

DATA70121

Statistics and Machine Learning 1

EDA and Regression

User ID: 11340063

Data Information

PimaDiabetes is a dataset from the National Institute of Diabetes and Digestive and Kidney Diseases in the US, containing 750 records of diagnostic tests for women, including Pregnancy, Blood Pressure, Glucose, Insulin, BMI, Skin Thickness, Age, and Diabetes Pedigree. The variable Outcome (1/0) indicates if the subject tested positive for diabetes. An Oral Glucose Tolerance Test (OGTT) measures plasma glucose concentration at 2 hours. Blood Pressure considers diastolic blood pressure. Skin Thickness stores the width of the skin over the triceps muscle. BMI measures weight and height. Insulin concentration is measured at 2 hours. A woman's diabetes pedigree score quantifies the genetic impact of her close relatives with and without diabetes, with higher scores indicating more diabetes diagnoses.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	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

Table 1: Data in PimaDiabetes dataset

Exploratory Data Analysis

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000
mean	3.844000	120.737333	68.982667	20.489333	80.378667	31.959067	0.473544	33.166667	0.346667
std	3.370085	32.019671	19.508814	15.918828	115.019198	7.927399	0.332119	11.708872	0.476226
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	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	36.500000	32.000000	0.377000	29.000000	0.000000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	0.628500	40.750000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Table 2: Central Tendency of PimaDiabetes dataset

The central tendency of different fields is displayed in Table 2. The fact that each field has a count of 750 indicates that there are no null values in the dataset. However, some fields, like BMI, Skin Thickness, Insulin, Blood Pressure, and Glucose, have minimum values of 0. These are considered as Null values since a value of 0 is not intended for them

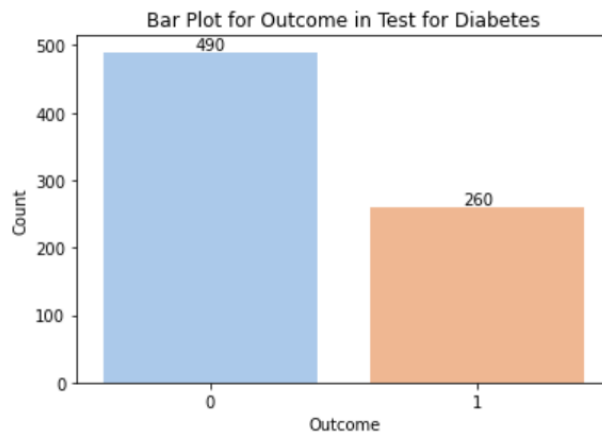


Figure 1: Bar Plot for Outcome in test for diabetes

Figure 1 shows that over 60% of the records in the dataset gives an outcome of 0 which shows that those people are not diagnosed with diabetes.

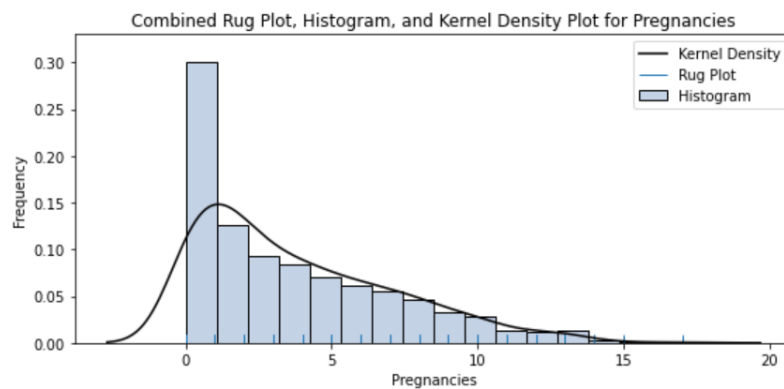


Figure 2: Combined Rug Plot, Histogram and Kernel Density Plot for Pregnancies

Figure 2 shows that the graph is right skewed and most of the women have less than 5 pregnancies in the dataset.

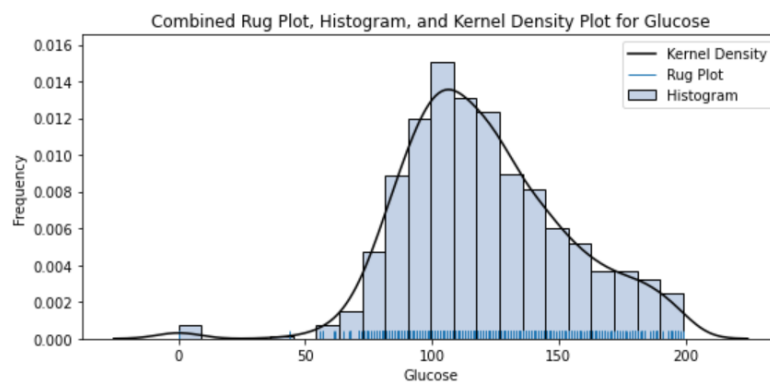


Figure 3: Combined Rug Plot, Histogram and Kernel Density Plot for Glucose

Figure 3 shows that Glucose contains some of the records with 0 as the value. The most frequent value comes in the range of 100 -150 mg/dl. The normal range of Glucose is found to be anywhere between 70 and 125 mg/dl. Figure 4 shows most women have normal diastolic pressure between 60 and 80. [1]

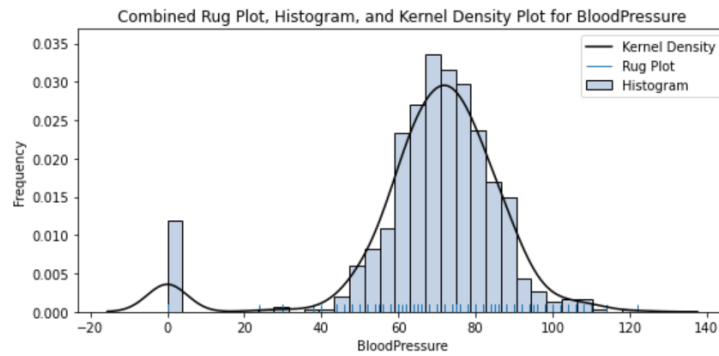


Figure 4: Combined Rug Plot, Histogram and Kernel Density Plot for Blood Pressure

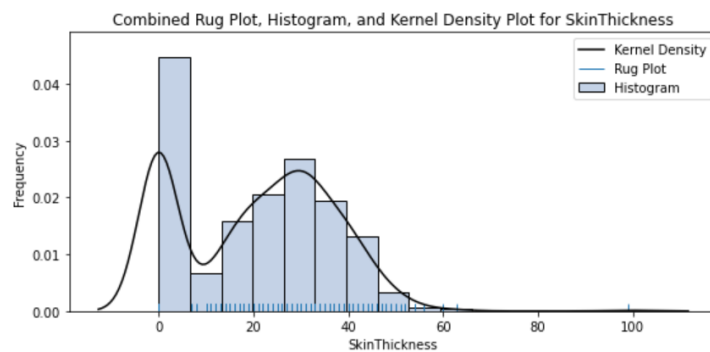


Figure 5: Combined Rug Plot, Histogram and Kernel Density Plot for Skin Thickness

When it comes to Skin Thickness, Figure 5 shows that it is bi-modal but the first peak in the graph is caused by records having a value of 0 which is inaccurate. Figure 6 shows that nearly 50% of data have a value of 0 and this should be imputed.

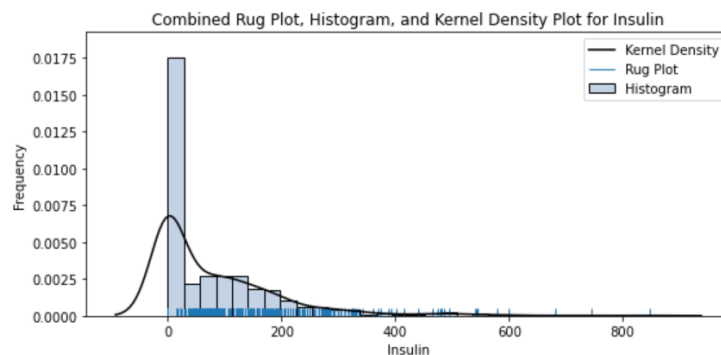


Figure 6: Combined Rug Plot, Histogram and Kernel Density Plot for Insulin

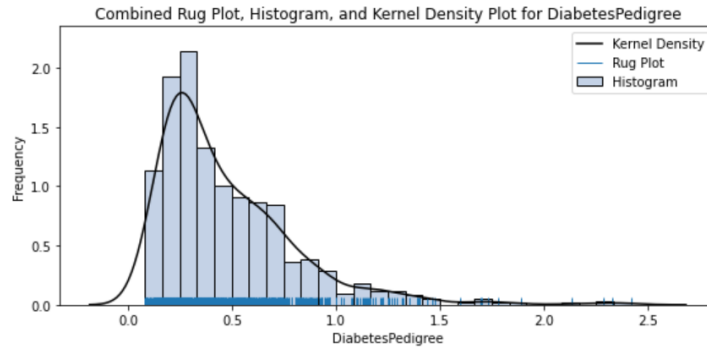


Figure 7: Combined Rug Plot, Histogram and Kernel Density Plot for Diabetes Pedigree

Figure 7 shows that most women have Diabetes Pedigree value between 0 and 0.5 which shows that most women do not have close relatives diagnosed with diabetes. From Figure 8 it is clear that most of the women fall under the age group between 20 and 40.

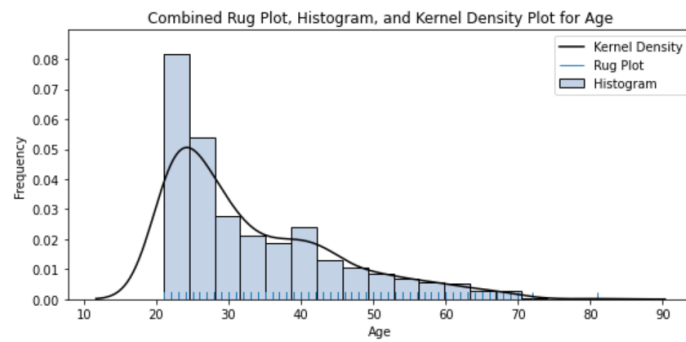


Figure 8: Combined Rug Plot, Histogram and Kernel Density Plot for Age

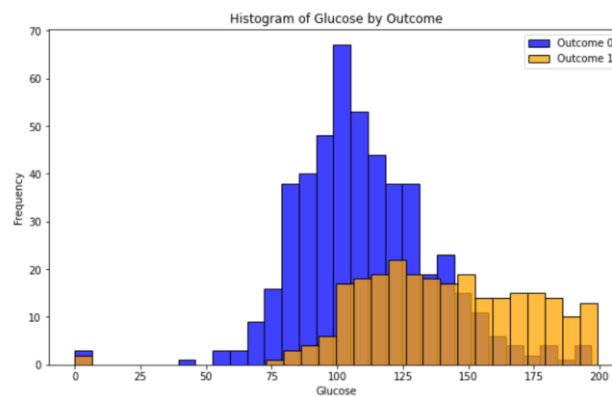


Figure 9: Histogram of Glucose by Outcome

Figure 9 shows that people who are diabetic have higher values of Glucose than non-diabetic people. From Figure 10 it is clear that people with greater diastolic pressure have diabetes.

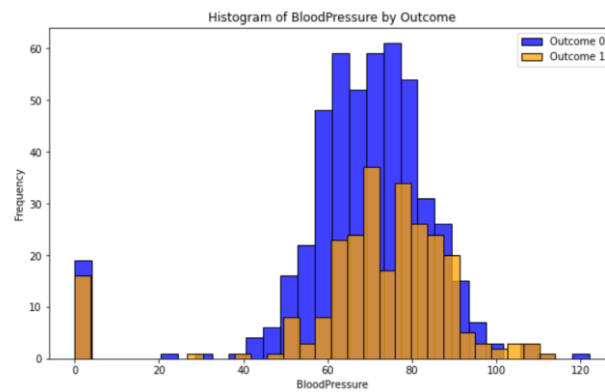


Figure 10: Histogram of Blood Pressure by Outcome

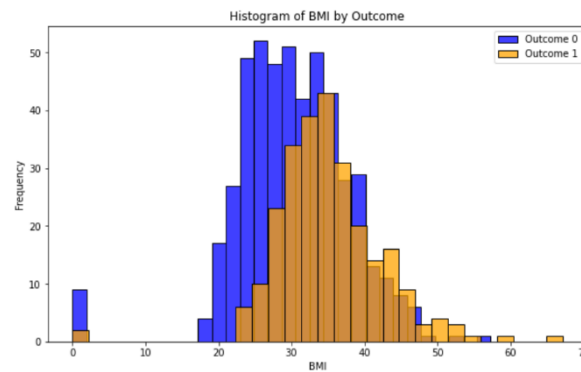


Figure 11: Histogram of BMI by Outcome

People who have a BMI between 30 and 40 are found to have diabetes from Figure 11. From Figure 12 it is clear that each column has outliers.[\[2\]](#) Hence pre-processing is required to get an optimized model to predict the Outcome.

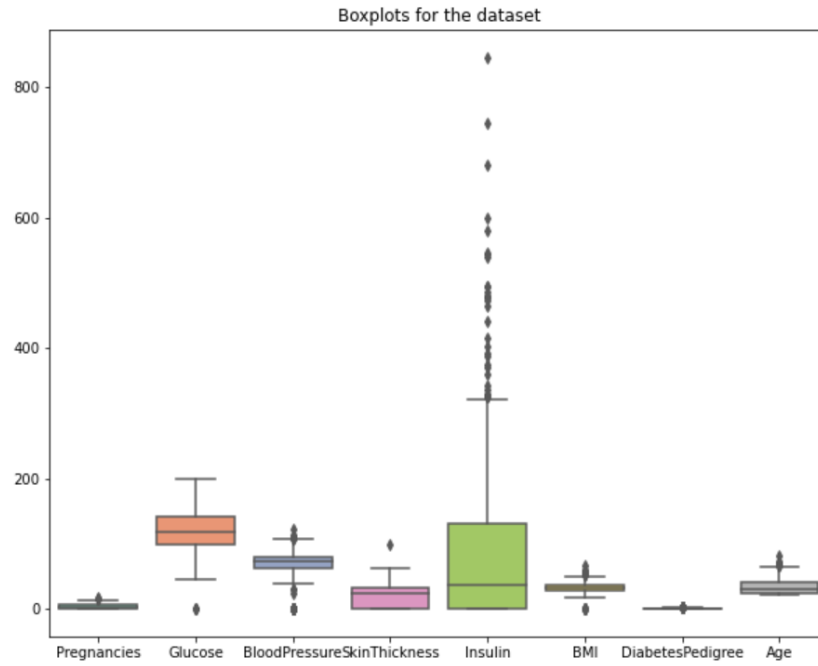


Figure 12: Boxplot for PimaDiabetes

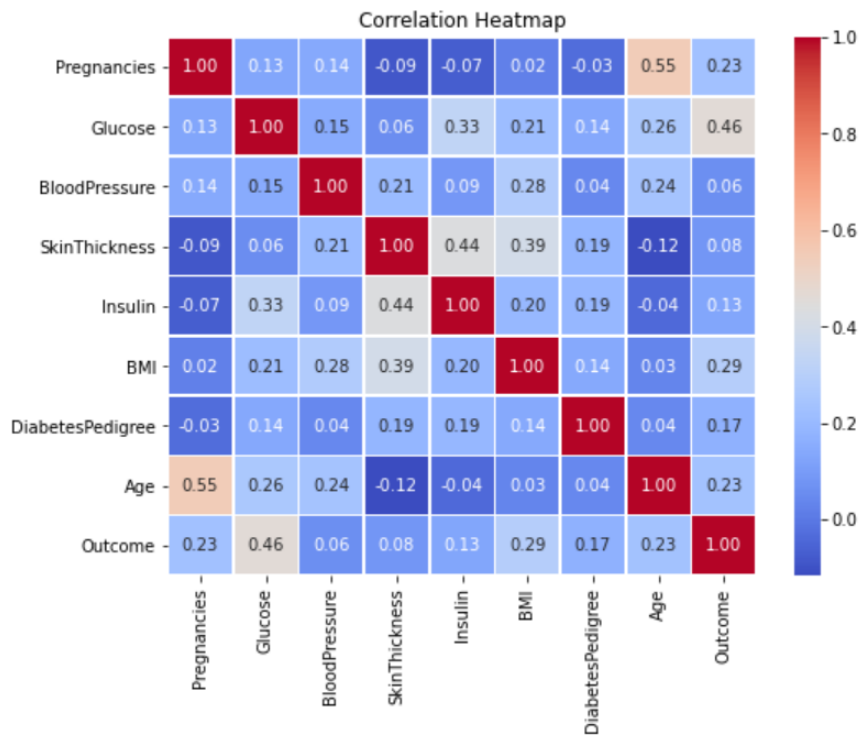


Figure 13: Heatmap of Correlation Matrix

The correlation heatmap from Figure 13 reveals that pregnancy and age have the highest correlation, while insulin and skin thickness and glucose and outcome have a stronger correlation.[\[3\]](#) Therefore, examining these relationships is crucial for a deeper understanding.

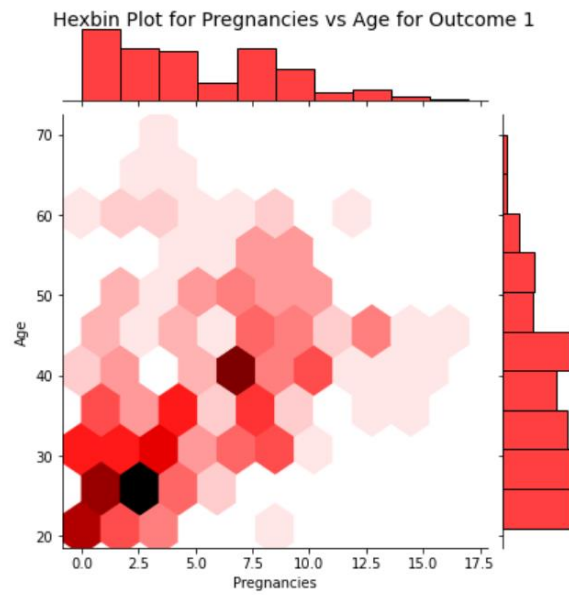


Figure 14: Hexbin plot for Pregnancies vs Age for Outcome 1

Based on the value of the outcome, a Hexbin is plotted after taking into account age and pregnancy.[4] Figure 14 shows that the greatest number of people with diabetes are found to be falling under the age group between 20 and 30 who have had less than 5 pregnancies. On the other hand, a lot of people between the age group of 20 and 30 with less than 4 pregnancies are also found to not have diabetes from Figure 15. However, both the graphs show that as age increases there is a tendency for the number of pregnancies also to increase.

