PROJECTREPORT

ON

DIABETICS PREDICTION

Submitted as a part of Curriculum of

Bachelor of Technology

In

Computer Science Engineering

By

Under the guidance of

Mr. D. KRISHNA

Associate professor

Department of Computer Science and Engineering

ACE Engineering College

(An Autonomous Institution)

NBA ACCREDITEDB.TECH COURSES: EEE, ECE, CSE & MECH

Ankushapur (V),Ghatkesar(M), Medchal.Dist.-501301

(Affiliated to Jawaharlal Nehru Technological University Hyderabad 2021-2025)

ACE
Engineering College
(with a Difference in Excellence)

CERTIFICATE

This is to certify that the project work entitled DIABETICS PREDICTION is being submitted by G S Madhu Bala (21AG1A05E8), Yadaram Harivardhan Reddy(21AG1A05J2), Arsha Sultana (21AG1A05D2), Vubbani Jagadish (21AG1A05J1), A Arun Kumar Reddy (21AG1A05D0), Varkala Akshitha (21AG1A05J0), Yalla Sushmitha Reddy (21AG1A05J3), G Ranjith Kumar (21AG1A05E7), Adusumilli Nikitha (21AG1A05D1), Duvvala Lavanya (21AG1A05E6) as a part of Curriculum of Degree of Bachelor of Technology in Computer Science and Engineering to the ACE Engineering College during the academic year 2021- 2025 is a record of bonafide work carried out by the under our guidance and supervision.

Internal Guide	**Head of Department**
Mr.D.Krishna	**Dr.M.V.VIJAYASARADHI**
Associate Professor	**Professor and Head of the Dept CSE**

ACKNOWLEDGEMENT

We would like to express our gratitude to all the people behind the screen who havehelped us to transform an idea into a real time application. We would like to expressourheart-felt gratitude to our parents without whom we would not have beenprivileged toachieveandfulfillourdreams.

A special thanks to our Secretary, **Prof. Y. V. GOPALA KRISHNA MURTHY**, for having founded such an esteemed institution. We are also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project. We profoundly thank **Dr. M.V. VIJAYA SARADHI**, Head of the Department of Computer Science & Engineering.

We are very thankful to our guide **D. KRISHNA**, Associate Professor who has been an excellent and also given continuous support for the completion of our project work. The satisfaction and euphoria the accompany the successful completion of the task would be great, but in complete without the mention of the people who made it possible, whose guidance and encouragement crown all the efforts with success. In this context, we would like to thank all the other staff members, both teaching and non-teaching, which have extended their timely help and easier our task.

G S Madhu Bala (21AG1A05E8)

Yadaram Harivardhan Reddy (21AG1A05J2)

Arsha Sultana (21AG1A05D2)

Vubbani Jagadish (21AG1A05J1)

A Arun Kumar Reddy (21AG1A05D0)

Varkala Akshitha (21AG1A05J0)

Yalla Sushmitha Reddy (21AG1A05J3)

G Ranjith Kumar (21AG1A05E7)

Adusumilli Nikitha (21AG1A05D1)

Duvvala Lavanya (21AG1A05E6)

DECLARATION

We hereby declare that project entitled “**Diabetes Prediction**” submitted as a part of Curriculum of Bachelor of Technology in Computer Science and Engineering. This dissertation is our original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles and no part of it has been published or sent for the publication at the time of submission.

G S Madhu Bala (21AG1A05E8), Yadaram Harivardhan Reddy (21AG1A05J2), Arsha Sultana (21AG1A05D2), Vubbani Jagadish (21AG1A05J1), A Arun Kumar Reddy (21AG1A05D0), Varkala Akshitha (21AG1A05J0), Yalla Sushmitha Reddy (21AG1A05J3), G Ranjith Kumar (21AG1A05E7), Adusumilli Nikitha (21AG1A05D1), Duvvala Lavanya (21AG1A05E6)

ABSTRACT

Diabetes: is a chronic disease that affects millions of people worldwide. Early diagnosis and treatment can prevent or delay the onset of complications such as blindness, kidney failure, and cardiovascular diseases. However, many people with diabetes are unaware of their condition or do not have access to adequate health care. Therefore, there is a need for developing effective and efficient methods to identify and monitor people at risk of diabetes.

In this project, we use logistic regression, a supervised machine learning algorithm, to predict whether a person has diabetes or not based on various features such as age, blood pressure, glucose level, body mass index, etc. We use Python and popular libraries such as Pandas, Scikit-Learn, and Matplotlib to perform data analysis, model building, and evaluation. The results show that logistic regression achieves an accuracy of 80% on the test data.The project demonstrates the potential of using logistic regression to assist in the diagnosis and management of diabetes using Python.

INDEX

1.1 : INTRODUCTION

Diabetes causes a large number of deaths each year and a large number of people living with the disease do not realize their health condition early enough.

