Predicting Diabetes (Classification Project) Using Machine Learning Algorithms

Group No: 4 | Bhavans Vivekananda Degree College

B.VAMSHI | SRIVANI | SRAVAN KUMAR | NITHIN CHANDRA

ABSTRACT

The project focuses on developing a model that accurately predicts diabetes in a person, leading to early prevention and care.

The study explores machine learning algorithms like Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machines, and Boosting to predict diabetes.

OBJECTIVE

To find the suitable machine learning model to predict diabetes in a person for giving proper medication and early prevention.

CONTENTS

Introduction

•Diabetes, a growing health crisis, demands early detection for effective management.

•Traditional diagnosis methods are often time-consuming and resource-intensive.

•Machine learning offers a promising solution by analyzing patient data to predict diabetes risk.

This project aims to leverage machine learning algorithms to develop a predictive model to assist healthcare professionals


Literature
Review

Literature Review-1

12th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, 2021.

Md.Mehedi Hassan et al. – Utilized Unsupervised and supervised approaches which yielded 99.57% and 99.03% accuracy for random forest respectively.

Literature Review-2

"Diabetes Detection Based on Machine Learning and Deep Learning Approaches" - Published by Multimedia Tools and Applications in August 2023, this review investigates the impacts of machine learning and deep learning approaches in diabetes identification/classification using non-invasive or anthropometric measurements.

Literature Review-3

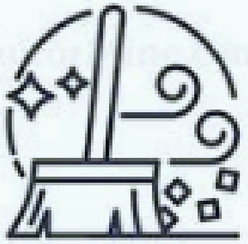
"Advances in Artificial Intelligence for Diabetes Prediction: Insights from a Systematic Literature Review" - Published by arXiv in December 2024, this review explores the use of machine learning in predicting diabetes, focusing on datasets, algorithms, training methods, and evaluation metrics.

Literature Review-4

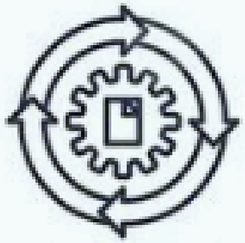
Jobeda Jamal Khanam, Simon Y. Foo -ICT Express, Volume 7, Issue 4, 2021,

Jobeda et al.– Analysed the diabetes data using an Artificial Neural Network with three hidden layers and 400 epochs, providing an accuracy of 88.6%

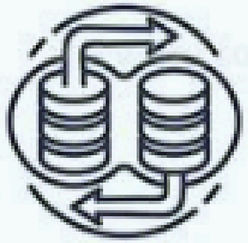
DATA PREPROCESSING



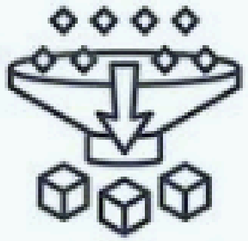
REMOVING NOISE



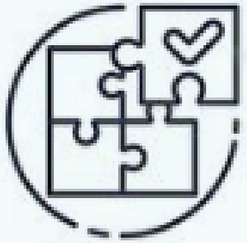
STANDARDIZATION



COMBINING SOURCES



SIMPLIFICATION



HANDLING MISSING VALUES

DATA

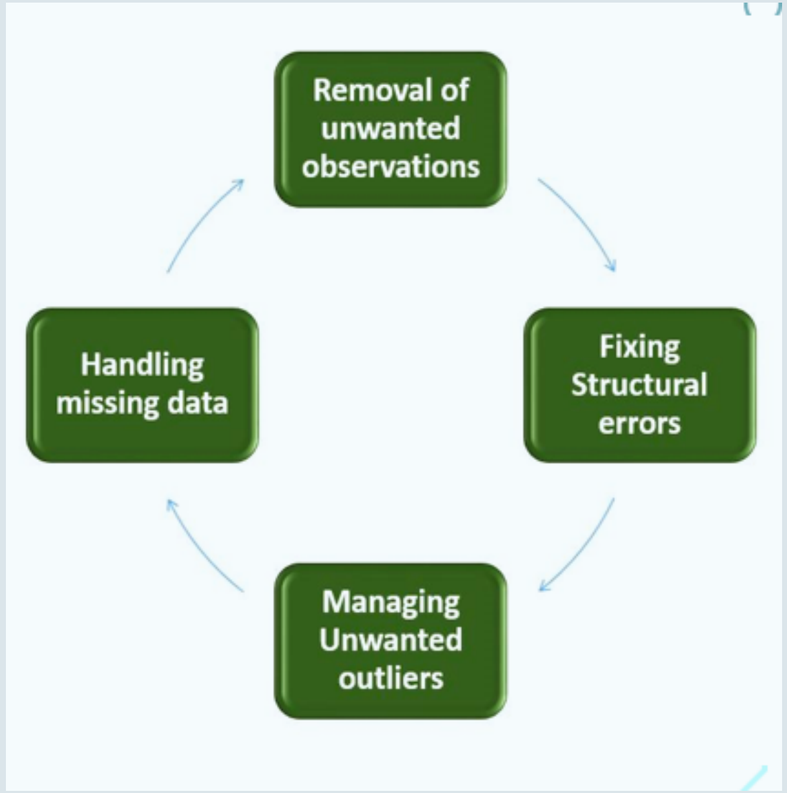
Dataset: Our Dataset consists of 9 variables and 769 records

source *https://drive.google.com/drive/folders/1vGSRCnhqSxEH53BgLqhh32F1qNKqfOim?usp=drive_link*

	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

DATA CLEANING

Data cleaning, also known as data cleansing, data scrubbing, data wrangling, or data munging, is the process of identifying and correcting inaccurate, incomplete, or irrelevant data in a dataset



•Checked for missing values and unique values.