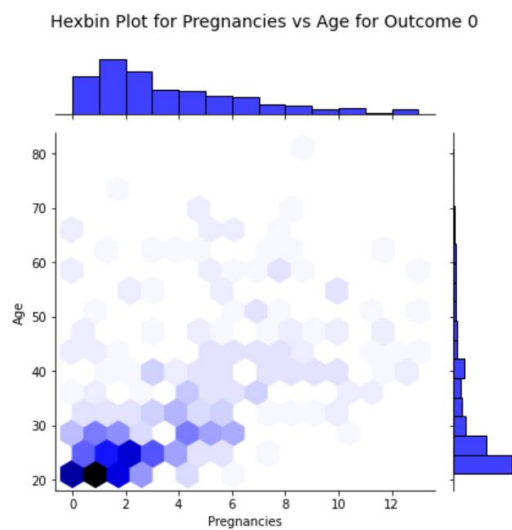


Figure 15: Hexbin plot for Pregnancies vs Age for Outcome 0

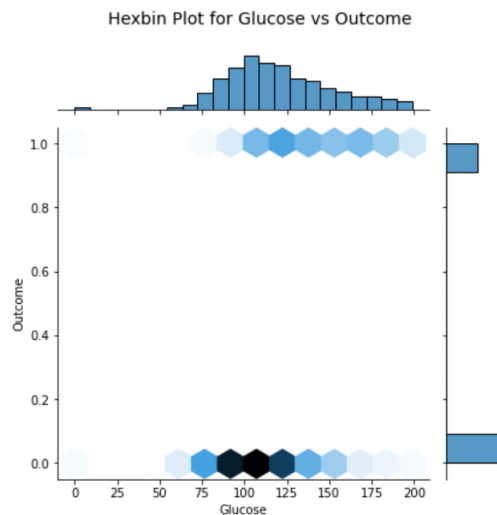


Figure 16: Hexbin plot for Glucose vs Outcome

From Figure 16, many of the women who do not have diabetes fall under the range of having glucose between 75 mg/dl and 125 mg/dl which is the normal range of glucose. Whereas, the greatest number of people with diabetes have over 125 mg/dl as their glucose level.



Figure 17: Hexbin plot for Skin Thickness vs Insulin

From Figure 17 it is clear that Skin Thickness and Insulin had higher correlation because of majority of their values in the records having a value of 0. This is inaccurate and these records should be imputed before being fitted in a model.

Probability of Developing Diabetes

A new field, `SevenOrMorePregnancies`, was created to determine the likelihood of diabetes in women with seven or more pregnancies. The data was divided into training and testing sets, and a Logistic Regression model was fitted using `SevenOrMorePregnancies` as the sole predictor. The model showed an accuracy of 0.6866 and an F-score of 0.405. The probability of developing diabetes was stored and the `predict_proba` function was used to determine the probability. The likelihood of getting diabetes with six or fewer pregnancies was 0.29464, and the probability of getting diabetes with seven or more pregnancies was 0.5787.

Regression Model for Predicting Outcome

The dataset was pre-processed using a KNN Imputer with a nearest neighbour value of 5 to impute 0 values for Glucose, Blood Pressure, Insulin, Skin Thickness, and BMI, and all fields except Outcome were standardized using a Standard Scaler. [\[5\]](#)

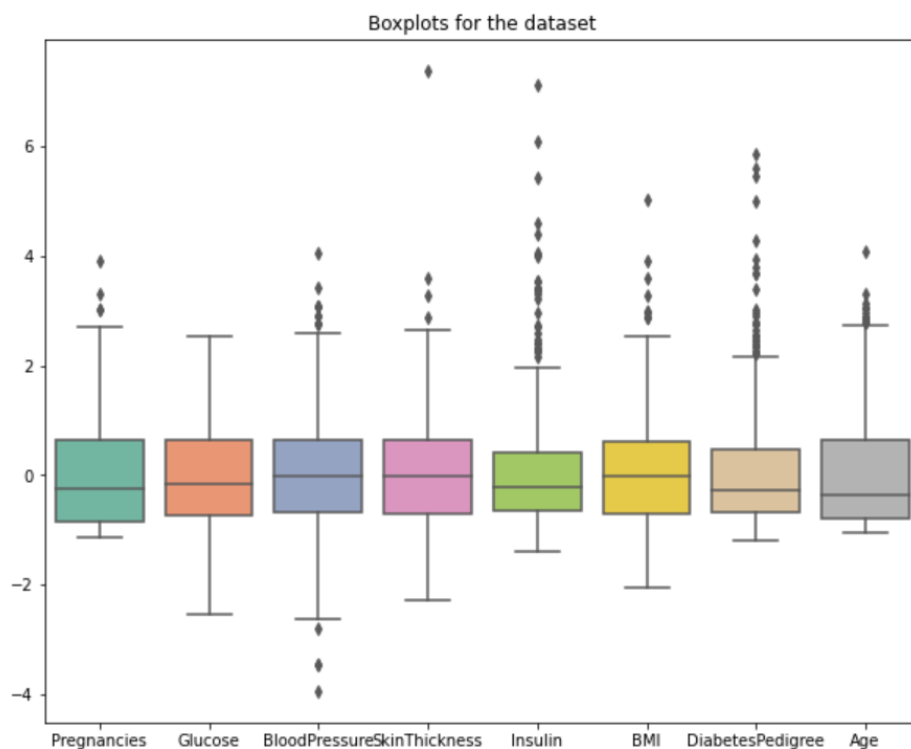


Figure 18: Boxplot for PimaDiabetes after Imputing and Scaling

Figure 18 displays a boxplot of fields post-imputing and scaling, revealing that all fields except field age have outliers, which requires removal. [\[6\]](#)

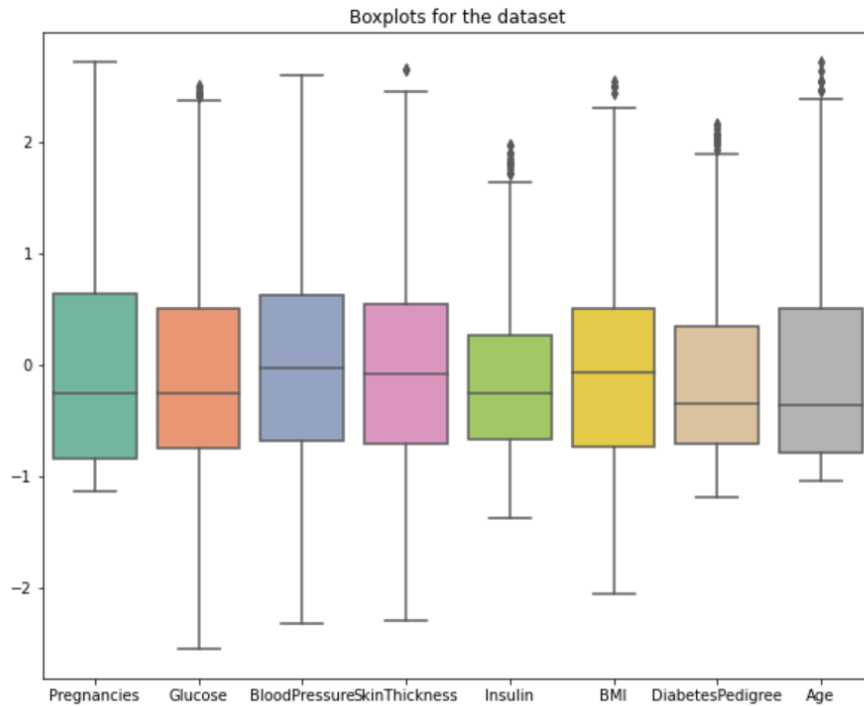


Figure 19: Boxplot after removing outliers from PimaDiabetes

Figure 19 displays a boxplot indicating a significant reduction in the number of outliers after removing them.

ToPredict is a dataset used for predicting outcomes and diabetes probability, consisting of 5 records with 0 values for Insulin and Skin Thickness in some of the records. Table 3 displays the entire dataset.

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

Table 3: ToPredict Dataset to predict the Outcome

The model uses KNN Imputer and Standard Scaler on the dataset, trained on the PimaDiabetes dataset, and logistic regression. The optimum feature combination is determined by using 80%

of the PimaDiabetes data as the training set and 20% as the testing set, with a feature significance graph initially produced. [7]

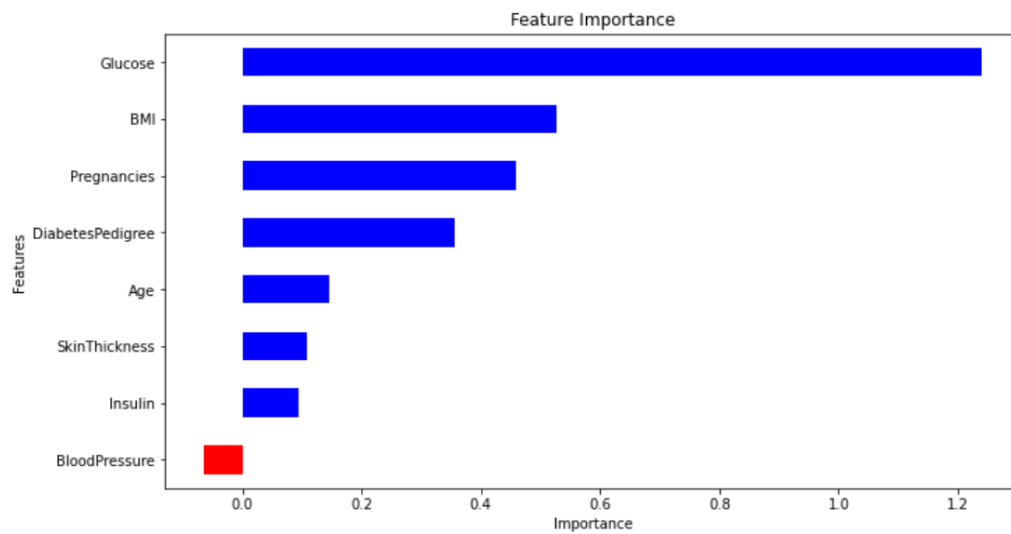


Figure 20: Feature Importance considering all variables as features

The model's accuracy is 0.712, with Blood pressure having a negative importance from Figure 20. Glucose, BMI, Pregnancies, Diabetes Pedigree, Age, and Insulin were used as features to increase accuracy to 0.719 for subsequent training. No feature has a negative impact on the model, as shown in Figure 21.

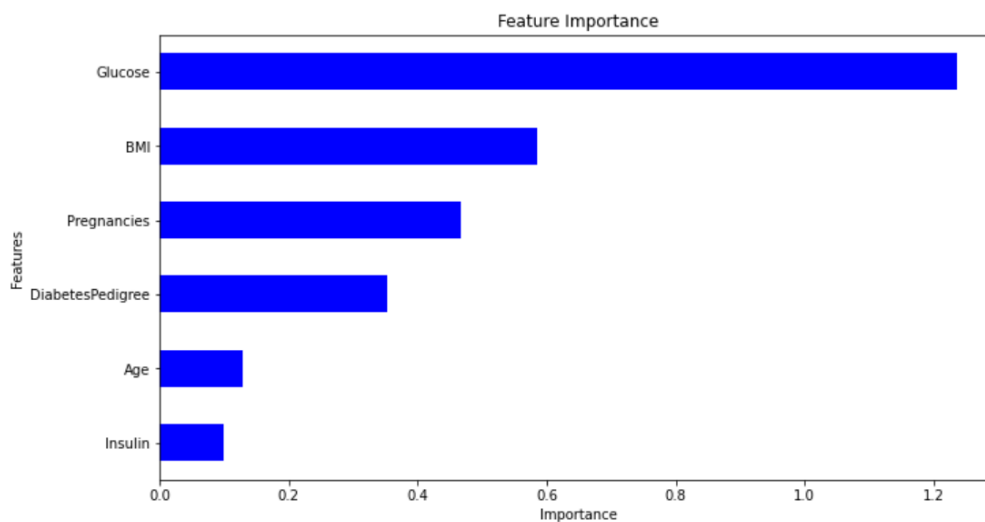


Figure 21: Feature Importance considering only Glucose, BMI, Pregnancies, DiabetesPedigree, Age and Insulin as Features

As a result, the model makes use of these features. To predict the outcome, it uses the ToPredict dataset after training on the whole PimaDiabetes dataset. The probability of developing diabetes was determined to be 0.389 using predict_proba ().

References

- [1] National Institute on Aging. (n.d.). *High Blood Pressure and Older Adults*. [online] Available at: <https://www.nia.nih.gov/health/high-blood-pressure/high-blood-pressure-and-older-adults#:~:text=Normal%20blood%20pressure%20for%20most>.
- [2] Nishida, K. (2019). *Introduction to Boxplot Chart in Exploratory*. [online] Medium. Available at: <https://blog.exploratory.io/introduction-to-boxplot-chart-in-exploratory-255c316a01ca> [Accessed 20 Nov. 2023].
- [3] Szabo, B. (2020). *How to Create a Seaborn Correlation Heatmap in Python?* [online] Medium. Available at: <https://medium.com/@szabo.bibor/how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e>.
- [4] He, S. (2023). *Mastering Hexbin Plotting in Python: A Beginner's Guide*. [online] Medium. Available at: <https://levelup.gitconnected.com/mastering-hexbin-plotting-in-python-a-beginners-guide-3626e0389c37#:~:text=Hexbin%20plots%20are%20a%20versatile> [Accessed 20 Nov. 2023].
- [5] Bhanupsingh (2023). *Handling Missing Data with KNN Imputer*. [online] Medium. Available at: <https://medium.com/@bhanupsingh484/handling-missing-data-with-knn-imputer-927d49b09015#:~:text=Applying%20KNN%20Imputer> [Accessed 20 Nov. 2023].
- [6] Bhandari, P. (2022). *How to Find Outliers / Meaning, Formula & Examples*. [online] Scribbr. Available at: <https://www.scribbr.co.uk/stats/statistical-outliers/>.
- [7] Serengil, S. (2021). *Feature Importance in Logistic Regression for Machine Learning Interpretability*. [online] Sefik Ilkin Serengil. Available at: <https://sefiks.com/2021/01/06/feature-importance-in-logistic-regression/>.

Appendix

```
In [39]: !jupyter nbconvert --to webpdf --allow-chromium-download Coursework.ipynb
```

```
[NbConvertApp] Converting notebook Coursework.ipynb to webpdf
[NbConvertApp] Building PDF
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 929033 bytes to Coursework.pdf
```

```
In [2]: pip install pandas seaborn
```

```
Requirement already satisfied: pandas in c:\users\reeth\anaconda3\lib\site-packages (1.5.3)Note: you may need to restart the kernel to use updated packages.
```

```
Requirement already satisfied: seaborn in c:\users\reeth\anaconda3\lib\site-packages (0.12.2)
```

```
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\reeth\anaconda3\lib\site-packages (from pandas) (2.8.2)
```

```
Requirement already satisfied: pytz>=2020.1 in c:\users\reeth\anaconda3\lib\site-packages (from pandas) (2022.7)
```

```
Requirement already satisfied: numpy>=1.21.0 in c:\users\reeth\anaconda3\lib\site-packages (from pandas) (1.24.3)
```

```
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\reeth\anaconda3\lib\site-packages (from seaborn) (3.7.1)
```

```
Requirement already satisfied: contourpy>=1.0.1 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
```

```
Requirement already satisfied: cycler>=0.10 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
```

```
Requirement already satisfied: fonttools>=4.22.0 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
```

```
Requirement already satisfied: packaging>=20.0 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
```

```
Requirement already satisfied: pillow>=6.2.0 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\reeth\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
```

```
Requirement already satisfied: six>=1.5 in c:\users\reeth\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
In [3]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.distributions.empirical_distribution import ECDF
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
```

```
In [4]: #Reading PimaDiabetes.csv file into df dataframe
df=pd.read_csv('PimaDiabetes.csv')
df.head()
```

```
Out[4]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	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

EDA using Visualization

In [5]: `df.describe()`

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000
mean	3.844000	120.737333	68.982667	20.489333	80.378667	31.959067	0.473544	33.166667
std	3.370085	32.019671	19.508814	15.918828	115.019198	7.927399	0.332119	11.708819
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.244000	24.000000
50%	3.000000	117.000000	72.000000	23.000000	36.500000	32.000000	0.377000	29.000000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	0.628500	40.750000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000

In [6]:

```

print("No of rows with value 0 in the following columns")
print("\nGlucose:")
print(len(df[df['Glucose']==0]))
print("\nBlood Pressure:")
print(len(df[df['BloodPressure']==0]))
print("\nSkin Thickness:")
print(len(df[df['SkinThickness']==0]))
print("\nInsulin:")
print(len(df[df['Insulin']==0]))
print("\nBMI:")
print(len(df[df['BMI']==0]))

```

No of rows with value 0 in the following columns

Glucose:
5

Blood Pressure:
35

Skin Thickness:
221

Insulin:
362

BMI:
11

In [7]:

```

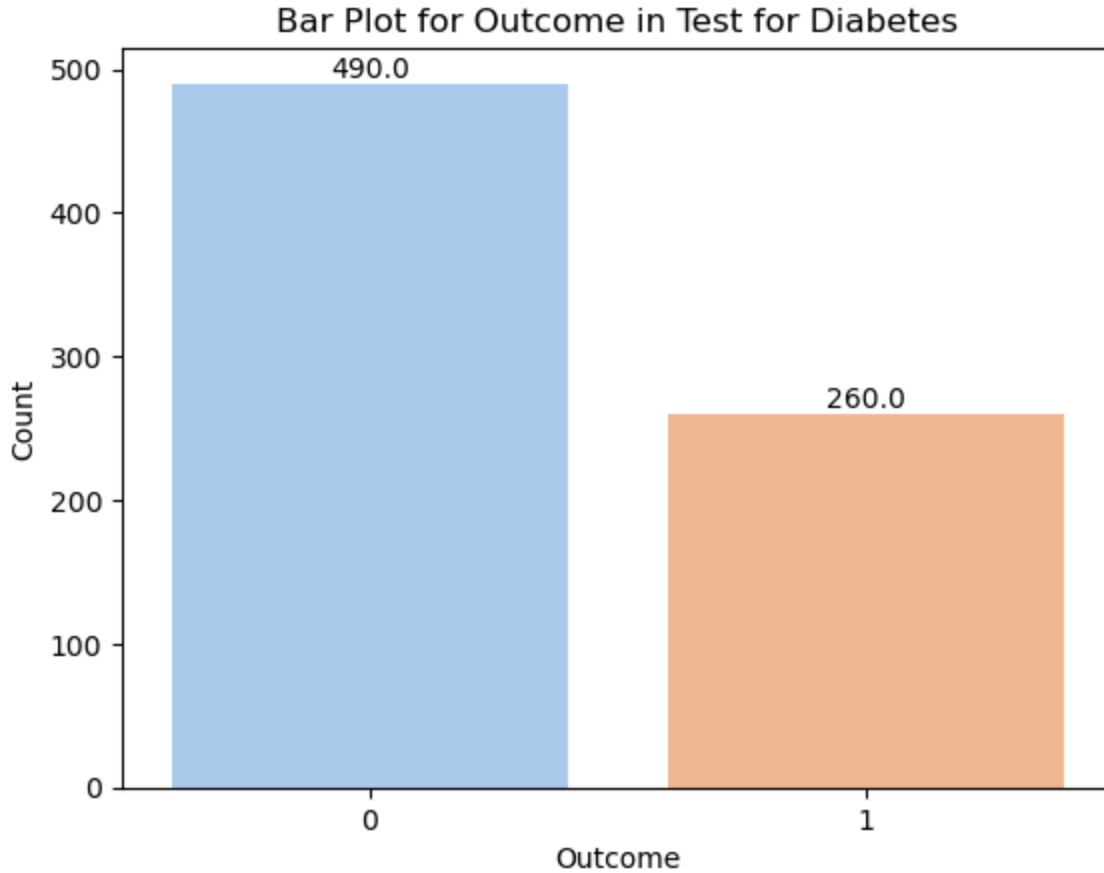
#Plotting Bar Plot to know how many women in the dataset have diabetes
bp=sns.countplot(x='Outcome', data=df, palette='pastel')
for i in bp.patches:
    x = i.get_x() + i.get_width() / 2
    y = i.get_height()

```

```

bp.annotate(y, (x, y), ha='center', va='bottom')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.title('Bar Plot for Outcome in Test for Diabetes')
plt.show()

```



```

In [8]: #Plotting a combined Rug Plot, Histogram and Kernel Density Plot for each column in the
for col in ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'Dia

plt.figure(figsize=(8, 4))
seaborn_version_str = sns.__version__
version_str_parts = seaborn_version_str.split('.')
if int(version_str_parts[1]) < 11:
    kde_axes = sns.distplot(df[col],
                            kde_kws={"label": "Kernel Density", "color": "black"},
                            hist_kws={"label": "Histogram", "color": "lightsteelblue"}
else:
    kde_axes = sns.kdeplot(df[col], color="black", label="Kernel Density")
    sns.histplot(df[col], stat="density", color="lightsteelblue", label="Histogram")

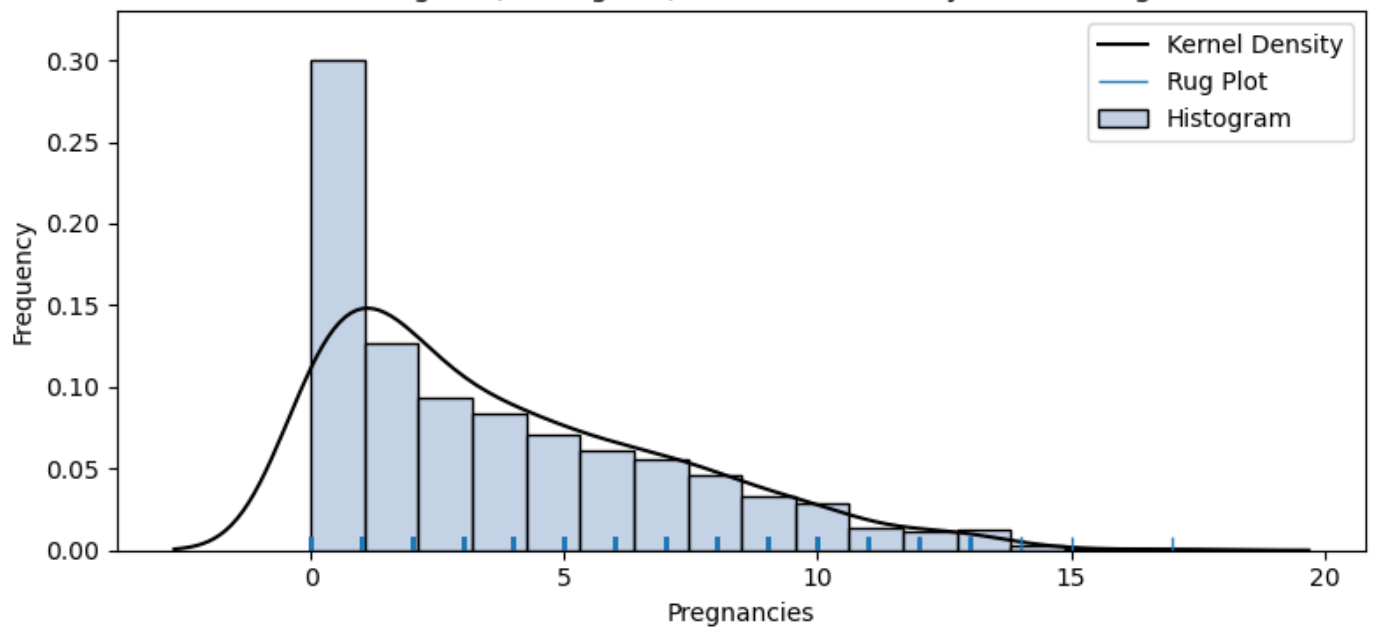
sns.rugplot(df[col], label="Rug Plot")

# Add labels and title
plt.xlabel(col)
plt.ylabel('Frequency')
plt.title('Combined Rug Plot, Histogram, and Kernel Density Plot for '+col)
plt.legend()
plt.tight_layout()

