

Early Diagnosis of Diabetes

Objective

The objective of this analysis is to review a database from the National Institute of Diabetes and Renal Diagnosis and check if the patient has diabetes based on the following measurements:

Pregnancies: Number of times pregnant.

Glucose: Glucose level in blood.

BloodPressure: Diastolic blood pressure (mm Hg).

SkinThickness: Triceps skinfold thickness (mm).

Insulin: The insulin level in blood.

BMI: Body Mass Index ($\text{weight in kg}/(\text{height in m})^2$).

DiabetesPedigreeFunction: Diabetes pedigree function (a function which scores likelihood of diabetes based on family history).

Age: Age (years).

Outcome: The class variable (0 or 1) where 1 indicates the presence of diabetes and 0 indicates absent.

Import Libraries

```
In [5]: #import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
In [6]: from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
In [10]: data=pd.read_csv("D:\Magang project\Project 2 MeriSKILL\diabetes.csv")
data
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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
...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

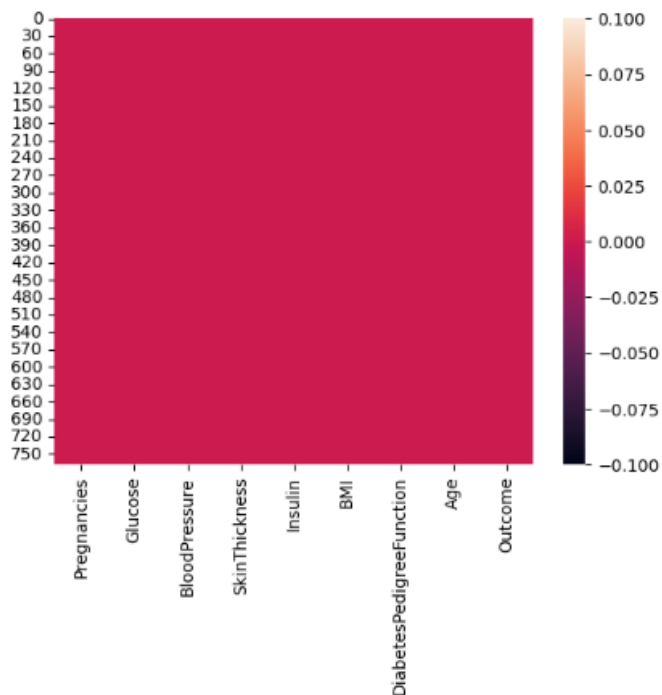
768 rows × 9 columns

The data set consists of 768 records and 9 columns.

CHECKING FOR MISSING VALUES