•Only for EDA:

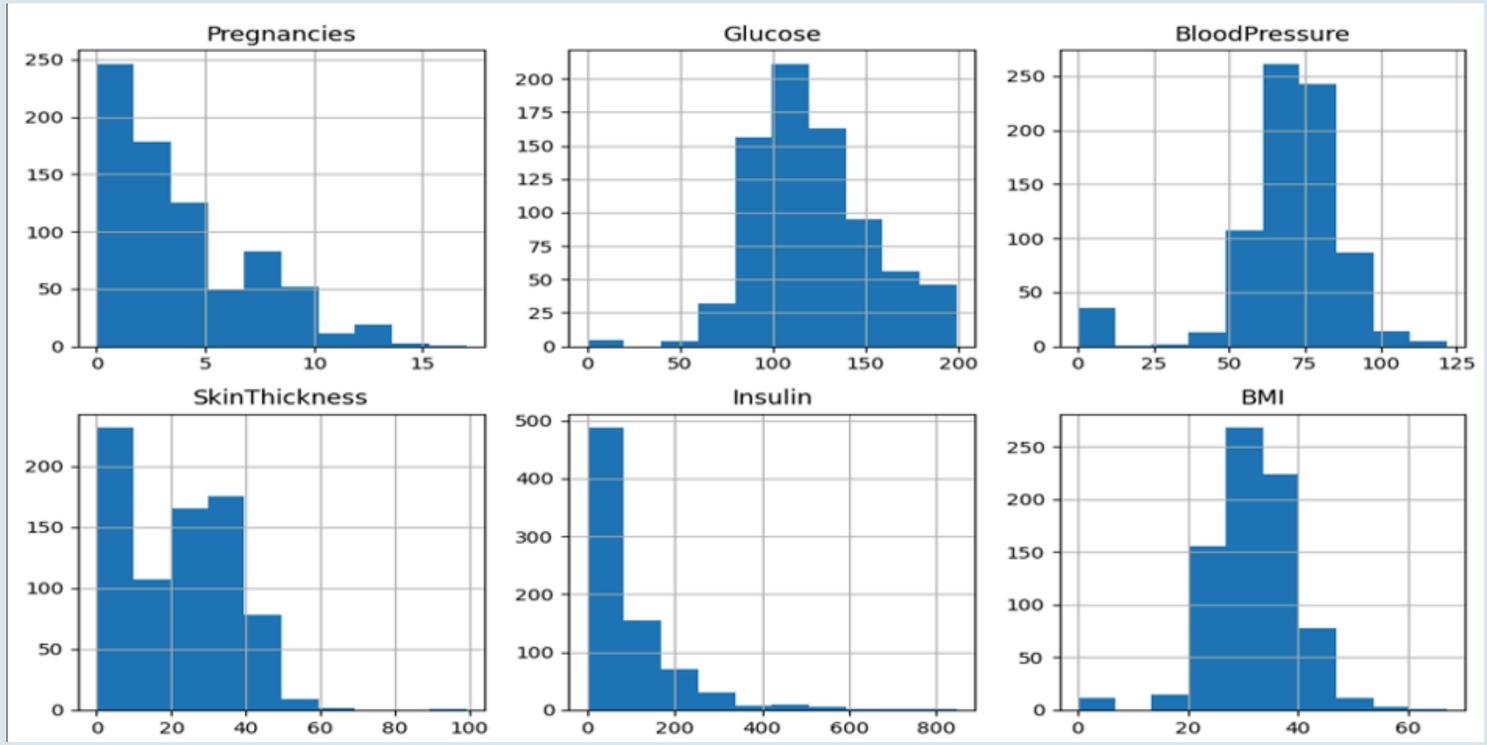
•Encoded the BMI variable: Enabled “Overweight” as 1 and “Normal Weight” as 0

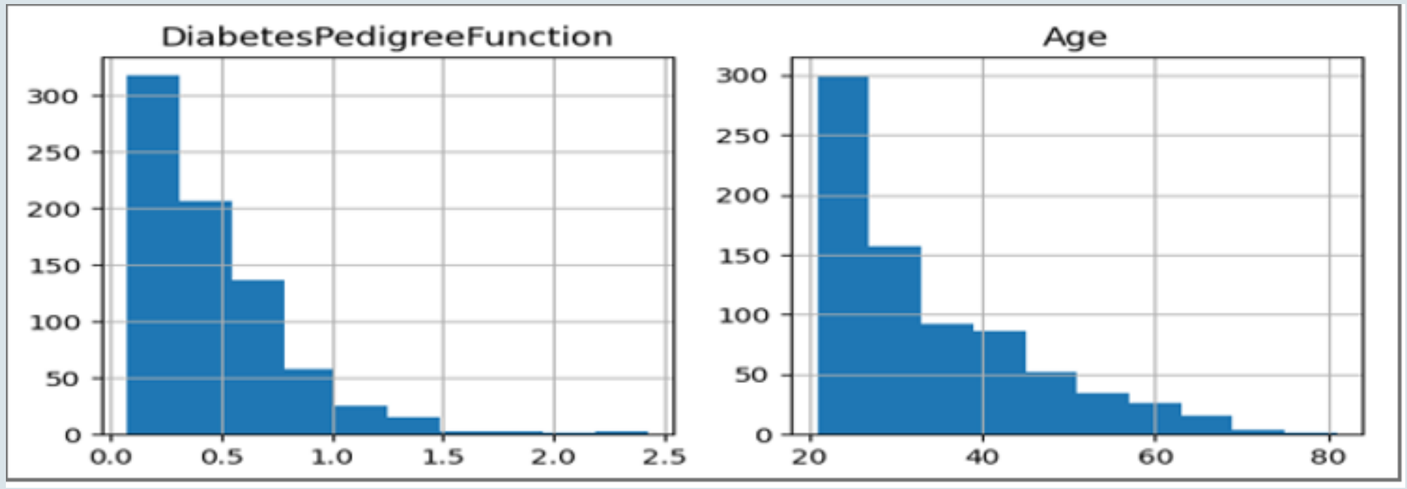
•Age has been converted into two categories: “Less than 60 years” and “Greater than 60 years”.