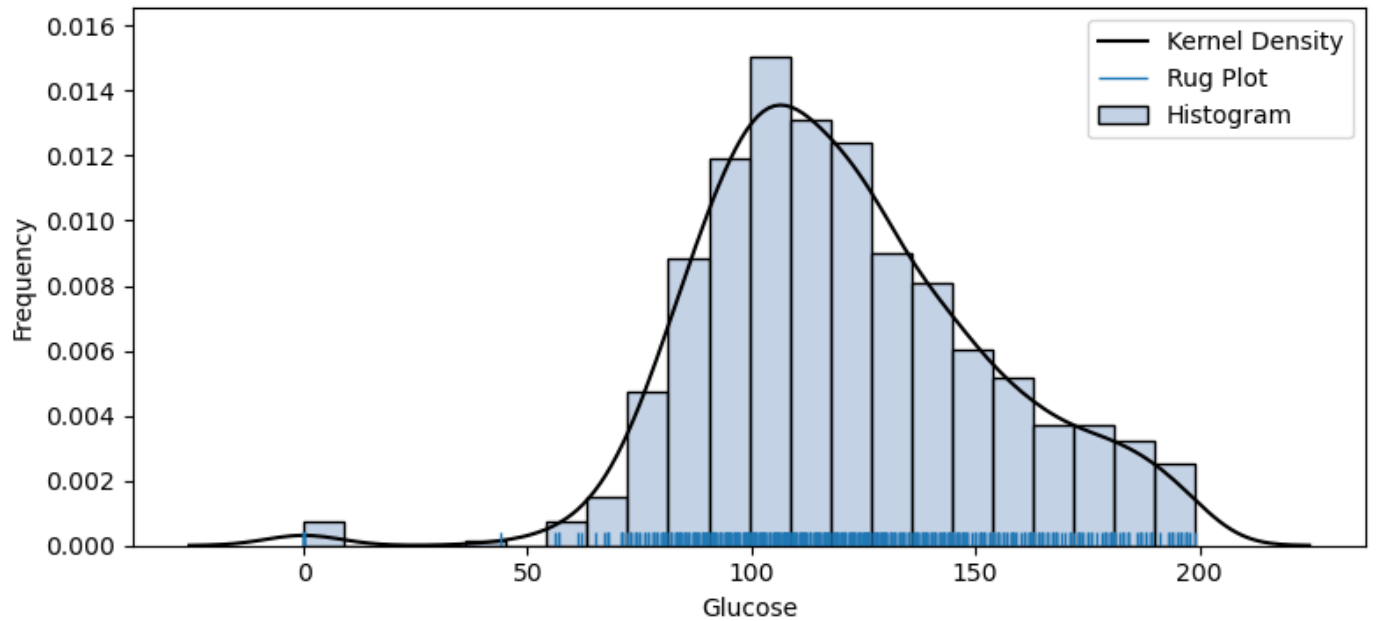
# Show the plot
plt.show()

```

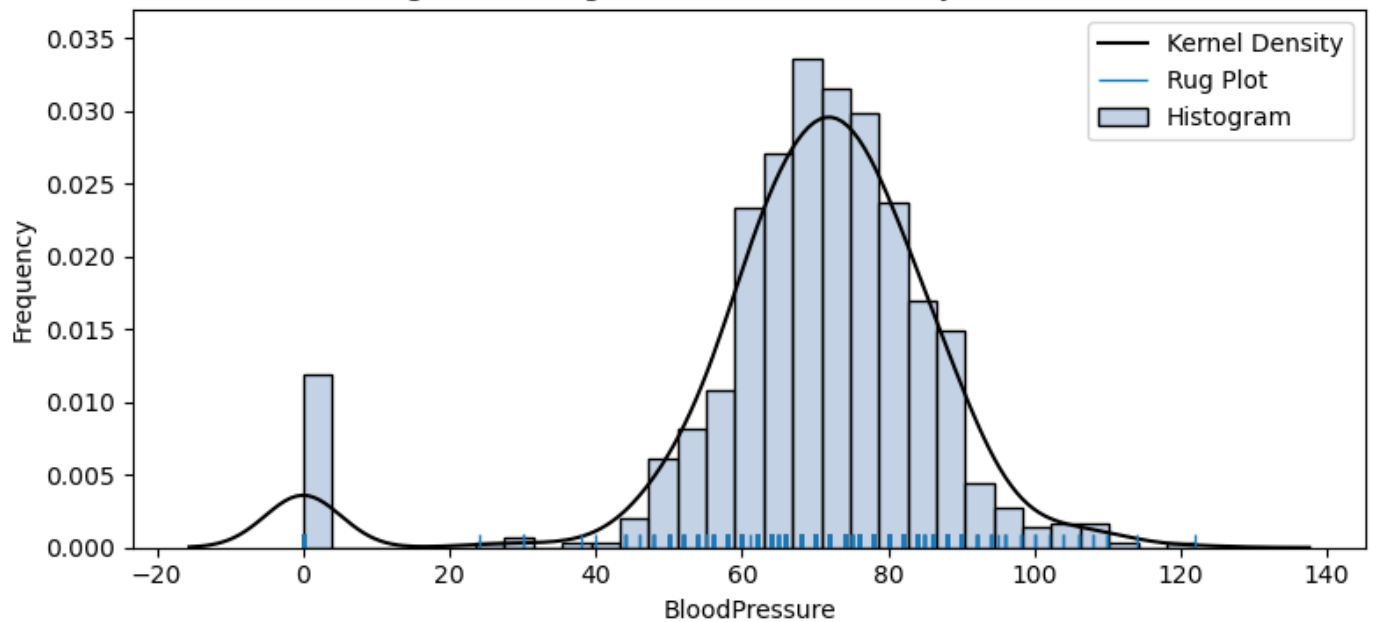

Combined Rug Plot, Histogram, and Kernel Density Plot for Pregnancies



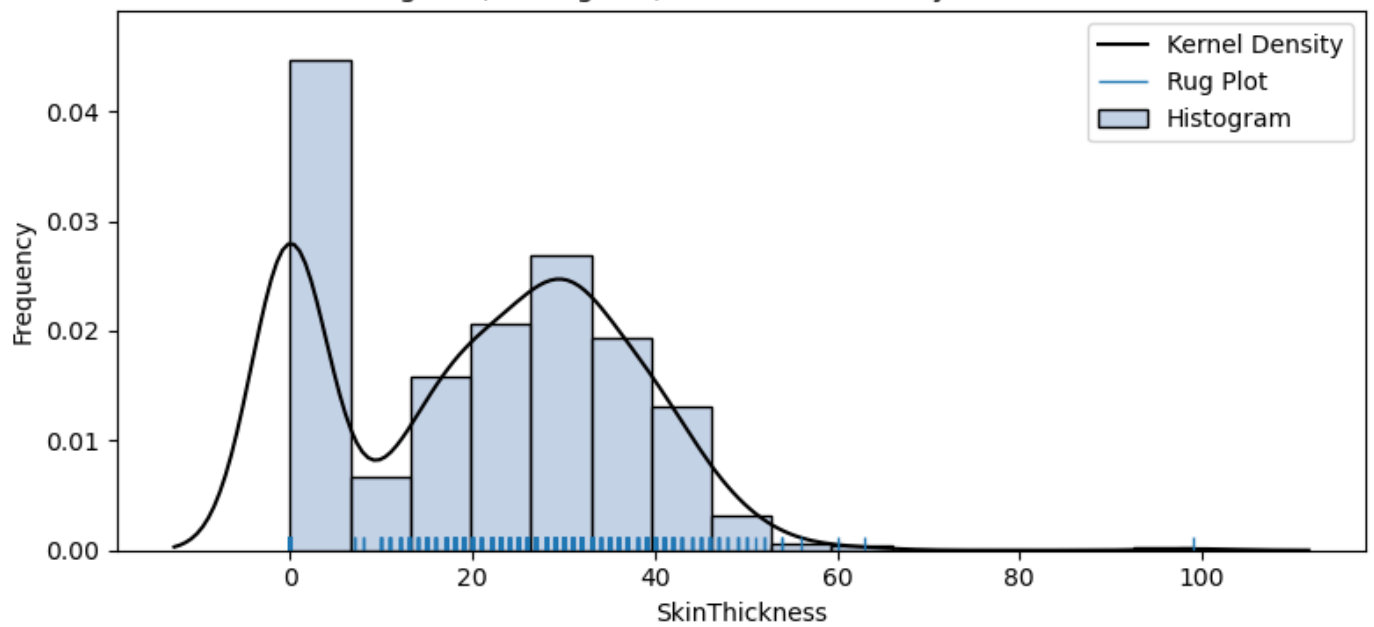
Combined Rug Plot, Histogram, and Kernel Density Plot for Glucose



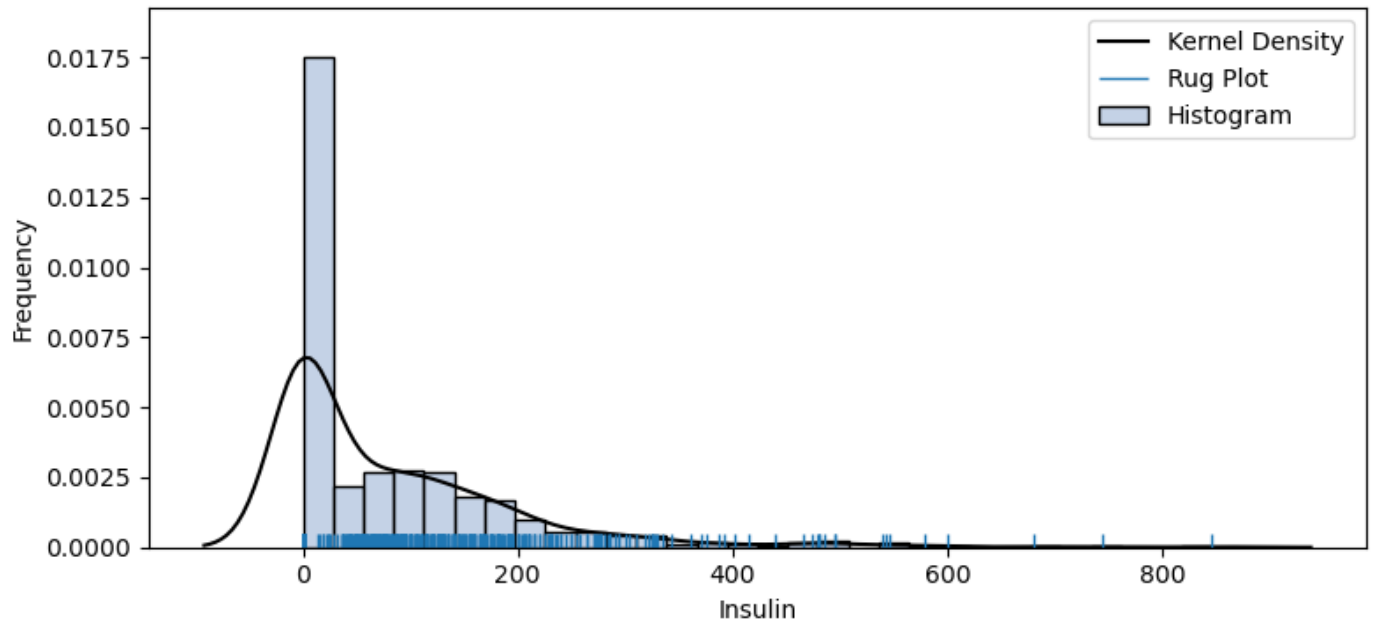
Combined Rug Plot, Histogram, and Kernel Density Plot for BloodPressure



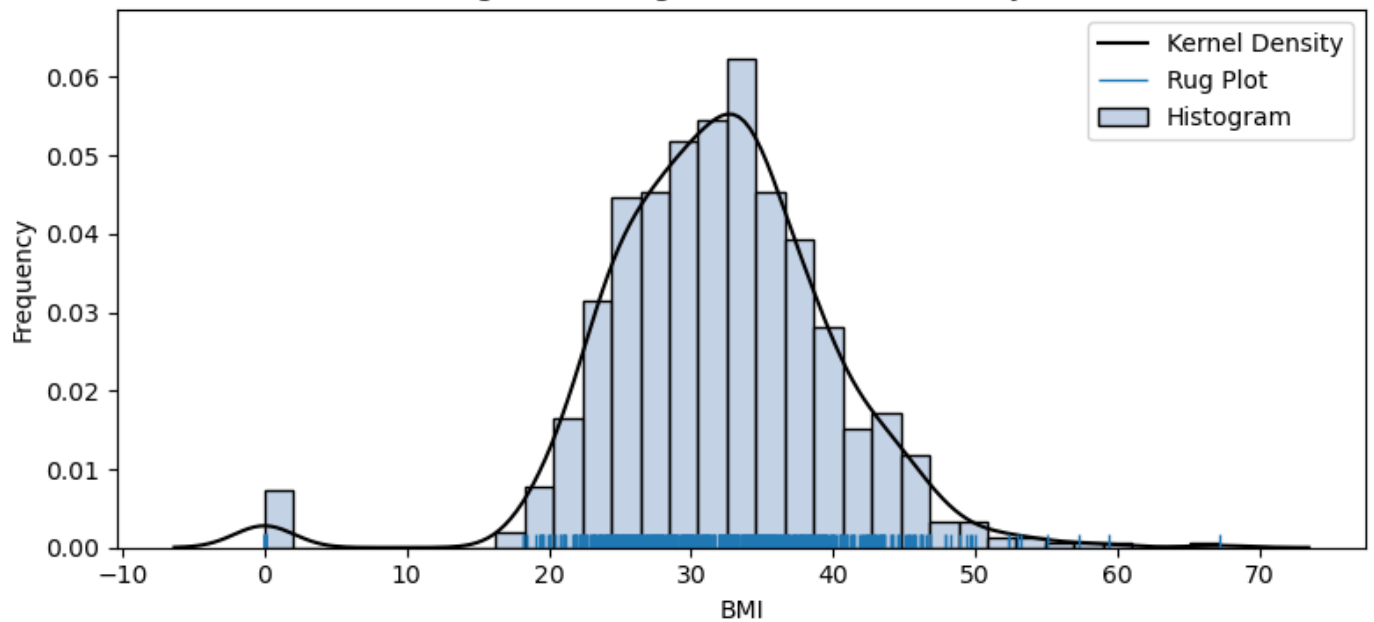
Combined Rug Plot, Histogram, and Kernel Density Plot for SkinThickness



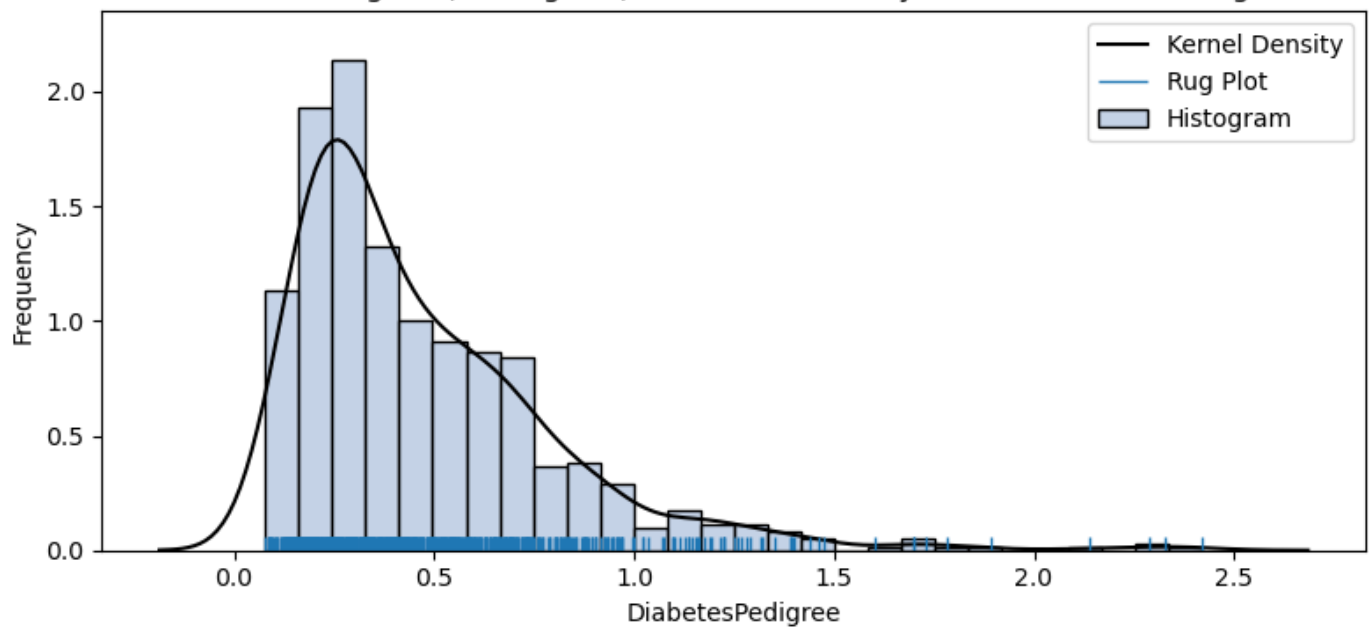
Combined Rug Plot, Histogram, and Kernel Density Plot for Insulin



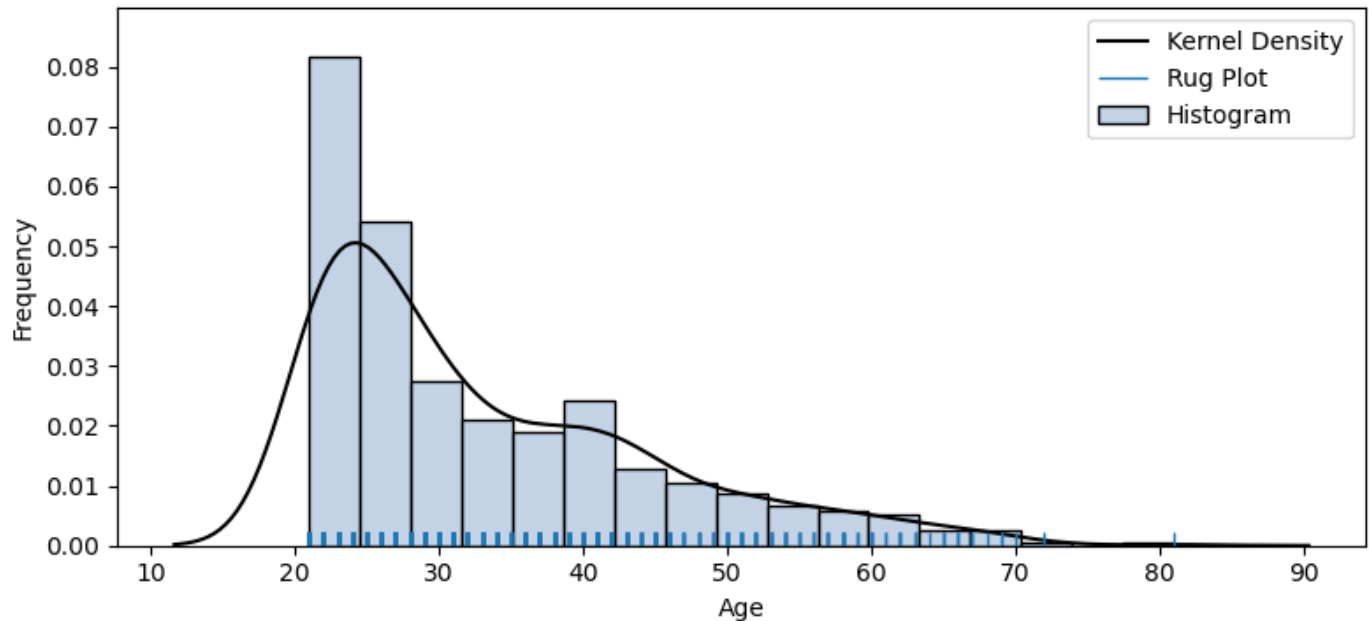
Combined Rug Plot, Histogram, and Kernel Density Plot for BMI



Combined Rug Plot, Histogram, and Kernel Density Plot for DiabetesPedigree



Combined Rug Plot, Histogram, and Kernel Density Plot for Age

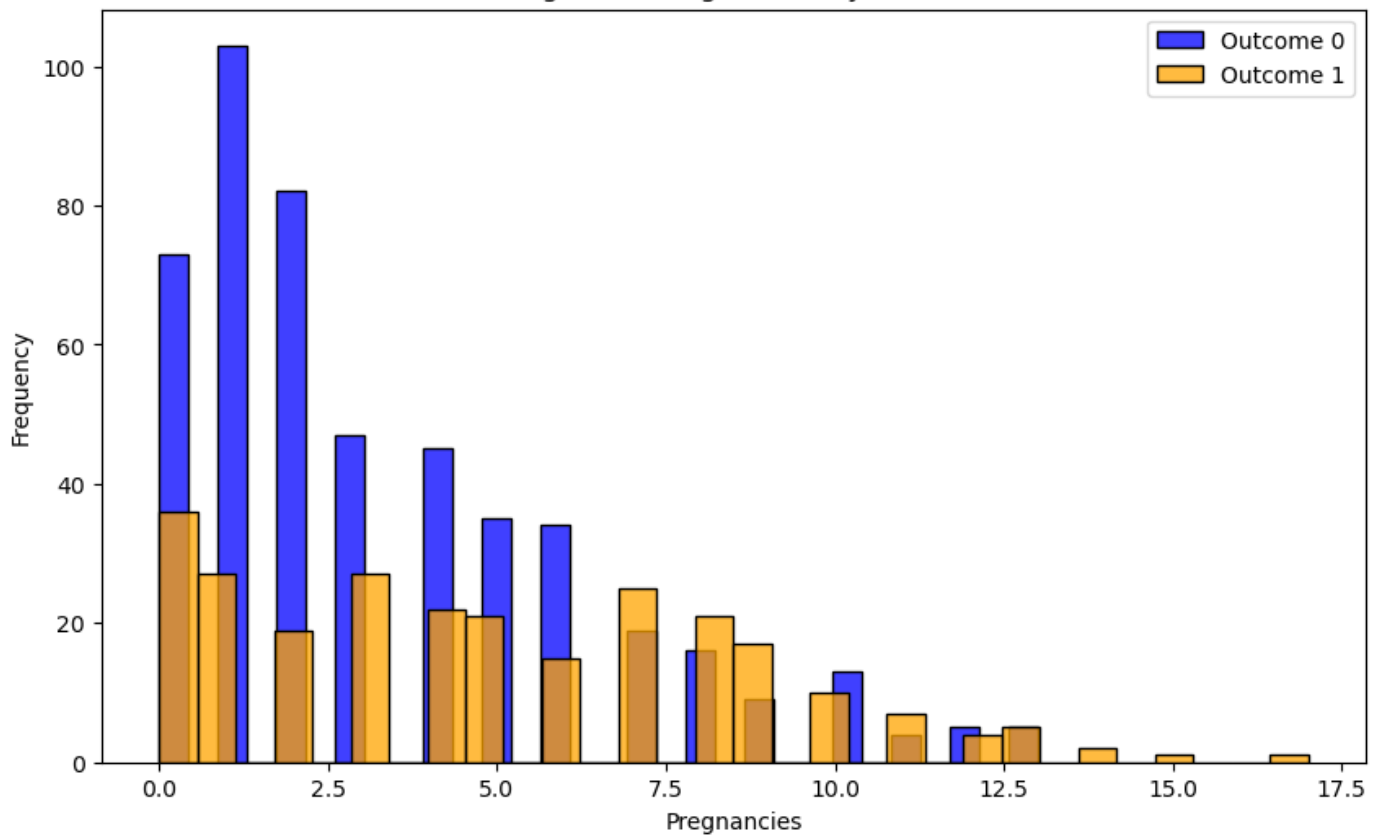


```
In [9]: outcome_0 = df[df['Outcome'] == 0]
outcome_1 = df[df['Outcome'] == 1]

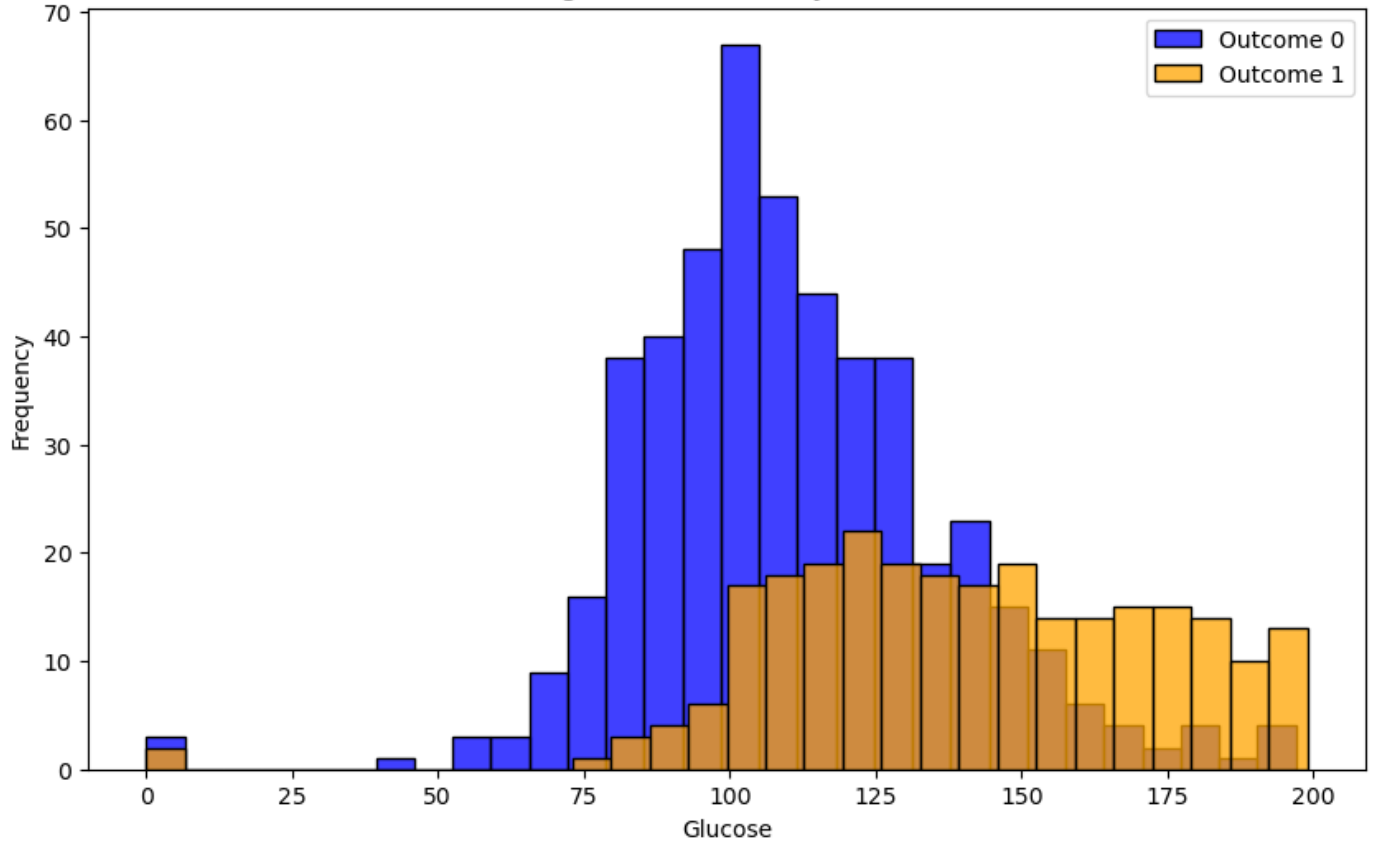
# Plot histograms
for col in ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'Dia
plt.figure(figsize=(10, 6))
sns.histplot(outcome_0[col], bins=30, kde=False, color='blue', label='Outcome 0')
sns.histplot(outcome_1[col], bins=30, kde=False, color='orange', label='Outcome 1')

