Diabetes Prediction using Supervised Learning Techniques

By
Oluyomi Alabi

Introduction

This project applied supervised learning techniques to a real-world "Diabetes" dataset from the National Institute of Diabetes and Digestive and Kidney Diseases,
and use data visualization tools to communicate the insights gained from the analysis.
The objective of the dataset is to diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset.

Project Goals

The ultimate goal of the project is to gain insights from the data sets and communicate these insights to stakeholders using appropriate visualizations and metrics to make informed decisions based on the business questions asked.

Process

A — Importing the data

Kaggle_Diabetes data set

B — Exploratory Data Analysis

Visualization to understand data

C — Preprocessing & Feature Engineering

Handling missing values,outliers, scaling etc

D — Training ML model

Evaluation metrics (accuracy, precision, recall, F1-score, and ROC-AUC)

E — Model Tuning

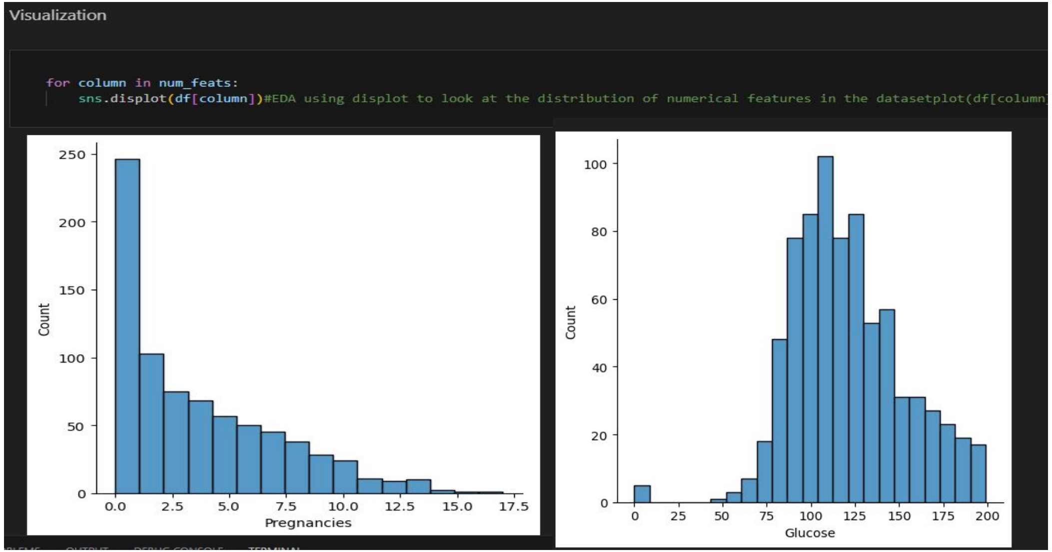
Hyperparameter Tuning & Cross validation

EDA QUESTIONS ANSWERED.

Are there any missing values in the dataset?

- How are the predictor variables related to the outcome variable?
- What is the correlation between the predictor variables?
- What is the distribution of each predictor variable?
- Are there any outliers in the predictor variables?
- How are the predictor variables related to each other?
- Is there any interaction effect between the predictor variables?
- What is the average age of the individuals in the dataset?
- What is the average glucose level for individuals with diabetes and without diabetes?
- What is the average BMI for individuals with diabetes and without diabetes?
- How does the distribution of the predictor variables differ for individuals with diabetes and without diabetes?
- Are there any differences in the predictor variables between males and females (if gender information is available)?