EXPLORATORY DATA ANALYSIS

HISTOGRAM

a histogram is a graphical representation of the distribution of data.

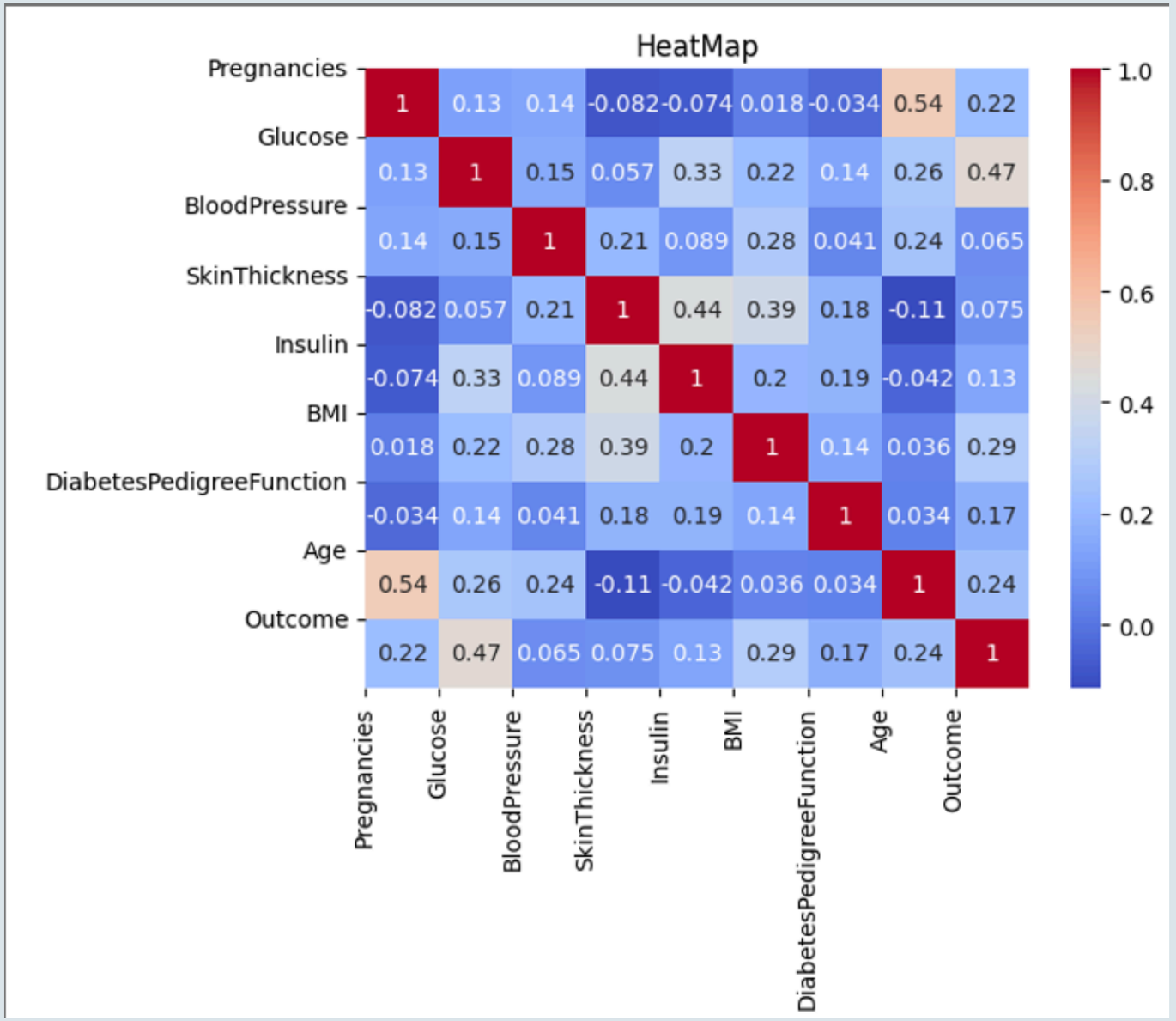