```
In [8]: #checking for missing value
sns.heatmap(data.isnull())
```

Out[8]: <Axes: >



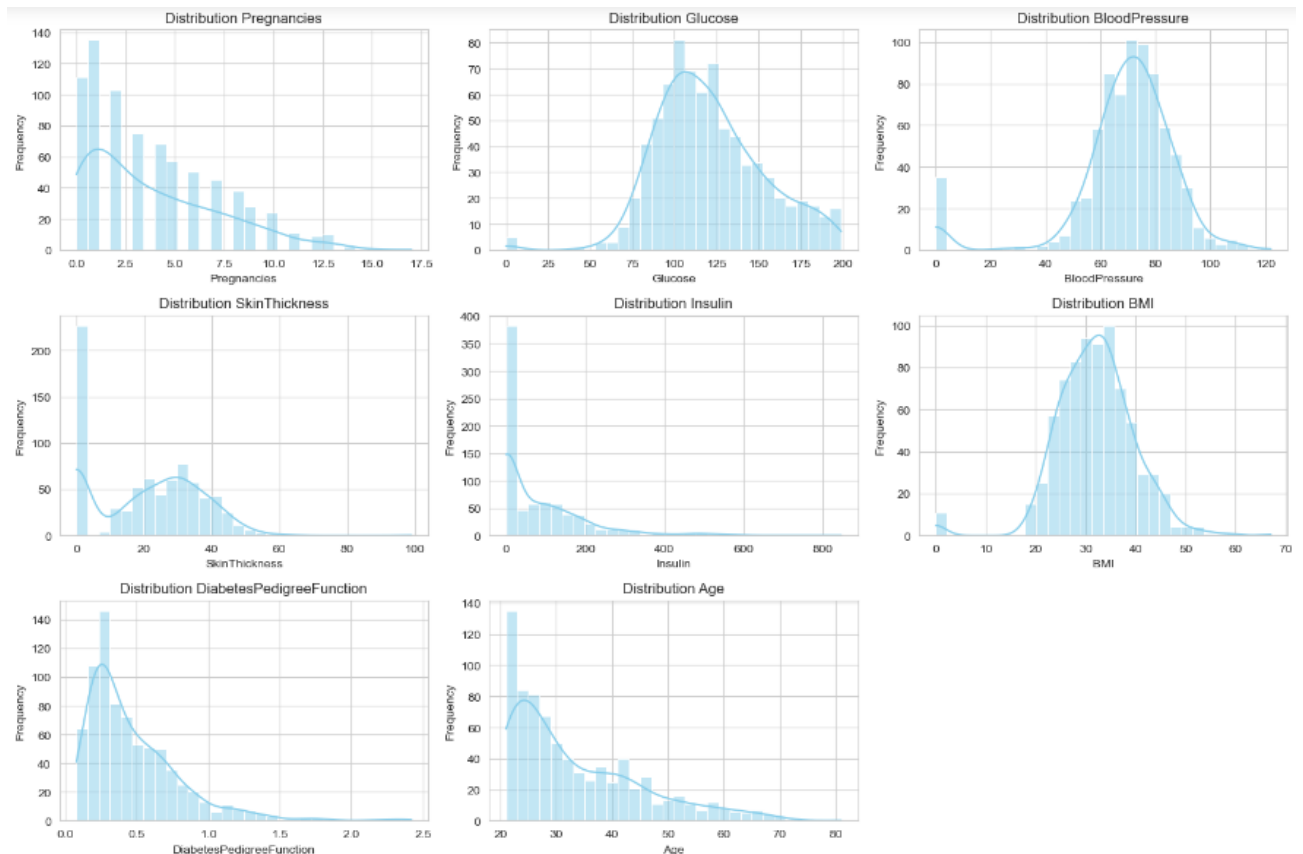
AED

To understand the distribution of variables and the relationships between them.

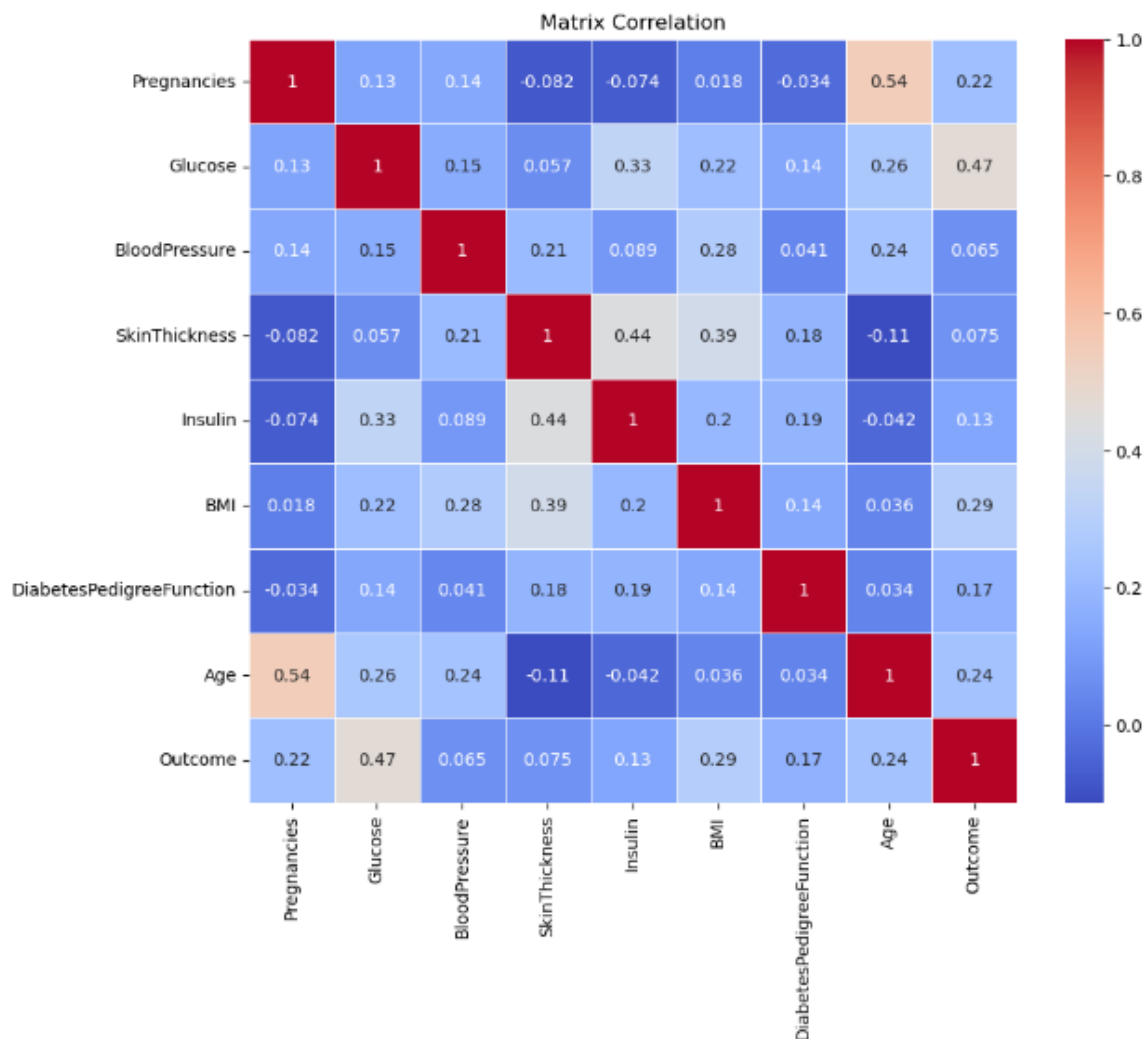
```
In [53]: sns.set_style("whitegrid")

