Diabetes Prediction using Supervised Learning Techniques

By
Oluyomi Alabi

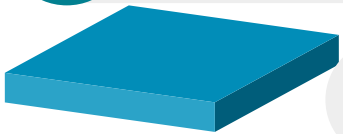
Introduction

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

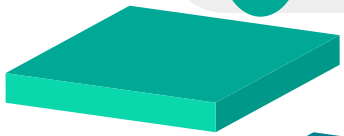
The ultimate goal of the project is to gain insights from the data sets and communicate these insights to stakeholders using appropriate visualizations and metrics to make informed decisions based on the business questions asked.

Process



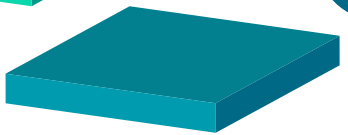
A **Importing the data**

Kaggle_Diabetes data set



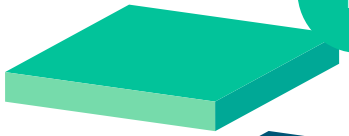
B **Exploratory Data Analysis**

Visualization to understand data



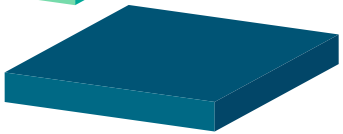
C **Preprocessing & Feature Engineering**

Handling missing values,outliers, scaling etc



D **Training ML model**

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



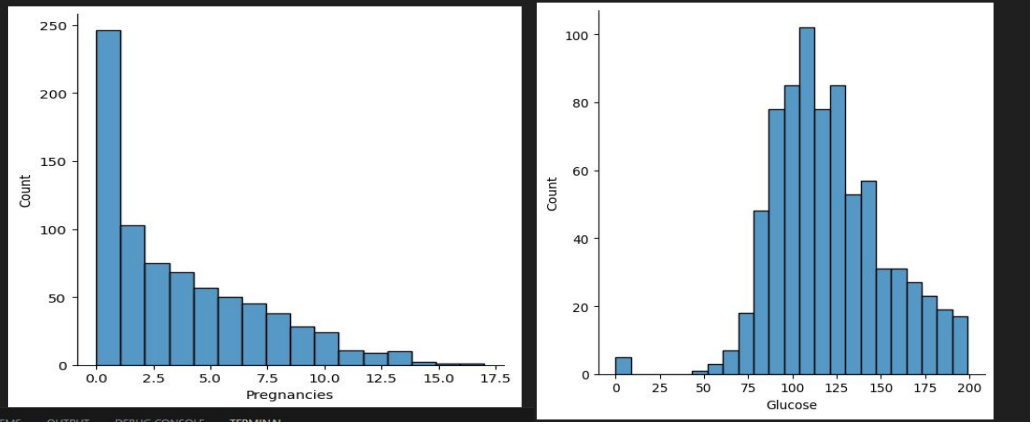
E **Model Tuning**

Hyperparameter Tuning & Cross validation

Exploratory Data Analysis Visualization

Visualization

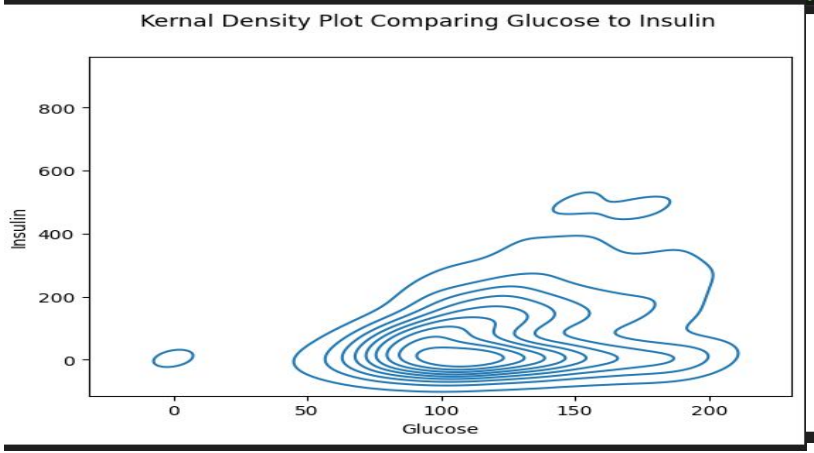
```
for column in num_feats:
    sns.displot(df[column])#EDA using displot to look at the distribution of numerical features in the datasetplot(df[column
```



Displot showing distribution of pregnancy and Glucose variables..