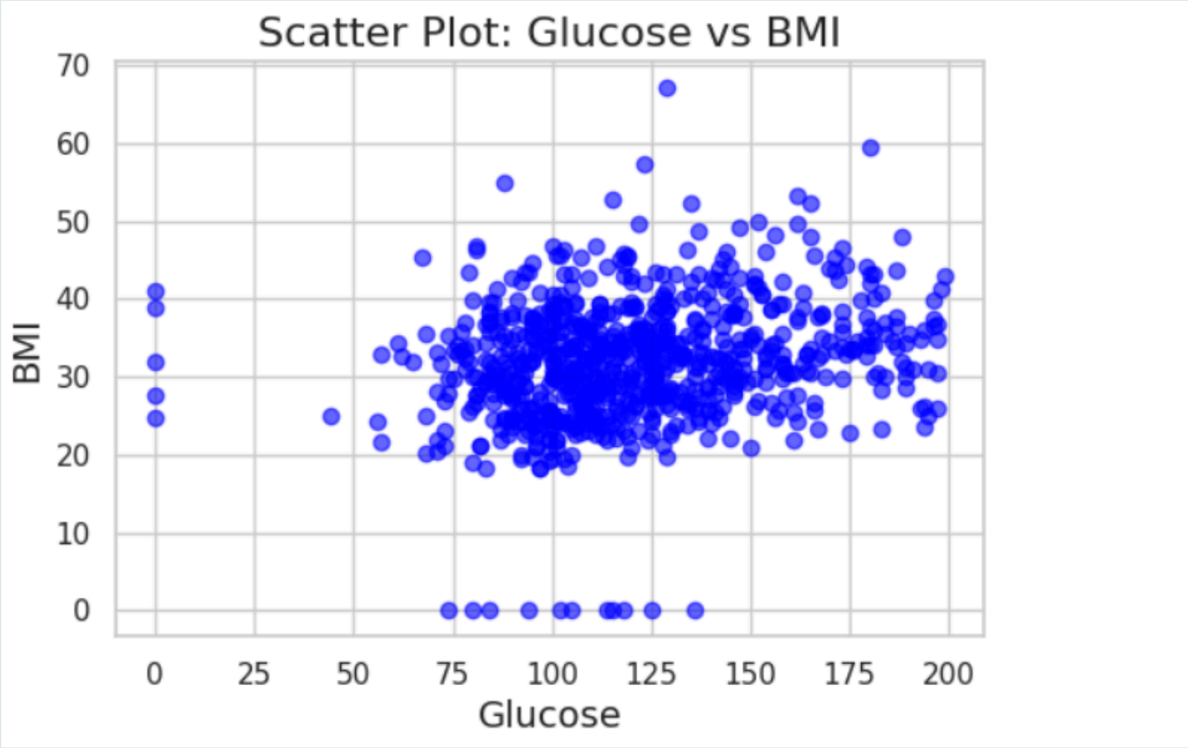
HEAT MAP

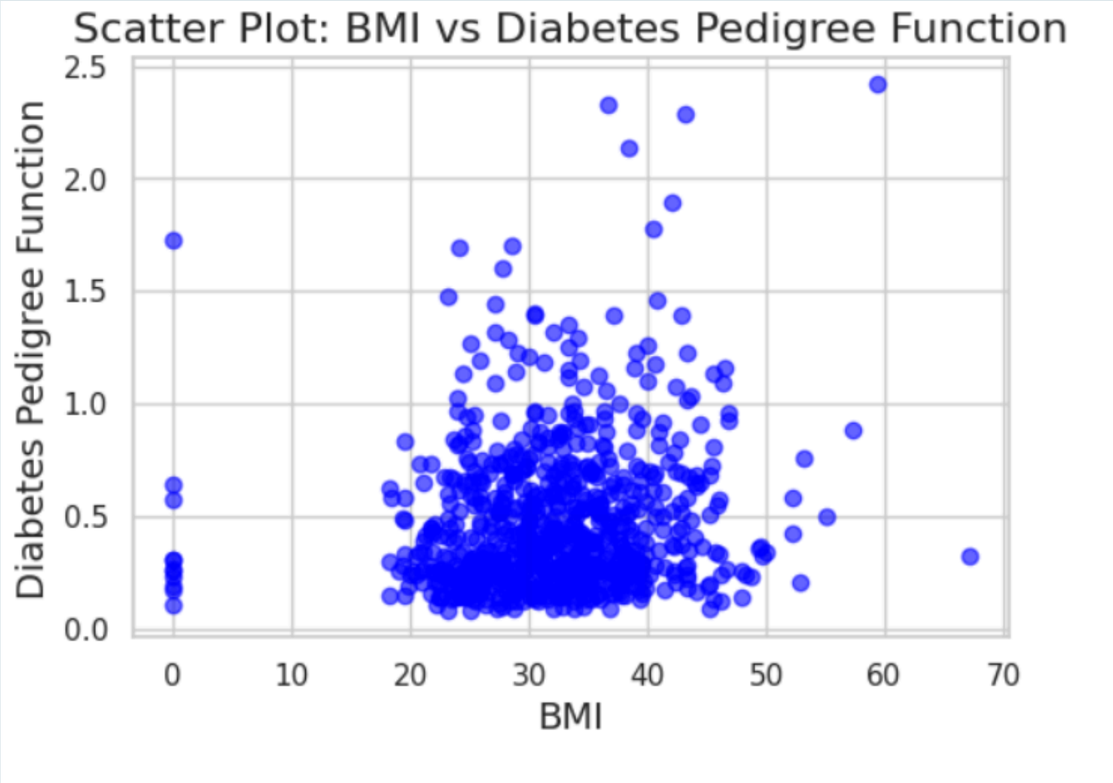
all variables are

moderately correlated.

