

Portfolio for Data Science COM618: Diabetes Data Analysis

Student Name: Matthew Wilcox

Student Number: Q16048563

Github Repository Link: [Data-Science-Assessment Github Link](#)

Introduction

Diabetes is a large, growing problem within the Medical community, with Diabetes being the direct cause of 1.6 million deaths in 2021 ^[^1], with the mortality rates increasing since 2000. Furthermore, 59% of adults (30 and over) were living without medication with Diabetes in 2022. These figures represent a challenge in the medical area with effectively screening for people who have the disease and also a problem in distributing care, leading to these increased mortality rates. This represents that the industry could benefit from some analysis of data on the problem in order to help them make informed decisions quicker to help take preventative measures to stop the disease/ distribute care to people with the disease to stop mortality rates rising further.

[^1]: [WHO Diabetes Fact Sheet](#)

Aims/Objectives of the coursework:

- This Coursework aims to provide an analytic insight into some of the variables that can lead to diabetes. This would be helpful in the Medical Sector as having access to data analysis of this problem could lead to better diagnoses of future patients, which could help set up preventative treatments beforehand to stop strain on medical services, but also allow patients to have access to quality healthcare leading to better quality of life, patient satisfaction and survival rate.
-

Methods

Dataset

This Diabetes Dataset was sourced from Kaggle ^[^2] and contains data from the National Institute of Diabetes and Digestive and Kidney Diseases. This Dataset was used to predict whether a patient had diabetes. This Classes patients in the Outcome variable as either 0 (Non Diabetic) or 1 (Diabetic). This dataset provides insight into many key features (BMI, Glucose, Blood Pressure etc) that can be a large indicator in the cause of diabetes. By leveraging this dataset using different data analysis techniques and data modelling, we can help identify indicators of the disease that can be used for prevention and management of the disease. This dataset has missing values , so will need to be preprocessed to ensure valuable insight can be obtained.

[^2]: [Diabetes Dataset \(Kaggle\)](#)

Analysis and Results

Dataset Preparation

This Dataset had to be prepared for Usage in Data Analysis by cleaning up any null / not included ie included as 0 data points. We first did this by checking for null values by using the `isnull()` function on the DataFrame, which returned 0. Then by checking for 0 values it was found that there are lots of missing values. These were turned to Nan using `diabetes_data_cleaned[columns_to_clean] = diabetes_data_cleaned[columns_to_clean].replace(0, np.nan)`. The proportion of values that were now null in each column of the dataset was then calculated and displayed , giving this output:

```
Proportion of missing values in the cleaned dataset: Id 0.000000
Pregnancies 0.000000
Glucose 0.650289
BloodPressure 4.515896
SkinThickness 28.901734
Insulin 48.049133
BMI 1.408960
DiabetesPedigreeFunction 0.000000
Age 0.000000
Outcome 0.000000
dtype: float64
```

Due to this Output , it was decided that 2 rows: **SkinThickness** and **Insulin** would be dropped. This was because imputing thousands of lines to data to something ie the mean would lead to a big skew in data, making the dataset overall unfit for purpose of data analysis as any output would be massive different to real life. **Id** was also dropped as it serves no analytical value for us.

The other Columns with missing data: **Glucose**, **BloodPressure** and **BMI** had the missing columns filled in as they had a lower proportion of missing values, therefore not effecting the data as much. This was done using the Mean(Average) values using the code below, with the example showing how it was done on the **BMI** column:

```
imputer = SimpleImputer(strategy='mean')
diabetes_data_cleaned[ 'BMI' ] =
imputer.fit_transform(diabetes_data_cleaned[[ 'BMI' ]])
```

This was done with the averages as this gives the biggest representation of the overall trend of the column from the data, which will help the replaced values being as accurate as possible.

```

1  # get skewness of the dataset
2  print(f"\n Skewness of the dataset: \n", diabetes_data_cleaned.skew())
3
4  # get kurtosis of the dataset
5  print(f"\n Kurtosis of the dataset: \n", diabetes_data_cleaned.kurtosis())
6
7  # handle skewness
8  # get columns with skewness greater than 1 (high skewness)
9  skewness = diabetes_data_cleaned.skew()
10 high_skewness = skewness[skewness > 1].index
11 print(f"\n Columns with high skewness: \n", high_skewness)
12
13 # handle kurtosis
14 # get columns with kurtosis greater than 3 (high kurtosis)
15 kurtosis = diabetes_data_cleaned.kurtosis()
16 high_kurtosis = kurtosis[kurtosis > 3].index
17 print(f"\n Columns with high kurtosis: \n", high_kurtosis)

```

```

Skewness of the dataset:
Pregnancies      0.959096
Glucose           0.518154
BloodPressure     0.192850
BMI               0.852292
DiabetesPedigreeFunction  1.842791
Age               1.166299
Outcome           0.657465
dtype: float64

Kurtosis of the dataset:
Pregnancies      0.333585
Glucose          -0.321336
BloodPressure     1.084821
BMI               2.406312
DiabetesPedigreeFunction  5.172935
Age               0.771859
Outcome          -1.568874
dtype: float64

Columns with high skewness:
Index(['DiabetesPedigreeFunction', 'Age'], dtype='object')

Columns with high kurtosis:
Index(['DiabetesPedigreeFunction'], dtype='object')

```

These Screenshots represent the checking of Skewness and Kurtosis values in the Dataset. These represent that **DiabetesPedigreeFunction**, **BMI(Outside 3 value of Kurtosis but included as it is high)** and **Age** have high skewness or kurtosis that could suggest outliers that could be handled in later steps.

Overall, this step (Data Preprocessing) has helped the data be more fit for its purpose as it has made the dataset more trimmed and has removed any null/unrepresentative(0) values that may skew any findings. This Step could have Been further improved on by using the Skewness and Kurtosis values in order to identify Outliers and handle them appropriately.

Exploratory Data Analysis/ Data Visualisation

The EDA of this Dataset aimed to see if we could get any valuable insights from the data.

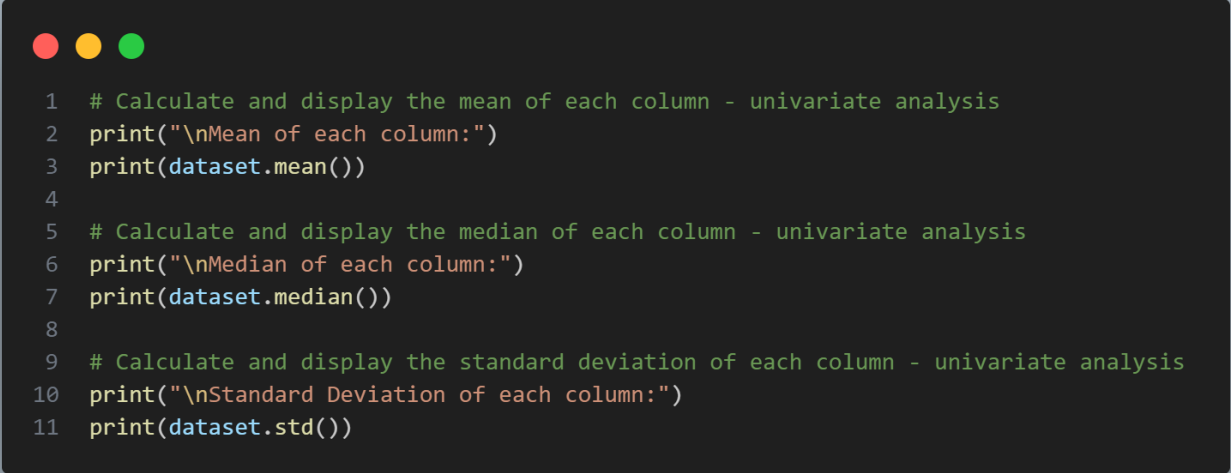
Independent Variables:

Glucose, BMI, Blood Pressure, Age, Pregnancies, DiabetesPedigreeFunction

Dependent:

Outcome

Univariate Analysis:



```
1 # Calculate and display the mean of each column - univariate analysis
2 print("\nMean of each column:")
3 print(dataset.mean())
4
5 # Calculate and display the median of each column - univariate analysis
6 print("\nMedian of each column:")
7 print(dataset.median())
8
9 # Calculate and display the standard deviation of each column - univariate analysis
10 print("\nStandard Deviation of each column:")
11 print(dataset.std())
```

```
Mean of each column:
Pregnancies      3.742775
Glucose          121.895273
BloodPressure    72.404086
BMI              32.596665
DiabetesPedigreeFunction  0.471193
Age              33.132225
Outcome          0.343931
dtype: float64

Median of each column:
Pregnancies      3.000
Glucose          118.000
BloodPressure    72.000
BMI              32.400
DiabetesPedigreeFunction  0.375
Age              29.000
Outcome          0.000
dtype: float64

Standard Deviation of each column:
Pregnancies      3.323801
Glucose          30.500960
BloodPressure    11.988255
BMI              7.103424
DiabetesPedigreeFunction  0.325669
Age              11.777230
Outcome          0.475104
dtype: float64
```

Summary of the dataset:				
	Pregnancies	Glucose	BloodPressure	BMI \
count	2768.000000	2768.000000	2768.000000	2768.000000
mean	3.742775	121.895273	72.404086	32.596665
std	3.323801	30.500960	11.988255	7.103424
min	0.000000	44.000000	24.000000	18.200000
25%	1.000000	99.000000	64.000000	27.575000
50%	3.000000	118.000000	72.000000	32.400000
75%	6.000000	141.000000	80.000000	36.625000
max	17.000000	199.000000	122.000000	80.600000
	DiabetesPedigreeFunction	Age	Outcome	
count	2768.000000	2768.000000	2768.000000	
mean	0.471193	33.132225	0.343931	
std	0.325669	11.777230	0.475104	
min	0.078000	21.000000	0.000000	
25%	0.244000	24.000000	0.000000	
50%	0.375000	29.000000	0.000000	
75%	0.624000	40.000000	1.000000	
max	2.420000	81.000000	1.000000	

These Snapshots represent analysis of the Mean, Median and Standard deviation of each variable, as well as a Dataset summary, where the min and max was used to calculate the range.

