



DATA SCIENCE

Major Project

Report

Project Title: *Diabetic Prediction*

Submitted to: Ms. Meghana Gowda V

Submitted by: Ms. Bhawna Goyal

Data Science August Batch 2023 – Mentor led



TEACHNOOK (TEACHSCAPE ONLINE LEARNING SERVICES PRIVATE LIMITED)

Content:

- Introduction
- About data
- Model pipeline
- Issues faced in model
- Result of model

Diabetes Prediction using Machine Learning



Introduction





About diabetes: Diabetes is a long-term condition that affects how your body handles blood sugar. Diabetes mellitus, sometimes known as diabetes, is a collection of disorders that can result in excessive amounts of glucose (a form of sugar) in your blood.

Diabetes develops when your pancreas fails to generate enough insulin or your body gets resistant to it. Its symptoms include feeling tired, hungry, or thirsty, and passing more urine (pee) than normal.

About Project: The Diabetes Prediction Project tries to estimate an individual's risk of diabetes based on several health factors. Diabetes identification is critical for prompt intervention and control. This study analyses and predicts diabetes risks using machine learning techniques. To avoid difficulties, early identification and care are crucial. Using machine learning approaches, this study addresses the requirement for an accurate diabetes prediction system. For this as per the given instruction, I have to use either classification & regression i.e., supervised learning. And I choose the classification algorithm to detect a person is suffered from diabetes or not.

Problem statement: The goal is to create a reliable diabetes predictive model that would aid healthcare providers in early detection and intervention.

Objective:

-  Analyze a large dataset with health-related factors.
-  To make predictions, use a variety of machine learning methods.
-  Analyze and compare the effectiveness of various models.
-  Determine important characteristics for diabetes prediction.

Literature Review:

- Techniques for Diagnosing Diabetes

The accuracy and timeliness of traditional diabetes detection techniques are frequently hampered. These techniques can be supplemented or replaced by machine learning algorithms, resulting in more trustworthy outcomes.

- Healthcare and Machine Learning

With applications ranging from illness prediction to therapy improvement, machine learning has shown to have tremendous promise in the healthcare sector. This effort adds to the body of knowledge in this field of study.



About dataset

Data collection & preprocessing:

To predict the patient is diabetic or not for that it is necessary to have data of some medical parameters.

When we want to create a model, it is necessary to have relevant & accurate data. That's the 1st step of any model "data gathering".

Data Source:

The 768 instances and 9 features utilized in this project were gathered from

<https://www.kaggle.com/datasets/aemyjutt/diabetesdataanslysis/data>

The output feature is an outcome that indicates to 0 and 1 whether or not a person has diabetes, and includes all data connected to diabetes in its 9 columns.

Other features of data are: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age

Data Cleaning:

A number of methods, including imputation and outlier removal, were used to deal with null values, outliers, and inconsistent data entries.

Sample of dataset:

	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

Feature Engineering:

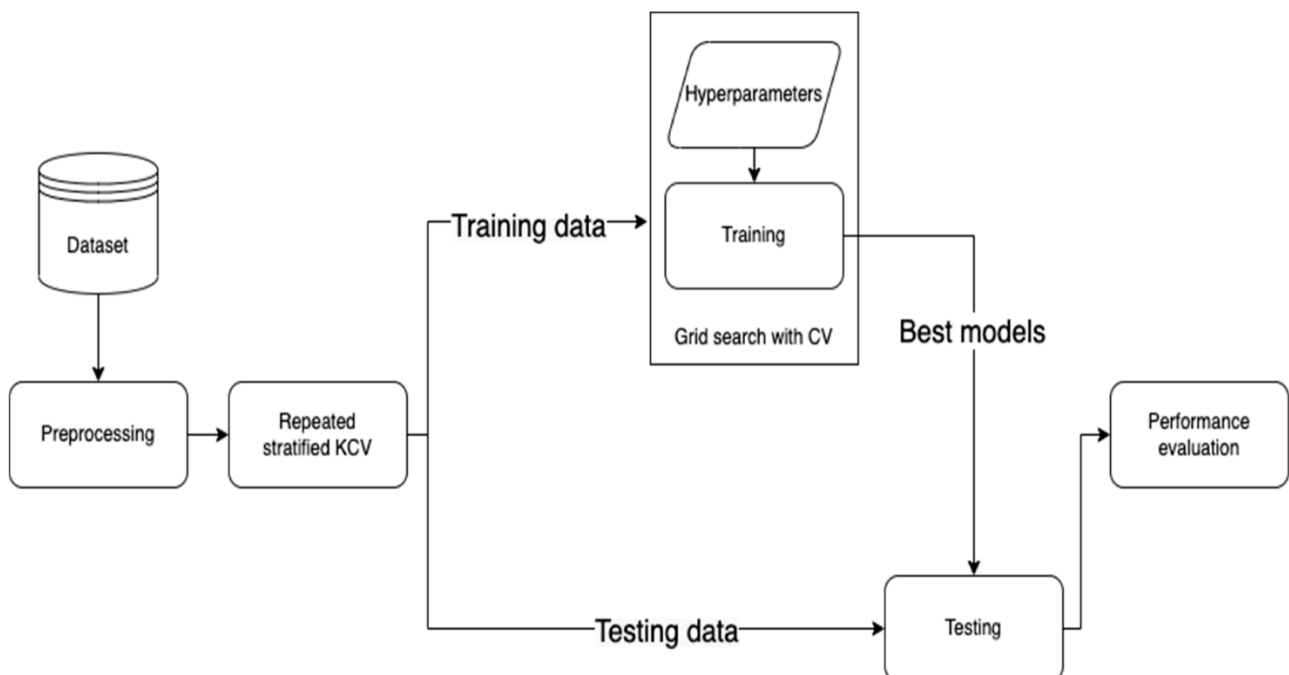
By converting uninformative characteristics into informative predictors, feature engineering increased the dataset's predictive ability.