Visualizations

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

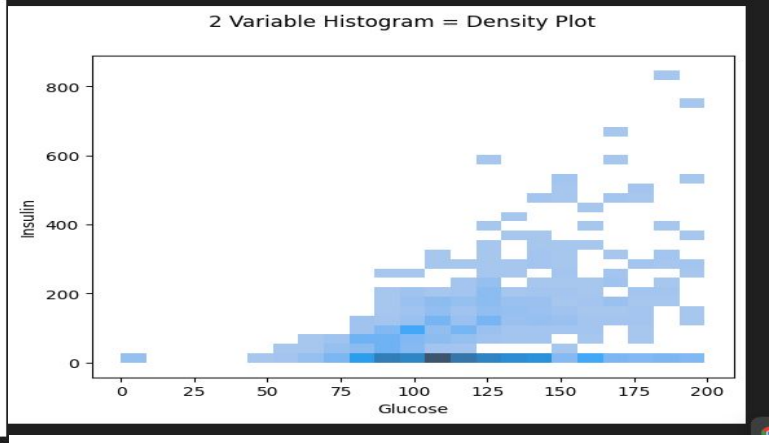



```
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(data=df,
             x="Glucose",
             y="Insulin")

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

plt.show()
```

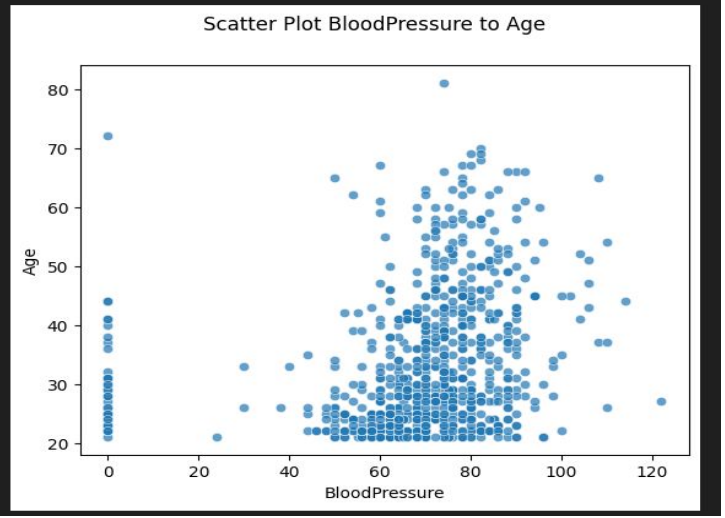
✓ 0.3s



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

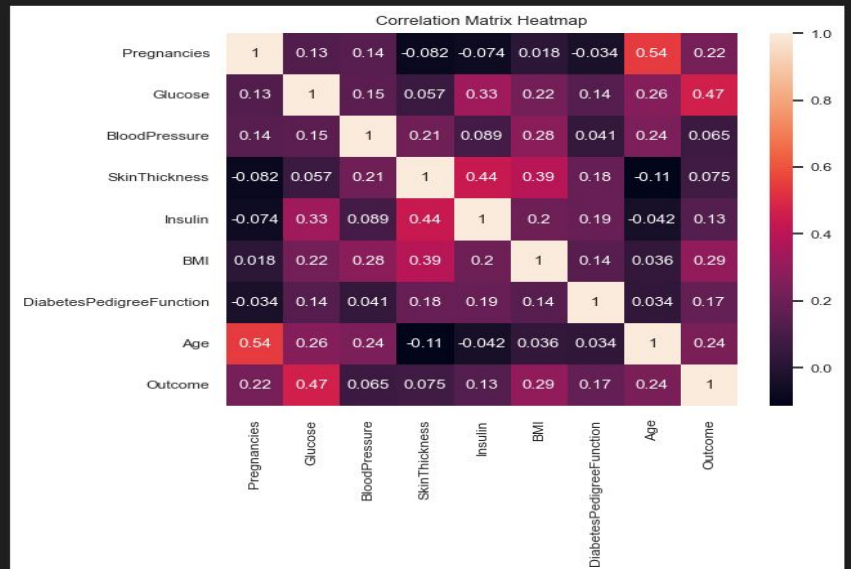
Scattered plot and Correlation matrix map showing correlation between different variables.

```
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')
```