Some standouts from this data:

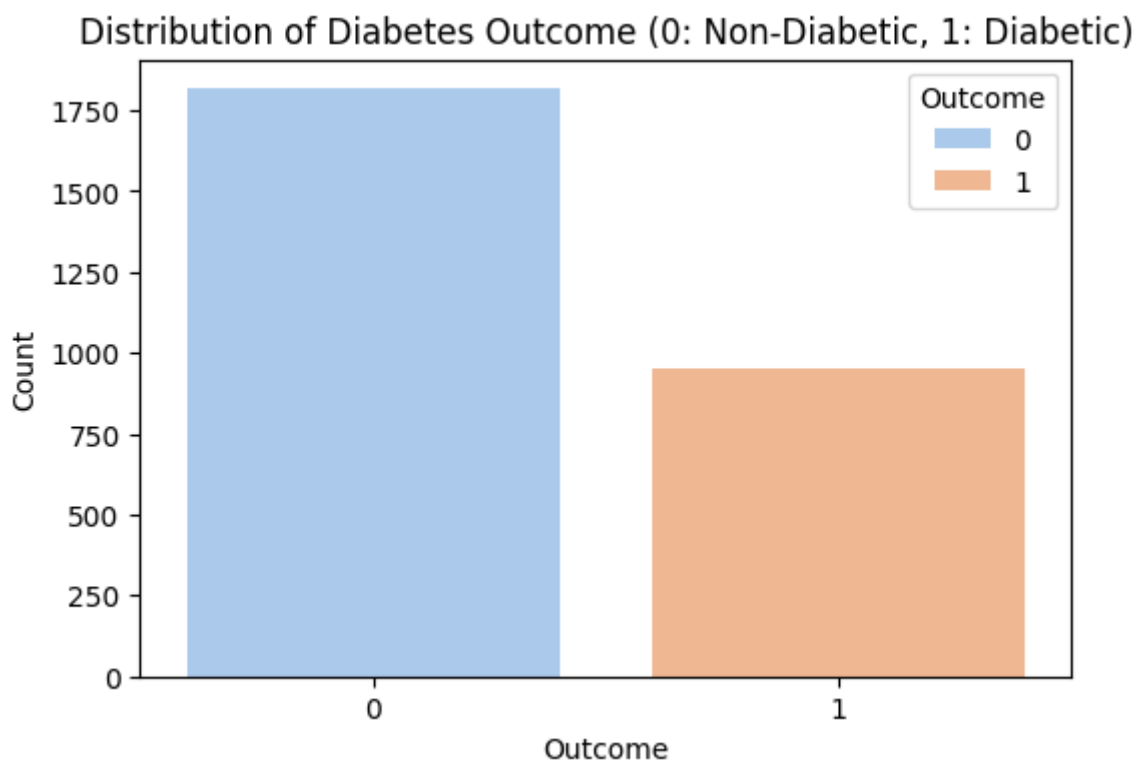
Glucose: The Wide range(44-199) and Significant Standard Deviation suggest extreme cases of Hypoglycemia(low) and hyperglycemia(high), which is a condition that can affect our outcome variable (diabetes or not).

BMI : a mean of 32.6 places the average patient in the obese category.

Pregnancies/Age : Both The range of both columns and the standard deviation of pregnancies (3,32) indicates high variation in age.



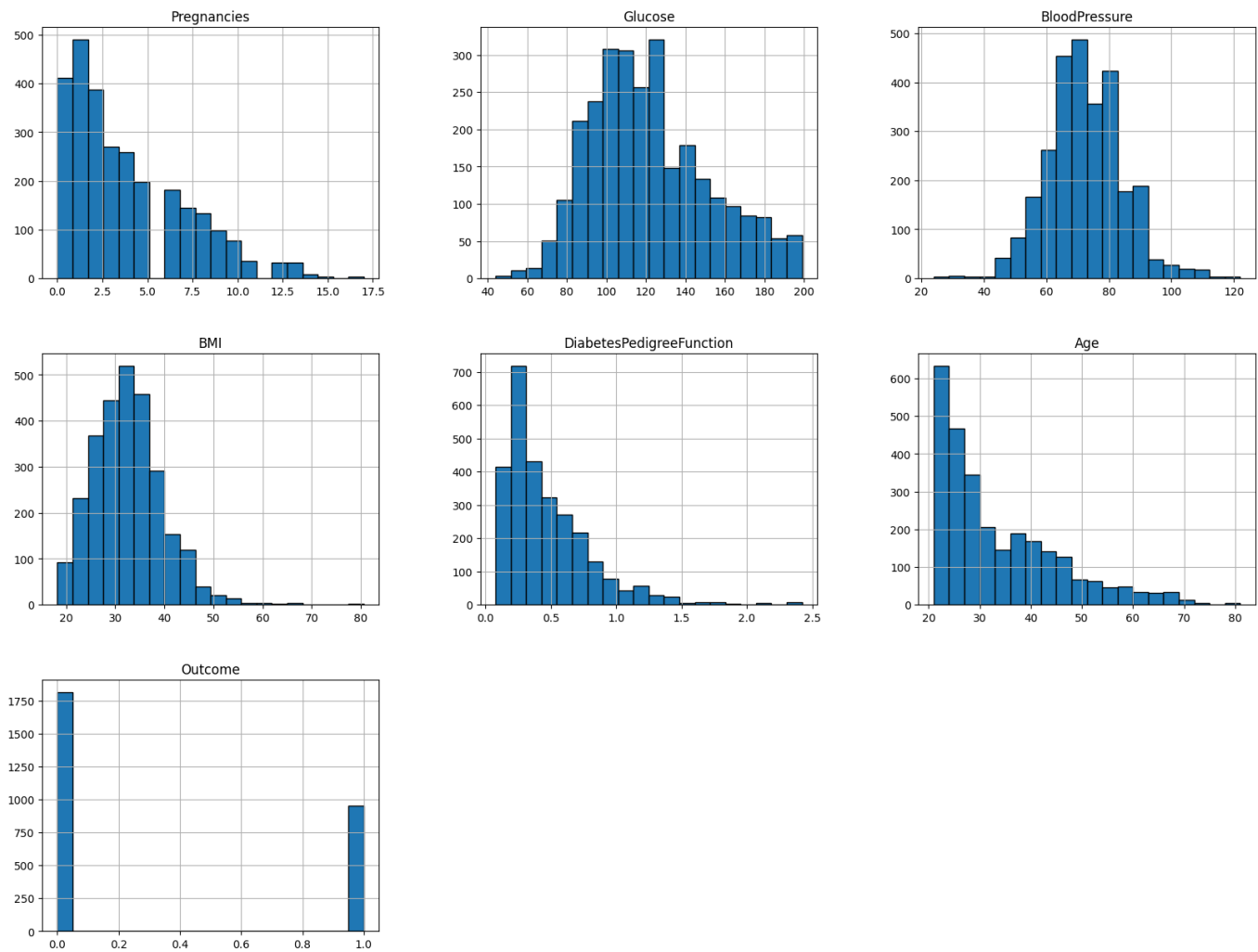
```
1 # Outcome distribution - count of each class
2 plt.figure(figsize=(6, 4))
3 sns.countplot(x='Outcome', data=dataset, hue='Outcome', palette='pastel')
4 plt.title('Distribution of Diabetes Outcome (0: Non-Diabetic, 1: Diabetic)')
5 plt.xlabel('Outcome')
6 plt.ylabel('Count')
7 plt.show()
```



