6.29808456e-01, -4.38881751e-01, -4.95560490e-02,
 1.08797157e+00, -6.62455397e-02],
[ 3.42325797e-01, -1.16438298e+00, -3.65173244e-01,
 2.44191322e-01, -5.74486246e-01, -2.82348237e-01,
 -2.94032364e-01, -7.82678784e-01],
[ 1.24421271e+00, 2.19552630e+00, -7.32098232e-01,
 -6.29808456e-01, -4.38881751e-01, -1.42968117e+00,
 8.29073737e-01, -6.62455397e-02],
[ 1.24421271e+00, 2.36871750e+00, 5.52139226e-01,
 -6.29808456e-01, -4.38881751e-01, 2.66081014e+00,
 -1.12177615e+00, 9.18850171e-01],
[ 9.43583739e-01, 1.05246438e+00, -5.48635738e-01,
 1.88362705e+00, 3.60176299e+00, 4.65912368e-01,
 9.96810362e-01, 8.29296016e-01],
[ 1.54484168e+00, 1.26029382e+00, 1.28598920e+00,
 -6.29808456e-01, -4.38881751e-01, -1.18026096e+00,
 -7.82656447e-01, 1.81439173e+00],
[-5.59561115e-01, -6.79447623e-01, -1.64941070e+00,
 2.44191322e-01, -5.05287778e-02, 9.81380785e-01,
 1.94591719e-01, -7.82678784e-01],
[-8.60190086e-01, -4.71618183e-01, 3.68676732e-01,
 -6.29808456e-01, -4.38881751e-01, 9.31496744e-01,
 -9.02989244e-01, -6.03570473e-01],
[-8.60190086e-01, -5.75532903e-01, 7.35601720e-01,
 -1.74655207e+00, -4.04970595e-01, -2.07817369e+00,
 1.69066580e-01, -9.61787095e-01],
[ 4.16968264e-02, -3.67703462e-01, -7.32098232e-01,
 2.11783216e+00, -1.42991861e-01, 4.82540381e-01,
 1.67869621e+00, -6.03570473e-01],
[-1.16081906e+00, 2.36871750e+00, 9.19064214e-01,
 -1.39524441e+00, 1.18231232e+00, 1.69560048e-02,
 8.65538221e-01, -9.61787095e-01],
[ 6.42954768e-01, 1.51870138e-01, 1.75174383e-03,
 -6.29808456e-01, -4.38881751e-01, -7.14676587e-01,
 -2.79446571e-01, -3.34908006e-01],
[-5.59561115e-01, -7.14085863e-01, -2.63767217e-01,
 -6.29808456e-01, -4.38881751e-01, -1.61258931e+00,
 -1.22752315e+00, -8.72232940e-01],
[-5.59561115e-01, 1.33171781e-02, 3.68676732e-01,
 1.29811429e+00, -5.05287778e-02, 1.29731304e+00,
 -8.37353173e-01, -3.34908006e-01],
[ 1.24421271e+00, 1.33171781e-02, 5.52139226e-01,
 -6.29808456e-01, -4.38881751e-01, -1.14700494e+00,
 -1.29942187e-01, 2.79948744e+00],
[-8.60190086e-01, -2.98426982e-01, -9.15560726e-01,
 -1.51234696e+00, 1.13608078e+00, -1.31328507e+00,
 -1.11812970e+00, -8.72232940e-01],
[-1.16081906e+00, -2.13210619e-02, -5.48635738e-01,

```