Model pipeline

The steps are needed for model:

1. Importing the necessary libraries
2. Data Preprocessing
 - a. Data Gathering
 - b. Handling null values
 - c. Noise removal
 - d. Outlier removal
3. EDA (Exploratory Data Analysis)
 - a. Data Visualization
 - b. Correlation
 - c. Graphical EDA
4. Data Modelling
 - i. Data Encoding
 - ii. Splitting of Data
 - iii. Creation of model
 - a. Logistic Regression
 - b. Random Forest
 - c. SVM
 - d. ANN
 - e. Stacking
5. Evaluating Model
6. Testing



i. Importing necessary libraries

▼ Import necessary libraries

```
[ ] 1 import sklearn
    2 import pandas as pd
    3 import matplotlib.pyplot as plt
    4 import seaborn as sns
    5 import numpy as np
    6
```

Pandas: A tool for analysing and manipulating data. It offers crucial data structures for processing structured data, such as DataFrame and Series.

Matplotlib & Seaborn: The powerful plotting toolkit Matplotlib, on which Seaborn is based, offers a high-level interface for creating visually appealing and educational statistical visuals.

NumPy: The core Python library for numerical computing. It supports the use of matrices, arrays, and mathematical operations on various data structures.

Scikit-Learn (sklearn): A machine learning framework called Scikit-Learn (sklearn) offers straightforward and effective tools for data mining and data analysis. You made use of a number of sklearn modules, including preprocessing, model choice, and evaluation metrics.

ii. Data Preprocessing

To make sure that the dataset is clear, devoid of missing values and outliers, and in an appropriate format for machine learning algorithms, these pretreatment processes are crucial.

i. Data Gathering

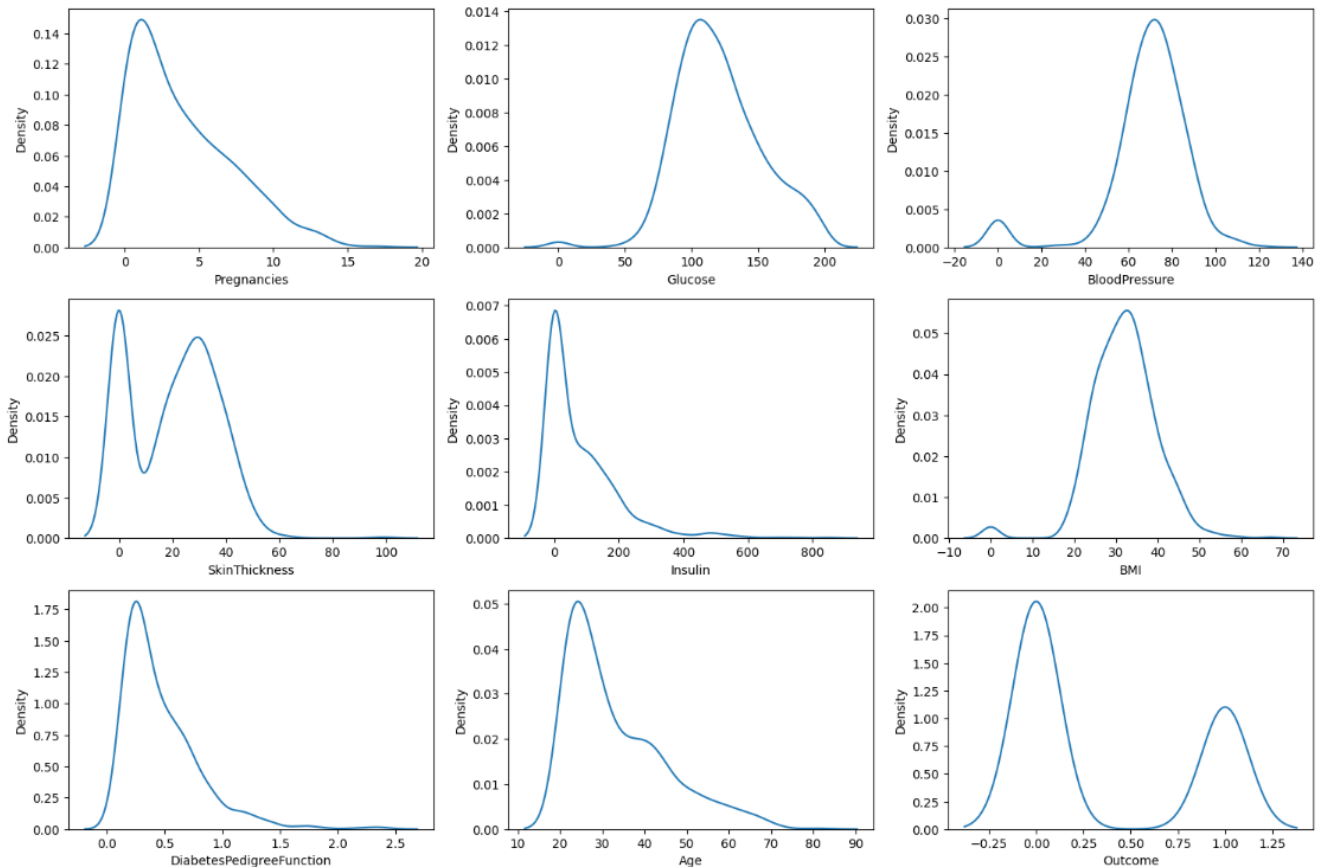
```
[ ] 1 df=pd.read_csv("/content/diabetes.csv")    #gather data into dataframe
```

```
[ ] 1 df.head()                                #display data
```

	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

Choosing the right data source was an important stage in the diabetes prediction research. The dataset used in this project came from an outside source, guaranteeing a wide-ranging and thorough collection of actual data.

Then I check the shape & distribution of data for further analysis & almost data having too much noise & outliers seen by the normal distribution curve.



ii. Handling Null values

To ensure that machine learning models can effectively learn from the data, handling null or missing values is a vital step in the preprocessing of the data.

