DATA70121

Statistics and Machine Learning - I

STATS PROJECT -1

STUDENT ID : **11356488**

TABLE OF CONTENTS

SECTION 1: DESCRIPTION OF THE DATASET

1.1 ORIGIN

The Pima Diabetes dataset originates from the National Institute of Diabetes and Digestive and Kidney Diseases(NIDDK) in the United States. The dataset consists of information recorded by 750 women.

1.2 OBJECTIVE OF PIMA DATASET

The purpose of the dataset is to investigate and analyse the factors associated with the development of diabetes in women.

1.3 KEY FEATURES

The dataset has 8 diagnostic features and 1 target feature indicating if a woman eventually tested positive for diabetes or not

S.no	Features	DESCRIPTION
1.	Pregnancies	The number of times the woman has been pregnant
2.	Glucose	The plasma glucose concentration(mg/dl) at 2 hours in an oral glucose tolerance test(OGTT)
3.	Blood Pressure	Diastolic blood pressure(mm Hg)
4.	Skin Thickness	Triceps skin fold thickness(mm)
5.	Serum Insulin	Insulin concentration (μ U/ml) at 2 hours in an OGTT
6.	BMI	Body mass index (weight in kg)/(height in m)2
7.	Diabetes Pedigree	A numerical score reflecting the genetic influence of both diabetic and non-diabetic relatives on diabetes risk.
8.	Age	Age in years
9.	Outcome	Binary variable indicating whether the women tested positive for diabetes(1) or not(0)

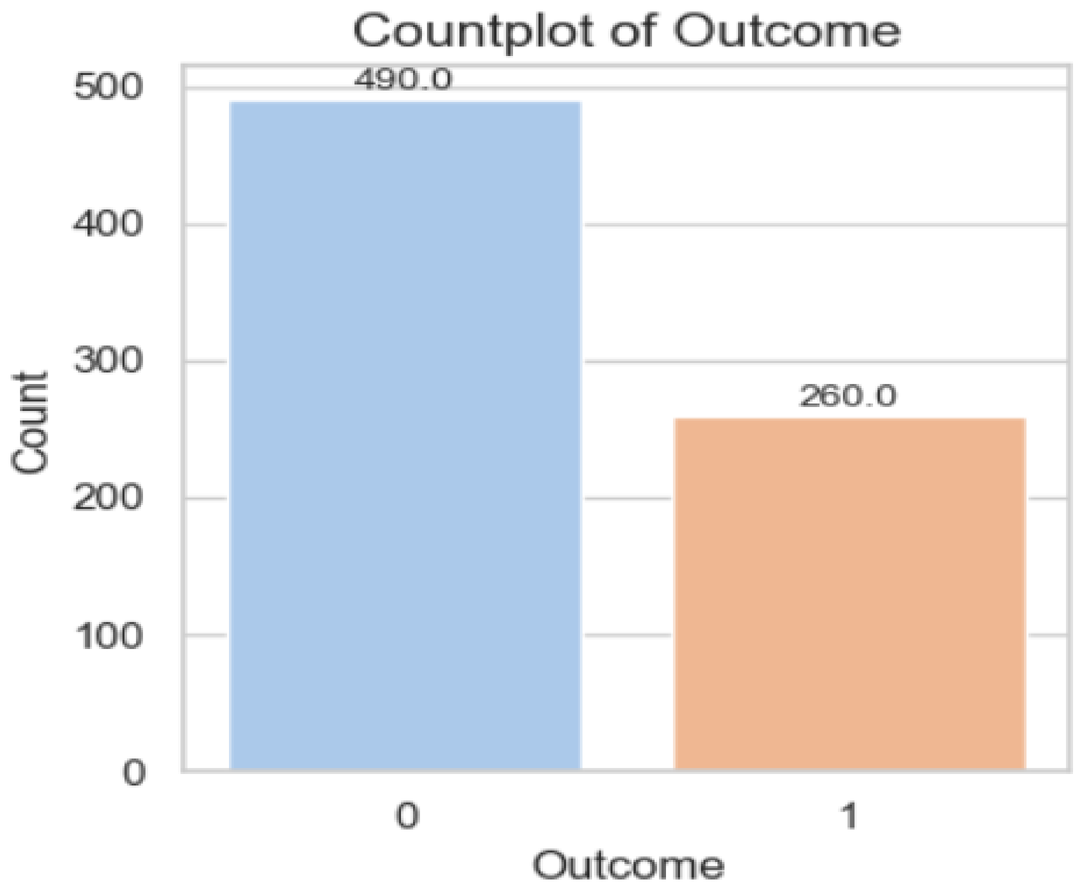
1.4 POTENTIAL DATA QUALITY ISSUES

- ➔ Missing Values : Few columns have missing or incorrect values affecting the analysis accuracy.
- ➔ Outliers : Extreme values in skewed variables, impacting model predictions
- ➔ Data Inconsistency: Varied united of features.
- ➔ Imbalanced Outcome variable : Imbalance in distribution of women having diabetes and not having diabetes