```

1.27088770e-01, -4.38881751e-01, 1.14766092e+00,
-6.76909444e-01, -9.61787095e-01],
[-5.59561115e-01, -6.44809383e-01, -1.28248571e+00,
1.06390919e+00, -2.81686485e-01, -1.67910137e+00,
-1.05614008e+00, -9.61787095e-01],
[ 1.24421271e+00, -6.79447623e-01, 3.68676732e-01,
-6.29808456e-01, -4.38881751e-01, 1.13103291e+00,
-9.28514382e-01, 8.29296016e-01],
[ 9.43583739e-01, 1.39884678e+00, -1.64941070e+00,
7.12601531e-01, 1.02820719e+00, -2.32464197e-01,
5.22772073e-01, 5.60633549e-01],
[ 9.43583739e-01, 2.33407926e+00, -2.01633569e+00,
8.29704083e-01, 4.37228868e+00, 3.32888260e-01,
1.39062679e+00, 1.12862771e-01],
[ 4.16968264e-02, 1.26029382e+00, 2.76945485e-01,
-6.29808456e-01, -4.38881751e-01, 2.72732220e+00,
-7.53484860e-01, -6.62455397e-02],
[-8.60190086e-01, -2.13210619e-02, 1.46945170e+00,
1.76652450e+00, 9.51154618e-01, 2.22848179e+00,
2.27409754e-01, -6.03570473e-01],
[-8.60190086e-01, -8.52638823e-01, -5.48635738e-01,
-1.51234696e+00, -1.08303320e+00, -2.04491766e+00,
-4.03425816e-01, -6.93124629e-01],
[-8.60190086e-01, -9.05975420e-02, 1.46945170e+00,
-2.24218887e-01, 5.65891773e-01, 4.32656341e-01,
-1.51820877e-01, 6.50187705e-01],
[-1.16081906e+00, 6.36805499e-01, -2.63767217e-01,
-6.29808456e-01, -4.38881751e-01, 7.31960583e-01,
1.78079676e+00, -6.93124629e-01],
[-2.58932144e-01, -1.30293594e+00, -1.81710750e-01,
-6.29808456e-01, -4.38881751e-01, -1.79549746e+00,
-2.02871155e-01, -6.93124629e-01],
[ 6.42954768e-01, -7.48724103e-01, -1.28248571e+00,
8.29704083e-01, 1.25936489e+00, 3.49516274e-01,
-5.33667711e-02, 9.18850171e-01],
[ 9.43583739e-01, 1.43348502e+00, 1.28598920e+00,
-6.29808456e-01, -4.38881751e-01, -2.49092210e-01,
-1.01967559e+00, 1.27706679e+00],
[ 6.42954768e-01, -2.98426982e-01, -7.32098232e-01,
1.53231940e+00, -4.38881751e-01, 3.82772301e-01,
-6.73262996e-01, -7.82678784e-01],
[-2.58932144e-01, 1.50276150e+00, -1.81710750e-01,
-9.26834200e-01, -5.05287778e-02, -4.95560490e-02,
-6.44091409e-01, -4.24462162e-01],
[-8.60190086e-01, 1.53739974e+00, 9.19064214e-01,
2.00072960e+00, -6.36128302e-01, 1.49980112e-01,
-3.77900677e-01, 1.54572926e+00],
[ 1.84547065e+00, 3.25061339e-01, -9.15560726e-01,
1.18101174e+00, -4.38881751e-01, 1.54673324e+00,
-1.32558389e-02, 4.71079394e-01],
[-2.58932144e-01, 2.09161158e+00, -7.32098232e-01,
-1.07116335e-01, -5.89896760e-01, 3.49516274e-01,
-6.33152063e-01, -6.03570473e-01],
[ 6.42954768e-01, 4.98252539e-01, 7.35601720e-01,
1.29811429e+00, 4.03325737e+00, 2.37813391e+00,
-7.53484860e-01, 1.18751264e+00],
[-1.16081906e+00, -1.82250954e+00, 3.68676732e-01,

```