1. Identifying Null values

In my model, I replace all 0 values with NAN (Not a number) to get better data

```
[ ] 1 df.replace({'Pregnancies':0,  
2           'Glucose':0,  
3           'BloodPressure':0,  
4           'SkinThickness':0,  
5           'Insulin':0,  
6           'BMI':0,  
7           'DiabetesPedigreeFunction':0,  
8           'Age':0},np.nan,inplace=True)
```

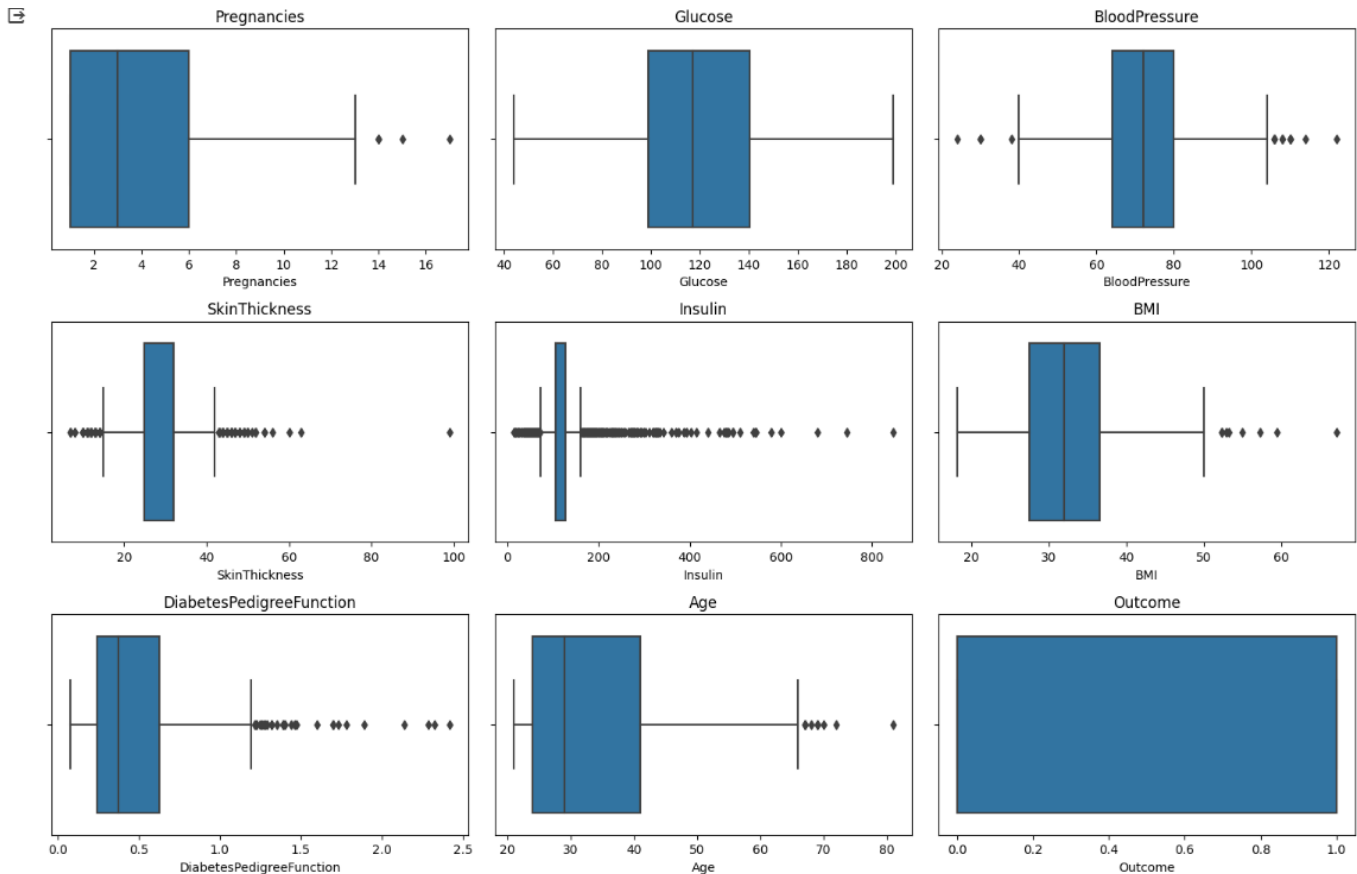
```
[ ] 1 df.isnull().sum()           #check null values  
  
Pregnancies      111  
Glucose           5  
BloodPressure     35  
SkinThickness    227  
Insulin          374  
BMI              11  
DiabetesPedigreeFunction  0  
Age              0  
Outcome          0  
dtype: int64
```

2. Filling null values

In this model, all the null value are replaced by the mode of its attribute, if I eliminate the null values then there may a huge loss of data & important values because total instance is 768 & instances with null value are 227

iii. Outlier Removal

Implementing the Interquartile Range (IQR) technique, outlier reduction was implemented.