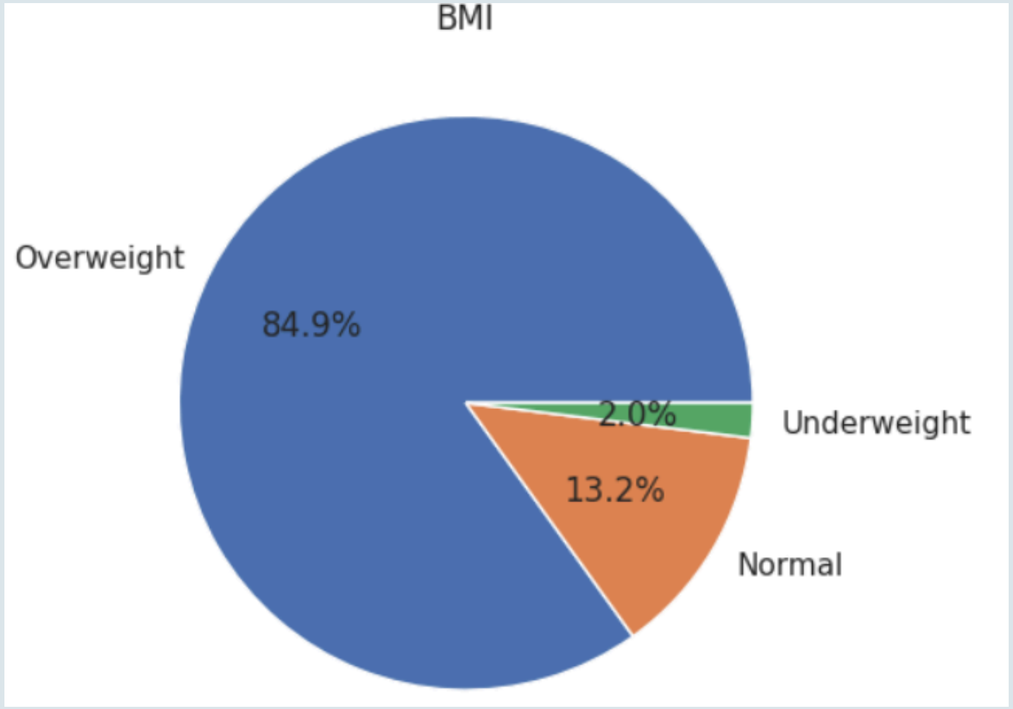


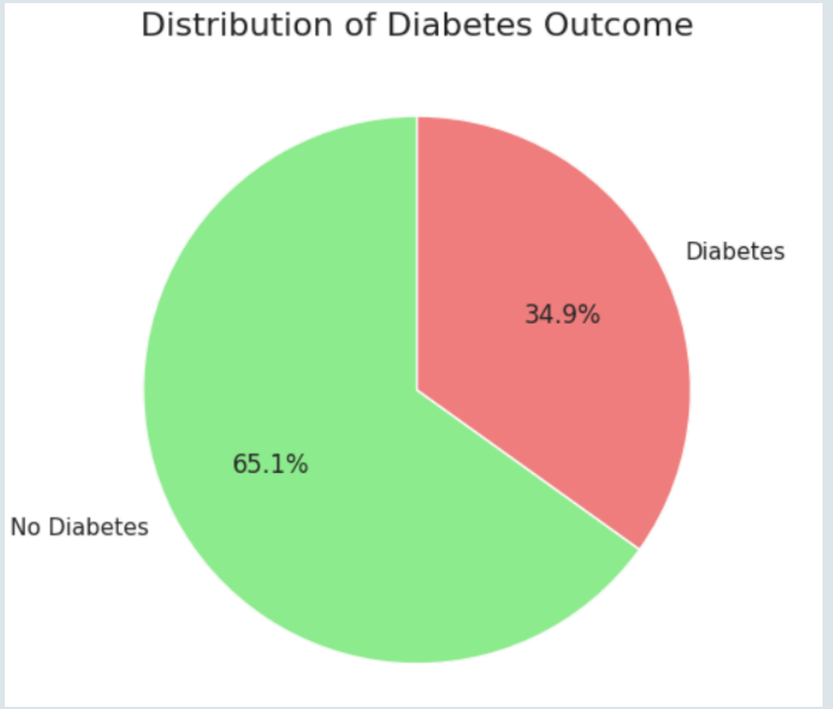
SCATTER PLOT





PIE CHART


BMI
Overweight
84.9%
2.0%
Underweight
13.2%
Normal


Distribution of Diabetes Outcome
Diabetes
34.9%
65.1%
No Diabetes

MULTICOLLINEARITY CHECK

MACHINE LEARNING ALGORITHMS

- Logistic Regression
- K Nearest Neighbour
- Support Vector Machine
- Decision Tree
- Boosting(XG Boosting, Adaptive Boosting)
- Random Forest

Logistic Regression

Logistic Regression in machine learning is a statistical method used for binary classification tasks, where the goal is to predict one of two possible outcomes(yes or no; 1 or 0; True or false).

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.779	0.656
75-25	0.776	0.656
70-30	0.784	0.654
60-40	0.750	0.666

K-Nearest Neighbour

K-Nearest Neighbors is a supervised learning algorithm which makes predictions based on the similarity between a new data point and the existing data points in the training dataset

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.753	0.617
75-25	0.771	0.625
70-30	0.771	0.632
60-40	0.740	0.620

Support Vector Machine

Support Vector Machine is a supervised classification algorithm that finds the optimal decision boundary (hyperplane) that maximizes the margin between different classes.

There are two types of SVM's - Linear SVM; Non-Linear SVM

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.779	0.688
75-25	0.781	0.677
70-30	0.792	0.688
60-40	0.773	0.662

Decision Tree

A Decision Tree is a supervised machine learning algorithm that models decisions and their possible consequences as a tree-like structure, where each node represents a decision based on a feature and each branch represents the outcome of that decision

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.786	0.656
75-25	0.776	0.677
70-30	0.758	0.671
60-40	0.731	0.675

Boosting (XG Boost)

XG Boost (Extreme Gradient Boosting) is an advanced, efficient version of gradient boosting that builds a strong model by combining many weak decision trees and optimizes performance through techniques like regularization, parallelization, and handling missing values

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.792	0.662
75-25	0.771	0.667
70-30	0.758	0.649
60-40	0.770	0.633

Boosting (Adaptive Boosting)

Adaptive Boosting is a machine learning algorithm that improves the accuracy of weak learners by focusing on the data points that are hardest to classify. It works by sequentially training weak models, adjusting the weights of misclassified data points, and combining the predictions of all models into a stronger classifier.