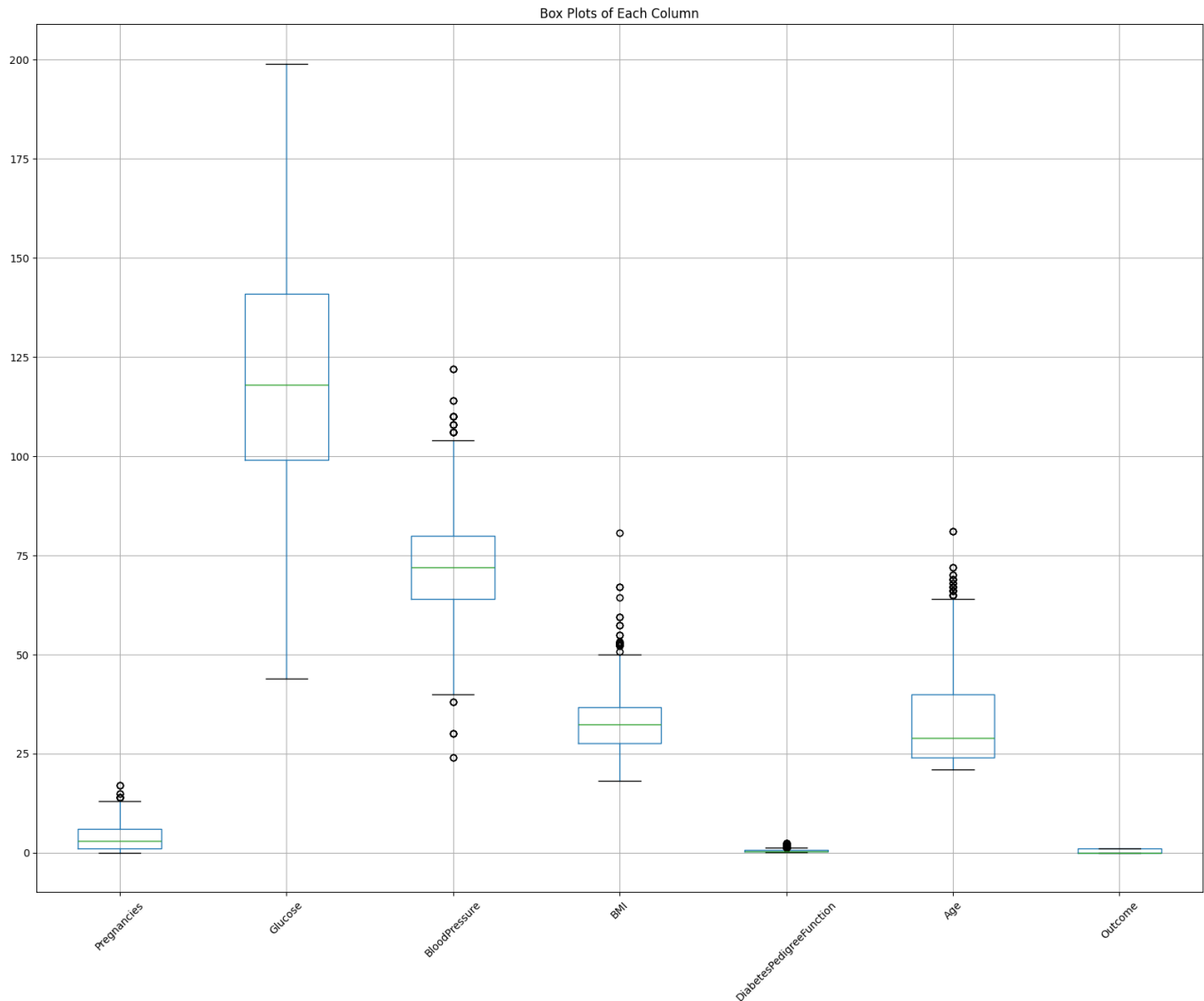
These Screenshots represent the Outcome distribution of the Dataset, which suggests 65% of the outcome variable is 0(Non-Diabetic), which shows a dataset imbalance. This could effect later classification models,

which could lead to us needing to do resampling/undersampling or use more complex models like Random Forest.

Histograms of Each Column



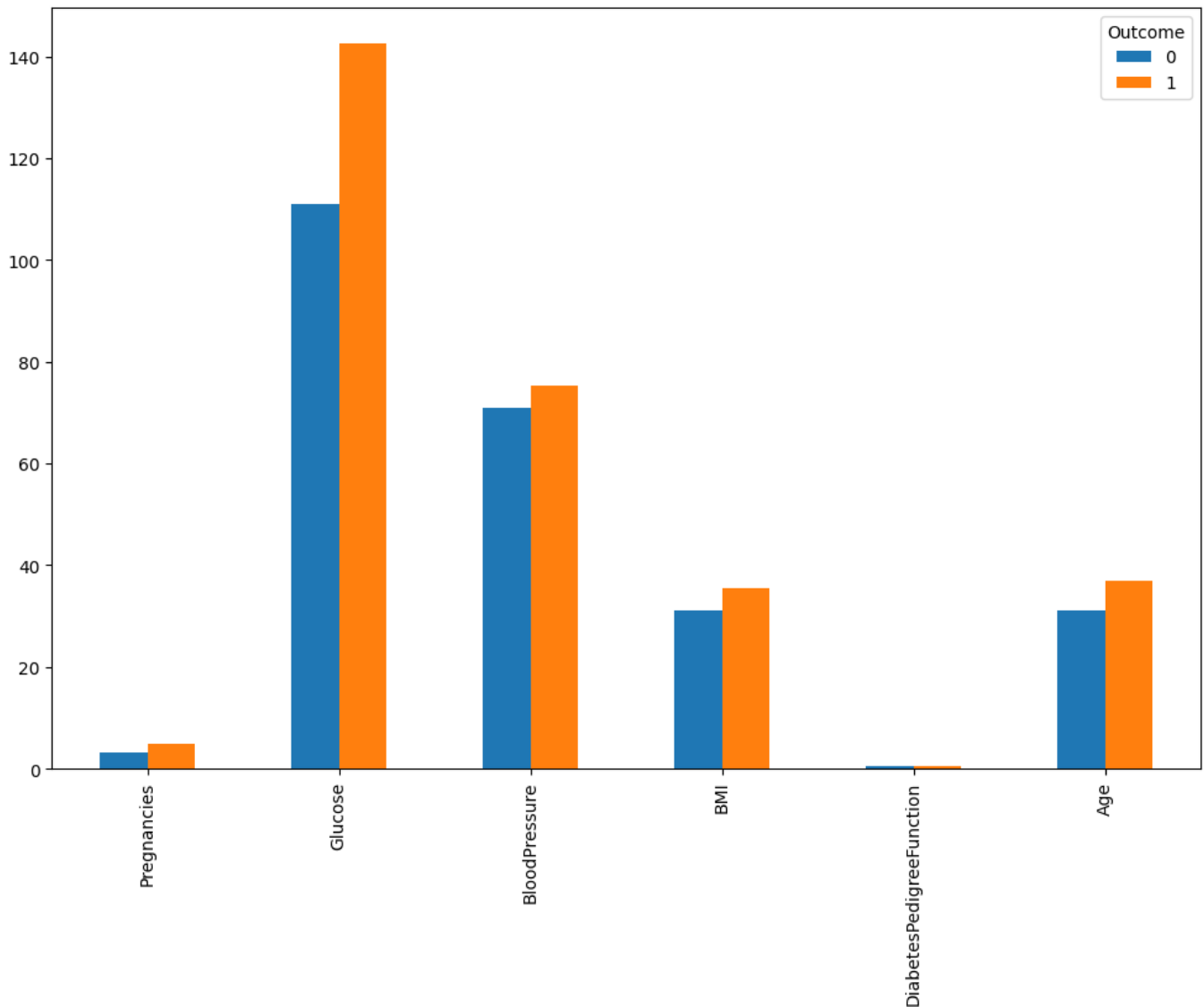
These Histograms represent the Skewness of the Variables and help us understand what values are most present.



These Box Plots show that **BloodPressure**, **BMI** and **Age** show a number of outliers (circle outside box plot).