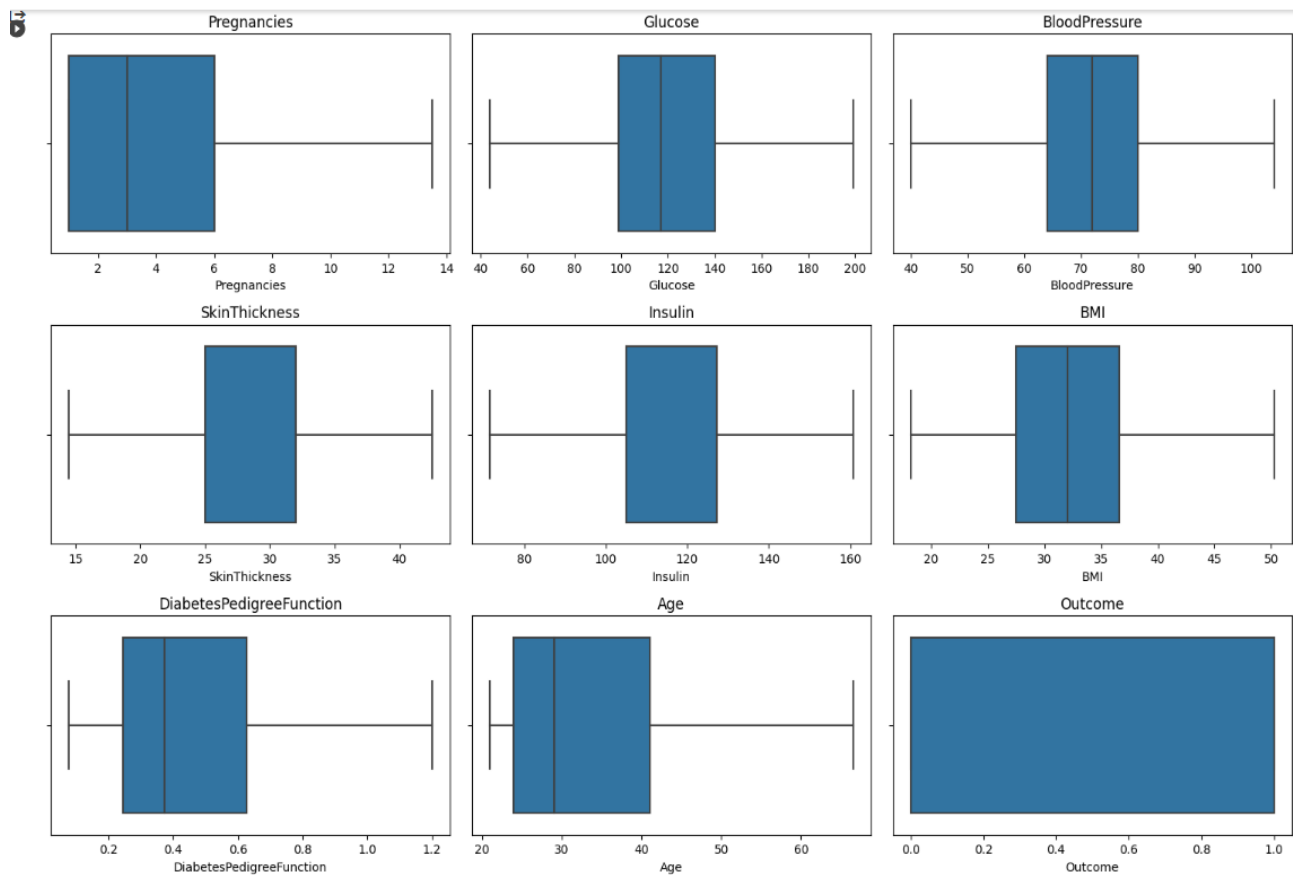
There are many no. of outliers in BMI, Pregnancies, Blood Pressure, Skin Thickness, Insulin, Diabetes Pedigree Function, Age.

To remove all this outlier, use IQR method to get the data range b/w 25 to 75 percentile & through the int & float type data i.e. BMI and pregnancies

Key formula to remove outlier

```
1 #remove outliers
2 #calculate the Q1, Q2, Q3 ... if data is not in inter quartile range then it eliminate
3 #make user defined function
4 def remove_outliers_iqr(data):
5     # Calculate the first quartile (Q1) and third quartile (Q3)
6     Q1 = np.percentile(data, 25)
7     Q3 = np.percentile(data, 75)
8     # Calculate the interquartile range (IQR)
9     IQR = Q3 - Q1
10    # Define the lower and upper bounds for outliers
11    lower_bound = Q1 - 1.5 * IQR
12    upper_bound = Q3 + 1.5 * IQR
13
14    # Remove outliers
15    data = np.where(data > upper_bound, upper_bound, np.where(data < lower_bound, lower_bound, data))
16
17    return data[(data >= lower_bound) & (data <= upper_bound)]
18
19 #apply for those coulms which contain outliers
20 for col in fea_outlier:
21     df[col]= remove_outliers_iqr(df[col])
```

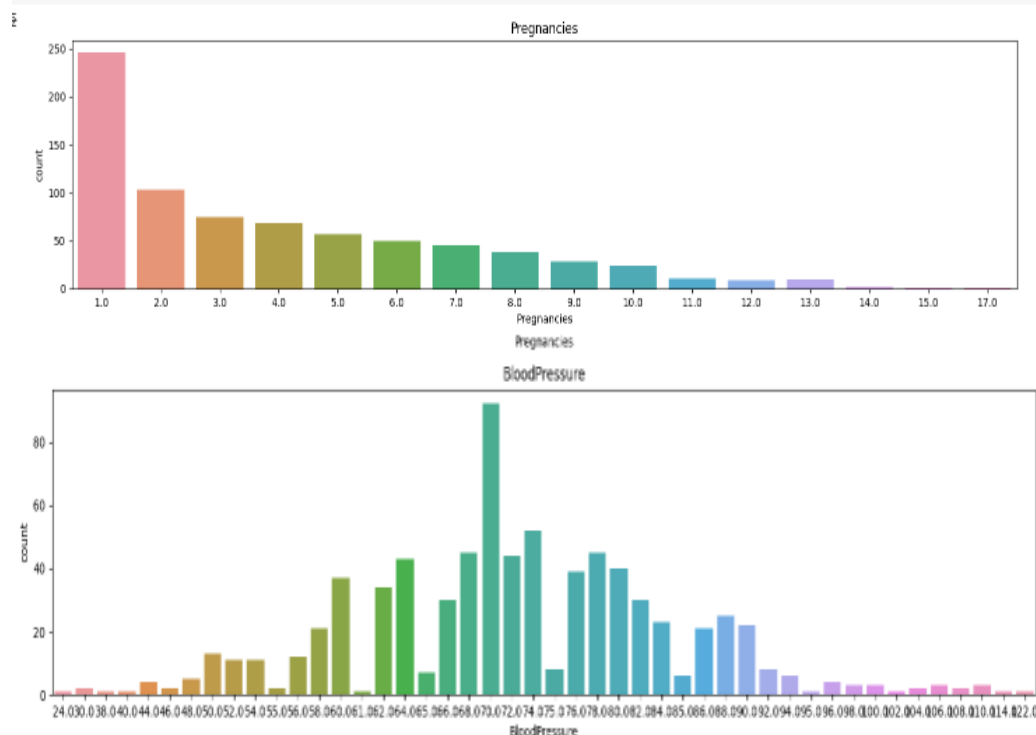
After removal all outlier data looks like