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.790	0.688
75-25 .	0.760	0.698
70-30	0.753	0.675
60-40	0.744	0.698

Random Forest

Random Forest is an ensemble learning algorithm that combines multiple decision trees to create a more accurate and stable model. It reduces overfitting by using random samples of data and features to build each tree and by averaging (or voting) across all trees for final predictions

Ratio	Model 1 (Before VIF) (Accuracy Score)	Model 2 (After VIF) (Accuracy Score)
80-20	0.799	0.656
75-25 .	0.786	0.646
70-30	0.788	0.662
60-40	0.788	0.656

Artificial Neural Network

SPLIT	Architecture	Optimizer	Epochs	Accuracy
80-20	128-64-32-1	SDG	500	nan
80-20	128-64-32-1	Adam	500	0.769
80-20	128-64-32-1	Adam	250	0.655
75-25	128-64-32-1	SGD	500	nan
75-25	128-64-32-1	Adam	500	0.764
75-25	128-64-32-1	Adam	500	0.672
70-30	128-64-32-1	SGD	250	nan
70-30	128-64-32-1	Adam	80	0.818
70-30	128-64-32-1	Adam	80	0.068
60-40	128-64-32-1	SGD	500	nan
60-40	128-64-32-1	Adam	250	0.695
60-40	128-64-32-1	Adam	250	0.652

FINDING THE BEST SPLIT

Algorithm	Before VIF (Highest Accuracy Score)		After VIF (Highest Accuracy Score)	
Logistic Regression	0.779	(80-20)	0.666	(60-40)
K-Nearest Neighbour	0.771	(80-20, 75-25)	0.632	(70-30)
Support Vector Machine	0.792	(70-30)	0.688	(70-30)
Decision Tree	0.786	(80-20)	0.677	(75-25)
XG Boost	0.792	(80-20)	0.667	(75-25)
Adaptive Boosting	0.790	(80-20)	0.698	(75-25)
Random Forest	0.779	(80-20)	0.662	(70-30)
Artificial Neural Network	0.818	(70-30)	0.681	(70-30)
	70-30 is the best split "Before VIF"		**75-25 is the best split "After VIF"**	

FINDING THE BEST ALGORITHM

Algorithm	Before VIF(Highest Accuracy Score)	After VIF (Highest Accuracy Score)
Logistic Regression	0.779	0.657
K-Nearest Neighbour	0.766	0.625
Support Vector Machine	0.779	0.680
Decision Tree	0.770	0.703
XG Boost	0.776	0.682
Adaptive Boosting	0.750	0.700
Random Forest	0.800	0.660
Artificial Neural Network	0.818	0.672

Artificial Neural Network Algorithm has the highest accuracy

Conclusion

In this diabetes prediction ML project, insights from data preprocessing, model selection, evaluation metrics, and deployment are essential for building an effective tool for diabetes prediction. By applying machine learning techniques, the model can aid healthcare professionals in early detection, which can lead to better management and treatment of diabetes. Future work can enhance the model's capabilities through more advanced techniques, real-time data integration, and careful consideration of ethical concerns.

FUTURE SCOPE

- The future of diabetes prediction research is very promising, with numerous opportunities to leverage cutting-edge technologies such as artificial intelligence, big data, and genomics. The integration of personalized healthcare, real-time monitoring, and ethical AI models will revolutionize diabetes management and prevention. Ultimately, the goal is to improve early detection, reduce complications, and help individuals live healthier lives through more accurate and individualized predictions and interventions.

Work Distribution

Team Member	Tasks
VAMSHI	Introduction and Literature Review
SRIVANI	Data Pre-Processing and EDA
SRAVAN KUMAR	Data Modelling and Evaluation
NITHIN CHANDRA	Algorithm Comparison & Summary

THANKYOU

Colab Notebook

Done By

Sravan kumar(107222546009)

Vamshi(107222546010)

Srivani (107222546011)

Nithin chandra (107222546012)

https://colab.research.google.com/drive/1TGwZXRQxxOZuHYt5VuDCR1FRAjR0LHvR?usp=sharing

Appendix

IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

uploading files to colab

```
[ ] from google.colab import files
    uploaded=files.upload()
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving diabetes.csv to diabetes.csv
```

reading the files

DATA PREPROCESSING

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

checking null values

```
data.isna().sum()
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

```
data.shape
```

```
(768, 9)
```

```
data.shape[1]
```

```
9
```

```
for i in range(data.shape[1]):
  print(i)
```

```
0
1
2
3
4
5
6
7
8
```

finding out unique values for each column and counting them

```
for i in range(data.shape[1]):
  print(data.iloc[:,i].unique())
  print(data.iloc[:,i].value_counts())
```

```
[ 6  1  8  0  5  3 10  2  4  7  9 11 13 15 17 12 14]
Pregnancies
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
```

```
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
...	...	...	...	...	...	...	...	...	...
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

```
data2=data.copy(deep=True)
```

No categorical data so no dummies

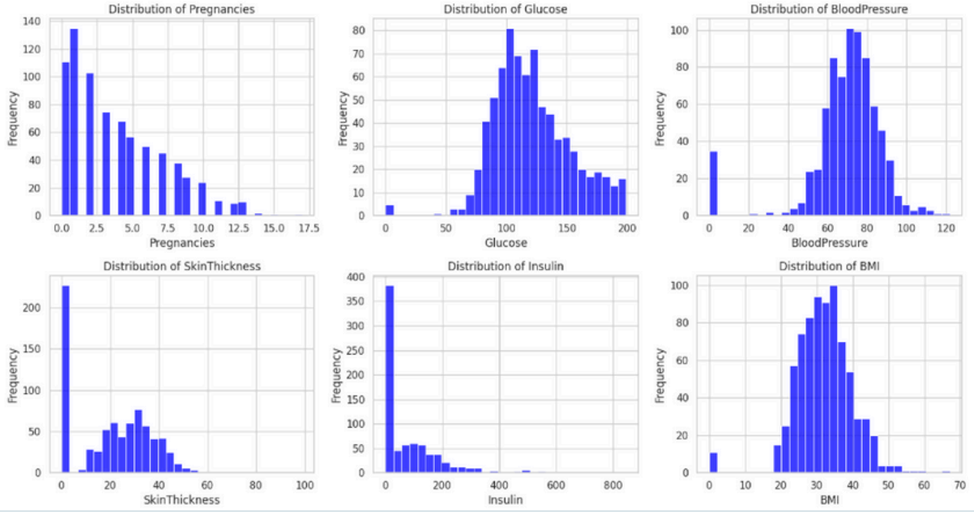
```
#Feature Variables
X=data.drop(['Outcome'],axis=1)
print(X)

#Target Variable
y=data['Outcome']
print(y)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6   
1              1       85             66             29        0  26.6   
2              8      183             64              0        0  23.3   
3              1       89             66             23       94  28.1   
4              0      137             40             35      168  43.1   
..           ...      ...            ...            ...      ...   ...   
763           10      101             76             48      180  32.9   
764            2      122             70             27        0  36.8   
765            5      121             72             23      112  26.2   
766            1      126             60              0        0  30.1   
767            1       93             70             31        0  30.4   

     DiabetesPedigreeFunction  Age  
0                       0.627   50  
1                       0.351   31  
2                       0.672   32  
3                       0.167   21  
4                       2.288   33  
..                        ...  ...  
763                     0.171   63  
764                     0.340   27  
765                     0.245   30  
766                     0.349   47  
767                     0.315   23  

[768 rows x 8 columns]
0      1
1      0
2      1
3      0
4      1
```