# Add labels and title
plt.xlabel(col)
plt.ylabel('Frequency')
plt.title('Histogram of ' + col + ' by Outcome')
plt.legend()
```

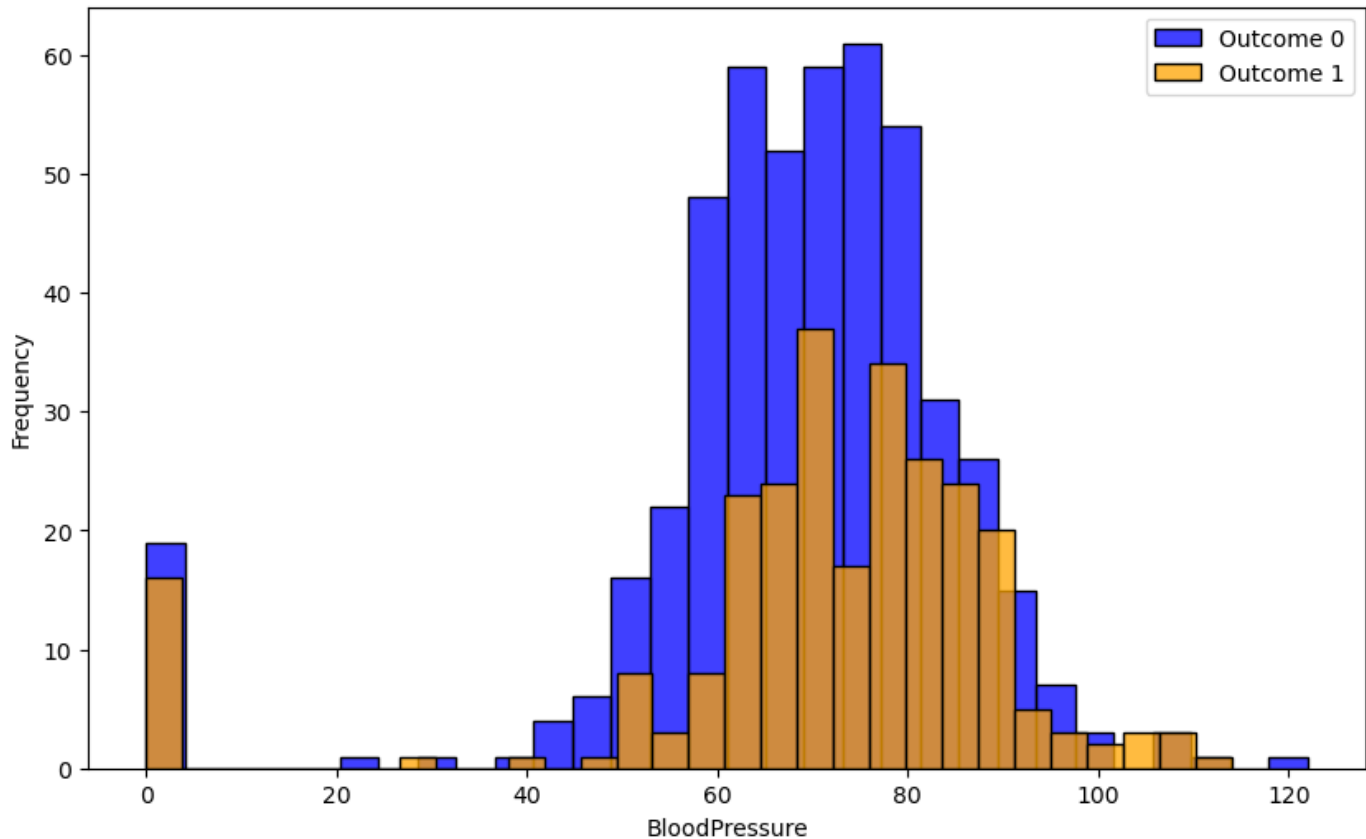
Histogram of Pregnancies by Outcome



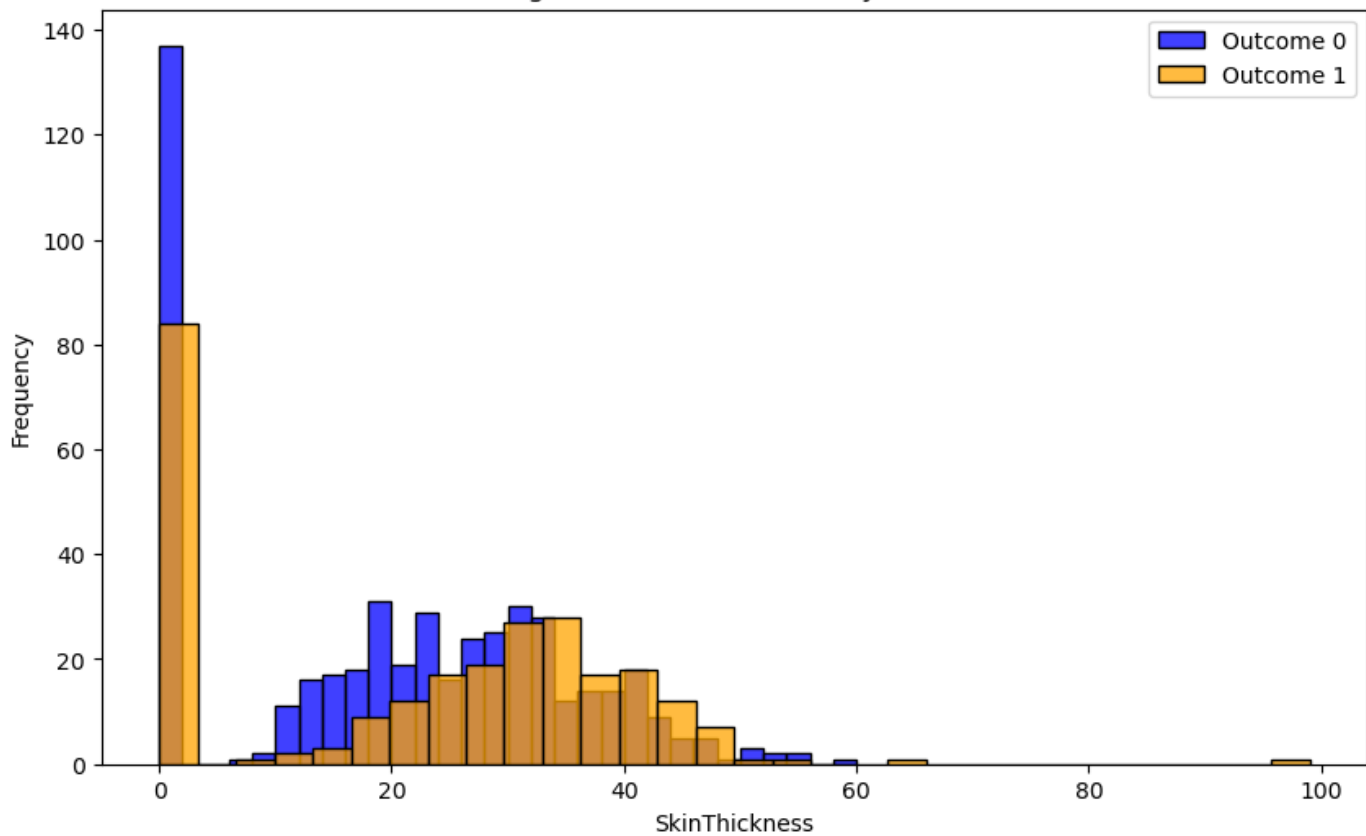
Histogram of Glucose by Outcome



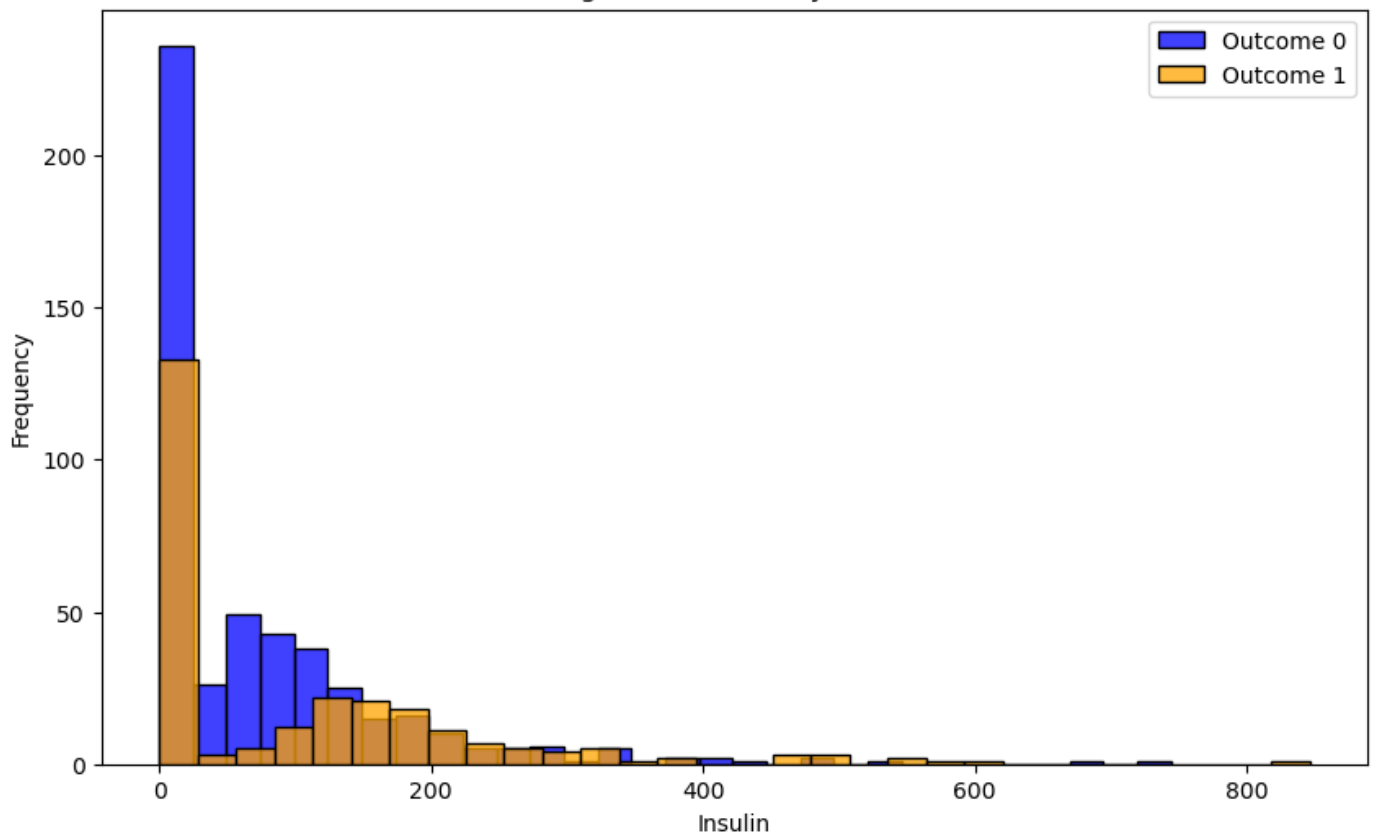
Histogram of BloodPressure by Outcome



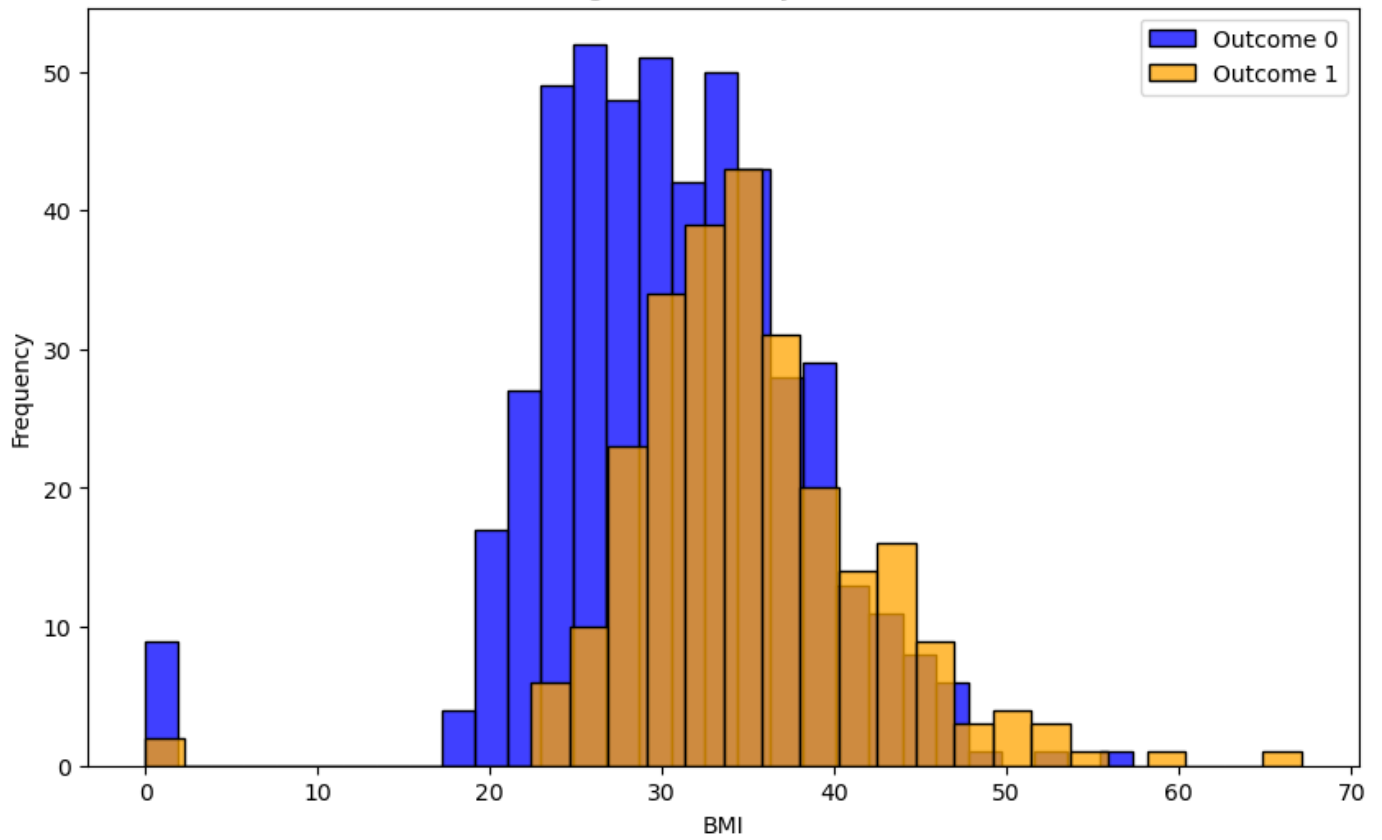
Histogram of SkinThickness by Outcome

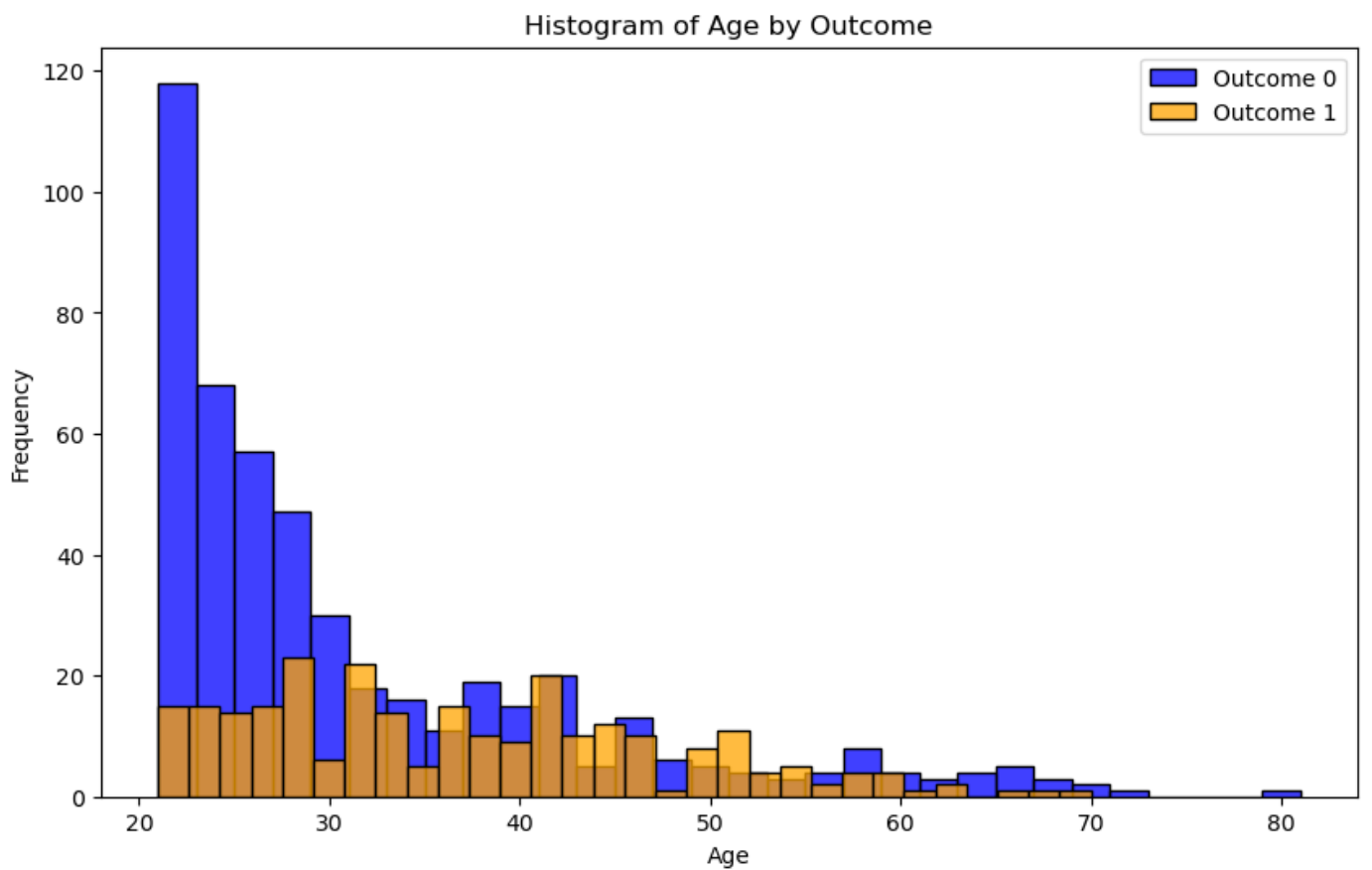
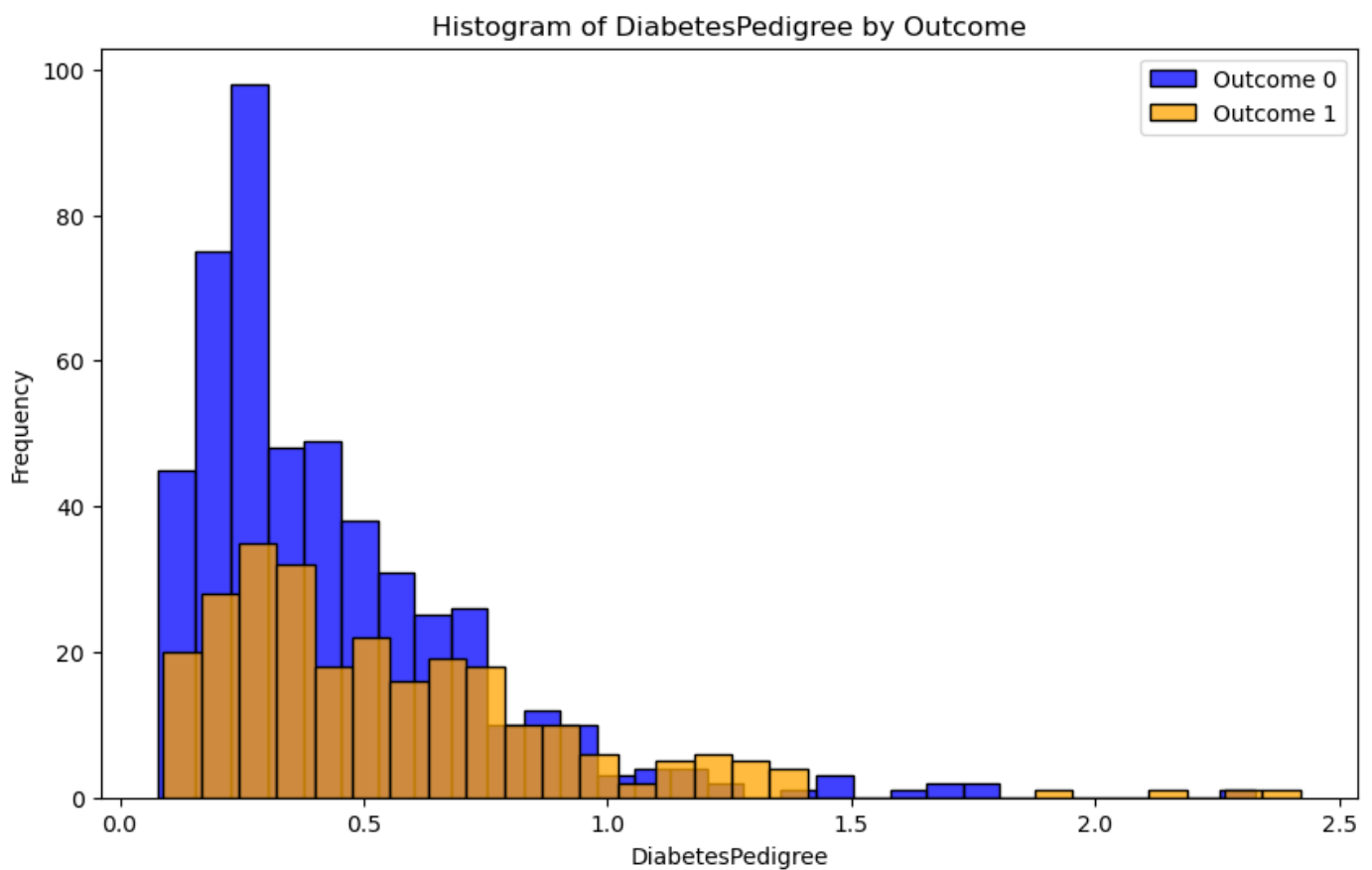


Histogram of Insulin by Outcome



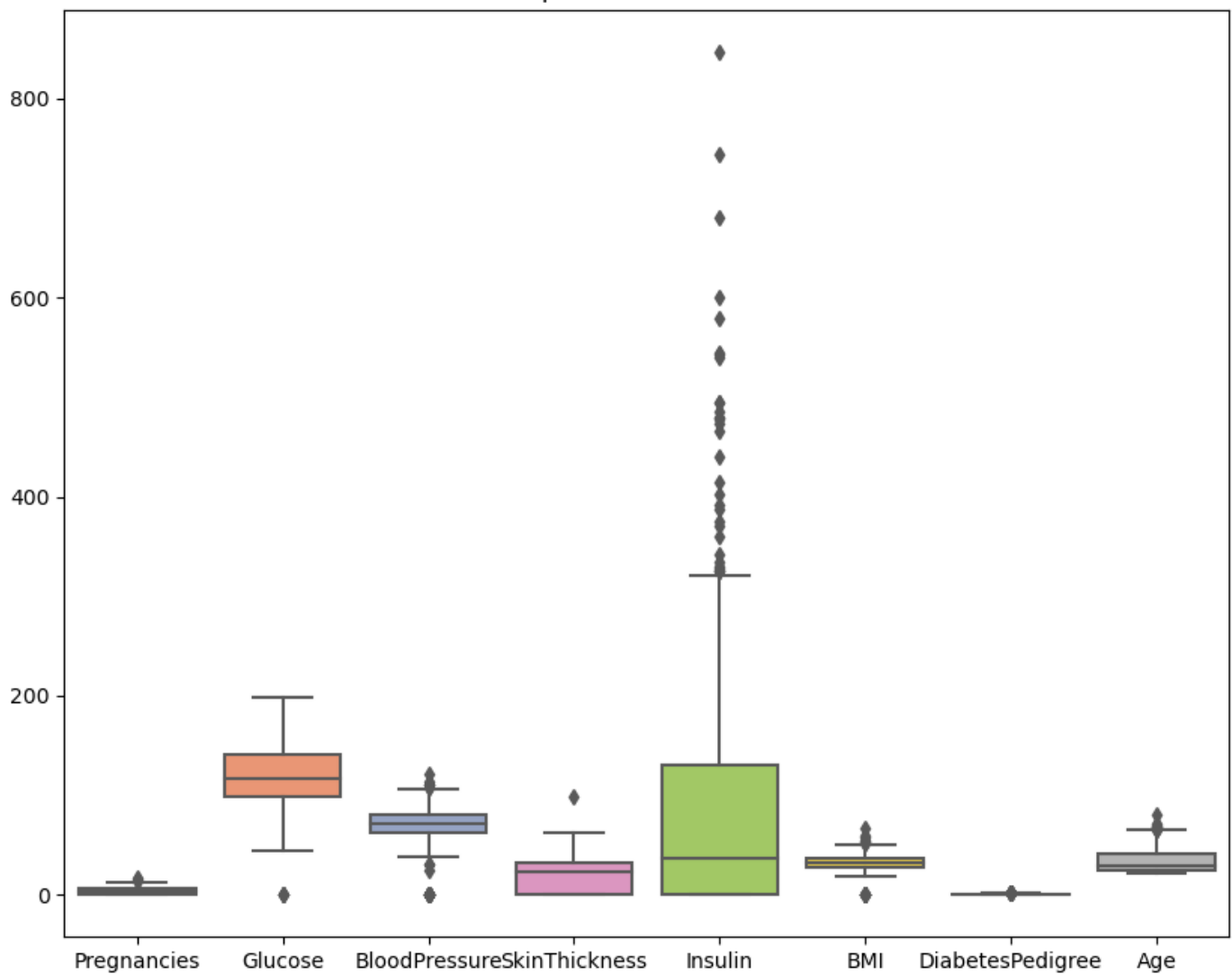
Histogram of BMI by Outcome



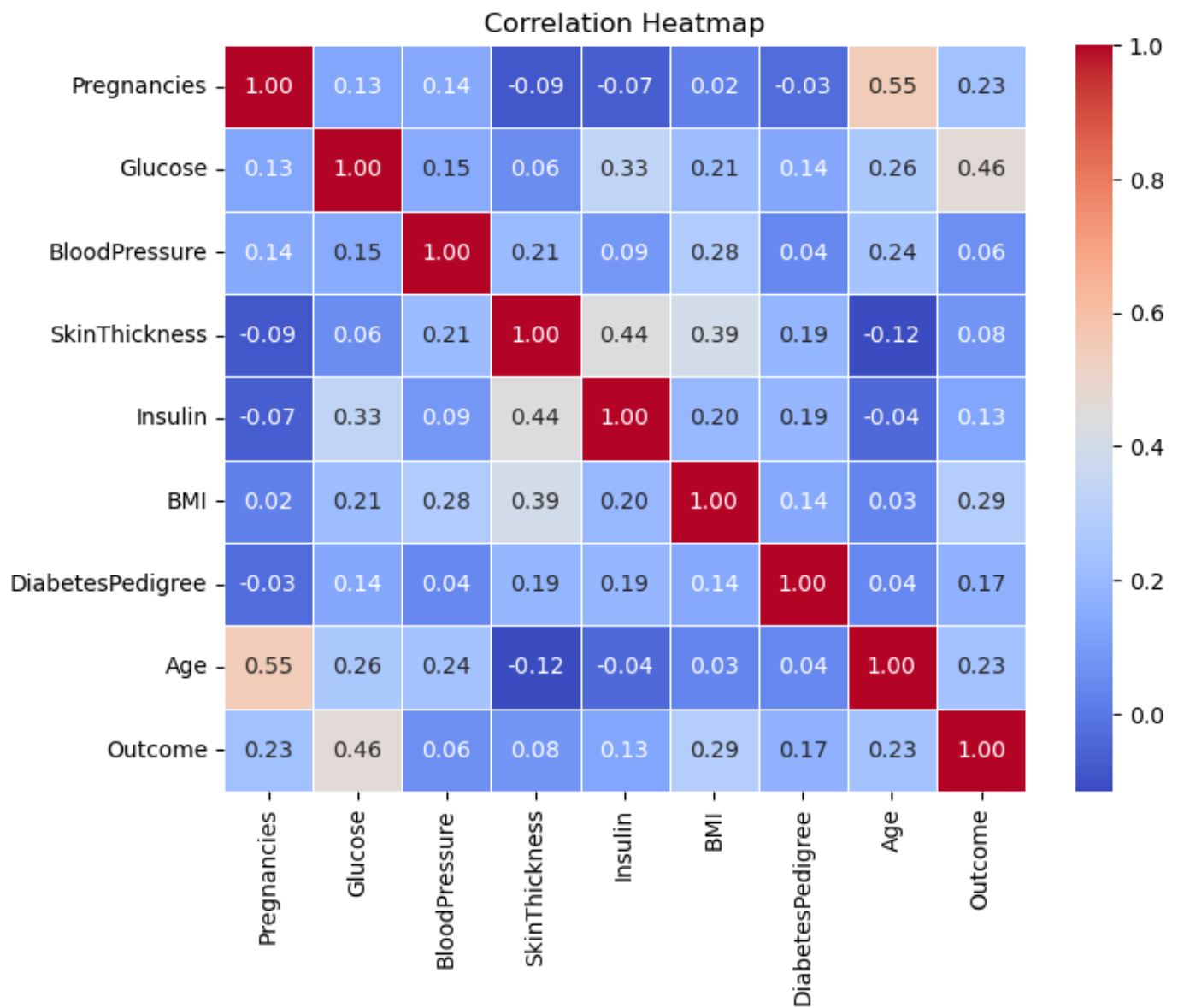