Diabetes prediction is a significant area in healthcare, where machine learning and data science techniques can make a real difference.

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Linear Regression Equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_n X_n$$

Where, y is a dependent variable and x1, x2 ... and Xn are explanatory variables.

Sigmoid Function:

$$p = \frac{1}{1 + e^{-y}}$$

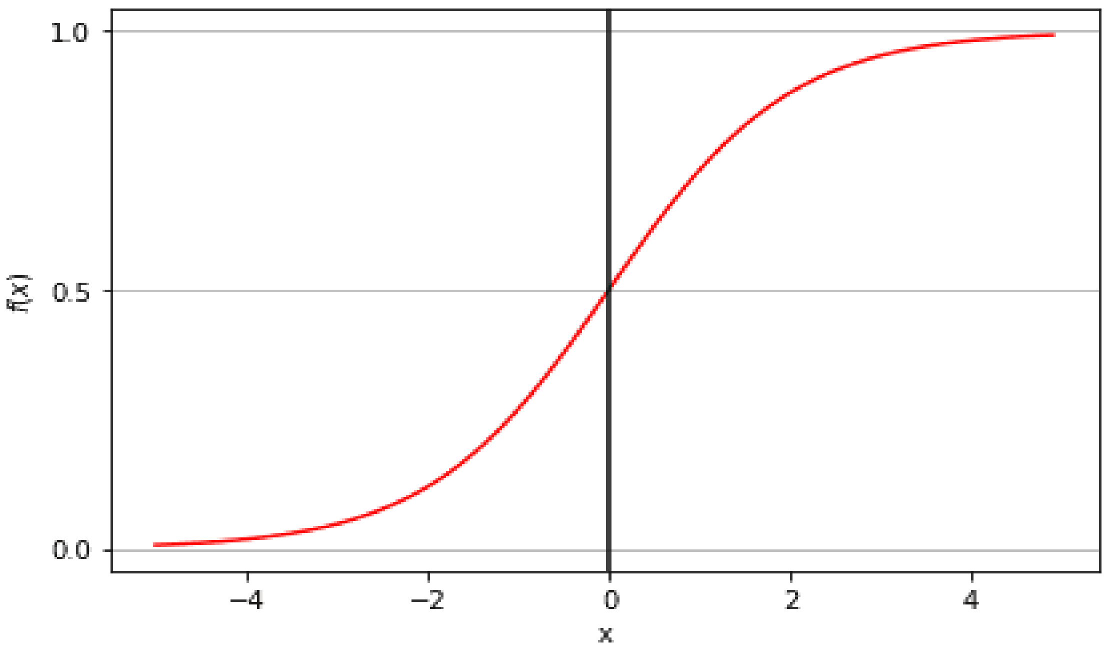
Apply Sigmoid function on linear regression:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_n X_n)}}$$

Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.

- No R Square, Model fitness is calculated through Concordance, KS-Statistics.



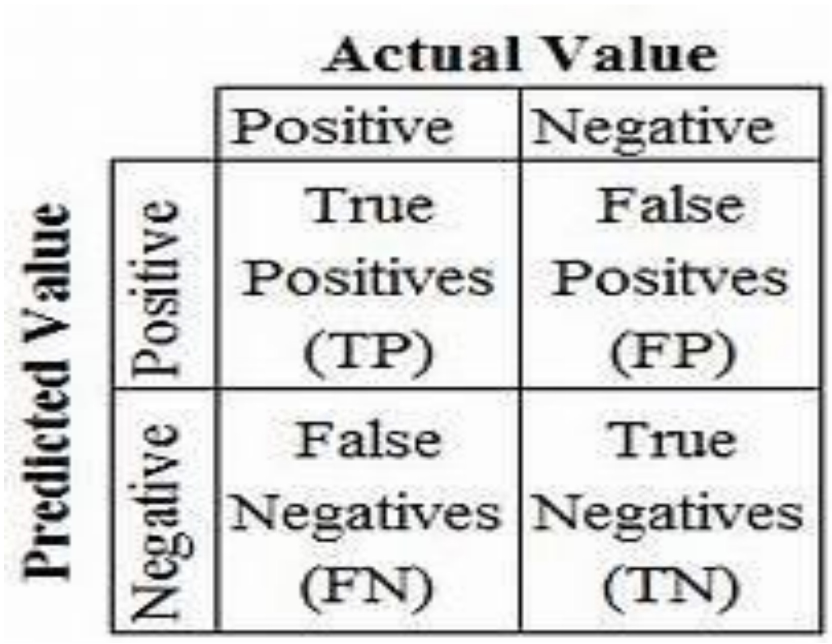
1.2: Application

Logistic regression is a powerful technique for binary classification problems, where the goal is to estimate the probability that an observation belongs to one of two possible classes. It is widely applied in various domains, such as machine learning, medicine, and social sciences. For instance, the Revised Trauma Score (RTS), which is commonly used to assess the severity of trauma in injured patients, was derived using logistic regression.[16] Many other medical tools used to diagnose or predict diseases (e.g. cancer; stroke) are based on logistic regression models that use features such as age, gender, weight, blood tests, etc. Another example could be to predict whether a voter in the United States will vote for the Democratic or Republican party, based on factors such as education, income, gender, ethnicity, state, previous voting behavior etc.