Bivariate/Mutivariate Analysis:

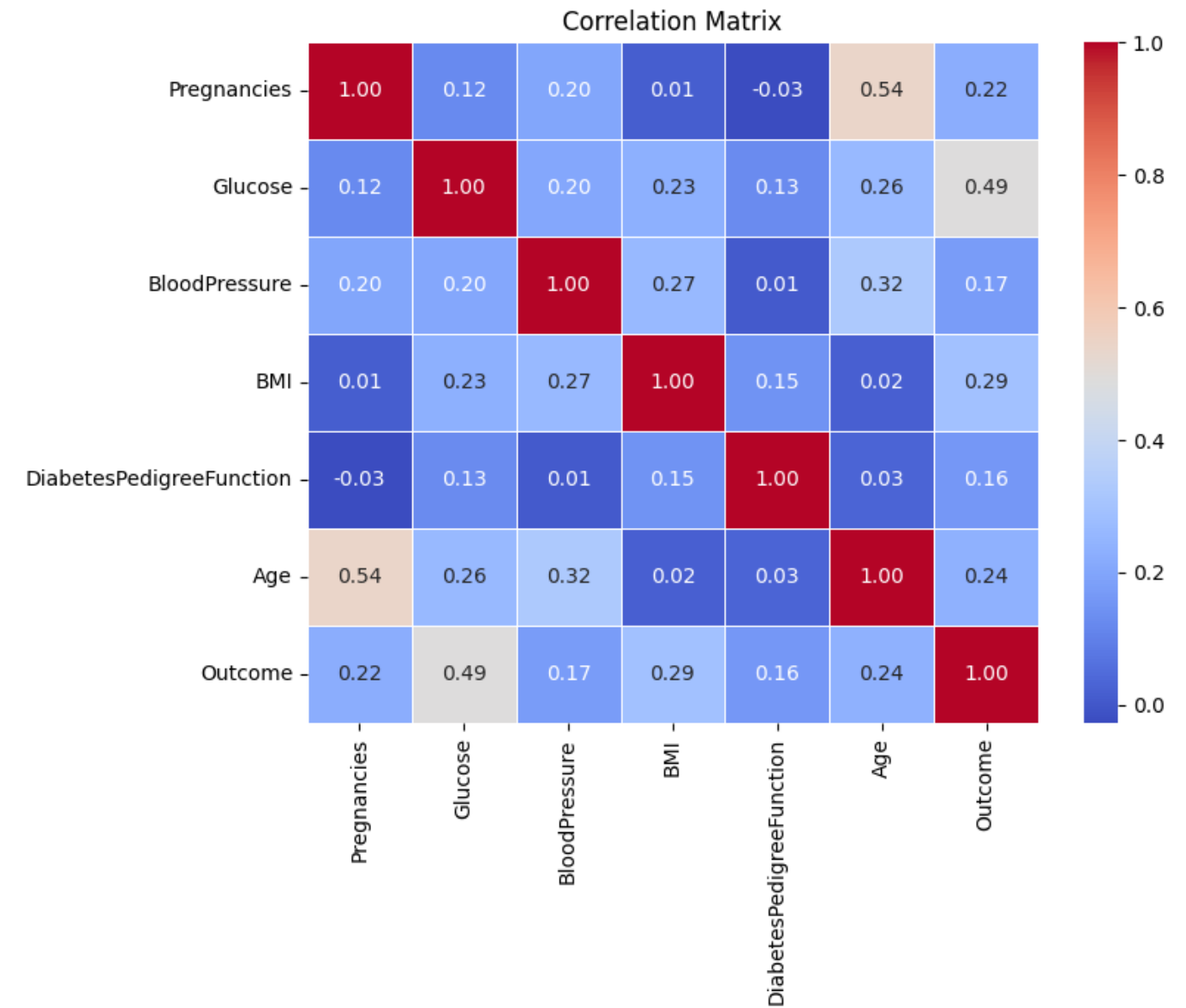
```
1 # Grouped averages by Outcome - example of bivariate analysis
2 grouped_means = dataset.groupby('Outcome').mean()
3 print(grouped_means)
4
5 #Plot the grouped means - example of bivariate analysis
6 plt.figure(figsize=(12, 8))
7 # transpose the dataframe to plot the means of each feature by Outcome
8
9
10 grouped_means.T.plot(kind='bar', figsize=(12, 8))
```

This represents how each variable may effect the outcome, with the graph showing that on average higher glucose, blood pressure, BMI and Age contribute to the 1/Diabetes Outcome.

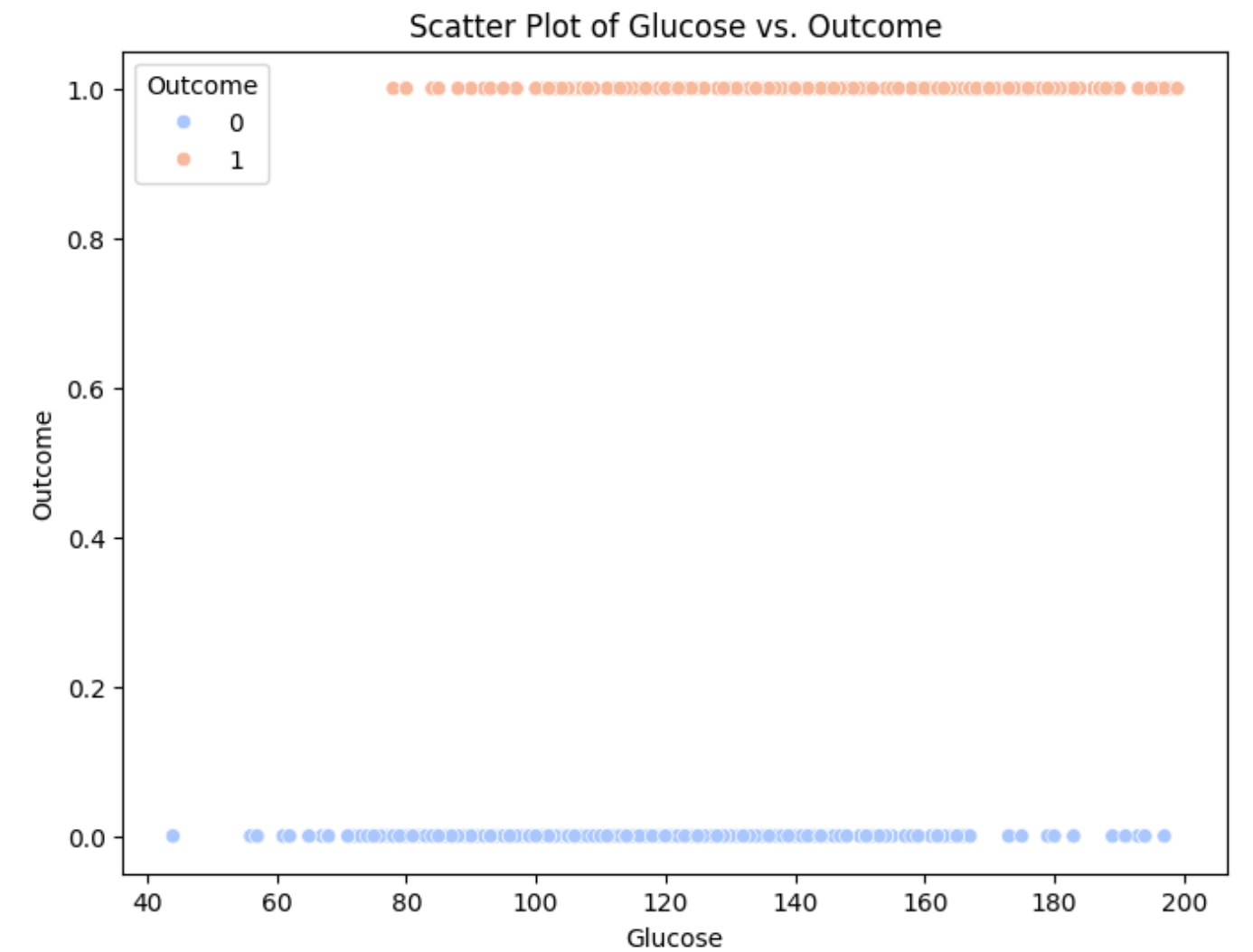


```
1 # Correlation matrix - example of multivariate analysis
2 plt.figure(figsize=(8, 6))
3 correlation_matrix = dataset.corr()
4 # fmt='.2f' to display values in 2 decimal places to give us more insight into the data
5 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
6 plt.title('Correlation Matrix')
7 plt.show()
```



This is the Correlation Matrix for the Dataset, which shows how much the different variables effect the others. From this, we can see that Glucose has a large impact on Outcome, following other Diabetes findings. Also, Variables such as BMI , Pregnancies Age have a higher correlation with the causation of Diabetes, with DPF and BloodPressure having the lowest impact. We can also see that BMI and BloodPressure and BMI and Glucose have a fair amount of correlation, which again follows Trends.

```
1 # Scatter plot of Glucose vs. Outcome - example of bivariate analysis
2 plt.figure(figsize=(8, 6))
3 sns.scatterplot(x='Glucose', y='Outcome', data=dataset, hue='Outcome', palette='coolwarm')
4 plt.title('Scatter Plot of Glucose vs. Outcome')
5 plt.xlabel('Glucose')
6 plt.ylabel('Outcome')
7 plt.show()
```



This scatterplot shows an example of a scatter plot done for analysis, which shows us that a higher amount of glucose at the higher end of the scale has a 1(diabetic) outcome.