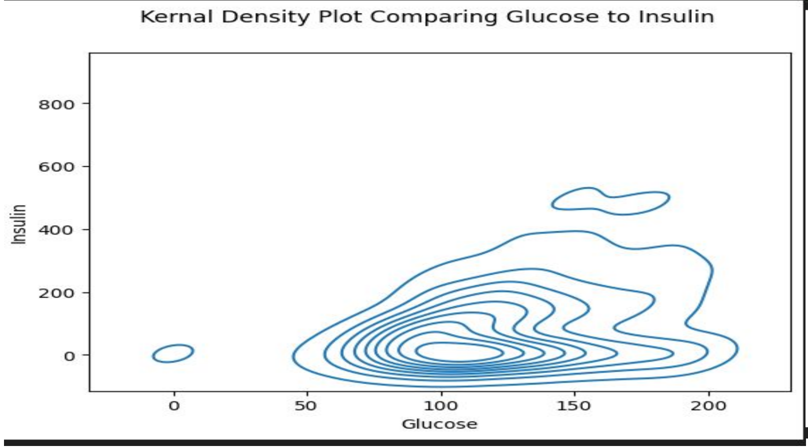
Exploratory Data Analysis Visualization



Displot showing distribution of pregnancy and Glucose variables..

Visualizations showing relationship between Glucose level and Insulin.

```
import matplotlib.pyplot as plt
sns.kdeplot(data=df,
            x='Glucose',
            y='Insulin')
plt.suptitle("Kernal Density Plot Comparing Glucose to Insulin")
plt.show()
```

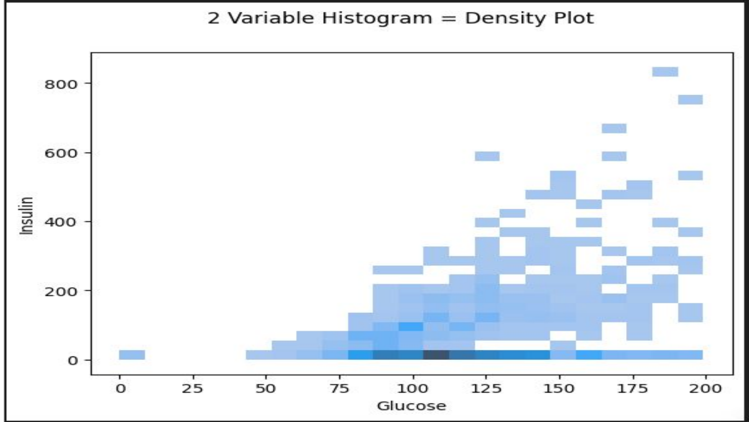



```
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(data=df,
             x="Glucose",
             y="Insulin")

plt.suptitle("2 Variable Histogram = Density Plot")

plt.show()
```

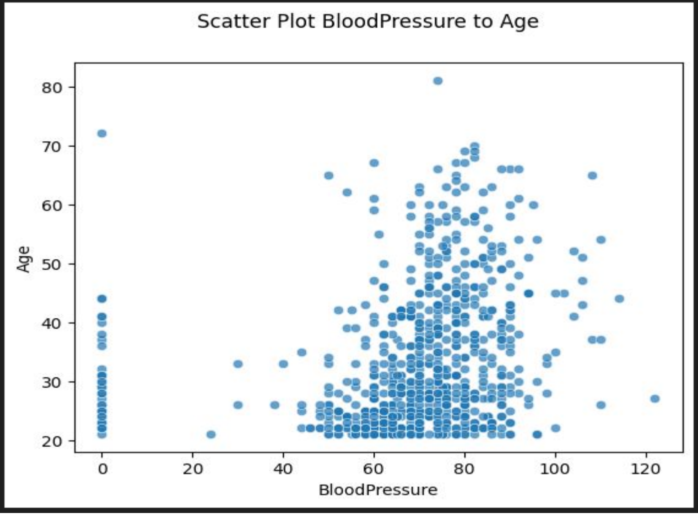
✓ 0.3s



Kernel Density plot   and Histogram Density plot comparing Glucose to insulin showing outliers.