SECTION 2: EXPLORATORY DATA ANALYSIS

2.1 UNIVARIATE ANALYSIS

2.1.1. Count Plot:

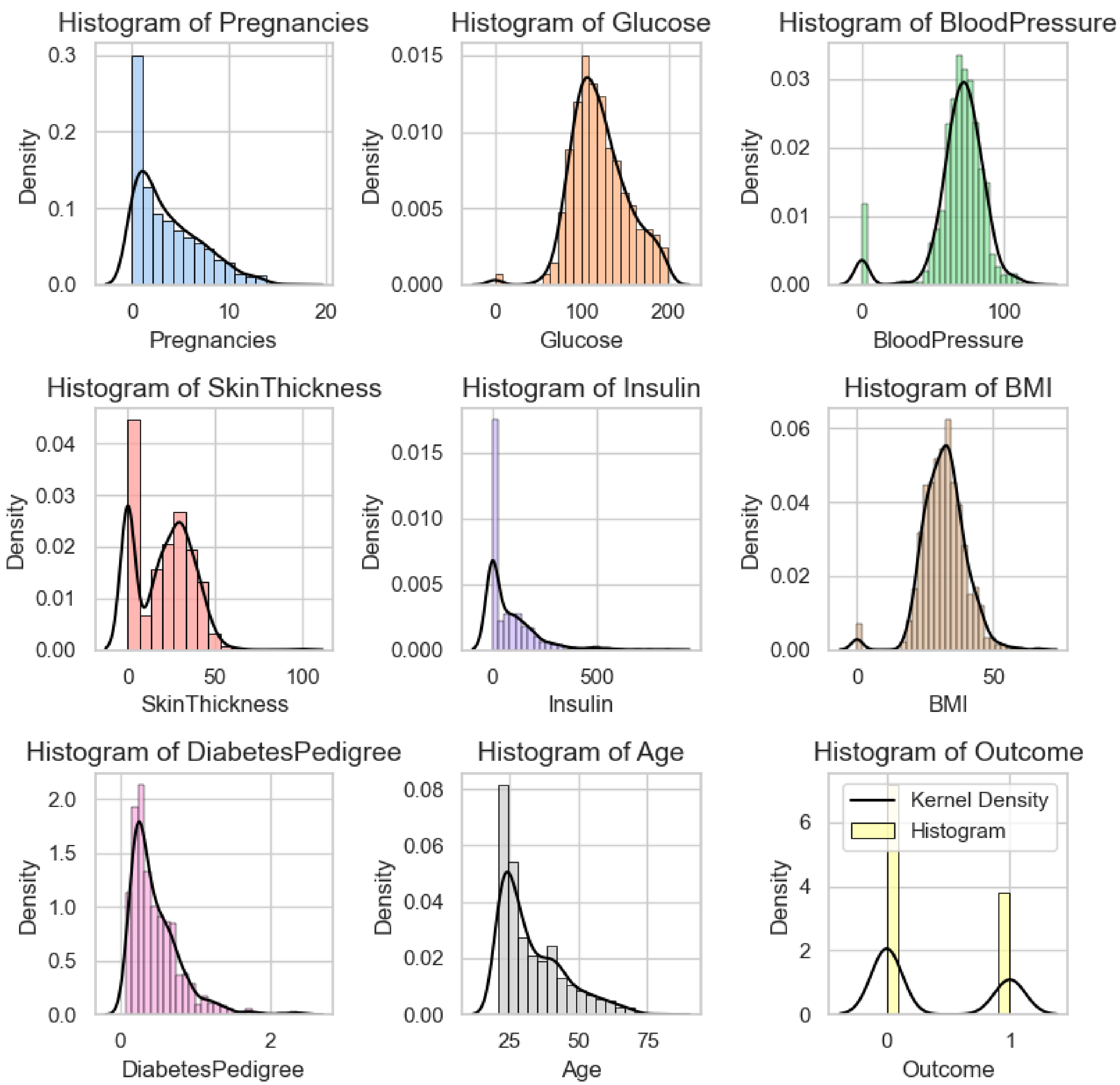


OBSERVATIONS:

Target Variable (Outcome) -

- The above graph shows that out of 750 women, 490 will not get diabetes but 260 will get diabetes. This clearly indicates an **unbalanced data.**

2.1.1 Histogram & Kernel Density Plots (KDE):



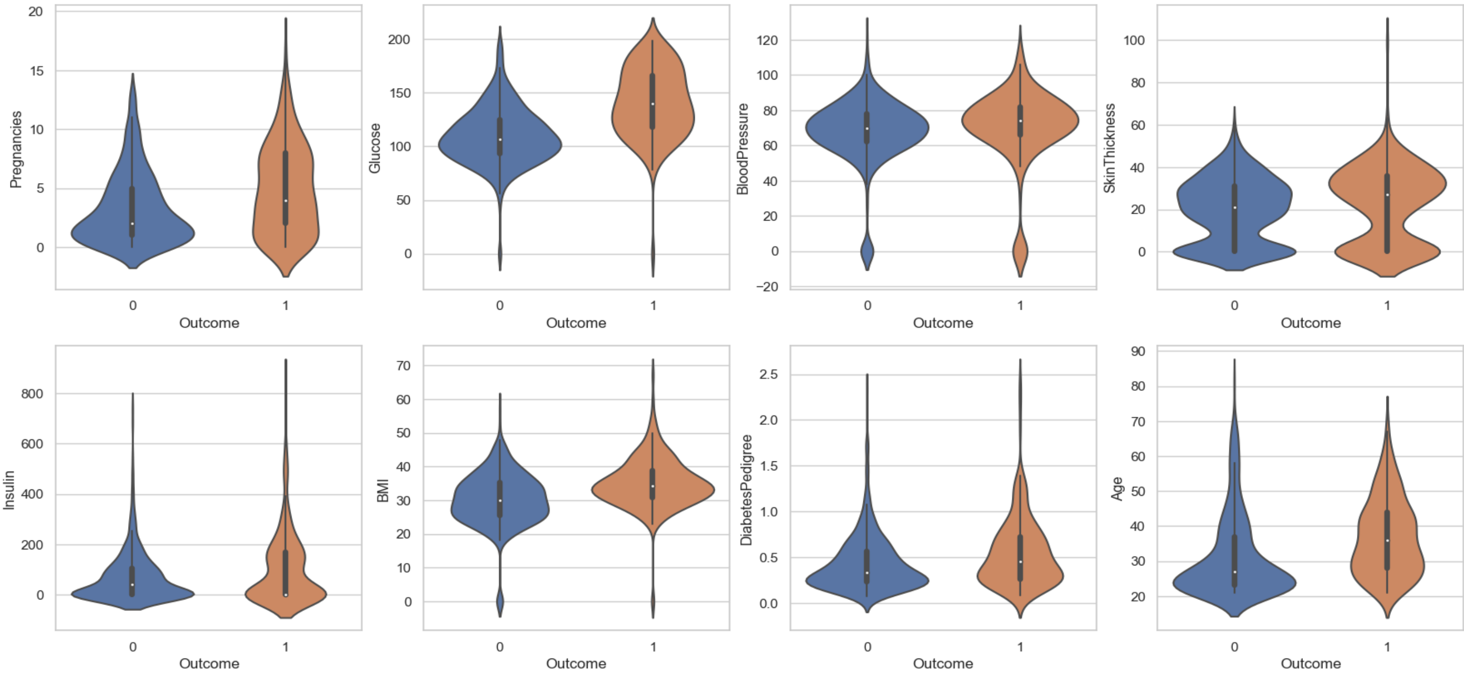
OBSERVATIONS:

➔ Distribution shape:
- ‘Glucose’ and BMI are normally distributed.
- Blood Pressure,Insulin & Skin Thickness are right skewed indicating lower levels in a significant population.
- Pregnancies and age are right skewed indicating fewer pregnancies and younger crowds.
- Most women have lower pedigree scores.

➔ Missing Values:
  - Based on medical domain knowledge we can say that 'Insulin' at 0 suggests potential diabetes risk.
  - Pregnancies and Outcome can be 0
  - Glucose , Blood Pressure, Skin Thickness and BMI at '0' are invalid.