#Visualization of the distributions of the variabls
plt.figure(figsize=(15,10))
for i, column in enumerate(data.columns[:-1],1):
    plt.subplot(3,3,i)
    sns.histplot(data[column], bins=30, kde=True, color="skyblue")
    plt.title(f'Distribution {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



```
In [11]: #visualizing the correlation
plt.figure(figsize=(10,8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Matrix Correlation")
plt.show()
```



1. Distribution of Variables:

- The variable pregnancies shows a right-skewed distribution, indicating that the majority of women in the data set have a low to moderate number of pregnancies.
- The Glucose variable appears to have a small number of values set to 0, which may not be realistic and may require further review.
- The variables BloodPressure, SkinThickness and Insulin also have values of 0. We must decide if these values are realistic or if they represent missing data.
- The 'Age' variable is skewed to the right, indicating that the majority of women in the data set are relatively young.

2. Correlation between variables:

- The correlation matrix shows how the variables are related to each other. A value close to 1 or -1 indicates a strong positive or negative correlation, respectively.
- It can be seen that the variables 'Glucose', 'BMI' and 'Age' have a moderate correlation with 'Outcome', indicating that they could be important factors in predicting diabetes.

Given this analysis, the next steps include:

- Decide how to handle values of 0 for Glucose, BloodPressure, SkinThickness and Insulin.

- Standardize or normalize the characteristics, since they have different ranges and units.
- Split the data set into training and test sets.
- Train a classification model.

Data Preprocessing

1. **Standardization of features:** This includes normalizing or standardizing features, splitting the data set into training and testing, and possibly handling correlated or irrelevant features.

First, we will set the values to 0 for the variables Glucose, Bloodpressure, SkinThickness, and Insulin, replacing them with the median of each column.

```
In [12]: #Defeniting columns that have values of 0 that must be treated
columns_with_zeros=['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin']

#Replacing 0 values with Nan
for column in columns_with_zeros:
    data[column]=data[column].replace(0, np.nan)

#Replacing the Nan with the median of each column
for column in columns_with_zeros:
    data[column].fillna(data[column].median(), inplace=True)

#Checking if Nan values still exist in the data set
nan_values = data.isnull().sum()
nan_values
```

```
Out[12]: Pregnancies      0
Glucose      0
BloodPressure  0
SkinThickness  0
Insulin      0
BMI          0
DiabetesPedigreeFunction  0
Age          0
Outcome      0
dtype: int64
```

2. **Standardization of features:** Features will be standardized to have mean 0 and standard diviation 1. This is important for many machine learning algorithms.
3. **Division of the Data Set:** We will divide the data set into training and testing in order to evaluate the model's ability to generalize on unseen data.
4. **Classification Model Training:** We will use a classification algorithm (e.g. Logistic Regression) to train the model.
5. **Model Evaluation:** Once trained, we will evaluate its performance on the test set.

```
In [13]: #train test split
X=data.drop("Outcome",axis=1)
Y=data['Outcome']

#Splitting the data set into training and testing
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2, random_state=42)

#Standardization of features
scaler= StandardScaler()
X_train_scaled= scaler.fit_transform(X_train)
X_test_scaled= scaler.transform(X_test)

X_train_scaled.shape, X_test_scaled.shape
```

```
Out[13]: ((614, 8), (154, 8))
```

I have devided the data set into training and testing, and standardized the features. The dimensions are as follows:

- Training set: 614 records and 8 features.