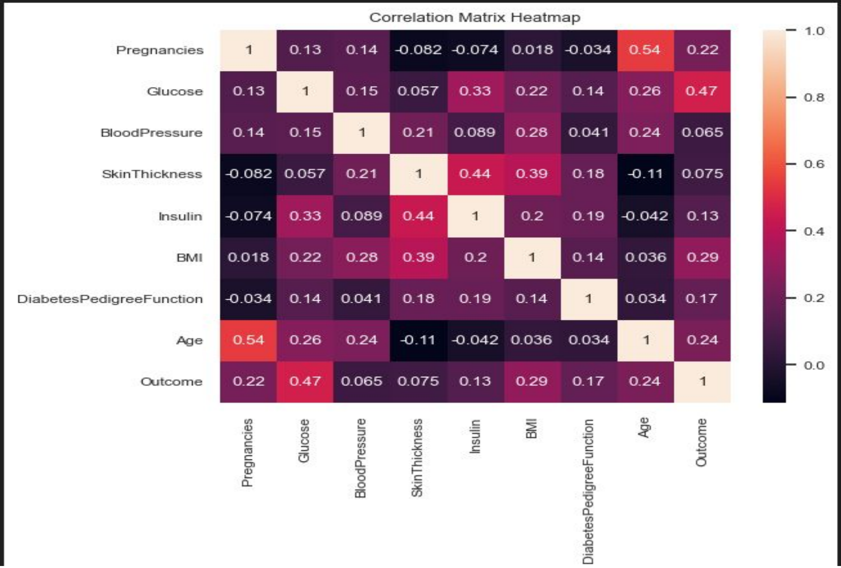
Scattered plot and Correlation matrix map showing correlation between different variables.

```
sns.scatterplot(data=df,
                x="BloodPressure",
                y="Age",
                alpha = 0.7)
plt.suptitle("Scatter Plot BloodPressure to Age")
plt.show()
```




```
sns.heatmap(data= corr, annot=True)
plt.title("Correlation Matrix Heatmap")
```

```
Text(0.5, 1.0, 'Correlation Matrix Heatmap')
```



Predictor variables Average grouped by Outcome showing individuals with or without Diabetes.

```
import pandas as pd
# define the columns to be grouped and calculated
columns_to_groupby = ['Outcome']
columns_to_calculate = ['SkinThickness', 'Insulin', 'Glucose', 'BloodPressure', 'BMI', 'DiabetesPedigreeFunction',  'Age']

# Calculate the average (mean) values for the specified columns for each group
Predictorvariables_average_results = df.groupby(columns_to_groupby)[columns_to_calculate].mean().reset_index()

print(Predictorvariables_average_results)
```

```
   Outcome  SkinThickness     Insulin     Glucose  BloodPressure        BMI  \
0        0      19.664000   68.792000  109.980000      68.184000  30.304200
1        1      22.164179  100.335821  141.257463      70.824627  35.142537

   DiabetesPedigreeFunction        Age
0                  0.429734  31.190000
1                  0.550500  37.067164
```

From this column calculation grouping by Outcome,Average glucose level of individuals without diabetes is 109.98 while the average glucose level with diabetes is 141.257463.
Average BMI for individuals without Diabetes is 30.30while Average BMI of individuals with Diabetes is 35.14

Distribution statistics showing statistical significant correlation between the feature variables

```
#Distribution statistics showing if there is a statistical significant correlation
# Calculate the correlation matrix
corr = df.corr()
corr
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

Summary statistics of each variable showing anomalies and potential outliers

```
#The summary statistics of each of the variables, we can  identify anomalies and potential outliers.
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
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	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

E.g BloodPressure cannot be zero except the patient is dead, same with Skin Thickness, BMI where mininum value is zero

This summary statistics shows the average age of the individuals in the dataset as
33 years,Overall Average glucose level as 120.89, Average BMI as 31.99.