2.2 BIVARIATE ANALYSIS

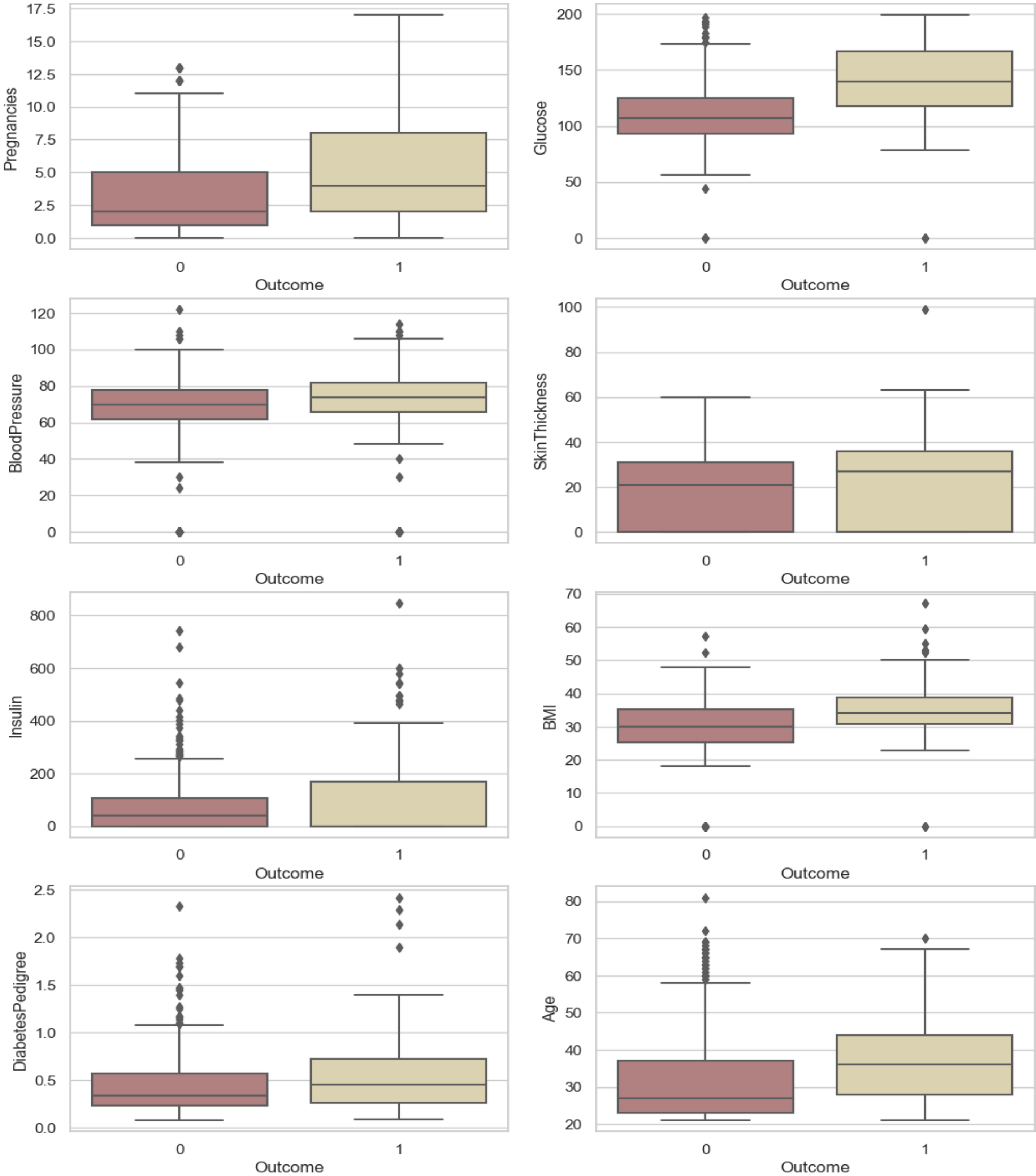
2.2.1 Violin plot



OBSERVATIONS

➔ Diabetic women cluster around **30-40 age**.
➔ **Higher BMI** increases diabetes risk in obese and overweight women.
➔ **Insulin** higher no.of.outliers.

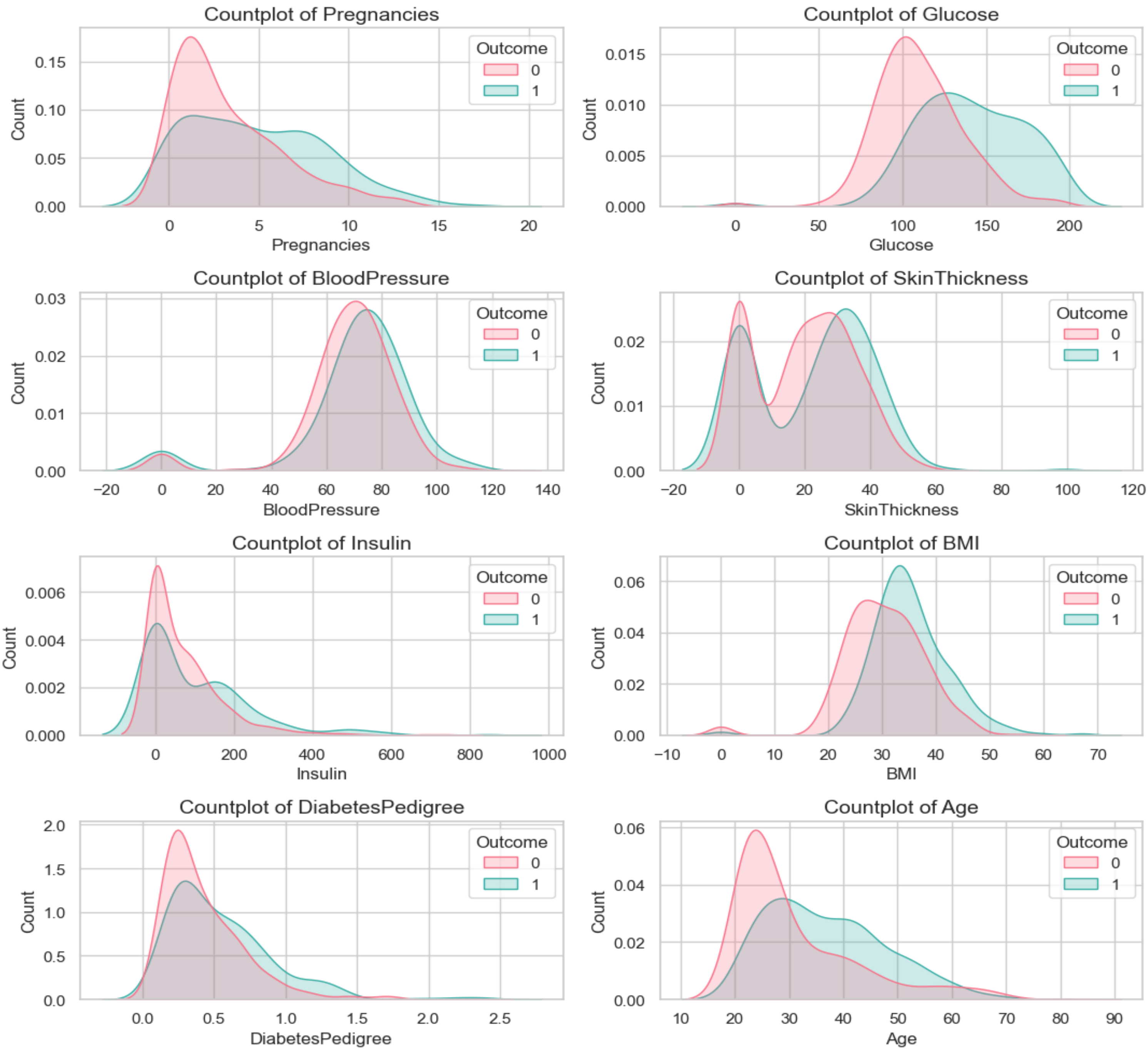
2.2.2 BoxPlot :