```
In [10]: #Boxplot for the dataset
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
plt.figure(figsize=(10, 8))
sns.boxplot(data=df[col],palette='Set2')
plt.title('Boxplots for the dataset')
plt.show()
```

Boxplots for the dataset



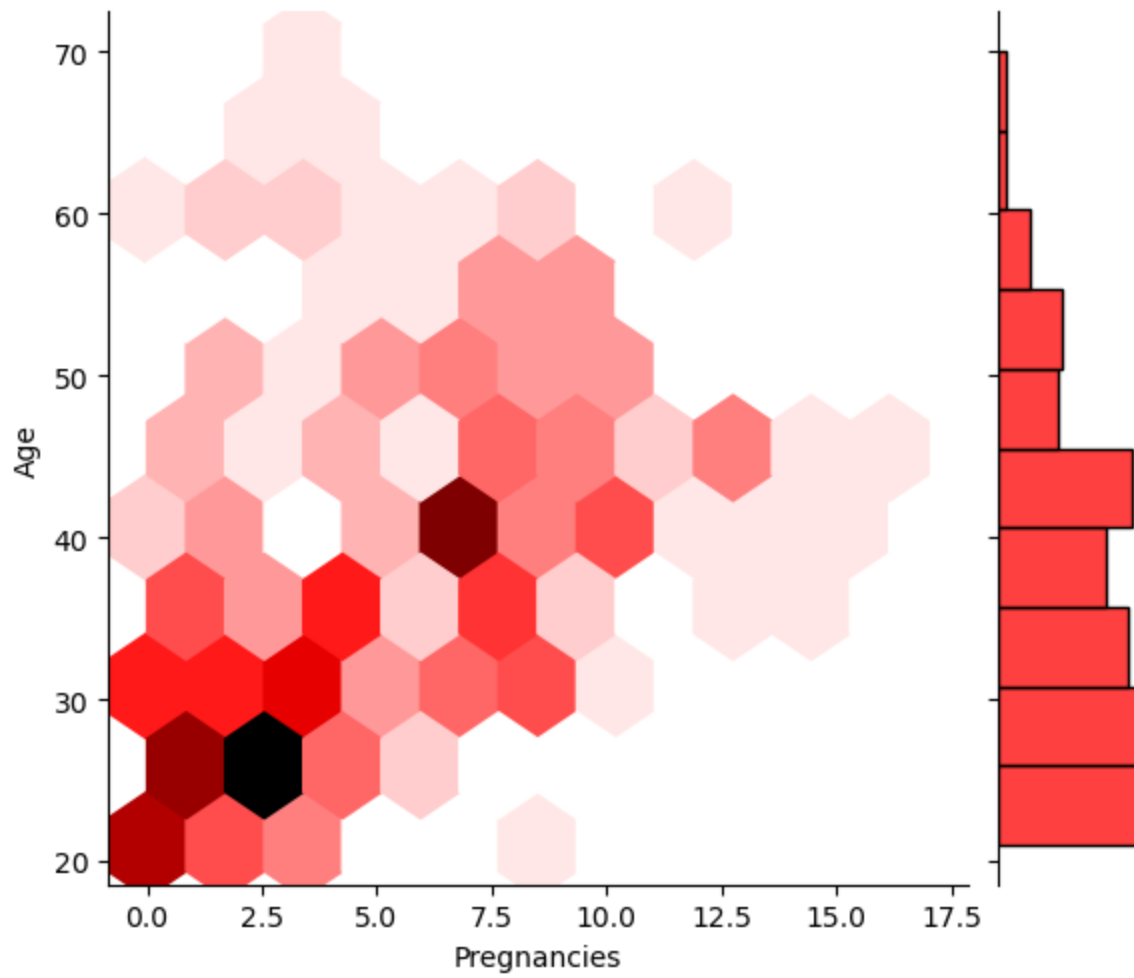
```
In [11]: #Plotting correlation heatmap for the dataset
corr=df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```

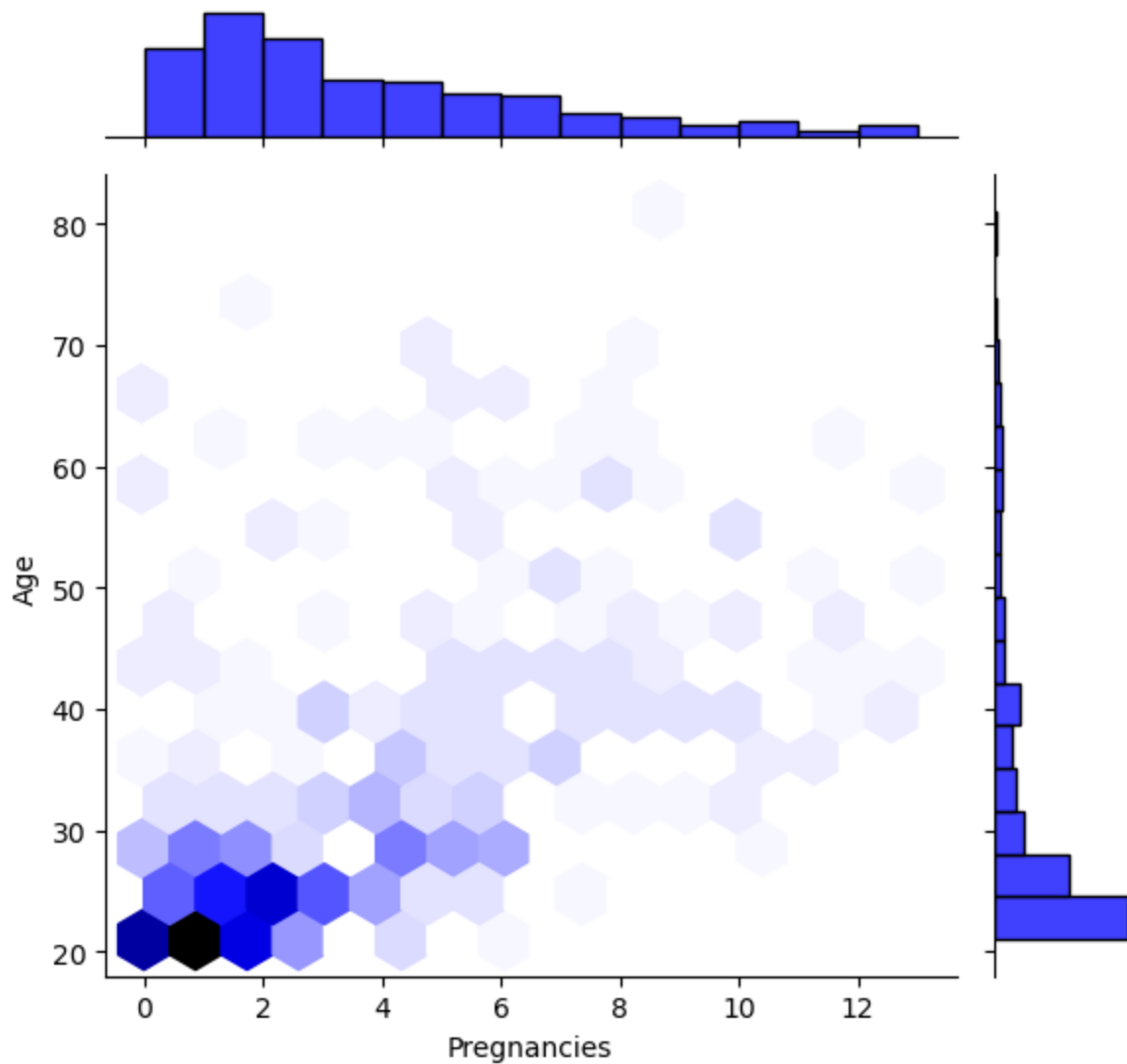
```
In [12]: #Hexbin Plot for Pregnancy vs Age for each group of Outcome
plt.figure(figsize=(8, 6))
for group, color in zip(df['Outcome'].unique(), ['red', 'blue']):
    sns.jointplot(x='Pregnancies', y='Age', data=df[df['Outcome'] == group], kind='hex',
    plt.xlabel('Pregnancies')
    plt.ylabel('Age')
    plt.suptitle('Hexbin Plot for Pregnancies vs Age for Outcome ' +str(group), x=0.5, y
plt.tight_layout()
plt.show()
```