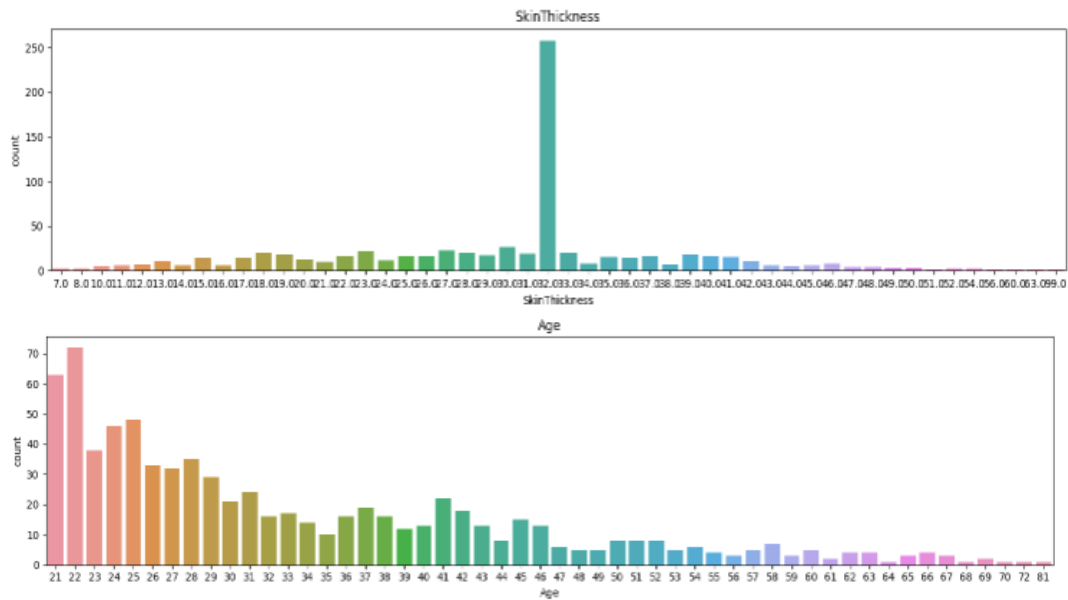


iii. EDA (Exploratory Data Analysis)

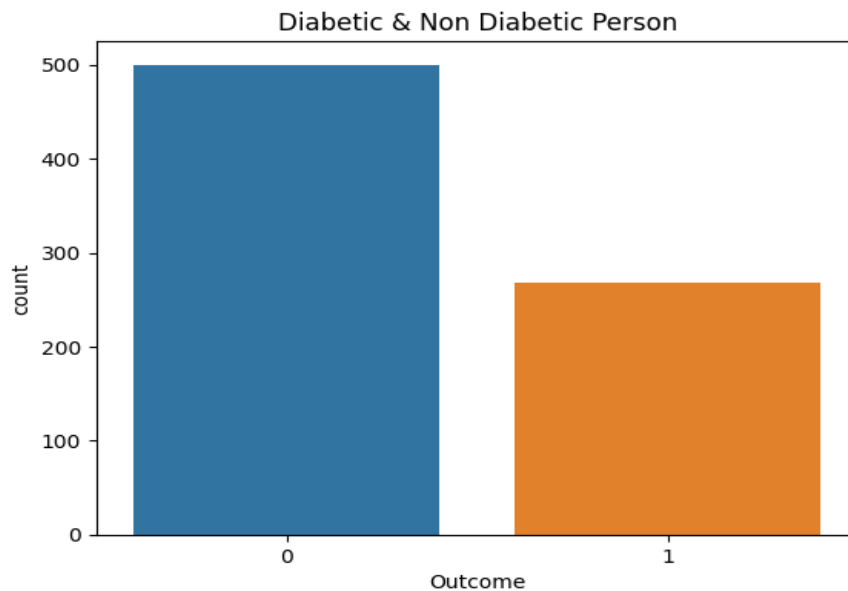
Before creating machine learning models, I conducted exploratory data analysis (EDA) and data visualization to obtain understanding of your dataset.

- Univariate Analysis: By plotting the count of certain parameters (pregnancies, blood pressure, skin thickness, and age) in this section, I performed univariate analysis. These charts shed light on the occurrence and distribution of several elements in the dataset.