EXPLORATORY DATA ANALYSIS

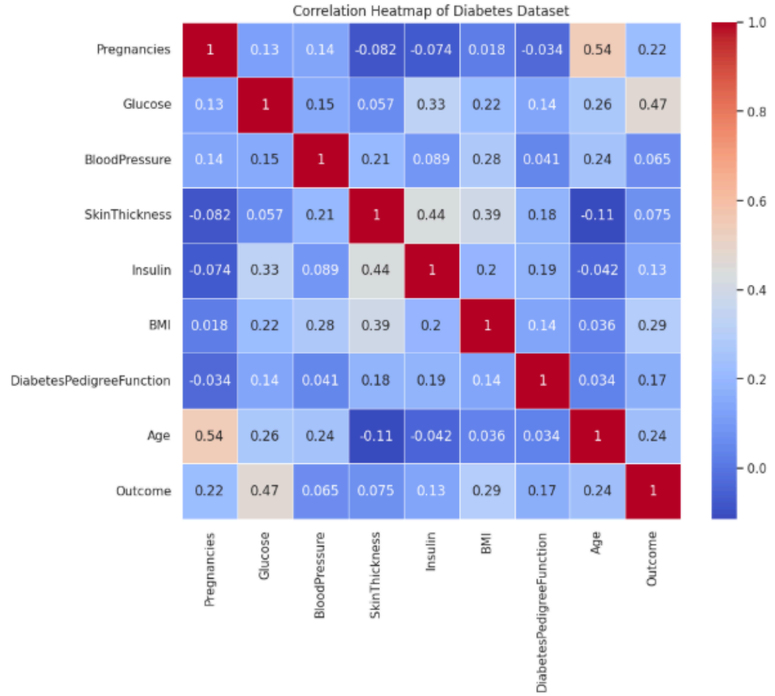
Distribution of each feature

```
columns = data.columns

sns.set(style="whitegrid")

plt.figure(figsize=(15, 12))
for i, column in enumerate(columns, 1):
    plt.subplot(3, 3, i)
    sns.histplot(data[column], kde=False, bins=30, color='blue')
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')


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



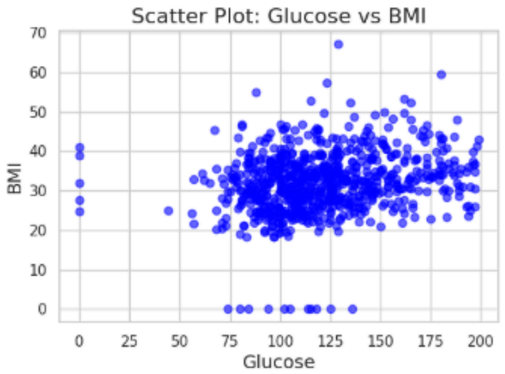

```
corr_matrix = data.corr()
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap of Diabetes Dataset')
plt.show()
```



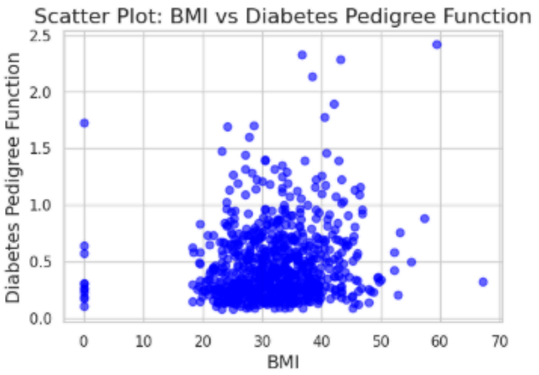
Scatter plot of Glucose vs BMI

```
[ ] plt.figure(figsize=(6, 4))
    plt.scatter(data['Glucose'], data['BMI'], color='blue', alpha=0.6)

    plt.xlabel('Glucose', fontsize=14)
    plt.ylabel('BMI', fontsize=14)
    plt.title('Scatter Plot: Glucose vs BMI', fontsize=16)

    plt.show()
```



Scatter plot (BMI vs DiabetesPedigreeFunction)

```
plt.figure(figsize=(6, 4))
plt.scatter(data['BMI'], data['DiabetesPedigreeFunction'], color='blue', alpha=0.6)

plt.xlabel('BMI', fontsize=14)
plt.ylabel('Diabetes Pedigree Function', fontsize=14)
plt.title('Scatter Plot: BMI vs Diabetes Pedigree Function', fontsize=16)

plt.show()
```



Logistic Regression BEFORE MULTICOLLINEARITY CHECKING

```
X = data2.drop('Outcome', axis=1)  # Select all columns except 'Outcome'
y = data2['Outcome']  # Select the 'Outcome' column