OBSERVATIONS:

- ➔ Higher **Glucose** links to higher diabetes risk
- ➔ Median pregnancies increase diabetes risk in women
- ➔ Skin thickness has overlapping diabetes outcomes(0/1)

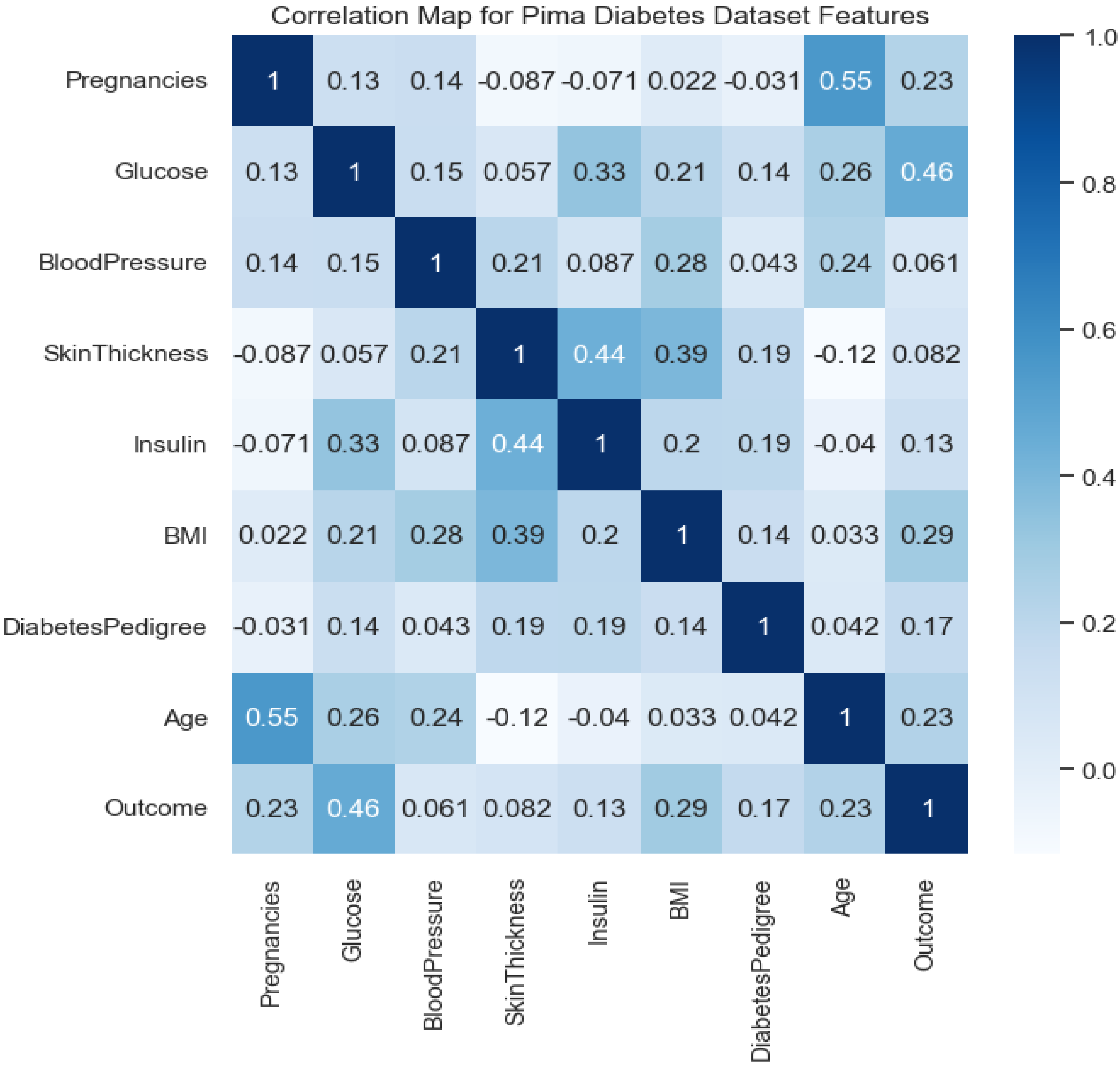
2.2.3 Box Plot



OBSERVATIONS:

- ➔ **Glucose** values between 120-150 indicate highest Diabetes risk.
- ➔ **Pregnancies** peak for middle values.
- ➔ **Blood Pressure, skin thickness,Insulin** are overlapping peaks, less informative features.

2.2.4 Heat Map (with correlation) - feature selection



OBSERVATIONS:

➔ Based on the correlation heat map and target:**Outcome** , the highest positive correlation is **Glucose** followed by **BMI**, **Age** and **Pregnancies.**

```
corr_matrix['Outcome'].sort_values(ascending=False)
```

```
Outcome             1.000000
Glucose             0.460310
BMI                 0.289832
Age                 0.232892
Pregnancies         0.229235
DiabetesPedigree    0.170688
Insulin             0.130928
SkinThickness       0.082205
BloodPressure       0.060860
Name: Outcome, dtype: float64
```

- ➔ **Age** and **Pregnancies** show identical correlation. Leading to **multicollinearity**.