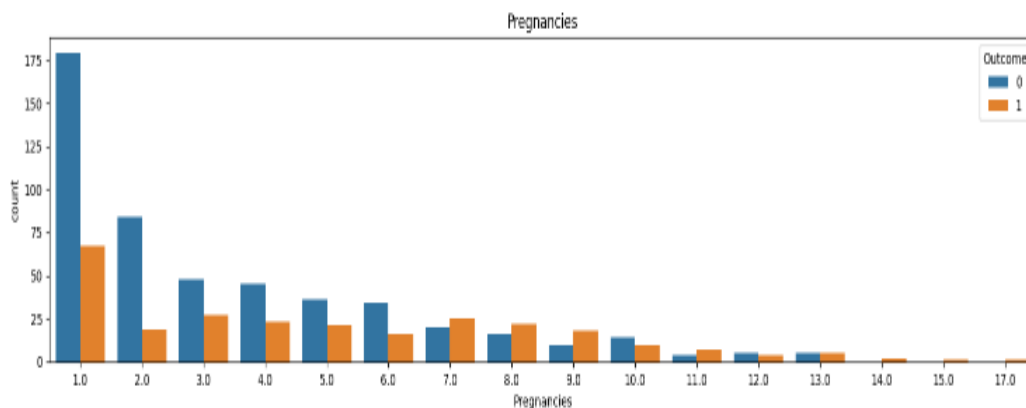


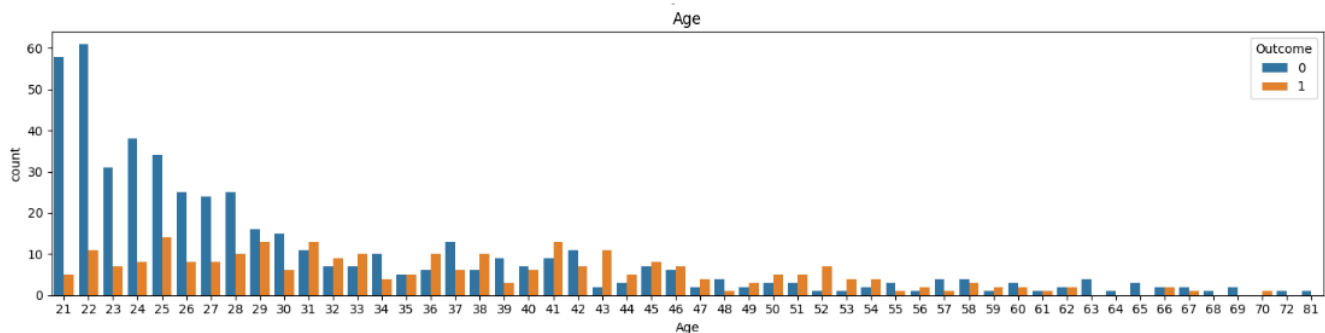
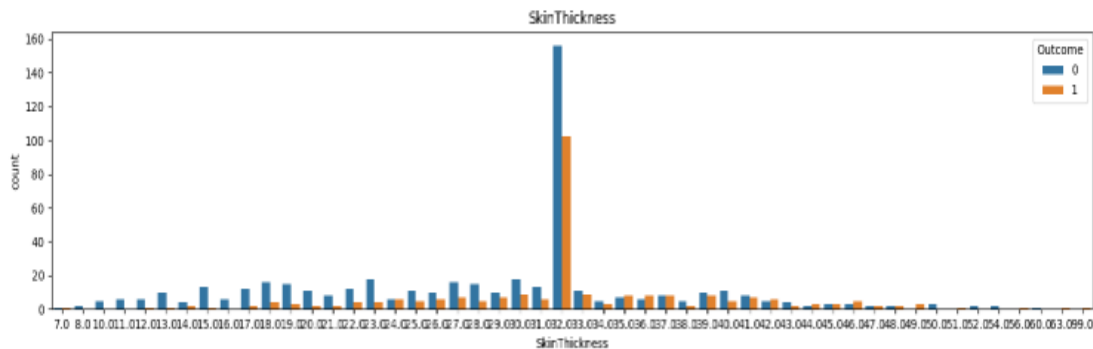
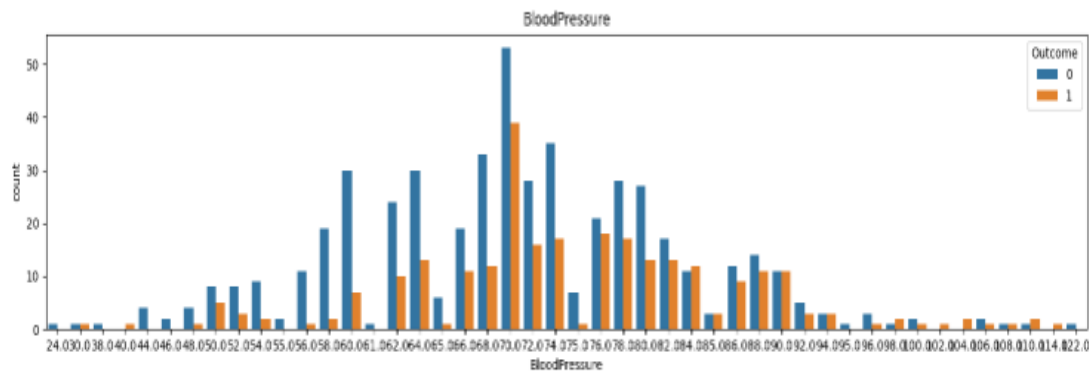


✚ Outcome Analysis: A countplot was used to display the number of diabetic and non-diabetic people. This aids in your comprehension of the dataset's distribution of the target variable (Outcome).

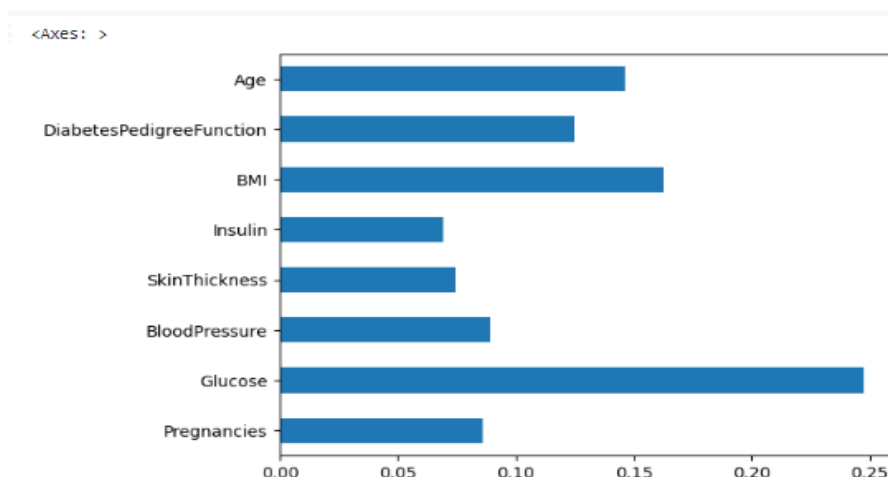


✚ Multivariate Analysis: By charting the number of features with respect to the Outcome variable, I expanded the scope of my study to multivariate space. These plots offer more in-depth perceptions of how several factors connect to the target variable.





📊 **Feature Importance Analysis:** I used a horizontal bar chart to display feature relevance after training your Random Forest model. The weights assigned to each feature in your machine learning model are displayed in this figure.