Data Modelling:

Modelling - Predict Dependent variable (Outcome) based on Dataset

The aim of the Model Use of this data was to analyse if the dataset was fit for purpose of classifying patients, as this will help to earlier catch disease symptoms/ trends and either implement measures to stop it becoming Diabetes, or give them the appropriate Medical care to ensure that the death rate does not rise as highly as it has in recent years. I have used 2 Models to analyse this , Logistic Regression and Random Forest. These have been used due to the Data being Categorical in nature, meaning other Machine learning models may not be fit for purpose.

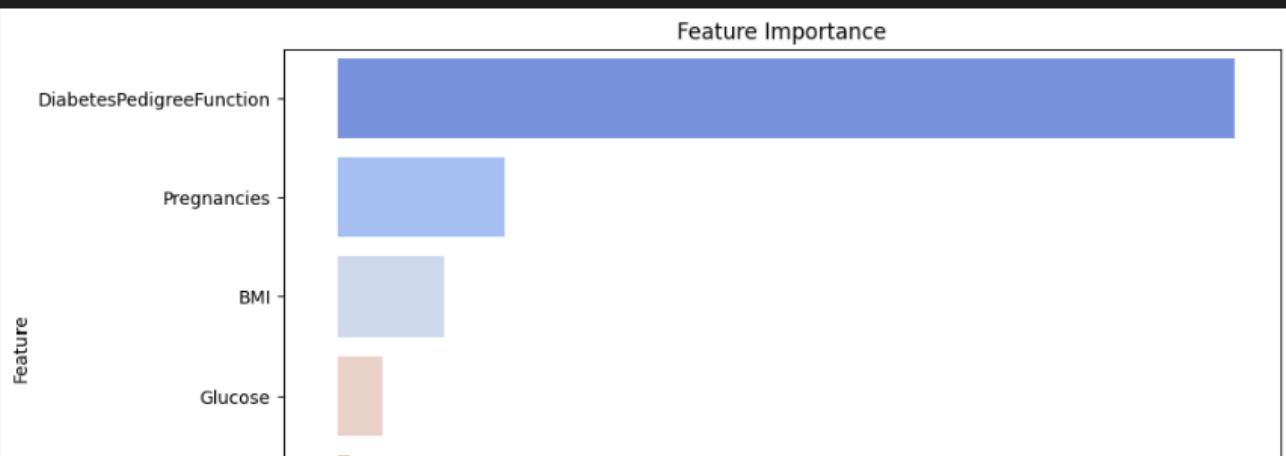
Logistic Regression:

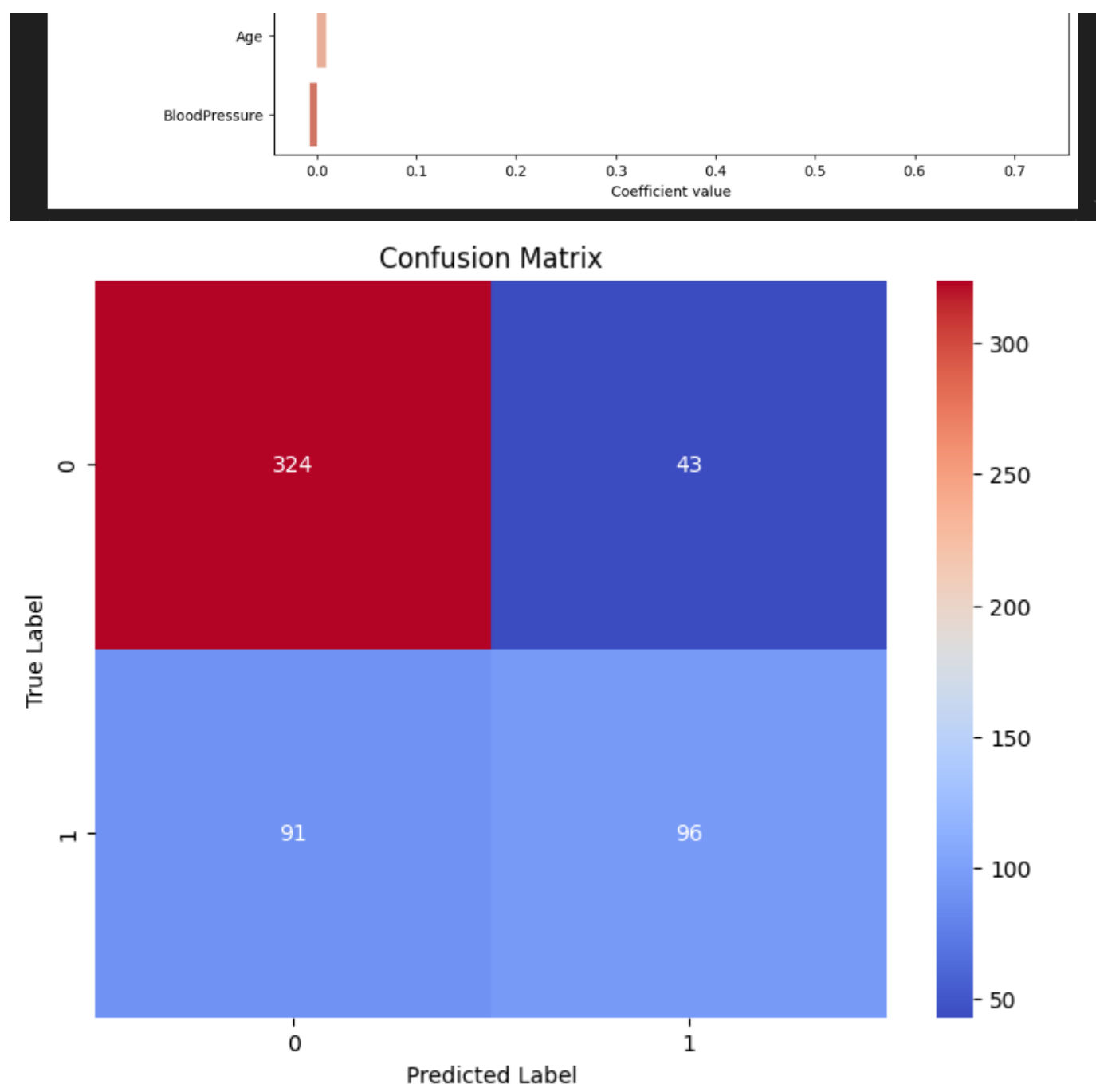

```

1  from sklearn.model_selection import train_test_split
2  from sklearn.linear_model import LogisticRegression
3  from sklearn.metrics import classification_report, confusion_matrix
4  from sklearn.metrics import accuracy_score
5
6  X = dataset[['Glucose', 'BMI', 'BloodPressure', 'Age', 'Pregnancies', 'DiabetesPedigreeFunction']]
7  y = dataset['Outcome']
8
9
10 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
11
12
13 log_reg = LogisticRegression()
14 log_reg.fit(X_train, y_train)
15
16 # Make predictions
17 y_pred = log_reg.predict(X_test)
18
19 # Evaluate the model
20 print(confusion_matrix(y_test, y_pred))
21 print(classification_report(y_test, y_pred))
22
23 #Accuracy Score
24 accuracy = accuracy_score(y_test, y_pred)
25 print(f'Accuracy Score: {accuracy:.2f}')
26
27 #get coefficients
28 coefficients = log_reg.coef_[0]
29
30
31 feature_importance = pd.DataFrame({
32     'Feature': X_train.columns,
33     'Coefficient': coefficients
34 })
35 # Calculate the absolute value of the coefficients for ranking
36 feature_importance['Abs_Coefficient'] = feature_importance['Coefficient'].abs()
37
38 # Sort the features by the absolute value of the coefficient
39 feature_importance = feature_importance.sort_values('Abs_Coefficient', ascending=False)
40
41 # plot the feature importance
42 plt.figure(figsize=(10, 6))
43 sns.barplot(x='Coefficient', y='Feature', data=feature_importance, palette='coolwarm', hue='Feature')
44 plt.title('Feature Importance')
45 plt.xlabel('Coefficient value')
46 plt.ylabel('Feature')
47 plt.show()

```

Accuracy Score: 0.76





This Logistic Regression Model was created using the sklearn library . We first seperated the independent and variables, then split the dataset into the test and training sets, then fit our model with the training data and predicted the test data based on that. The model was then evaluated as: Accuracy: 0.76 / 76%

[[324 43]				
[91 96]]				
	precision	recall	f1-score	support
0	0.78	0.88	0.83	367
1	0.69	0.51	0.59	187
accuracy			0.76	554
macro avg	0.74	0.70	0.71	554
weighted avg	0.75	0.76	0.75	554