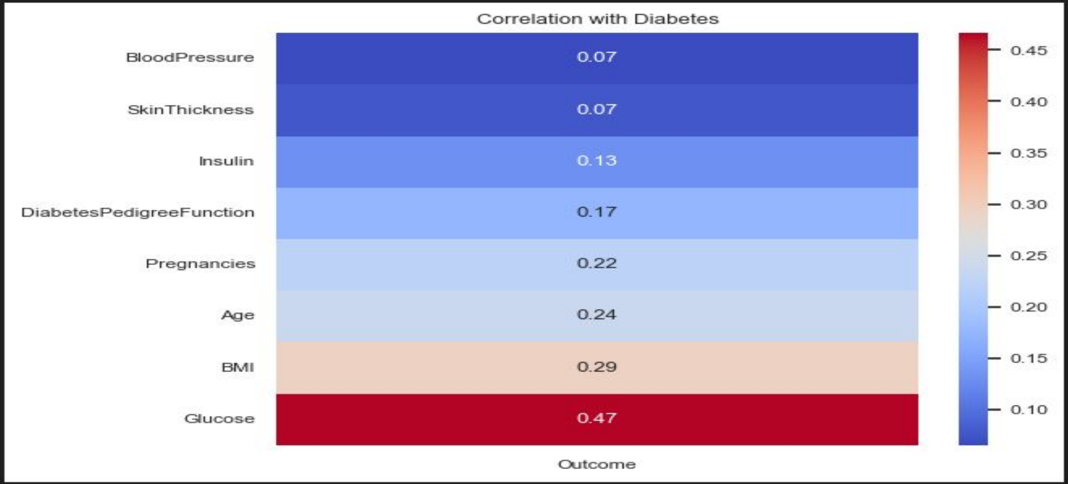
This statistics also show some anomalies eg Blood Pressure of a living being cannot be zero, same with Skin Thickness, BMI,Glucose level. This indicates outliers.

Various feature variable correlation indicating Diabetes in target variable(Outcome).

```
corr = df.corr()
target_corr = corr['Outcome'].drop('Outcome')

target_corr_sorted = target_corr.sort_values(ascending=True)

sns.set(font_scale=0.8)
sns.set_style("white")
sns.set_palette("PuBuGn_d")
sns.heatmap(target_corr_sorted.to_frame(), cmap="coolwarm", annot=True, fmt='.2f')
plt.title('Correlation with Diabetes')
plt.show()
```



With a correlation of 0.47, Glucose is the most strongly correlated feature with the outcome. This suggests that higher glucose levels could be a significant indicator of diabetes.

BMI can also be an important factor in having diabetes.

Results showing no missing values in the dataset.

```
# check missing values in variables

df.isnull().sum()
```