# 2. Ensure X and y have the same number of samples
print(X.shape)
print(y.shape)


from sklearn.model_selection import train_test_split

X_train1,X_test1,y_train1,y_test1=train_test_split(X,y,test_size=0.20,train_size=0.80)
X_train2,X_test2,y_train2,y_test2=train_test_split(X,y,test_size=0.25,train_size=0.75)
X_train3,X_test3,y_train3,y_test3=train_test_split(X,y,test_size=0.30,train_size=0.70)
X_train4,X_test4,y_train4,y_test4=train_test_split(X,y,test_size=0.40,train_size=0.60)
```

```
(768, 8)
(768,)
```

80 - 20 SPLIT

```
logreg = LogisticRegression(C=1e9)
logreg.fit(X_train1, y_train1)
predictions = logreg.predict(X_test1)
print(predictions)
```

```
[0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0
 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1
 0 0 0 0 0 0]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import confusion_matrix
z=confusion_matrix(y_test1, predictions)
z
```

```
array([[89, 12],
       [26, 27]])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test1,predictions)
```

```
0.7532467532467533
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test1,predictions))
```

```
              precision    recall  f1-score   support

           0       0.77      0.88      0.82       101
           1       0.69      0.51      0.59        53

    accuracy                           0.75       154
   macro avg       0.73      0.70      0.71       154
weighted avg       0.75      0.75      0.74       154
```

MULTICOLLINEARITY checking using VIF

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return(vif)


calc_vif(X)
```

	variables	VIF
0	Pregnancies	3.275748
1	Glucose	16.725078
2	BloodPressure	14.619512
3	SkinThickness	4.008696
4	Insulin	2.063689
5	BMI	18.408884
6	DiabetesPedigreeFunction	3.195626
7	Age	13.492985

```
calc_vif(X.drop('BMI', axis=1))
```

	variables	VIF
0	Pregnancies	3.272901
1	Glucose	13.573749
2	BloodPressure	12.372453
3	SkinThickness	3.475236
4	Insulin	2.033589
5	DiabetesPedigreeFunction	3.151598
6	Age	13.381319

```
calc_vif(X.drop(['BMI','Glucose'], axis=1))
```

	variables	VIF
0	Pregnancies	3.266130
1	BloodPressure	9.940457
2	SkinThickness	3.475014
3	Insulin	1.859553
4	DiabetesPedigreeFunction	3.026709
5	Age	10.896809

KNN before MULTICOLLINEARITY checking

```
from sklearn.neighbors import KNeighborsClassifier
```

```
model=KNeighborsClassifier(n_neighbors=20)
```

80 - 20 SPLIT

```
model.fit(X_train1,y_train1)
```

```
KNeighborsClassifier(n_neighbors=20)
```

```
y_pred1=model.predict(X_test1)
y_pred1
```

```
array([0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1])
```

```
knn =pd.DataFrame({'Actual':y_test1,'Predicted':y_pred1})
knn
```

	Actual	Predicted
91	0	0
45	1	1
628	0	0
201	0	0
329	0	0
...	...	...
505	0	0
532	0	0
533	0	0
199	1	0
166	0	1

154 rows × 2 columns

Evaluation Metric

```
accuracy_score(y_test1,y_pred1)
```

```
0.7402597402597403
```

SVM before MULTICOLLINEARITY checkig

```
from sklearn.svm import SVC
```

```
model = SVC(kernel='linear')
```

80 - 20 SPLIT

```
model.fit(X_train1, y_train1)
```

```
SVC(kernel='linear')
```

```
y_pred1 = model.predict(X_test1)
y_pred1
```

```
array([0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0])
```

```
svm = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
svm
```

	Predicted	Actual
91	0	0
45	1	1
628	0	0
201	1	0
329	0	0
...	...	...
505	0	0
532	0	0
533	0	0
199	0	1
166	0	0

154 rows × 2 columns

```
accuracy_score(y_test1,y_pred1)
```

```
0.7467532467532467
```

```
cm = confusion_matrix(y_test1,y_pred1)
cm
```

```
array([[87, 14],
       [25, 28]])
```

KNN after MULTICOLLINEARITY checking

80 - 20 SPLIT

```
model.fit(X_train1_nomulti, y_train1)
```

```
SVC
SVC(kernel='linear')
```

```
y_pred1 = model.predict(X_test1_nomulti)
y_pred1
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
knn = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
knn
```

	Predicted	Actual
668	0	0
324	0	0
624	0	0
690	0	0
473	0	0
...	...	...
355	0	1
534	0	0
344	0	0
296	0	1
462	0	0

154 rows × 2 columns

Evaluation Metric

```
accuracy_score(y_test1,y_pred1)
```

```
0.6428571428571429
```

```
cm = confusion_matrix(y_test1,y_pred1)
cm
```

```
array([[99,  0],
       [55,  0]])
```

SVM after MULTICOLLINEARITY checking

```
from sklearn.svm import SVC
```

```
model1 = SVC(kernel='linear')
```

80 - 20 SPLIT

```
model1.fit(X_train1_nomulti, y_train1)
```

```
SVC
SVC(kernel='linear')
```

```
y_pred1 = model1.predict(X_test1_nomulti)
y_pred1
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
svm = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
svm
```

	Predicted	Actual
668	0	0
324	0	0
624	0	0
690	0	0
473	0	0
...	...	...
355	0	1
534	0	0
344	0	0
296	0	1
462	0	0

154 rows × 2 columns

```
accuracy_score(y_test1,y_pred1)
```

```
0.6428571428571429
```

```
cm = confusion_matrix(y_test1,y_pred1)
cm
```

```
array([[99,  0],
       [55,  0]])
```

Thank you