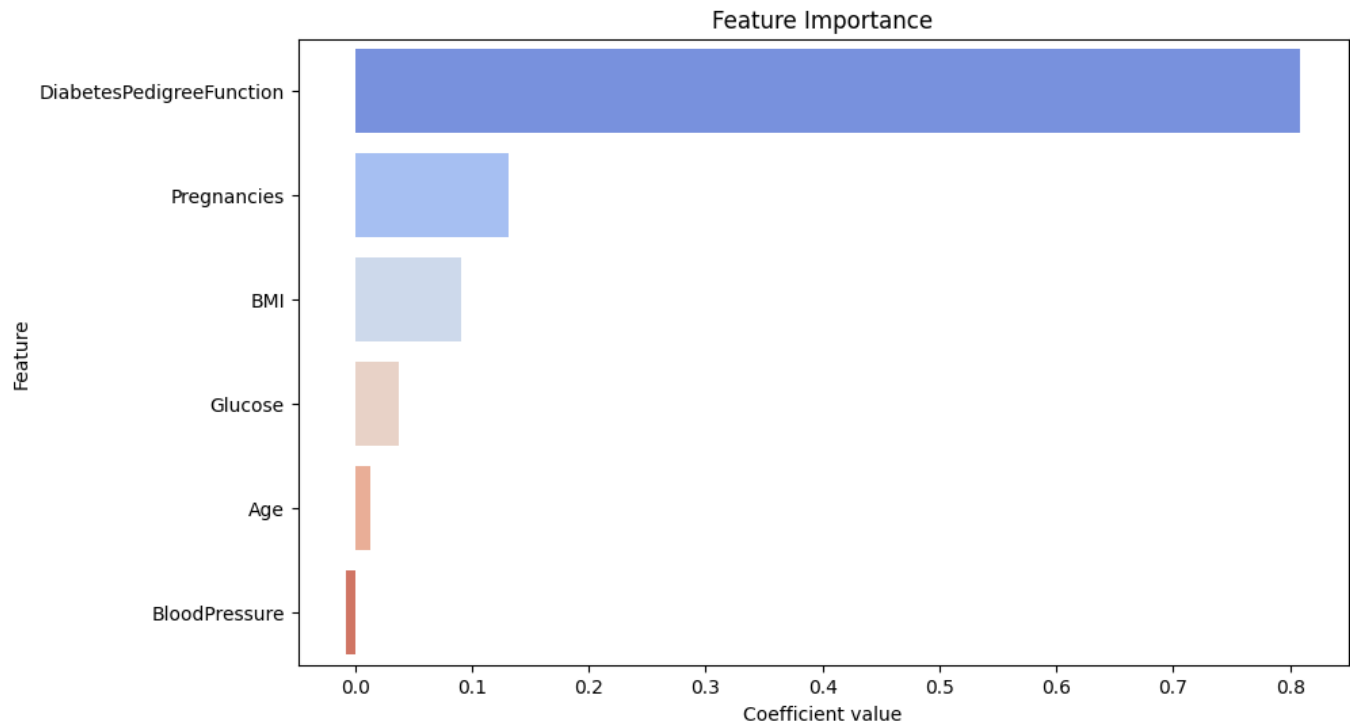
The model Misclassified 43 false positives and 91 false negatives, showing the model is less accurate on the 1 outcome due to less support/ training data (187 to 367). This is also shown from the much lower precision, recall and f1-score. When

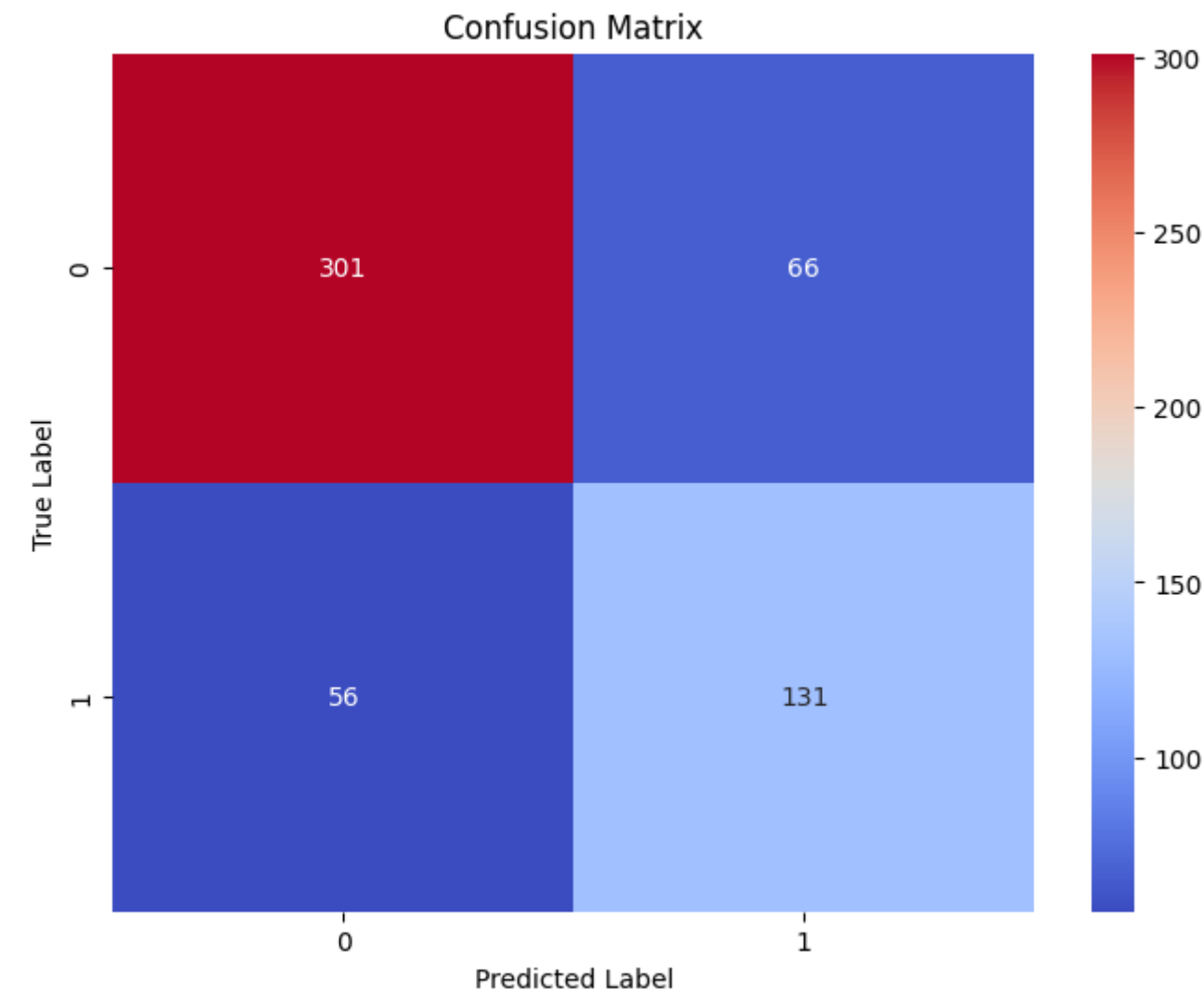
attempting to address this with `log_reg=LogisticRegression(class_weight='balanced')` , this improved the model as such:

```
[[301  66]
 [ 56 131]]
```

	precision	recall	f1-score	support
0	0.84	0.82	0.83	367
1	0.66	0.70	0.68	187
accuracy			0.78	554
macro avg	0.75	0.76	0.76	554
weighted avg	0.78	0.78	0.78	554

Accuracy Score: 0.78





These show that while it did improve performance by 2% and the performance on the 1 outcomes recall and f1-score, the 0 Outcome ended up with more misclassified rows (false-positives) and decreased the recall.

While this performance is ok, we want to improve it to make it as useful as possible, due to this we will try RandomForest, which is a classifier model that due to its ensemble learning^[3] will deal better with the imbalanced dataset.

^[3]: [Logistic regression vs random forest](#)

Random Forest Classifier

For this Model, we have used the RandomForestClassifier from sklearn


```

1  from sklearn.ensemble import RandomForestClassifier
2
3  X = dataset[['Glucose', 'BMI', 'BloodPressure', 'Age', 'Pregnancies', 'DiabetesPedigreeFunction']]
4  y = dataset['Outcome']
5  #print(X)  debugging
6
7  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
8
9  rf = RandomForestClassifier(random_state=42)
10 rf.fit(X_train, y_train)
11
12 y_pred_rf = rf.predict(X_test)
13
14 # Evaluate the model
15 print(confusion_matrix(y_test, y_pred_rf))
16 print(classification_report(y_test, y_pred_rf))
17
18 accuracy_rf = accuracy_score(y_test, y_pred_rf)
19 print(f'Random Forest Accuracy Score: {accuracy_rf:.2f}')
20
21 importances = rf.feature_importances_
22 features = X_train.columns
23
24 #print(importances) debugging
25
26 # create data frame to pair feature names with their importance (This was made after a plotting error)
27 feature_importances = pd.DataFrame(
28     {'Feature': features,
29      'Importance': importances
30     }
31 )
32 print(feature_importances) #debugging
33
34 # feature importances
35 plt.figure(figsize=(8, 6))
36 sns.barplot(x='Importance', y='Feature', data=feature_importances, palette='viridis', hue=features)
37 plt.title('Feature Importance')
38 plt.xlabel('Importance')
39 plt.ylabel('Feature')
40 plt.show()
41
42
43 # Confusion matrix heatmap
44 sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Blues', cbar=False)
45 plt.title('Confusion Matrix')
46 plt.xlabel('Predicted Label')
47 plt.ylabel('True Label')
48 plt.show()