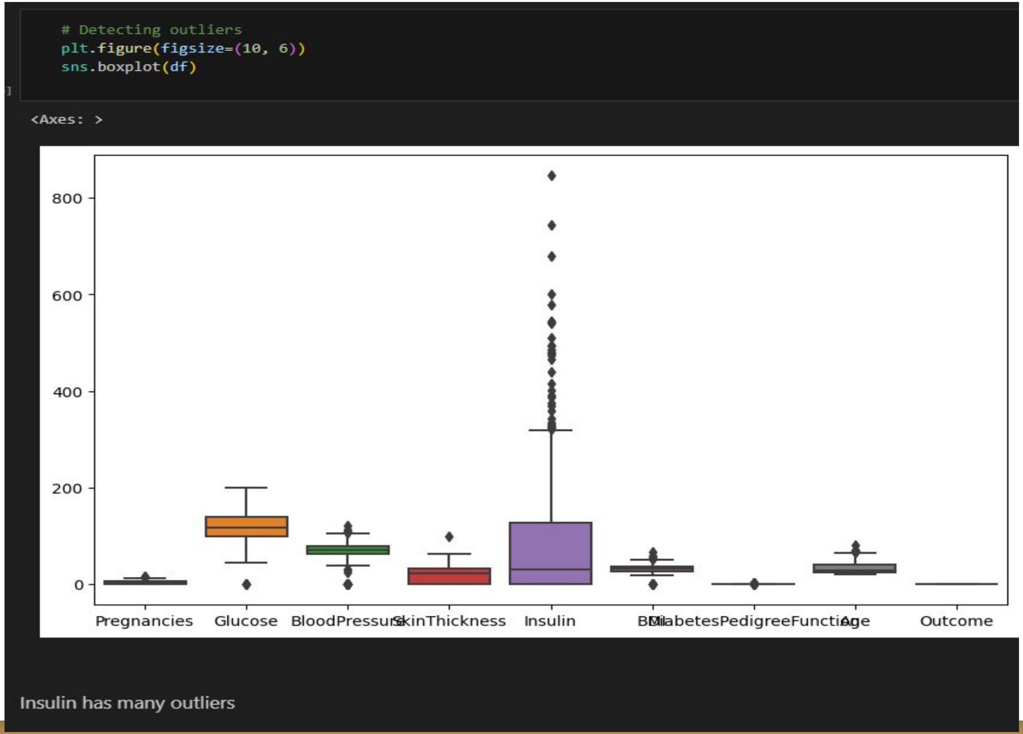
✓ 0.0s

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

Answer: There are no missing values in the dataset.

There are no missing values in the dataset

Handling outliers in preprocessing and feature engineering



It can bee seen that Insulin has many outliers more than other feature variables, this was handled by transforming using log transformation.

```
cols_with_outliers = ['Glucose','Insulin',
'SkinThickness','BloodPressure', 'BMI',
'Age']
df[cols_with_outliers] =
df[cols_with_outliers].apply(lambda x:
np.log1p(x))
```

Logistic regression model showing evaluation metrics results

Accuracy

```python
# import accuracy_score from sklearn
from sklearn.metrics import accuracy_score

# compute accuracy
accuracy = accuracy_score(y_test,y_pred)

# print accuracy
print(accuracy)
```

```
0.7445887445887446
```

F1-Score

```python
# import f1_score from sklearn
from sklearn.metrics import f1_score

# compute F1-score
f1_score = f1_score(y_test,y_pred)

# print F1-score
print(f1_score)
```

```
0.6143790849673203
```

Recall

```python
from sklearn.metrics import recall_score

recall = recall_score(y_test, y_pred)
print("Recall:", recall)
```

```
Recall: 0.5875
```

Precision_score

```python
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred)
print("Precision:", precision)
```

```
Precision: 0.6438356164383562
```

AUC_score

```python
# import roc_auc_score from sklearn
from sklearn.metrics import roc_auc_score

roc_auc = roc_auc_score(y_test, y_pred)

print("ROC-AUC:", roc_auc)
```

```
ROC-AUC: 0.7076572847682119
```

Confusion_matrix

```python
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[125  26]
 [ 33  47]]
```

Random Forest Classifier model with model Evaluation metrics Results

```python
# instantiate the classifier
rfc = RandomForestClassifier(random_state=0)

# fit the model
rfc.fit(X_train, y_train)

# Predict the Test set results
y_pred = rfc.predict(X_test)

# Check accuracy score
from sklearn.metrics import accuracy_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("ROC-AUC:", roc_auc)
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_rep)
```

```
Model accuracy score with 10 decision-trees : 0.7446
Precision: 0.6296296296296297
Recall: 0.6375
F1-score: 0.6335403726708074
ROC-AUC: 0.719412251655629

Confusion Matrix:
[[121  30]
 [ 29  51]]

Classification Report:
```

```
Model accuracy score with 10 decision-trees : 0.7446
Precision: 0.6296296296296297
Recall: 0.6375
F1-score: 0.6335403726708074
ROC-AUC: 0.719412251655629

Confusion Matrix:
[[121  30]
 [ 29  51]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.80      0.80       151
           1       0.63      0.64      0.63        80

    accuracy                           0.74       231
   macro avg       0.72      0.72      0.72       231
weighted avg       0.75      0.74      0.74       231
```

Result explanation

Based on these metrics, we can see that both models have a similar accuracy score (0.7446), but the Random Forest Classifier has a slightly higher ROC-AUC score (0.7194) compared to the Logistic Regression model (0.7077). Additionally, the Random Forest Classifier has slightly higher precision and recall values.