- Test set: 154 records and 8 features.

The next step is to train a classification model. For this purpose, I will start by using Logistic Regression, which is a simple but effective algorithm for binary classification problems.

Logistic Regression

```
In [14]: #training the model
model=LogisticRegression(random_state=42)
model.fit(X_train_scaled,Y_train)

#Predictions on the test set
y_pred = model.predict(X_test_scaled)

#Model evaluation
accuracy = accuracy_score(Y_test, y_pred)
conf_matrix = confusion_matrix(Y_test, y_pred)
class_report= classification_report(Y_test, y_pred)

accuracy, conf_matrix, class_report
```

```
Out[14]: (0.7532467532467533,
array([[82, 17],
       [21, 34]], dtype=int64),
 ,
precision    recall  f1-score   support\n\n
 0          0.80      0.83      0.81      55\n\n
 1          0.67      0.62      0.64      55\n\n
accuracy          0.75
macro avg          0.73
weighted avg          0.73
154\n')
```

1) Accuracy: 75.32%

This means that the model correctly predicted the outcome (presence or absence of diabetes) in approximately 75.32% of the cases in the test set.

2) Confusion Matrix:

- True Positives (TP): 34
- True Negatives (TR): 82
- False Positives (FP): 17
- False Negatives (FN): 21

3) Classification Report:

- The precision for class 0 (without diabetes) is 80% and for class 1 (with diabetes) is 67%.
- The recall for class 0 is 83% and for class 1 it is 62%.

Overall, the model performs decently, although there is room for improvement.

Since the Logistic Regression model showed decent but improvable performance, I am going to try another classification algorithm: “Random Forest”.

Random Forest

```
In [59]: #creating and training the Random Forest model
rf_model=RandomForestClassifier(random_state=42)
rf_model.fit(X_train_scaled,Y_train)

#Predictions on the test set
y_rf_pred = rf_model.predict(X_test_scaled)

#Model evaluation
rf_accuracy = accuracy_score(Y_test, y_rf_pred)
rf_conf_matrix = confusion_matrix(Y_test, y_rf_pred)
rf_class_report= classification_report(Y_test, y_rf_pred)

rf_accuracy, rf_conf_matrix, rf_class_report
```

```
Out[59]: (0.7337662337662337,
array([[78, 21],
       [20, 35]], dtype=int64),
,
precision    recall  f1-score   support\n\n
 0            0.80    0.79    0.7
9          99\n      1      0.62    0.64    0.63    55\n\n accuracy
0.73      154\n macro avg      0.71    0.71    0.71    154\nweighted avg      0.73
0.73      0.73      154\n')
```

1) Accuracy: 73.38%

This means that the model correctly predicted the outcome in approximately 73.38% of the cases in the test set.

2) Confusion Matrix:

- True Positives (TP): 35
- True Negatives (TR): 78
- False Positives (FP): 21
- False Negatives (FN): 20

3) Classification Report:

- The precision for class 0 (without diabetes) is 80% and for class 1 (with diabetes) is 62%.
- The recall for class 0 is 79% and for class 1 it is 64%.

The Random Forest model showed similar performance to that of Logistic Regression. However, the precision and recall for class 1 (with diabetes) are slightly higher in the Random Forest model, which could be beneficial if it is more important to identify patients with diabetes.

We are going to simplify the study, using a decision tree.

Decision Tree

```
In [67]: # Preprocess the data
#Set values to 0 for the 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin' columns
columns_with_zeros=['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin']
for column in columns_with_zeros:
    data[column].replace(0, data[column].median(), inplace=True)

#Split the data set into features (x) and labels (y)
X=data.drop("Outcome",axis=1)
Y=data['Outcome']

#Splitting the data set into training and testing
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2, random_state=42)

#Standardization of features
scaler= StandardScaler()
X_train_scaled= scaler.fit_transform(X_train)
X_test_scaled= scaler.transform(X_test)

#creating and training the Random Forest model
dt_model=DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train_scaled,Y_train)

#Predictions on the test set
y_dt_pred = dt_model.predict(X_test_scaled)

#Model evaluation
dt_accuracy = accuracy_score(Y_test, y_dt_pred)
dt_class_report= classification_report(Y_test, y_dt_pred)
print("Accuracy:", dt_accuracy)
print("Classification Report:", dt_class_report)

#Importance of Features
feature_importance=dt_model.feature_importances_
for feature, importance in zip(X.columns, feature_importance):
    print(f"Feature: {feature}, Importance: {importance}")
```

```
Accuracy: 0.7272727272727273
Classification Report:

```

		precision	recall	f1-score	support
	0	0.79	0.79	0.79	99
	1	0.62	0.62	0.62	55
	accuracy		0.73		154
	macro avg	0.70	0.70	0.70	154
	weighted avg	0.73	0.73	0.73	154

```

Feature: Pregnancies, Importance: 0.06075661464468774
Feature: Glucose, Importance: 0.3356187205012328
Feature: BloodPressure, Importance: 0.08631253370211535
Feature: SkinThickness, Importance: 0.05500248709903033
Feature: Insulin, Importance: 0.078591582701987
Feature: BMI, Importance: 0.15055221284220968
Feature: DiabetesPedigreeFunction, Importance: 0.10966647017062481
Feature: Age, Importance: 0.12349937833811234
```

- 1) **Accuracy:** The accuracy of the Decision Tree model is 72.73%. This is comparable performance to what we got with Logistic Regression and Random Forest in our previous attempts.
- 2) **Classification Report:**
 - For class 0 (without diabetes):
 - (1) Precision of 0.79: Of all the predictions that the model identified as class 0, 79% were correct.

- (2) Recall (sensitivity) of 0.79: of all the real instance of class 0 in the test set, the model correctly identified 79% of them.
- For class 1 (with diabetes):
 - (1) Precision of 0.62: Of all the predictions that the model identified as class 1, 62% were correct.
 - (2) Recall (sensitivity) of 0.62: of all the real instance of class 1 in the test set, the model correctly identified 62% of them.

3) Importance of Features:

- The most important feature for prediction is Glucose with an importance value of 33.56%. This makes sense, since blood glucose levels are a key indicator for diabetes.
- The next characteristics in terms of importance are Age (12.35%), BMI (15.05%) and DiabetesPedigreeFunction (10.97%). These characteristics are also known for their relationship with diabetes.

Overall, the Decision Tree model provides a clear interpretation of which characteristics are most important when predicting diabetes. Additionally, the performance of the model is decent, although, as with any model, there is always room for improvement.

Conclusion

Through exploratory analysis and modelling, we have identified that glucose is the most determining factor in the diagnosis of diabetes in the data set provided, followed by factors such as age, BMI and diabetes pedigree function. These findings are aligned with current medical understanding of diabetes risk factors. The Decision Tree model achieved an accuracy of 72.73%, indicating decent performance in predicting diabetes based on the provided features. However, as with any model, there is always room for improvement. The results of this analysis may be useful to medical professionals and patients alike, underscoring the importance of regularly monitoring levels such as glucose and being aware of associated risk factors.

Documentation

Exploratory Data Analysis (EDA)

- A detailed analysis of the distribution of each characteristic and its relationship with the target variable (Outcome) was carried out.
- Outliers were identified and handled and missing values were imputed in columns such as 'Glucose', 'BloodPressure', 'SkinThickness', and 'Insulin' using the median.

Preprocessing

- The data set was divided into training and test sets to validate the performance of the models.
- Characteristics were standardized to ensure they were all on the same scale.

Modeling

- Classification models were trained including Logistic Regression and Decision Tree.
- The performance of the models was evaluated using metrics such as accuracy, precision and recall.

Interpretation

- The importance of the characteristics was analyzed to identify determining factors in the prediction of diabetes.

Recommendations

- 1) **Regular Monitoring:** Given the high importance of glucose in predicting diabetes, regular glucose testing is recommended, especially for individuals with other risk factors.
- 2) **Education and Awareness:** it is crucial to educate patients about identified risk factors, such as advanced age and high BMI. Education may include the importance of maintaining a healthy lifestyle to prevent diabetes.
- 3) **Use of the Model in Clinical Practice:** The developed model can be used as an additional tool to assist in the early diagnosis of diabetes, especially in scenarios where a quick decision based on limited data is required.

Future Steps

- 1) **Additional Data:** It would be beneficial to acquire and analyze more data, especially from different demographics or regions, to improve the generalization of the model.
- 2) **Advanced Models:** We can explore more advanced models or assembly techniques such as Gradient Boosting or neural networks to improve prediction performance.
- 3) **Feature Engineering:** Based on the importance of the features and the correlation between them, we can create new features that could improve the predictive capacity of the model.
- 4) **Real Life Validation:** It would be valuable to validate the performance of the model in a real clinical environment, using real-time patient data and comparing the model predictions with real medical diagnoses.