```

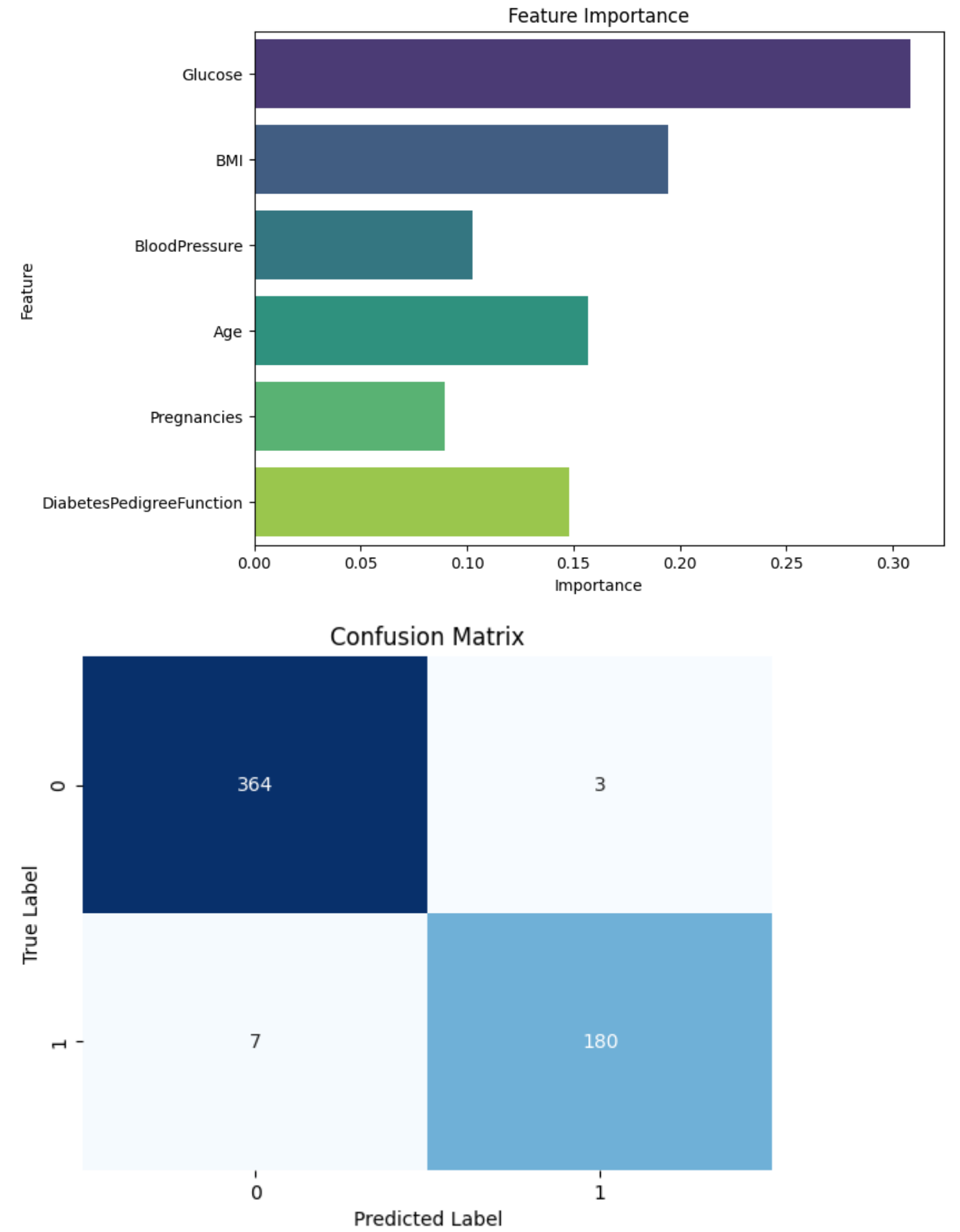
```

[[364   3]
 [  7 180]]

```

	precision	recall	f1-score	support
0	0.98	0.99	0.99	367
1	0.98	0.96	0.97	187
accuracy			0.98	554
macro avg	0.98	0.98	0.98	554
weighted avg	0.98	0.98	0.98	554

Random Forest Accuracy Score: 0.98



This Model performs much better then the Logistic regression model, with only 3 false positives and 7 false negatives and an accuracy of 0.98, also having both outcomes having over 95% recall and f1 score. The feature importance graph shows that pregnancies has the lowest impact, while glucose has the highest on the outcome variable.

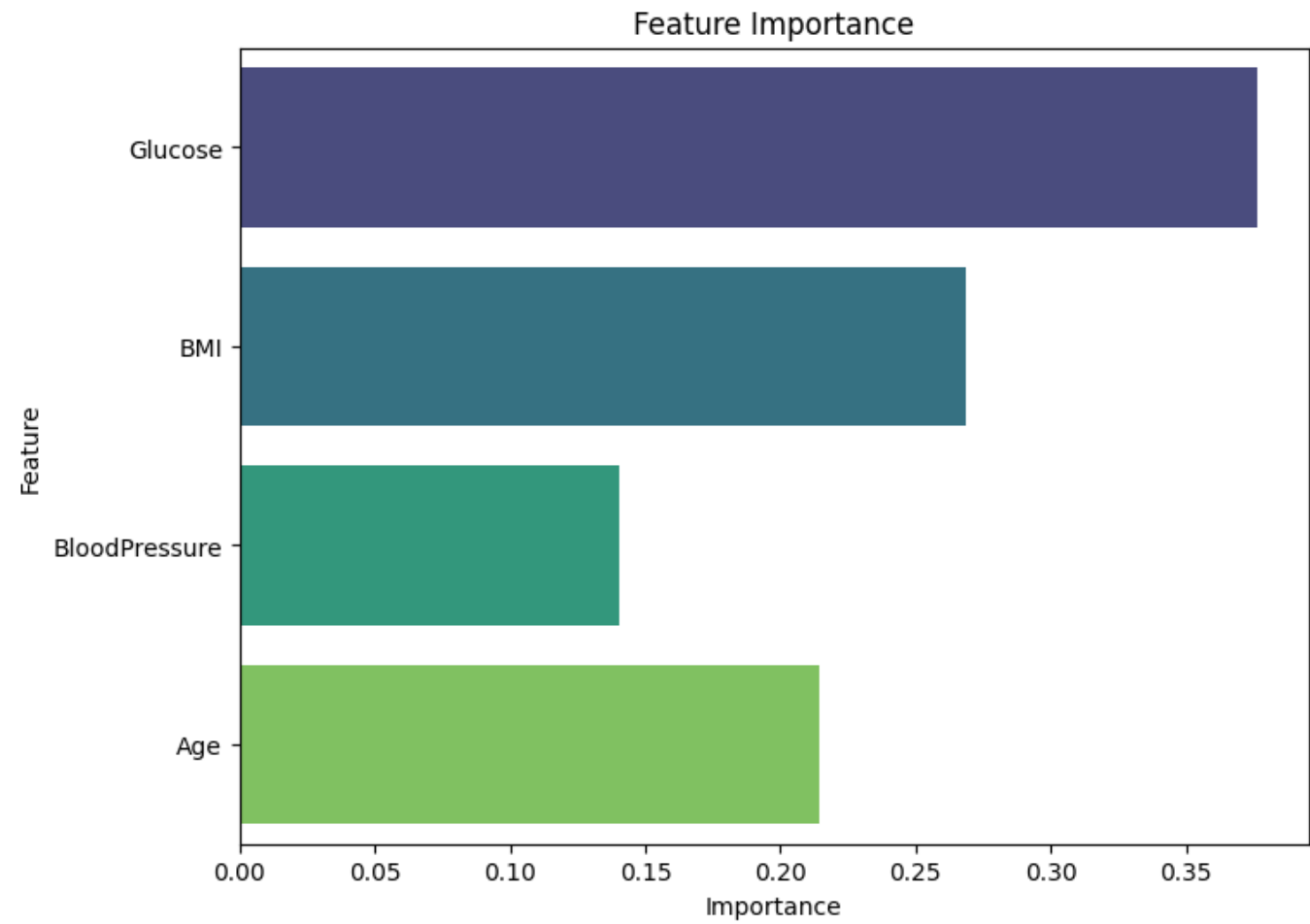
Over a few iterations of the model, we go the highest performance by removing pregnancies and DPF. We removed them due to Pregnancies low feature importance and DPF low score in the Correlation matrix from earlier. This led to the following scores.