```
-6.29808456e-01, -4.38881751e-01, 2.22848179e+00,
-9.13928589e-01, 1.18751264e+00],
[ 3.94987345e+00, 1.50276150e+00, 1.75174383e-03,
1.76652450e+00, 8.81658462e-02, 1.49684920e+00,
1.35780875e+00, 1.27706679e+00],
[ 4.16968264e-02, -2.63788742e-01, 5.52139226e-01,
1.64942195e+00, -4.38881751e-01, 1.24742900e+00,
-7.60777757e-01, 4.71079394e-01],
[-2.58932144e-01, -1.37221242e+00, -2.63767217e-01,
-6.29808456e-01, -4.38881751e-01, 1.57218944e-02,
-9.86857556e-01, -9.61787095e-01],
[-1.16081906e+00, -4.36979943e-01, 3.68676732e-01,
-6.29808456e-01, -4.38881751e-01, 2.22848179e+00,
8.80124014e-01, -7.82678784e-01],
[ 9.43583739e-01, 3.25061339e-01, -3.65173244e-01,
2.70334492e+00, 2.57681498e-01, 1.09777688e+00,
-2.05487356e-02, 9.18850171e-01],
[-1.16081906e+00, -8.87277063e-01, -2.63767217e-01,
-6.29808456e-01, -4.38881751e-01, 1.57218944e-02,
-6.87848789e-01, -6.93124629e-01],
[-1.16081906e+00, -9.91191783e-01, 7.35601720e-01,
-6.29808456e-01, -4.38881751e-01, 8.34680586e-02,
5.70175902e-01, -5.14016317e-01],
[-5.59561115e-01, -8.87277063e-01, -3.65173244e-01,
-9.26834200e-01, -4.97433677e-01, -9.80724802e-01,
4.24317967e-01, -1.05134125e+00],
[ 9.43583739e-01, -1.94512262e-01, -7.32098232e-01,
-6.29808456e-01, -4.38881751e-01, -7.47932614e-01,
1.04786064e+00, 1.12862771e-01],
[-8.60190086e-01, -1.59874022e-01, -1.81710750e-01,
4.78396426e-01, -1.89223402e-01, 4.49284354e-01,
3.07631619e-01, -6.62455397e-02]])
```

4.0 Model

1.0 Logistic Regression

In [47]:

```
1 ### Creating a Logistic regression object
2 logistic_reg=LogisticRegression()
3 logistic_reg
```

Out[47]: LogisticRegression()

In [48]:

```
1 ### Passing independant and dependant training data to the model
2 logistic_reg.fit(X_train,y_train)
```

Out[48]: LogisticRegression()

1.1 Using Above Model to get prediction for test data

```
In [49]: 1 logistic_reg_pred=logistic_reg.predict(X_test)
          2 logistic_reg_pred
```

1.2.0 Performance Metrics

1.2.1 Confusion Matrix

```
In [50]: 1 confusion_mat=confusion_matrix(y_test, logistic_reg_pred)
         2 confusion mat
```

```
Out[50]: array([[56,  8],  
                 [16, 24]], dtype=int64)
```

```
In [51]: 1 truly_positive=confusion_mat[0][0]
          2 falsely_positive=confusion_mat[0][1]
          3 falsely_negative=confusion_mat[1][0]
          4 truly_negative=confusion_mat[1][1]
```

1.2.2 Accuracy Score

```
In [52]: 1 ### accuracy using accuracy_score
          2 accuracy=round(accuracy_score(y_test, logistic_reg_pred),4)
          3 accuracy
```

Out[52]: 0.7692

```
In [53]: 1 ### manual calcualtion for accuracy
2 accuracy_manual=round(((truly_positive+truly_negative)/(truly_positive+false
3 print("Accuracy of our model is {}".format(accuracy_manual))
```

Accuracy of our model is 0.7692

1.2.3 Precision Score

```
In [54]: 1 precision_manual_diabetic=round(truly_positive/(truly_positive+false_posit  
2 print("Precision of our model is {}".format(precision_manual_diabetic))
```

Precision of our model is 0.875

1.2.4 Recall Score

```
In [55]: 1 recall_manual_diabetic=round(truly_positive/(truly_positive+falsely_negative)
2 print("Recall of our model is {}".format(recall_manual_diabetic))
```

Recall of our model is 0.7778

1.2.5 F-1 Score

1. Giving equal importance to falsely positive and falsely negative

```
In [56]: 1 f1_score=2*(precision_manual_diabetic*recall_manual_diabetic)/(precision_man
2 print("F-1 Score of our model is {} ".format(round(f1_score,4)))
```

F-1 Score of our model is 0.8235

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```