<Figure size 800x600 with 0 Axes>

Hexbin Plot for Pregnancies vs Age for Outcome 1



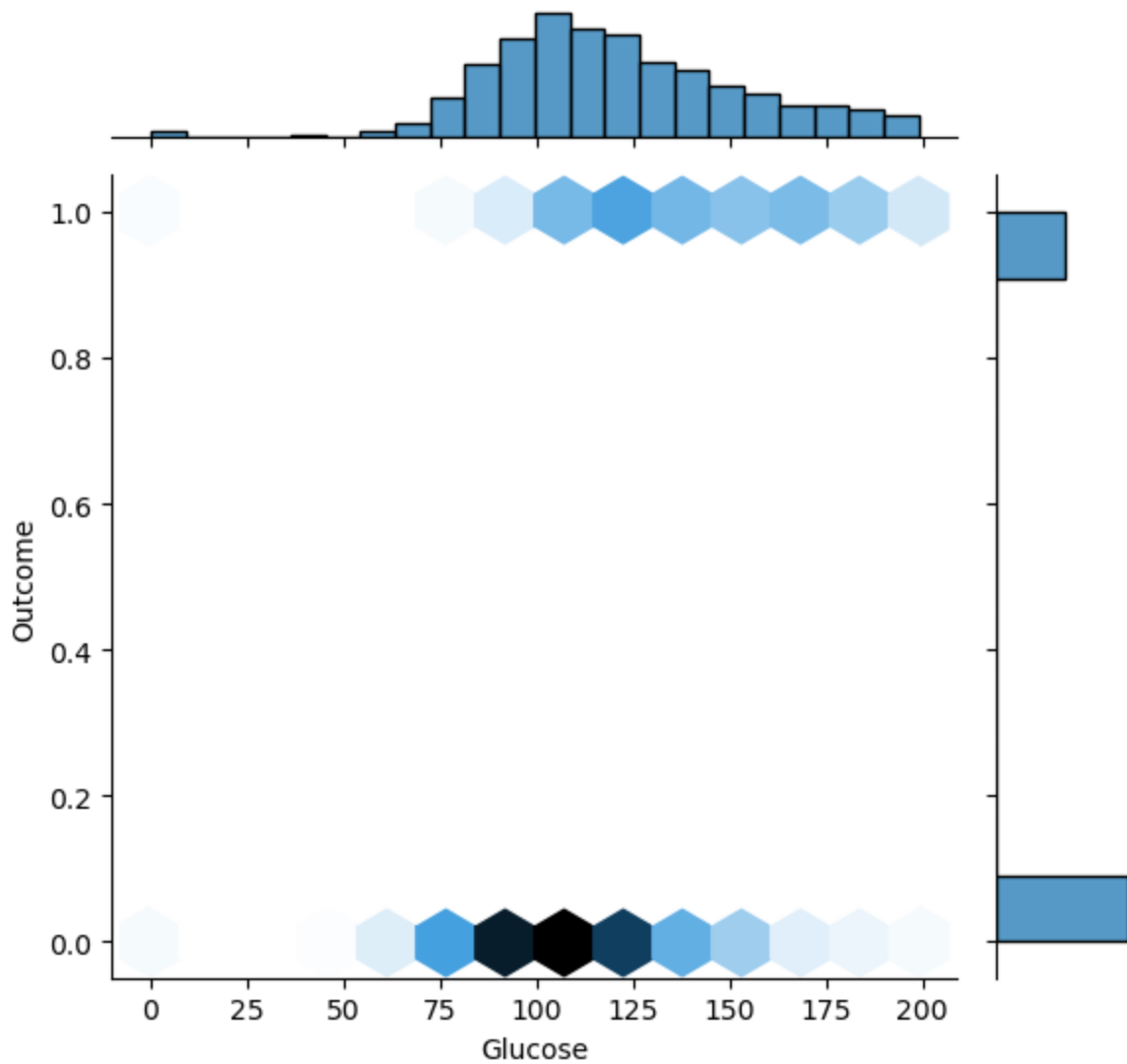
Hexbin Plot for Pregnancies vs Age for Outcome 0



```
In [13]: #Hexbin Plot for Glucose vs Outcome
plt.figure(figsize=(8, 6))
sns.jointplot(x=df['Glucose'], y=df['Outcome'], kind='hex')
plt.xlabel('Glucose')
plt.ylabel('Outcome')
plt.suptitle('Hexbin Plot for Glucose vs Outcome', x=0.5, y=1, ha='center', fontsize=14)
plt.tight_layout()
plt.show()
```

<Figure size 800x600 with 0 Axes>

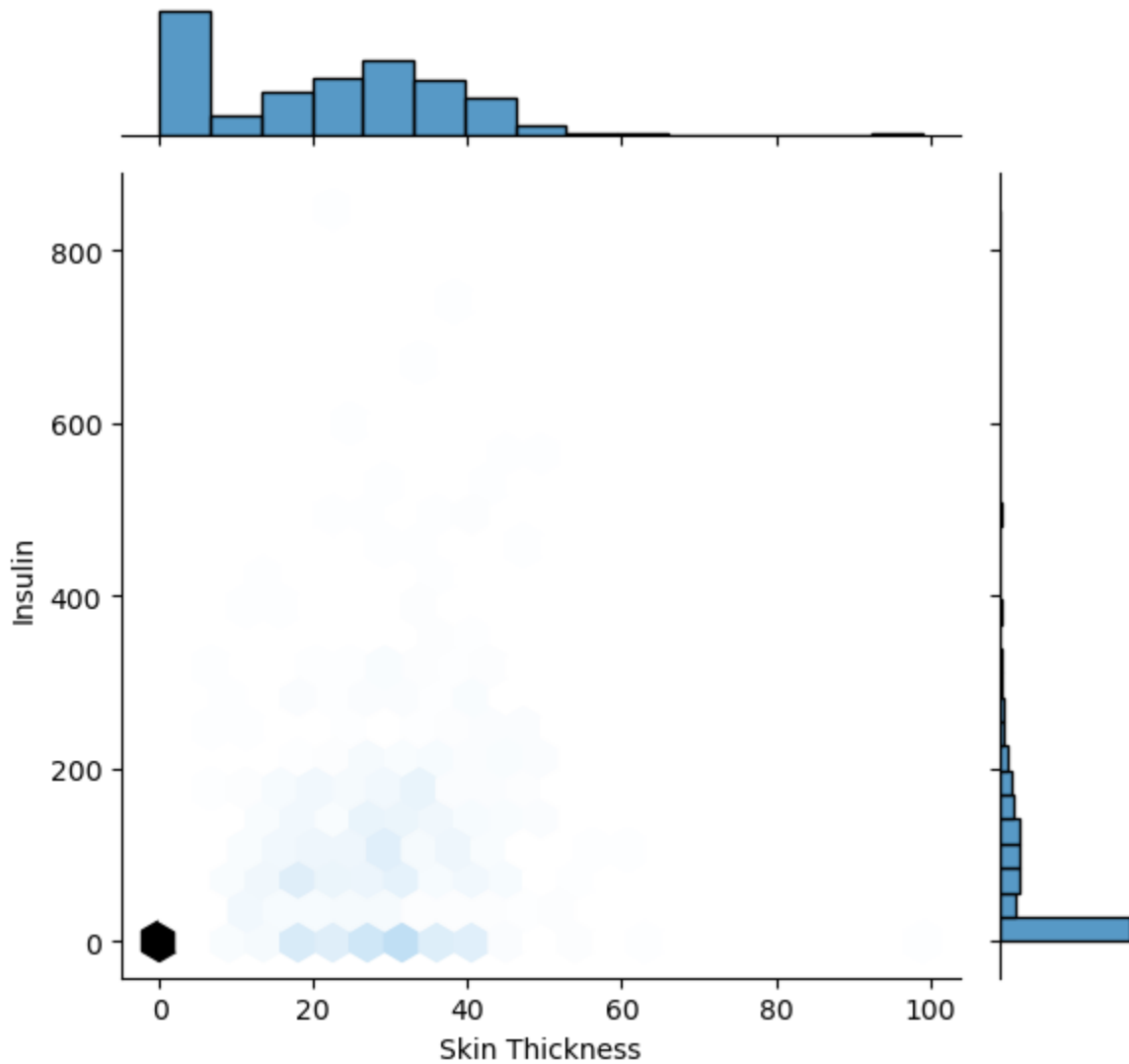
Hexbin Plot for Glucose vs Outcome



```
In [14]: #Hexbin Plot for Skin Thickness vs Insulin
plt.figure(figsize=(8, 6))
sns.jointplot(x=df['SkinThickness'], y=df['Insulin'], kind='hex')
plt.xlabel('Skin Thickness')
plt.ylabel('Insulin')
plt.suptitle('Hexbin Plot for Skin Thickness vs Insulin', x=0.5, y=1, ha='center', fonts
plt.tight_layout()
plt.show()
```

<Figure size 800x600 with 0 Axes>

Hexbin Plot for Skin Thickness vs Insulin



Question 3

```
In [15]: #Create a new column called SevenOrMorePregnancies
df['SevenOrMorePregnancies']=(df['Pregnancies']>=7).astype(int)
df.head(10)
```

```
Out[15]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome	SevenOrMorePregnancies
0	6	148	72	35	0	33.6	0.627	50	1	0
1	1	85	66	29	0	26.6	0.351	31	0	0
2	8	183	64	0	0	23.3	0.672	32	1	1
3	1	89	66	23	94	28.1	0.167	21	0	0
4	0	137	40	35	168	43.1	2.288	33	1	1
5	5	116	74	0	0	25.6	0.201	30	0	0
6	3	78	50	32	88	31.0	0.248	26	1	1
7	10	115	0	0	0	35.3	0.134	29	0	0
8	2	197	70	45	543	30.5	0.158	53	1	1
9	8	125	96	0	0	0.0	0.232	54	1	1

```
In [16]: #Splitting the dataset into training and testing set
X = df[['SevenOrMorePregnancies']]
Y = df['Outcome']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
In [17]: #Using Logistic Regression and fitting the model
model = LogisticRegression()
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)

print('Accuracy: ' + str(accuracy_score(Y_test, Y_pred)))
print('F-score: ' + str(f1_score(Y_test, Y_pred)))
```

```
Accuracy: 0.6866666666666666
F-score: 0.4050632911392405
```

```
In [18]: #Finding the probability of getting diabetes

#Storing the probability for the second class which is when Outcome is 1 into p
p=model.predict_proba(X_test)[:, 1]

#Probability of getting diabetes with six or fewer pregnancies
six_less=p[X_test['SevenOrMorePregnancies'] == 0].mean()

#Probability of getting diabetes with seven or more pregnancies
seven_more = p[X_test['SevenOrMorePregnancies'] == 1].mean()

print('Probability of diabetes with six or fewer pregnancies: ' + str(six_less))
print('Probability of diabetes with seven or more pregnancies: ' + str(seven_more))
```

```
Probability of diabetes with six or fewer pregnancies: 0.29464902942023524
Probability of diabetes with seven or more pregnancies: 0.5787003497647526
```

Question 4

Preprocessing of PimaDiabetes.csv

```
In [19]: #Columns to fill 0 values
col = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI' ]

# 0 is replaced with NaN
df[col] = df[col].replace(0, np.nan)

#Copy of df with the selected columns
i = df[col].copy()

# KNN Imputer
knn = KNNImputer(n_neighbors=5)
i_arr = knn.fit_transform(i)

# Convert the array to dataframe
imputed = pd.DataFrame(i_arr, columns=col)

# Update df with the new values from dataframe imputed
df.loc[:, col] = imputed

df.describe()
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000
mean	3.844000	121.612000	72.304533	28.782933	149.704267	32.434507	0.473544	33.166667
std	3.370085	30.480089	12.237632	9.508398	97.765886	6.912747	0.332119	11.708819
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000
25%	1.000000	99.000000	64.000000	22.000000	85.300000	27.500000	0.244000	24.000000
50%	3.000000	117.000000	72.000000	28.500000	129.200000	32.300000	0.377000	29.000000
75%	6.000000	141.000000	80.000000	35.000000	189.500000	36.600000	0.628500	40.750000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000

In [21]:

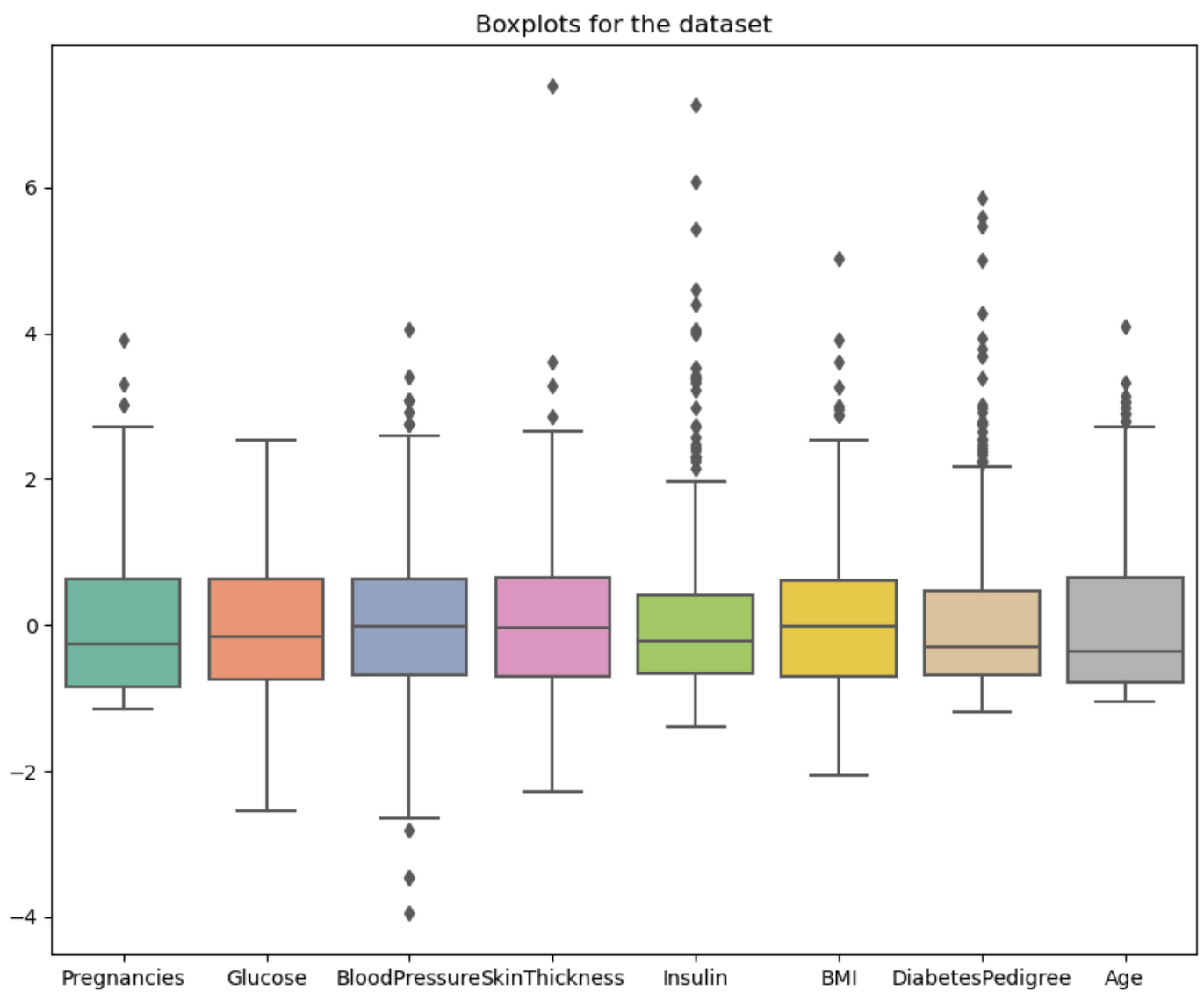
```
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigree']
X=df[col]

#StandardScaler
s=StandardScaler()
X_scale = s.fit_transform(X)

# Replace the values in df
df[col] = X_scale
```

In [22]:

```
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigree']
plt.figure(figsize=(10, 8))
sns.boxplot(data=df[col],palette='Set2')
plt.title('Boxplots for the dataset')
plt.show()
```



```
In [23]: def outliers(col):
    Q1 = col.quantile(0.25)
    Q3 = col.quantile(0.75)
    IQR = Q3 - Q1
    l=Q1 - 1.5 * IQR
    u=Q3 + 1.5 * IQR
    return col.apply(lambda x: x if (x>=l and x<=u) else None)
```

```
In [24]: col=['Pregnancies','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigree','Age']
for c in col:
    df[c] = outliers(df[c])
df = df.dropna()
```