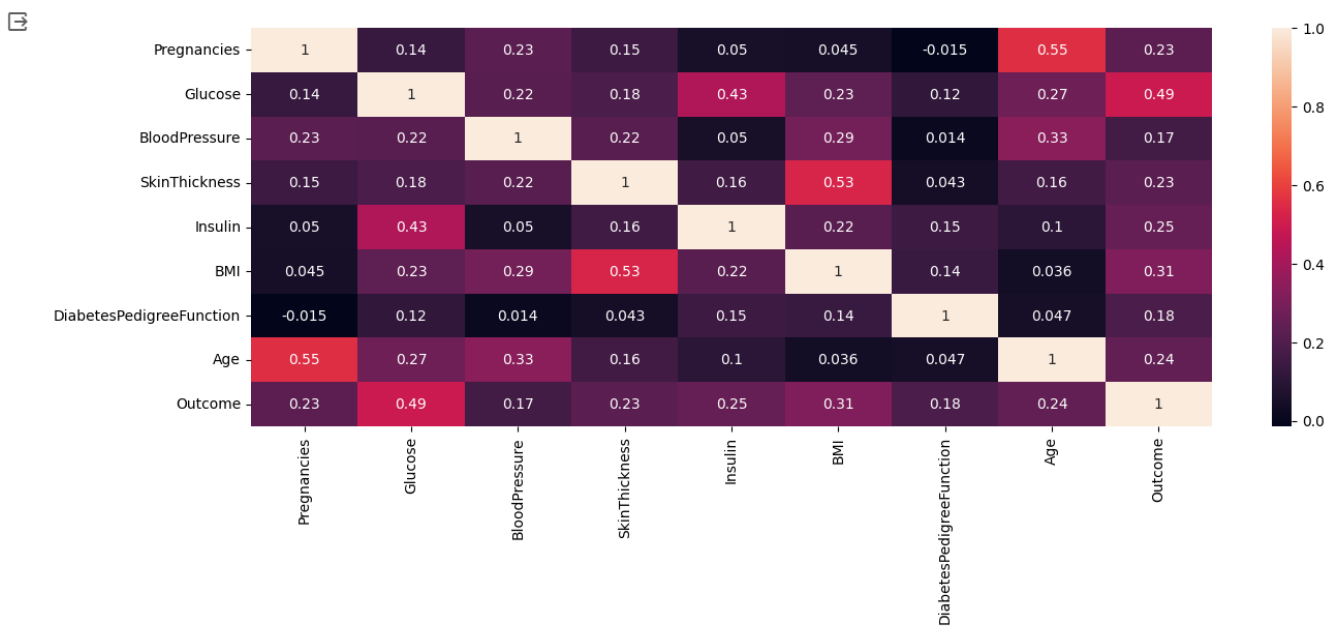


📊 **Statistical Insights :**
Summary Statistics:- Provide summary statistics for numerical features like glucose, blood pressure, body mass index, etc., such as mean, median, standard deviation, minimum, maximum, and quartiles. These statistics provide a broad overview of the data's central tendency and distribution.

```
[ ] 1 df.describe() #non graphical EDA
```

	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	3.989583	121.539062	72.295573	29.994792	130.932292	32.450911	0.471876	33.240885	0.348958
std	3.219464	30.490660	12.106756	8.886506	88.700443	6.875366	0.331329	11.760232	0.476951
min	1.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	25.000000	105.000000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	32.000000	105.000000	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

Correlation Analysis:- Calculate the correlation matrix to comprehend the linear relationships between numerical features using the correlation analysis method. This matrix can be shown using a heatmap to reveal feature correlations.



With this analysis we see the most occur element individually as well as with output feature ('Outcome'). Through the correlation matrix see the various dependencies of feature

iv. Data Modelling

- Encoding: After all the analysis, next step is to make model, but actually applying the accurate algorithm it is necessary to encode the data through any of the technique in my project I use LabelEncoder.
 - Splitting Data: To apply the model the next step is divide the data b/w input & output variable as I used the supervised learning so that it works upon the labelled data
- Divide data into x (independent variable) & y (dependent variable)

```
1 x=df.drop('Outcome',axis=1) #Independent variable
2 y=df['Outcome'] #Dependent variable
```

After separate the input & output data, now its also necessary to split the into train & test with ratio either 8:2 or 7:3 to get the best result.

For this I import the module of train_test_split from sklearn library.

```
1 from sklearn.model_selection import train_test_split
2 x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.33,random_state=20)
```

- **Creation of Model**

To get the best result I used the different algorithms & also create meta model by combining the random forest, SVM & logistic Regression.

- **Logistic Regression:** For binary classification tasks, a common approach is logistic regression. It simulates the likelihood that a specific instance falls into a specified category.

Syntax: from sklearn.linear_model import LogisticRegression
 lr=LogisticRegression() #lr is object of Logistic Regression

Accuracy: 0.7559055118110236

- **Support Vector Machine:** SVM is a strong classification technique that determines the optimum hyperplane for dividing a dataset into classes. Because of its many kernel functions, it is effective in high-dimensional spaces and adaptable.

Syntax: from sklearn.svm import SVC
 sv = SVC() #sv is object of support vector machine

Accuracy: 0.7755905511811023

- **Random Forest:** Random Forest is a form of ensemble learning that combines many decision trees to produce a more accurate and stable model.

Syntax: from sklearn.ensemble import RandomForestClassifier
 RF = RandomForestClassifier() #RF is object of Random Forest Classifier

Accuracy: 0.78346527

- **ANN:** Artificial neural networks especially deep networks, in particular, are capable of detecting complicated patterns in data. They are made up of interconnected layers of nodes that are trained to represent data attributes.

```
1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras.layers import Dense
3 from tensorflow.keras.layers import Dropout
4
5 # Initialize the model
6 model = Sequential()
7
8 # Add input layer and first hidden layer
9 model.add(Dense(units=128, activation='relu', input_dim=x_train.shape[1]))
10 model.add(Dropout(0.2)) # Dropout layer with a dropout rate of 0.2
11 # Add more hidden layers if needed
12 model.add(Dense(units=64, activation='relu'))
13 model.add(Dropout(0.2)) # Dropout layer with a dropout rate of 0.2
14 # Add the output layer with sigmoid activation for binary classification
15 model.add(Dense(units=1, activation='sigmoid'))
16 model.add(Dropout(0.2)) # Dropout layer with a dropout rate of 0.2
17 # Compile the model
18 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
19 #train
20 # Train the model
21 model.fit(x_train, y_train, epochs=100, batch_size=32, validation_data=(x_test, y_test))
22 yNN_train_pred=model.predict(x_train)
23 yNN_test_pred=model.predict(x_test)
```

Accuracy: 0.7795275449752808

To create neural network, needed to import tensorflow with keras module to make layers. In this one is input layer, one is hidden layer & other is output layer in which I use the sigmoid activation function & ahead two layers use the relu activation function. And also added a dropout layer in b/w all these layers.