2.3 Multivariate Analysis

2.3.1 Pair Plot

OBSERVATIONS:

1. **Glucose & BMI** are strongly correlated with diabetes , showing clear segregation.
2. **Age & BMI** overlap with diabetes women having higher BMI around ages 30-50.
3. **Pregnancies** peak at ages 20-40 for both diabetes outcomes.
4. **Blood Pressure and Age , Diabetes pedigree** and **Age** lack a clear relationship.
5. **Glucose and Insulin** show positive correlation.

DATA PREPROCESSING

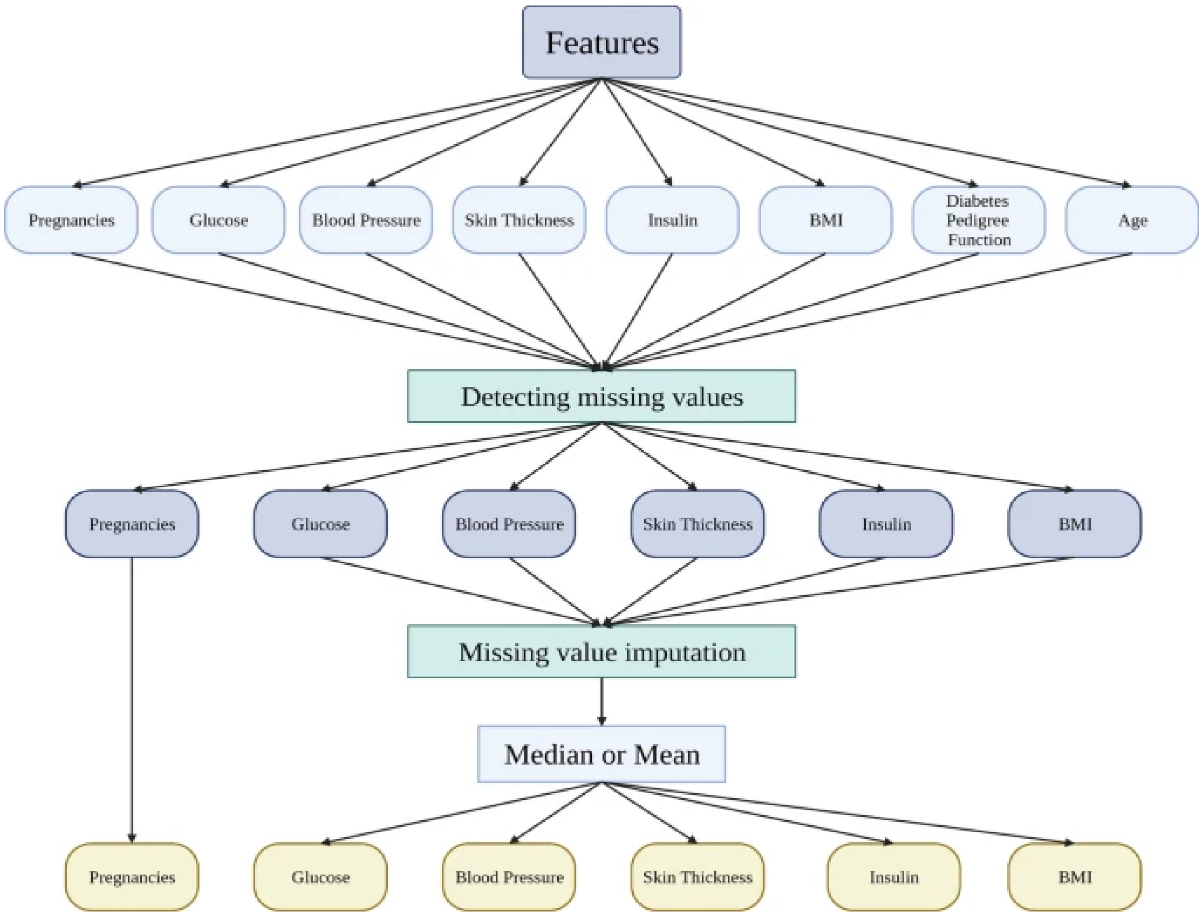
HANDLING MISSING VALUES:

1. The ‘0’ values in medical data need proper imputation without dropping.
2. Features like : Skin thickness, Blood Pressure,Glucose, BMI with 0 are invalid values.
3. Insulin with 0 values is rare but based on medical data , remains untouched.

➔ NaN value count

```
: Pregnancies          0
  Glucose              5
  BloodPressure       35
  SkinThickness      221
  Insulin              0
  BMI                 11
  DiabetesPedigree     0
  Age                  0
  Outcome              0
  dtype: int64
```

MEAN MEDIAN IMPUTATION:



- For skewed data, **median** imputation is unbiased by outliers..
- **Mean** imputation suits normal distribution.

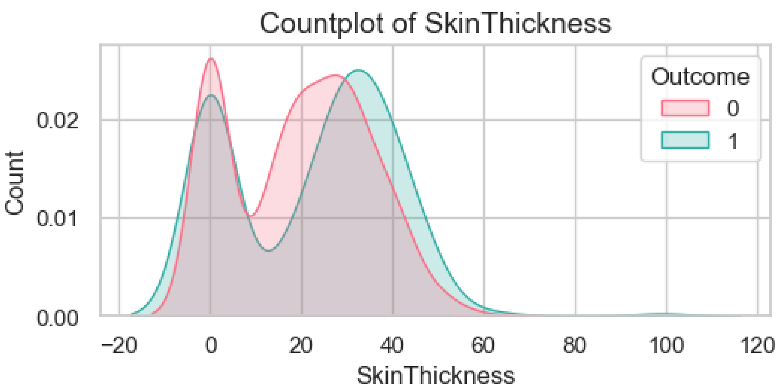
Skewness Estimate:

```
705]: X = df_diab.drop(['Outcome'], axis =1)
for i, column in enumerate(X.columns) :
    print(column+"\nKurtosis :"+ str(kurtosis(X[column]))+"\nSkewness :"+str(skew(X[column])))
    print("\n")
```

```
Pregnancies
Kurtosis :0.18334697034089675
Skewness :0.9088167295788591


Glucose
Kurtosis :0.6389200613297836
Skewness :0.16684145457194843


BloodPressure
Kurtosis :5.0227860259333905
Skewness :-1.827487419154484


SkinThickness
Kurtosis :-0.49200852838534326
Skewness :0.11902092466028916


Insulin
Kurtosis :7.173751770703026
Skewness :2.2556138228073612


BMI
Kurtosis :3.2480503189239256
Skewness :-0.4258331470070181


DiabetesPedigree
Kurtosis :5.5884028699133825
Skewness :1.917792507562979


Age
Kurtosis :0.6576218884569602
Skewness :1.132309726022226
```