Overall, the Random Forest Classifier appears to perform slightly better than the Logistic Regression model based on these evaluation metrics.

model tuning and cross-validation, to improve and optimize the model's performance.

```python
#Model performance evaluation matrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

# To Print the evaluation metrics
print("Best Hyperparameters:", best_params)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("ROC-AUC:", roc_auc)
```

```
Best Hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}
Accuracy: 0.7532467532467533
Precision: 0.6385542168674698
Recall: 0.6625
F1-score: 0.6503067484662576
ROC-AUC: 0.7319122516556292
```

Performing Cross-validation to get a more robust estimate of the model's performance.

```python
#Cross-validation
cross_validation_scores = cross_val_score(best_rf_model, X_train, y_train, cv=5, scoring='accuracy')
mean_cv_accuracy = cross_validation_scores.mean()
print("Mean Cross-Validation Accuracy:", mean_cv_accuracy)
```

```
Mean Cross-Validation Accuracy: 0.7745932848736586
```

This was done to improve and optimize the RandomForestClassifier model's performance. By Hyperparameter Tuning the number of estimators to 100. Evaluation metrics accuracy Increased.

Accuracy: 0.7532467532467533

Precision: 0.6385542168674698 Recall: 0.6625 F1-score: 0.6503067484662576

ROC-AUC: 0.7319122516556292

Mean Cross-Validation Accuracy: 0.7745932848736586

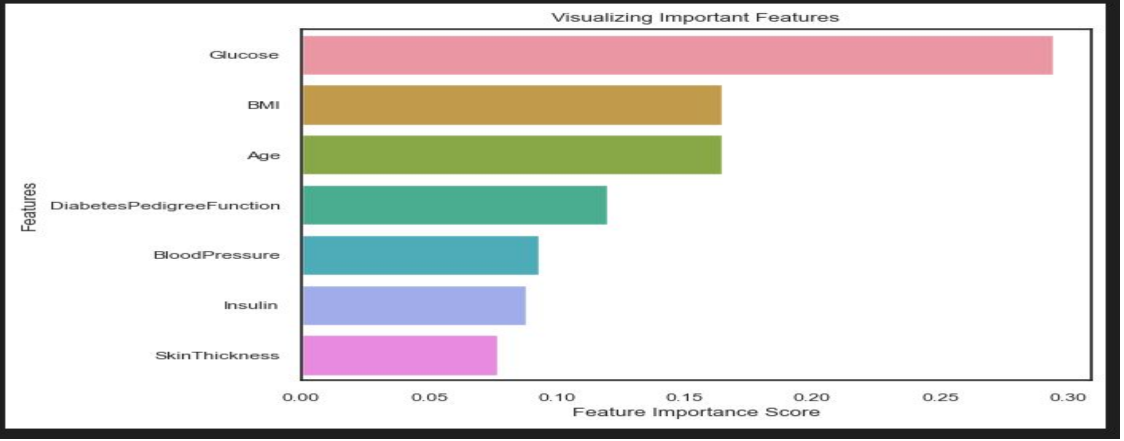
Feature scores showing the order of Importance or the diagnostic variables after tuning

```
sns.barplot(x=feature_scores, y=feature_scores.index)

# Add labels to the graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')

# Add title to the graph
plt.title("Visualizing Important Features")

# Visualize the graph
plt.show()
```



It can be seen that Glucose level is the most important feature in this diagnostic model while skin Thickness is the least feature affecting the accuracy of this model.

conclusion

From the machine learning developed, and exploratory data analysis conducted,the following were observed.

- Random Forest Classifier appear to have a better accuracy than Logistic regression model in predicting whether a patient have Diabetes or not.
- Cross validation gave more robust estimate of the mode performances with increased accuracy as `0.7745932848736586`
- Glucose level is the most important feature in predicting whether a patient have Diabetes or not.
- The least important feature that doesn’t really affect the prediction is Skin thickness.

Challenges

- Time constraints to explore available data using other models

THANK YOU