- Stacking: The benefits of numerous models are combined in stacking to build a more accurate and robust final model. It makes use of the skills of various algorithms to capture various parts of the data.

Syntax: `from sklearn.ensemble import StackingClassifier`
`stacking_classifier = StackingClassifier(estimators=[(RF,sv)],`
`final_estimator=lr) #stacking_classifier is object of Stacking`

Accuracy: 0.7716535433070866

v. Evaluating model:

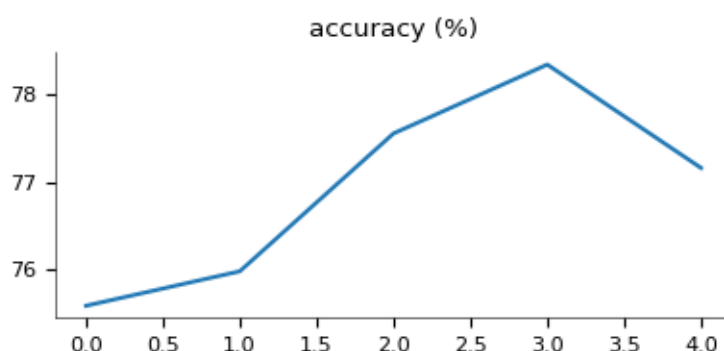
In this project evaluate the model on various parameters like confusion matrix, R squared value & RMSE (Root Mean Squared Error)

Syntax for evaluating

```
print('Train data results')
print('RMSE:', np.sqrt(mean_squared_error(ySC_train_pred, y_train)))
print('R-squared:', r2_score(ySC_train_pred, y_train))
print('Test data results')
print('RMSE:', np.sqrt(mean_squared_error(ySC_test_pred, y_test)))
print('R-squared:', r2_score(ySC_test_pred, y_test))
print("\n Classification Report:\n", classification_report(y_test, ySC_test_pred))
print("Confusion Matrix : \n", confusion_matrix(y_test, ySC_test_pred))
```

By evaluating all the models, the result is

	algo	accuracy (%)
0	Logistic_Regression	75.590551
1	Artificial_Neural_Network	75.984251
2	SVM	77.559055
3	Random_Forest	78.346457
4	Stacking	77.165354



All of the models appear to have nearly the same accuracy. However, Stacking, SVM, and Random Forest are superior since they have fewer false negatives than the others.

i. Testing Output:

To test the output, I use the random forest & stacking both but with new user given data Random Forest performs well on that & give the accurate result.

3	1.0	89	66.0	23.0	94.000	28.1	0.167	21.0	0
4	1.0	137	40.0	35.0	160.625	43.1	1.200	33.0	1
5	5.0	116	74.0	32.0	105.000	25.6	0.201	30.0	0
6	3.0	78	50.0	32.0	88.000	31.0	0.248	26.0	1

```
[ ] 1 #Detect either you are diabetic patient or not
    2 stacking_classifier.predict([[3.0,78, 50.0, 32.0, 88.000, 31.0, 0.248, 26.0 ]])

array([0])
```

```
[ ] 1
```

Result of stacking might be somewhat wrong

Because in dataset its positive result but model show negative

As I gave the 6th row input to get output but stacking result is somewhat wrong because accuracy is not excellent it just a good score.

But when I use Random forest it give appropriate & accurate result, as shown in fig.

```
[ ] 1 from sklearn.preprocessing import StandardScaler
    2 input=(2,197,70,45,543,30.5,0.158,53)
    3 input2array=np.asarray(input)
    4 input_reshape=input2array.reshape(1,-1)
    5 sd_data=scale.transform(input_reshape)
    6 prediction=RF.predict(sd_data)
    7 prediction
    8 if prediction == 0:
    9     print("Patient is not diabetic ")
   10 else:
   11     print("Patient is diabetic ")
   12
```

Patient is diabetic

Issues Faced in model

- ❖ This dataset contains lot of null value & outliers, so to remove them it's a bigger challenge & also to deal with these values.
- ❖ The main thing is to detect the best model because as you in almost all classification models have nearby 77% accuracy & not feasible to deploy in real world scenario, after all the work done & through every research I got to this accuracy.
- ❖ To increase accuracy, we can minimize the test size but that's not be a good choice to get the accurate result.
- ❖ At last through every detection, I got the best one model is Random Forest it's accuracy is slightly lower than stacking & SVM but with user given input Random forest do well & give accurate result then other.

Link of project :

<https://colab.research.google.com/drive/1eJxl9H2hRV3EL8dseThqgtvh5r4Idfb9?usp=sharing>


Results of model

Finally, model detects either a patient is diabetic or not. As I give the value for different

- **No. of Pregnancies = 5**
- **Glucose level = 180**
- **Blood Pressure = 100**
- **Skin thickness level = 60**
- **Insulin level = 570**
- **BMI = 25.5**
- **Diabetes Pedigree Function = 0.128**
- **Age = 40**

Result is:

```
1 from sklearn.preprocessing import StandardScaler
2 input=(5,180,100,60,570,25.5,0.128,40)
3 input2array=np.asarray(input)
4 input_reshape=input2array.reshape(1,-1)
5 sd_data=scale.transform(input_reshape)
6 prediction=RF.predict(sd_data)
7 prediction
8 if prediction == 0:
9     print("Patient is not diabetic ")
10 else:
11     print("Patient is diabetic ")
12
```

 Patient is diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:4:
warnings.warn(