Median Imputation

```
#Skewed Distributed - SkinThickness,Insulin,BMI
# Median Imputation
df_diab_copy['SkinThickness'].fillna(df_diab_copy['SkinThickness'].median(), inplace=True)
df_diab_copy['BMI'].fillna(df_diab_copy['BMI'].median(), inplace=True)
```



Mean Imputations

```
#Normally Distributed - Glucsose, BloodPressure
# Blood Pressure
df_diab_copy['BloodPressure'].fillna(df_diab_copy['BloodPressure'].mean(), inplace=True)
temp = df_diab_copy[df_diab_copy['BloodPressure'].notnull()]
temp = temp[['BloodPressure', 'Outcome']].groupby(['Outcome'])[['BloodPressure']].median().reset_index()
print("Glucose Mean grouped by Outcome : \n",temp)
df_diab_copy.loc[(df_diab_copy['Outcome'] == 0 ) & (df_diab_copy['BloodPressure'].isna()), 'BloodPressure'] = 72
df_diab_copy.loc[(df_diab_copy['Outcome'] == 1 ) & (df_diab_copy['BloodPressure'].isna()), 'BloodPressure'] = 74

## Glucose -  Grouped by Outcome and then mean value for glucose is calculated based on the 0 and 1 outcomes.
temp = df_diab_copy[df_diab_copy['Glucose'].notnull()]
temp = temp[['Glucose', 'Outcome']].groupby(['Outcome'])[['Glucose']].median().reset_index()
print("Glucose Mean grouped by Outcome : \n",temp)
df_diab_copy.loc[(df_diab_copy['Outcome'] == 0 ) & (df_diab_copy['Glucose'].isna()), 'Glucose'] = 107
df_diab_copy.loc[(df_diab_copy['Outcome'] == 1 ) & (df_diab_copy['Glucose'].isna()), 'Glucose'] = 140
```

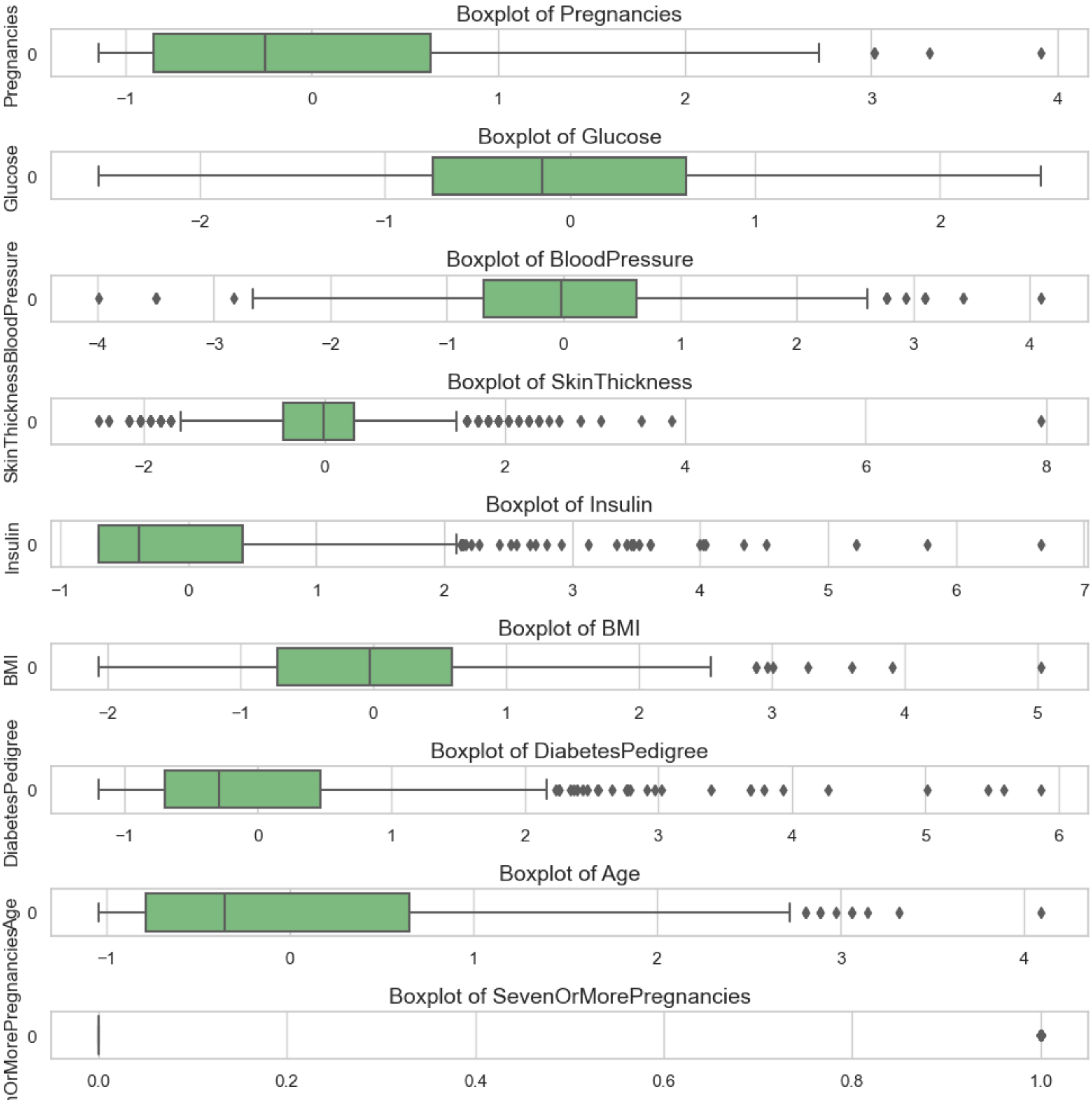
```
Glucose Mean grouped by Outcome :
    Outcome  BloodPressure
0        0           72.0
1        1           74.0
Glucose Mean grouped by Outcome :
    Outcome  Glucose
0        0    107.0
1        1    140.0
```

```
df_diab_copy.isna().sum()
```

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

OUTLIER DETECTION and HANDLING:

- In **Medical data,** outliers contribute to rare cases and hence should not be removed.