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

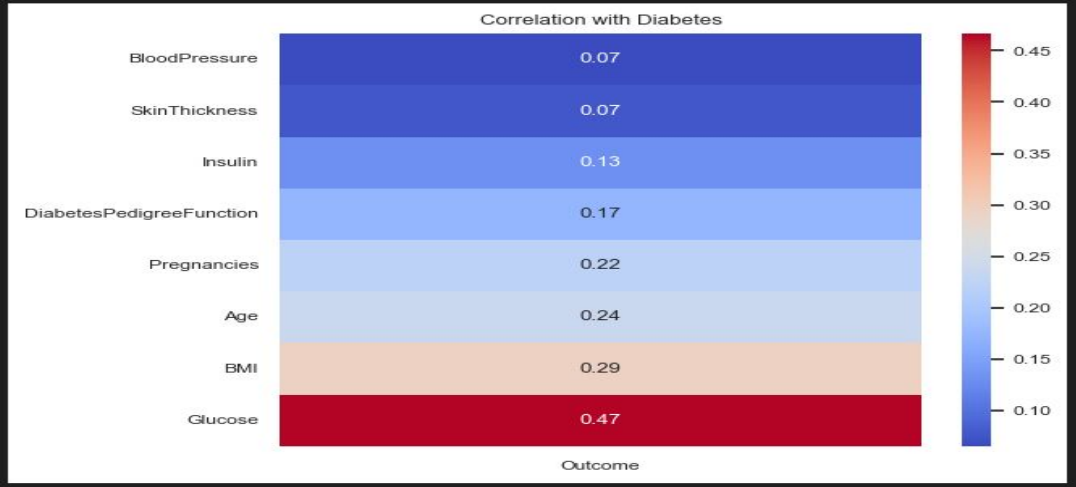
E.g BloodPressure cannot be zero except the patient is dead, same with Skin Thickness, BMI where mininum value is zero

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()
```



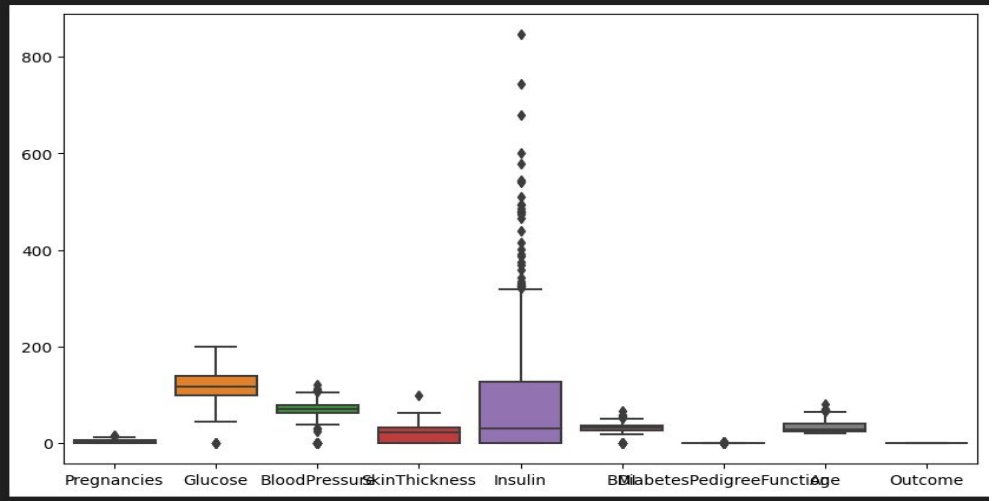
With a correlation of 0.47, Glucose is the most strongly correlated feature with the outcome. This suggests that higher glucose levels could be a significant indicator of diabetes.

BMI can be also important factor in diabetes.

Handling outliers in preprocessing and feature engineering

```
# Detecting outliers
plt.figure(figsize=(10, 6))
sns.boxplot(df)
```

<Axes: >



Insulin has many outliers

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 with evaluation metrics results

```
## Accuracy
```

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

# compute accuracy
accuracy = accuracy_score(y_test,y_pred)

# print accuracy
print(accuracy)
```

```
64]
0.7445887445887446
```

```
## F1-Score
```

```
# 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)
```

```
65]
0.6143790849673203
```

```
## Recall
```

```
from sklearn.metrics import recall_score

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