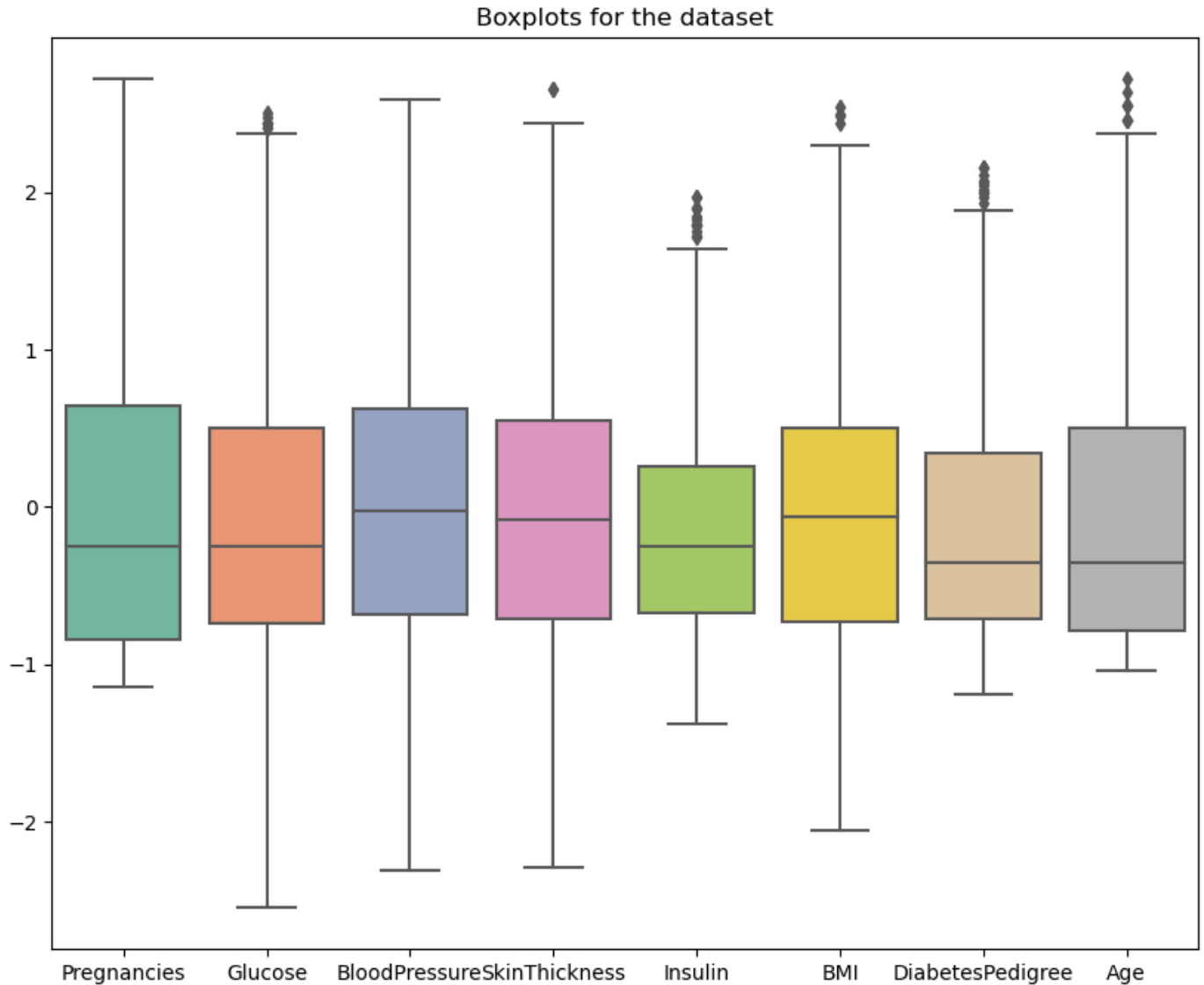
```
In [25]: df.describe()
```

```
Out[25]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000
mean	-0.021163	-0.086996	-0.029981	-0.056533	-0.144136	-0.060981	-0.137816	-0.079311
std	0.966896	0.946568	0.896331	0.938913	0.703973	0.927727	0.750830	0.907411
min	-1.141385	-2.548017	-2.314453	-2.292444	-1.378744	-2.060542	-1.191764	-1.039711
25%	-0.844459	-0.742356	-0.679059	-0.713838	-0.673017	-0.728779	-0.709688	-0.783411
50%	-0.250606	-0.249903	-0.024902	-0.082396	-0.252857	-0.062898	-0.354157	-0.356011

75%	0.640173	0.505191	0.629256	0.549046	0.258910	0.501654	0.338827	0.4985
max	2.718658	2.507833	2.591728	2.653853	1.968213	2.542725	2.161676	2.7205

```
In [26]: #Boxplot after Standardizing the dataset and removing the Outliers
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
plt.figure(figsize=(10, 8))
sns.boxplot(data=df[col],palette='Set2')
plt.title('Boxplots for the dataset')
plt.show()
```



Preprocessing of ToPredict.csv

```
In [27]: test=pd.read_csv('ToPredict.csv')
test.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	5.000000	5.00000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
mean	3.200000	140.00000	75.200000	27.800000	116.800000	34.380000	0.466200	29.200000
std	3.114482	28.62691	9.757049	17.268468	222.128791	6.803822	0.410425	9.093954
min	0.000000	108.00000	62.000000	0.000000	0.000000	26.000000	0.222000	22.000000
25%	1.000000	121.00000	70.000000	24.000000	0.000000	31.200000	0.223000	25.000000

50%	3.000000	136.00000	78.000000	32.000000	0.000000	32.400000	0.261000	26.000000
75%	4.000000	154.00000	78.000000	39.000000	74.000000	39.000000	0.443000	28.000000
max	8.000000	181.00000	88.000000	44.000000	510.000000	43.300000	1.182000	45.000000

In [28]: `test.head(10)`

Out[28]:

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

In [29]:

```
col=['SkinThickness', 'Insulin' ]

# 0 is replaced with NaN
test[col] = test[col].replace(0, np.nan)

#Copy of test with the selected columns
i = test[col].copy()

# KNN Imputer
knn = KNNImputer(n_neighbors=5)
i_arr = knn.fit_transform(i)

# Convert the array to dataframe
imputed = pd.DataFrame(i_arr, columns=col)

# Update test with the new values from dataframe imputed
test.loc[:, col] = imputed

test.describe()
```

Out[29]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	5.000000	5.00000	5.000000	5.00000	5.000000	5.000000	5.000000	5.000000
mean	3.200000	140.00000	75.200000	34.75000	292.000000	34.380000	0.466200	29.200000
std	3.114482	28.62691	9.757049	7.52911	154.149278	6.803822	0.410425	9.093954
min	0.000000	108.00000	62.000000	24.00000	74.000000	26.000000	0.222000	22.000000
25%	1.000000	121.00000	70.000000	32.00000	292.000000	31.200000	0.223000	25.000000
50%	3.000000	136.00000	78.000000	34.75000	292.000000	32.400000	0.261000	26.000000
75%	4.000000	154.00000	78.000000	39.00000	292.000000	39.000000	0.443000	28.000000
max	8.000000	181.00000	88.000000	44.00000	510.000000	43.300000	1.182000	45.000000

In [30]: `test.head()`

Out[30]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	4	136	70	34.75	292.0	31.2	1.182	22
1	1	121	78	39.00	74.0	39.0	0.261	28

2	3	108	62	24.00	292.0	26.0	0.223	25
3	0	181	88	44.00	510.0	43.3	0.222	26
4	8	154	78	32.00	292.0	32.4	0.443	45

```
In [31]: col=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPe
X=test[col]

#StandardScaler
s=StandardScaler()
X_scale = s.fit_transform(X)

# Replace the values in df
test[col] = X_scale
```

```
In [32]: test.head()
```

```
Out[32]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	0.287183	-0.156221	-0.595854	0.000000	0.000000	-0.522552	1.949902	-0.885186
1	-0.789754	-0.742052	0.320844	0.631103	-1.581139	0.759179	-0.558983	-0.147531
2	-0.071796	-1.249771	-1.512553	-1.596319	0.000000	-1.377038	-0.662498	-0.516359
3	-1.148733	1.601269	1.466718	1.373577	1.581139	1.465774	-0.665222	-0.393416
4	1.723100	0.546775	0.320844	-0.408361	0.000000	-0.325362	-0.063199	1.942492

Training the model

```
In [33]: #Splitting data into training and testing set
col=['Glucose', 'Insulin', 'Age', 'BMI', 'DiabetesPedigree', 'SkinThickness', 'BloodPressure',
X=df[col]
Y=df['Outcome']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

#Logistic Regression model
model=LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

print('Accuracy: ' + str(accuracy_score(Y_test, Y_pred)))
print('F-score: ' + str(f1_score(Y_test, Y_pred)))

Accuracy: 0.7121212121212122
F-score: 0.5128205128205129
```

```
In [34]: #Getting the coefficients
c=list(model.coef_[0])
labels=list(X_train.columns)

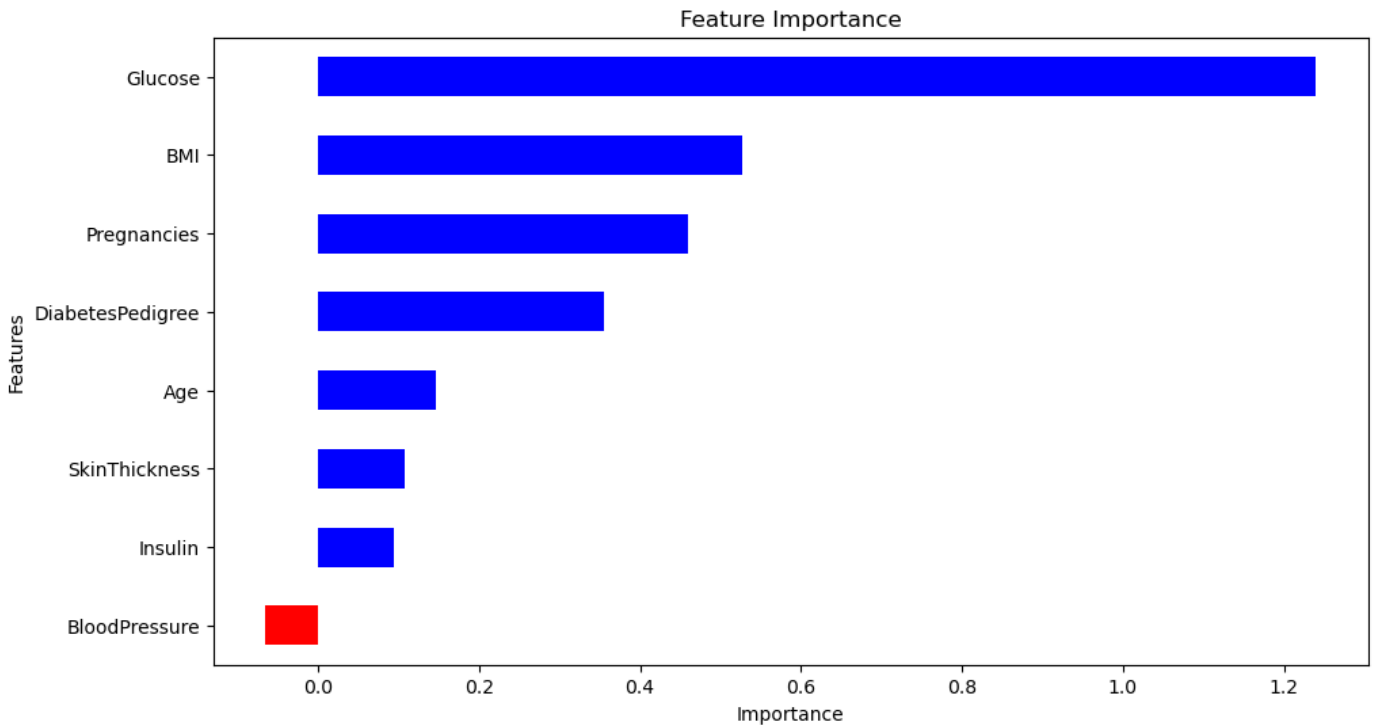
#Creating a dataframe that stores the coefficients
f=pd.DataFrame()
f['Features'] = labels
f['importance'] = c

f.sort_values(by=['importance'], ascending=True, inplace=True)
f['positive'] = f['importance'] > 0
f.set_index('Features', inplace=True)

#Plotting the graph for Feature Importance
```

```
f.importance.plot(kind='barh', figsize=(11, 6), color = f.positive.map({True: 'blue', False: 'red'}))
plt.title('Feature Importance')
plt.xlabel('Importance')
```

Out[34]: Text(0.5, 0, 'Importance')



```
In [35]: #New set of variables after removing Blood Pressure
col=['Glucose','Pregnancies','BMI','DiabetesPedigree','Age','Insulin']
X=df[col]
Y=df['Outcome']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

#Logistic Regression Model
model=LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

print('Accuracy: ' + str(accuracy_score(Y_test, Y_pred)))
print('F-score: ' + str(f1_score(Y_test, Y_pred)))
```

Accuracy: 0.7196969696969697

F-score: 0.5316455696202531

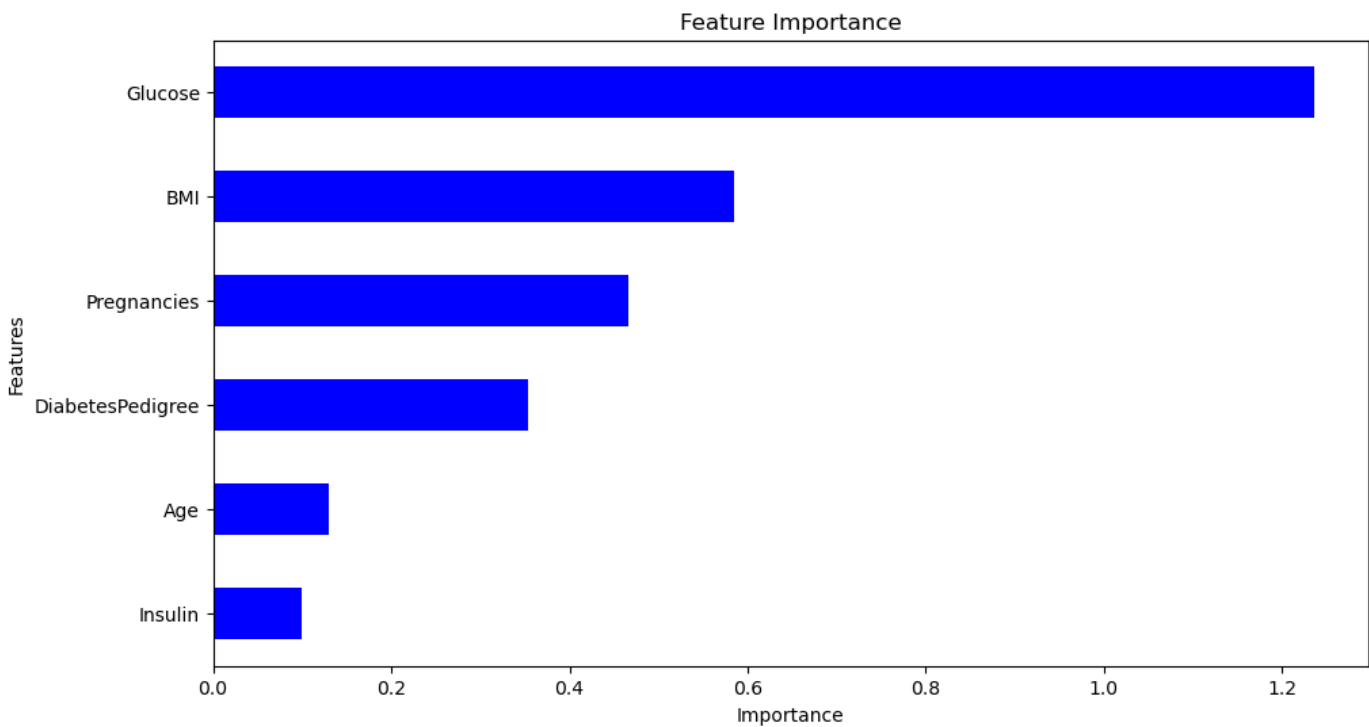
```
In [36]: #Getting the coefficients
c=list(model.coef_[0])
labels=list(X_train.columns)

#Creating a dataframe that stores the coefficients
f=pd.DataFrame()
f['Features'] = labels
f['importance'] = c

f.sort_values(by=['importance'], ascending=True, inplace=True)
f['positive'] = f['importance'] > 0
f.set_index('Features', inplace=True)

#Plotting the graph for Feature Importance
f.importance.plot(kind='barh', figsize=(11, 6), color = f.positive.map({True: 'blue', False: 'red'}))
plt.title('Feature Importance')
plt.xlabel('Importance')
```

Out[36]: Text(0.5, 0, 'Importance')



```
In [37]: #Training the model with the new set of variables
col=['Glucose','Pregnancies','BMI','DiabetesPedigree','SkinThickness','Age','Insulin']
X=df[col]
Y=df['Outcome']

#Training the Logistic regression model
model=LogisticRegression()
model.fit(X,Y)

#Predicting the Outcome based on the model trained
test['Outcome']=model.predict(test[['Glucose','Pregnancies','BMI','DiabetesPedigree','Sk
test.head(10)
```

```
Out[37]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outco
0	0.287183	-0.156221	-0.595854	0.000000	0.000000	-0.522552	1.949902	-0.885186	
1	-0.789754	-0.742052	0.320844	0.631103	-1.581139	0.759179	-0.558983	-0.147531	
2	-0.071796	-1.249771	-1.512553	-1.596319	0.000000	-1.377038	-0.662498	-0.516359	
3	-1.148733	1.601269	1.466718	1.373577	1.581139	1.465774	-0.665222	-0.393416	
4	1.723100	0.546775	0.320844	-0.408361	0.000000	-0.325362	-0.063199	1.942492	

```
In [38]: #Finding the probability with Outcome 1
p=model.predict_proba(test[['Glucose','Pregnancies','BMI','DiabetesPedigree','SkinThickn

prob_diabetes = p.mean()

print("Probability of getting diabetes: " + str(prob_diabetes))

Probability of getting diabetes: 0.38903014688944976
```