FEATURE SCALING:

1. **Standard Scaler(z-score normalization)**:

   Brings the data to a standard scale and assumes the distribution to be normal and make the mean =0 and Standard deviation =1

   $$z = (x - u) / s$$

   Where, x= mean of the sample
   u = mean of population
   s=standard deviation of sample

Scaled Features:

[744]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	SevenOrMorePregnancies	Outcome
0	0.640173	0.868724	-0.029631	0.676801	-0.699295	0.169254	0.462359	1.438616	0	1
1	-0.844459	-1.199563	-0.524255	-0.003933	-0.699295	-0.845805	-0.369222	-0.185168	0	0
2	1.234026	2.017772	-0.689129	-0.003933	-0.699295	-1.324333	0.597943	-0.099706	1	1
3	-0.844459	-1.068243	-0.524255	-0.684667	0.118506	-0.628293	-0.923609	-1.039792	0	0
4	-1.141385	0.507595	-2.667624	0.676801	0.762306	1.546834	5.466909	-0.014244	0	1
...	...	...	...	...	...	...	...	...	...	...
745	2.421732	-0.707113	0.959616	0.449890	0.214206	-0.352777	0.043556	1.096766	1	0
746	-0.844459	0.835894	1.783988	1.357535	-0.699295	2.445887	-0.348131	-0.527018	0	1
747	-0.844459	-1.330882	0.135243	1.357535	-0.203394	2.010861	1.875444	-0.099706	0	0
748	-0.250606	2.149092	-0.194506	-0.798123	1.040706	0.575278	-0.197482	0.242143	0	1
749	0.640173	1.328343	-0.854004	-0.003933	-0.699295	-1.179325	-0.890466	1.438616	0	1

750 rows × 10 columns

FEATURE ADDITION

➔ Added **'SevenOrMorePregnancies'** binary column based on pregnancies >=7., dependent on **'Pregnancies'** feature.

➔ **Logic used in python:**

```
# Adding a new column SevenOrMorePregnancies
df_diab_copy['SevenOrMorePregnancies'] = (df_diab_copy['Pregnancies'] >= 7).astype('int64')
```

➔ **Dataframe after feature addition:**

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome	SevenOrMorePregnancies
0	6	148.0	72.0	35.0	0	33.6	0.627	50	1	0
1	1	85.0	66.0	29.0	0	26.6	0.351	31	0	0
2	8	183.0	64.0	29.0	0	23.3	0.672	32	1	1
3	1	89.0	66.0	23.0	94	28.1	0.167	21	0	0
4	0	137.0	40.0	35.0	168	43.1	2.288	33	1	0
...	...	...	...	...	...	...	...	...	...	...
745	12	100.0	84.0	33.0	105	30.0	0.488	46	0	1
746	1	147.0	94.0	41.0	0	49.3	0.358	27	1	0
747	1	81.0	74.0	41.0	57	46.3	1.096	32	0	0
748	3	187.0	70.0	22.0	200	36.4	0.408	36	1	0
749	6	162.0	62.0	29.0	0	24.3	0.178	50	1	0

750 rows × 10 columns

SECTION 4 : REGRESSION MODEL WITH SINGLE PREDICTOR

Model : Logistic Regression model

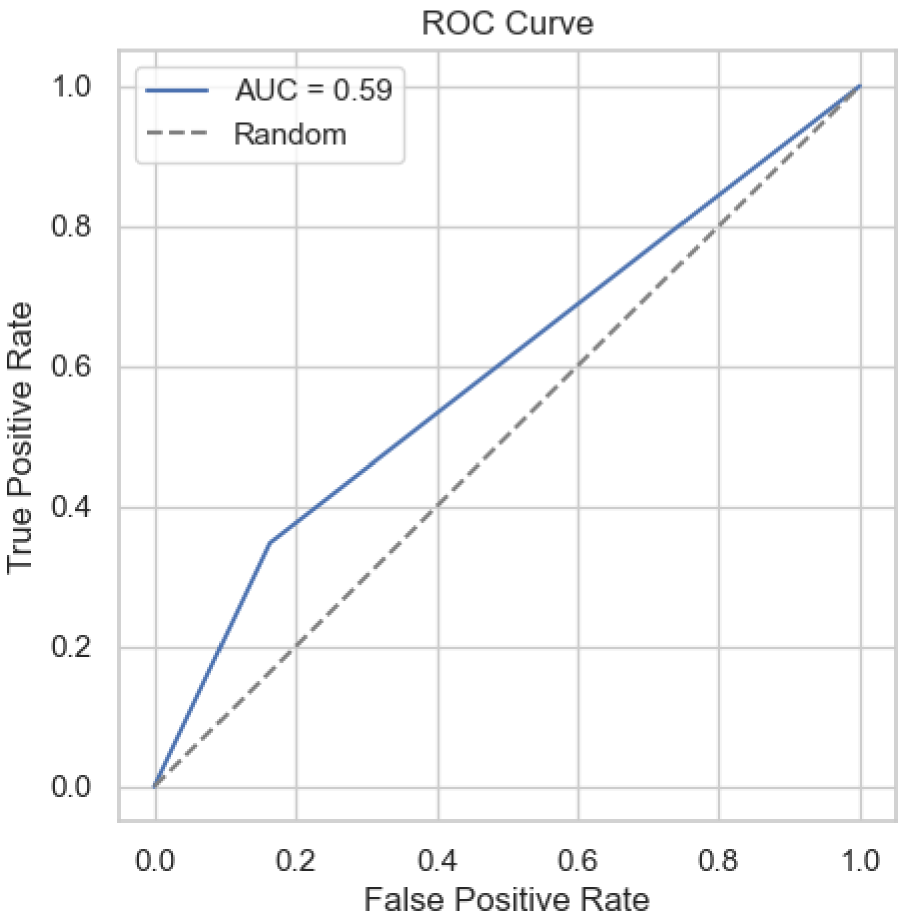
Predictor : 'SevenOrMorePregnancies'

Target : 'Outcome'

Analysis : The model performs poorly on training as well as testing dataset indicating single predictor alone is not a good selection

Results-

- ➔ **Accuracy_Score : **0.686666
- ➔ **AUC Score** : 0.59 , poor score



PREDICTING PROBABILITY

model.predict_proba(X_test)[:,-1] -> Predicts probability for class 1 diabetes.

4.1 Probability of diabetes for <=6 pregnancies.

```
]: #Predicting probability based on testing data
y_probs = model.predict_proba(X_test)[:,-1]

#Probability for diabetes based on no. of pregnancies
prob_six_or_lesser = model.predict_proba([[0]])[0,1]


# Print the results
print(f"Probability of diabetes with six or fewer pregnancies: {prob_six_or_lesser:.4f}")
```

Outcome: P(prob_six_or_lesser) : **0.2946**

4.2 Probability of diabetes for >= 7 pregnancies

```
#Predicting probability based on testing data
y_probs = model.predict_proba(X_test)[:,-1]

#Probability for diabetes based on no. of pregnancies

prob_seven_or_more = model.predict_proba([[1]])[0,1]

# Print the results
print(f"Probability of diabetes with seven or more pregnancies: {prob_seven_or_more:.4f}")
```

Outcome: P(prob_seven_or_more) : **0.5787**

REGRESSION MODELS AND PERFORMANCE ANALYSIS

Selected Predictors : Glucose, **BMI** and **Age**. - Selected from correlation heat map during EDA process.