```
68]
Recall: 0.5875
```

```
## Precision_score
```

```
from sklearn.metrics import precision_score

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

```
71]
Precision: 0.6438356164383562
```

```
## AUC_score
```

```
# 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
```

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

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

Random Forest Classifier model with model Evaluation metrics Results

```
# 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)
```

[105]

```
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]]
```

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:

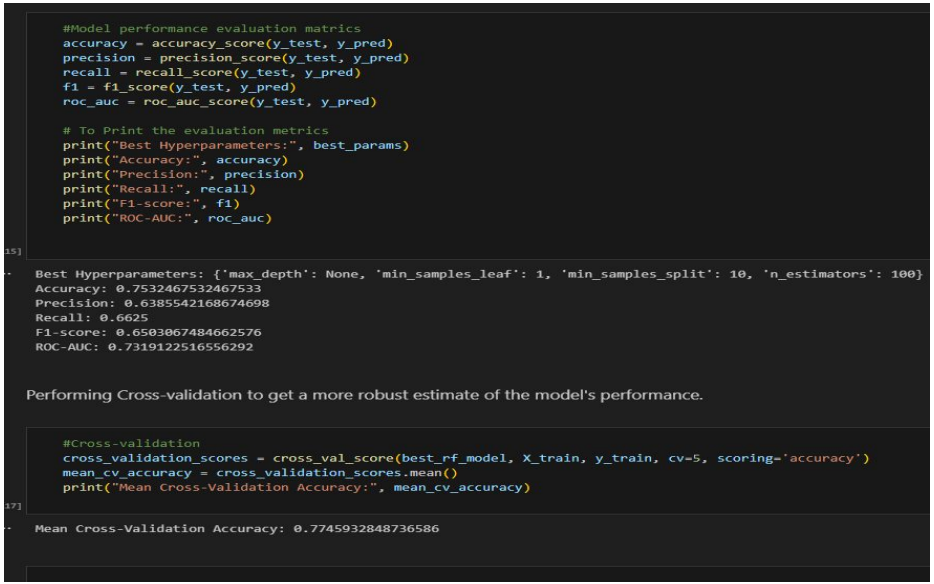
	precision	recall	f1-score	support
0	0.81	0.80	0.80	151
1	0.63	0.64	0.63	80
accuracy			0.74	231
macro avg	0.72	0.72	0.72	231
weighted avg	0.75	0.74	0.74	231

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.



```
#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.

```
#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

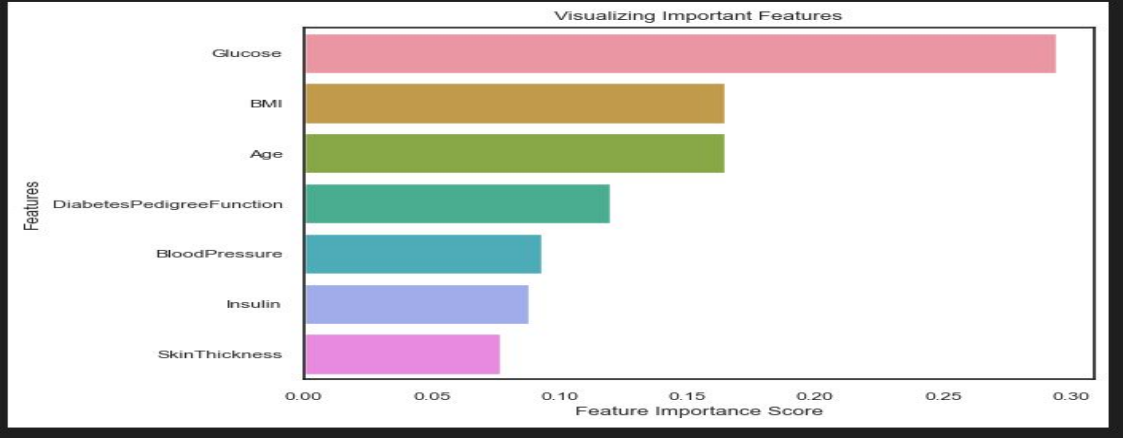
Feature scores showing the order of Importance or the diagnostic variables after tuning

```
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.

Challenges

- Time constraints to explore available data using other models

THANK YOU