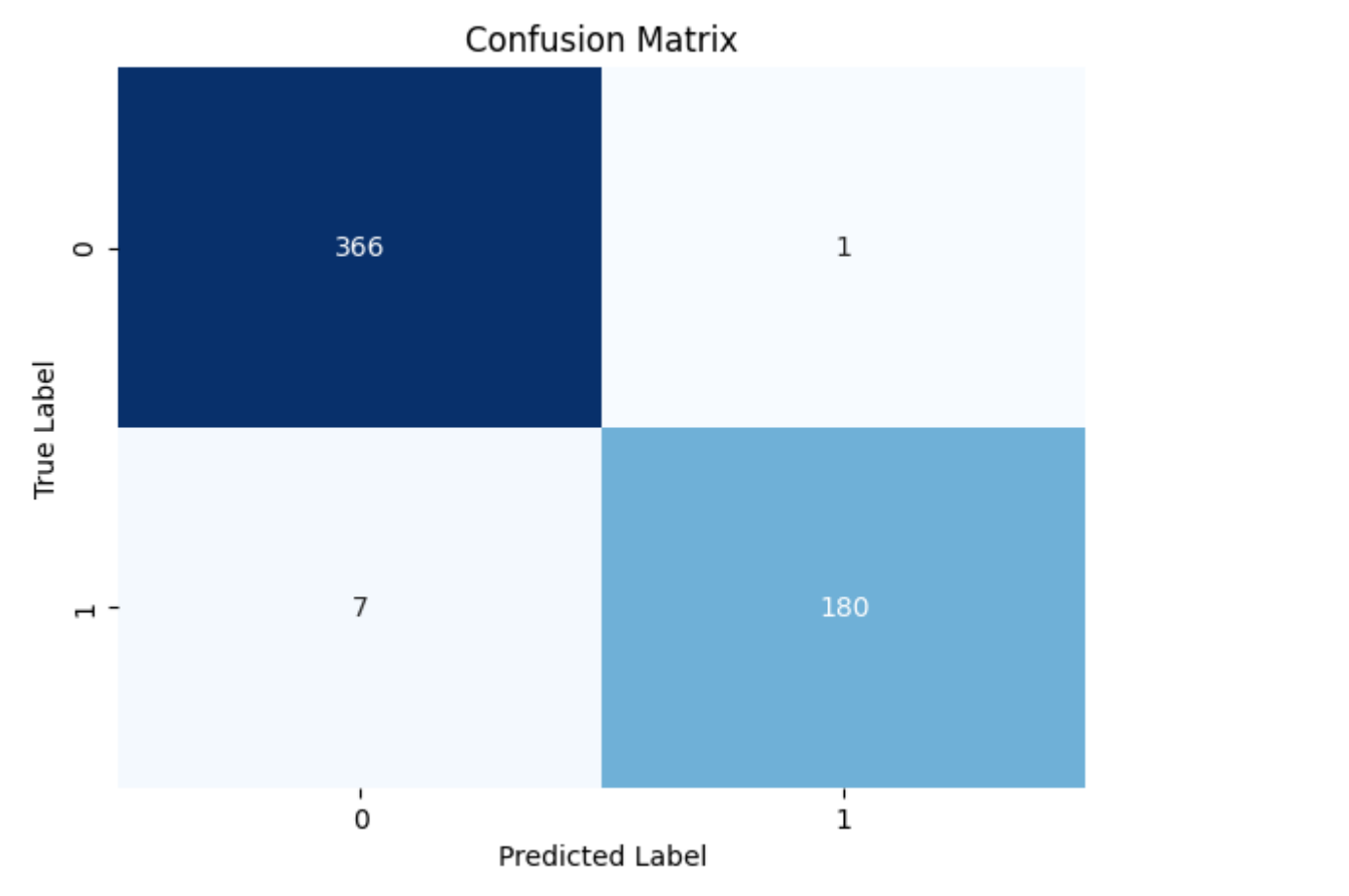
```
1  from sklearn.ensemble import RandomForestClassifier
2
3  X = dataset[['Glucose', 'BMI', 'BloodPressure', 'Age']]
4  y = dataset['Outcome']
5  #print(X)  debugging
6
7  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
8
9  rf = RandomForestClassifier(random_state=42)
10 rf.fit(X_train, y_train)
11
12 y_pred_rf = rf.predict(X_test)
13
14 # Evaluate the model
15 print(confusion_matrix(y_test, y_pred_rf))
16 print(classification_report(y_test, y_pred_rf))
17
18 accuracy_rf = accuracy_score(y_test, y_pred_rf)
19 print(f'Random Forest Accuracy Score: {accuracy_rf:.2f}')
20
21 importances = rf.feature_importances_
22 features = X_train.columns
23
24 #print(importances) debugging
25
26 # create data frame to pair feature names with their importance (This was made after a plotting error)
27 feature_importances = pd.DataFrame(
28     {'Feature': features,
29      'Importance': importances
30     }
31 )
32 #print(feature_importances) debugging
33
34 # feature importances
35 plt.figure(figsize=(8, 6))
36 sns.barplot(x='Importance', y='Feature', data=feature_importances, palette='viridis', hue=features)
37 plt.title('Feature Importance')
38 plt.xlabel('Importance')
39 plt.ylabel('Feature')
40 plt.show()
41
42
43 # Confusion matrix heatmap
44 sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Blues', cbar=False)
45 plt.title('Confusion Matrix')
46 plt.xlabel('Predicted Label')
47 plt.ylabel('True Label')
48 plt.show()
```

```
[[366  1]
 [ 7 180]]
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	367
1	0.99	0.96	0.98	187
accuracy			0.99	554
macro avg	0.99	0.98	0.98	554
weighted avg	0.99	0.99	0.99	554

Random Forest Accuracy Score: 0.99





Evaluation

These results for the final model show that the model is very good at evaluating our data based on 4 variables, glucose, bmi blood pressure and age. These would be good indicators for BMI in the medical field, so this could be a useful model for future research. This model gained 99% accuracy with only 8 rows misclassified and had high scores in precision, recall and f-1 score across both outcomes. The feature importance of glucose being so high tracks with real life diabetes studies and also our EDA from earlier, showing the model is classifying correctly. Overall, this model has been very successful at classifying patients.

Limitations and Challenges

Some of the limitations with this dataset include: Missing values - having missing values in any dataset isn't ideal as it creates a lack of real life, accurate data. We got around this in our data Preprocessing/ Preparation Phase, however 2 variables were dropped from the table during this and other variables having missing values imputed from the average (mean). Due to this, we may lack understanding of how those other variables contribute, which can affect healthcare and also the imputed values may skew any models trained as they are not true organic values gained from study. Misbalanced data - This data has more Values for Outcome 0 (1816) then outcome 1 (952). This has effected the analytical power of Models such as LogisticRegression as it struggled to identify the 1 outcome as well as 0 as it had less training rows.

Conclusion

The analysis of this data identified Glucose, BMI , Blood Pressure and Age as significant indicators of diabetes, aligning with medical research. Through preprocessing the original dataset, EDA and machine learning

algorithms , using feature selections i created an effective model with Random Forest Classifier, which had 99% accuracy of classification. This project showed the importance of quality data obtained through preprocessing visualisation in order to gain early insights into the data. Limitations such as imbalanced outcome amounts and missing values highlighted a challenge in healthcare datasets which may prevent some from gaining meaningful insights. Reflecting on this work , I gained experience in handling data , preprocessing it to increase usability, visualise data to analyse trends and creation of classifications models. Future work could be done on : Feature imbalance, Outlier detection and removal and further analysis of data, which would overall contribute to better tools for diagnoses, prevention and management of Diabetes.

Word Count (Excluding references): 2076

Reference List (Harvard Style)

[1] World Health Organization (2021) Diabetes Fact Sheet. Available at: <https://www.who.int/news-room/fact-sheets/detail/diabetes> (Accessed: 27 December 2024). [2] Mehmet Akturk (Mathchi) (n.d.) Diabetes Data Set. Available at: <https://www.kaggle.com/datasets/mathchi/diabetes-data-set> (Accessed: 27 December 2024). [3] GeeksforGeeks, 2024. Logistic Regression vs Random Forest Classifier. [online] Available at: <https://www.geeksforgeeks.org/logistic-regression-vs-random-forest-classifier/> [Accessed 30 December 2024]. [4] Pandas Documentation, 2024. pandas.DataFrame.T. [online] Available at: <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.T.html> [Accessed 30 December 2024]. [5] Tryolabs, 2017. Pandas & Seaborn: A guide to handle & visualize data elegantly. [online] Available at: <https://tryolabs.com/blog/2017/03/16/pandas-seaborn-a-guide-to-handle-visualize-data-elegantly> [Accessed 30 December 2024]. [6] Statology, 2024. sklearn regression coefficients. [online] Available at: <https://www.statology.org/sklearn-regression-coefficients/> [Accessed 30 December 2024].