Models Analysed along with their performance:

	Model	Training Score	Accuracy	Precision
0	**Logistic Regression**	75.500000	82.000000	74.509804
1	Decision Tree	100.000000	66.666667	51.851852
2	KNN	81.833333	70.666667	56.666667
3	Random Forest Classifier	100.000000	73.333333	60.000000

OBSERVATIONS:

- **Logistic Regression** performs best with **82%** accuracy and highest precision.
- Decision Tree and Random Forest exhibit overfitting, with poorer testing accuracy.
- KNN has a low precision , poor model.

Selected Model : Logistic Regression

LOGISTIC REGRESSION MODEL

Module : from sklearn.linear_model import LogisticRegression

Selected Predictors : Glucose, **BMI** and **Age**.
(Based on extensive EDA, it was clear that Glucose then BMI has the best correlation with target variable and between age and pregnancies , the model trained better with 'age')

Dataset : Applied Train-test-split module to split 20% test and 80% training data. Stratify = y applied to make sure the outcome is balanced well during the data split.

Formula:

$$P(Y_i) = \frac{1}{1 + e^{-(b_0 + b_1 X_{1i})}}$$

where

- $P(Y_i)$ is the predicted probability that Y is true for case i;
- e is a mathematical constant of roughly 2.72;
- b_0 is a constant estimated from the data;
- b_1 is a b-coefficient estimated from the data;
- X_i is the observed score on variable X for case i.

Train-test-split

```
#Splitting the data into input features and outcome
#X = features_transformed.drop('Outcome',axis=1)

X = features_transformed[['Glucose','BMI','Age',]]
y= features_transformed['Outcome']

#Splitting the data into training and testing data
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2, random_state=10, stratify=y)
```

Model Training :

```
# Initializing and fitting the Logistic regression model
model_lr = LogisticRegression()
model_lr.fit(X_train,y_train)
```

Model Prediction :

```
#Predicting the test data based on the trained model
y_pred = model_lr.predict(X_test)
```

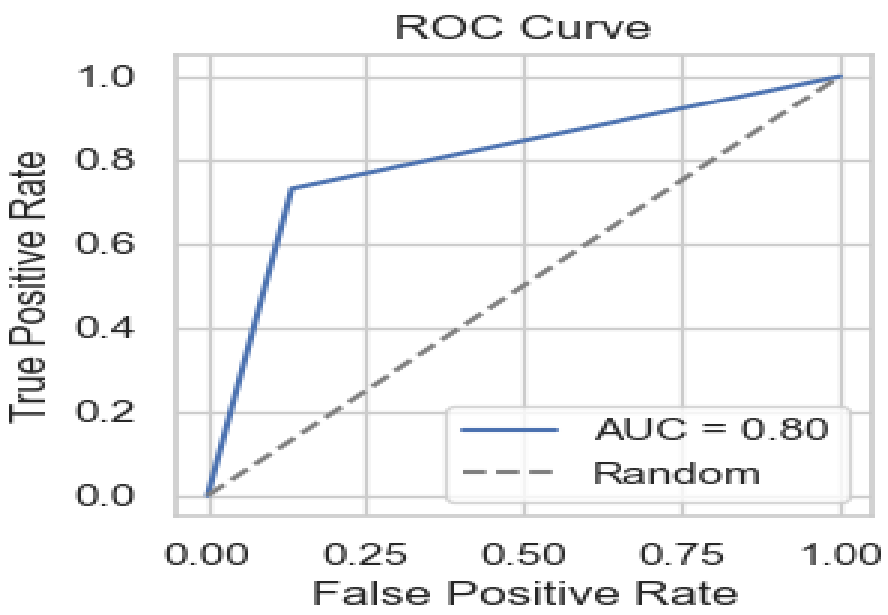
Evaluating Model Performance :

1. **Coefficients for 3 predictors:** Shows how much weight each has in predicting the Outcome. (log-odds)
    - Glucose : 1.0233
    - BMI: 0.6038
    - Age: 0.3686
    - Intercept: -0.8181
2. **Training - Testing Accuracy:**
    - **Training_Score : 75.5%**
    - **Testing_Accuracy : 82.0%**
3. **Classification Report:** Indicates Higher Precision and recall , good model

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.87	0.86	98
1	0.75	0.73	0.74	52
accuracy			0.82	150
macro avg	0.80	0.80	0.80	150
weighted avg	0.82	0.82	0.82	150

4. ROC Curve and AUC Score: 0.8 shows **strong** discriminatory power between class 1 and 0 for diabetes outcome.



➔ **Predicting Diabetes probability on UNSEEN Data : ToPredict.csv**

OUTCOME:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	predicted_outcome	probability
0	4	136	70	0	0	31.2	1.182	22	0	0.165244
1	1	121	78	39	74	39.0	0.261	28	0	0.236242
2	3	108	62	24	0	26.0	0.223	25	0	0.042341
3	0	181	88	44	510	43.3	0.222	26	1	0.826427
4	8	154	78	32	0	32.4	0.443	45	1	0.564862

REFERENCES

- https://pandas.pydata.org/docs/
- https://blog.tdg.international/understanding-logistic-regression-predictions-and-performance-metrics-5b4b6d1e5d2f
- https://duckduckgo.com/?q=logistic+regression+sklearn&atb=v397-1&ia=web
- https://datasciencehorizons.com/exploratory-data-analysis-eda-techniques-a-step-by-step-tutorial-with-python/
- https://www.diabetes.org.uk/guide-to-diabetes/managing-your-diabetes/blood-pressure
- https://pycaret.readthedocs.io/en/latest/api/classification.html
- https://www.sciencedirect.com/science/article/pii/S0169260722001596
- https://www.clinfo.eu/mean-median/
- https://ieeexplore.ieee.org/abstract/document/9317070
- https://www.kaggle.com/code/vipulgandhi/how-to-choose-right-metric-for-evaluating-ml-model
- https://iopscience.iop.org/article/10.1088/1757-899X/1070/1/012059
- https://link.springer.com/article/10.1007/s42452-019-0383-x
- https://medium.com/analytics-vidhya/analyzing-pima-indian-diabetes-dataset-36d02a8a10e5
- https://www.kaggle.com/code/ohseokkim/linear-nonlinear-scaling?scriptVersionId=87730271
- Step by Step Diabetes Classification-KNN-detailed | Kaggle
- https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-023-05467-x