The method can also be employed in engineering, for example, to predict the likelihood of a component or a system failing or malfunctioning. It is also useful in marketing applications, such as predicting whether a consumer will buy or not buy a product, or whether they will continue or cancel a subscription, etc. In economics, it can be used to predict the probability of a person being employed or unemployed, and in business, it can be used to predict the probability of a borrower repaying or defaulting on a loan. Hidden Markov models, a generalization of logistic regression to sequential data, are used in speech recognition and natural language processing.

1.3 : Confusion matrix

A confusion matrix is a table that summarizes the performance of a classification model on a set of test data. It shows how many instances of each class were correctly or incorrectly predicted by the model. A confusion matrix can help evaluate the accuracy, precision, recall, and other metrics of a model.



		Actual Value	
		Positive	Negative
Predicted Value	Positive	True Positives (TP)	False Positves (FP)
	Negative	False Negatives (FN)	True Negatives (TN)

TP stands for true positive, FN stands for false negative, FP stands for false positive, and TN stands for true negative. These terms indicate whether the model's prediction matched the actual label of the instance or not.

The image shows the number of instances of each class that were predicted as each class by the model. The diagonal entries show the correct predictions, while the off-diagonal entries show the incorrect predictions. The image also shows the accuracy, precision, and recall for each class, as well as the overall accuracy of the model.

2: DATA PREPARATION

The data was collected and made available by "National Institute of Diabetes and Digestive and Kidney Diseases" as part of the Pima Indians Diabetes Database.

Dataset link: https://www.kaggle.com/uciml/pima-indians-diabetes-database

The objective of the dataset is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

Target variable or Dependent variable: Outcome(0 if non-diabetic, 1 if diabetic)

- Predictors or Independent variable:

✓ Pregnancies: Number of times the patient was pregnant.

✓ Glucose: Plasma glucose concentration over two hours in an oral glucose tolerance test.

✓ **Blood Pressure:** Diastolic blood pressure (mm Hg).

✓ **Skin Thickness:** Triceps skin fold thickness (mm).

✓ **Insulin:** Two-Hour serum insulin (mu U/ml).

✓ **BMI:** Body mass index (weight in kg/(height in m)^2).

✓ **Diabetes Pedigree Function/DPF:** A function that scores the likelihood of diabetes based on family history.

✓ **Age:** In years.

Code:

Import libraries:

```
import pandas as pd    #for data manipulation and analysis
import numpy as np      #for numerical operations and array manipulation
import matplotlib.pyplot as plt   #for creating visualization
import seaborn as sns    #for advanced data visualization
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
```

Importing the dataset:

```
data = pd.read_csv ("diabetes2.csv")  #reads the dataset from csv file
```

Creating a DataFrame:

```
df = pd.DataFrame (data)  #creates a dataframe from the imported data
df.head()                  #displays the first few rows
```

Output:

	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

3: Data Exploration:

```
print ("Shape of data {}" . format (df.shape))       #provides the dimensions of the dataframe
print ("Number of rows: {}" . format (df.shape [0]))
print ("Number of columns: {}" . format (df.shape [1]))
```

Output:

```
Shape of data (768, 9)
Number of rows: 768
Number of columns: 9
```

```
df.info ()    #provides information about each column
```

Output:

```
<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
```

```
df.describe() #summarizes the columns with mean,standard deviation,quartiles etc..,
```

Output:

	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

```
df.columns  #list the column names
```

Output:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
df["SkinThickness"]=df["SkinThickness"].replace(0,df["SkinThickness"].mean()) #replacing zeros in SkinThickness with mean values
```

```
df
```

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35.000000	0	33.6	0.627	50	1
1	1	85	66	29.000000	0	26.6	0.351	31	0
2	8	183	64	20.536458	0	23.3	0.672	32	1
3	1	89	66	23.000000	94	28.1	0.167	21	0
4	0	137	40	35.000000	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48.000000	180	32.9	0.171	63	0
764	2	122	70	27.000000	0	36.8	0.340	27	0
765	5	121	72	23.000000	112	26.2	0.245	30	0
766	1	126	60	20.536458	0	30.1	0.349	47	1
767	1	93	70	31.000000	0	30.4	0.315	23	0

768 rows × 9 columns

```
df["Insulin"]=df["Insulin"].replace(0,df["Insulin"].mean())#replacing zeros in Insulin with mean values
```

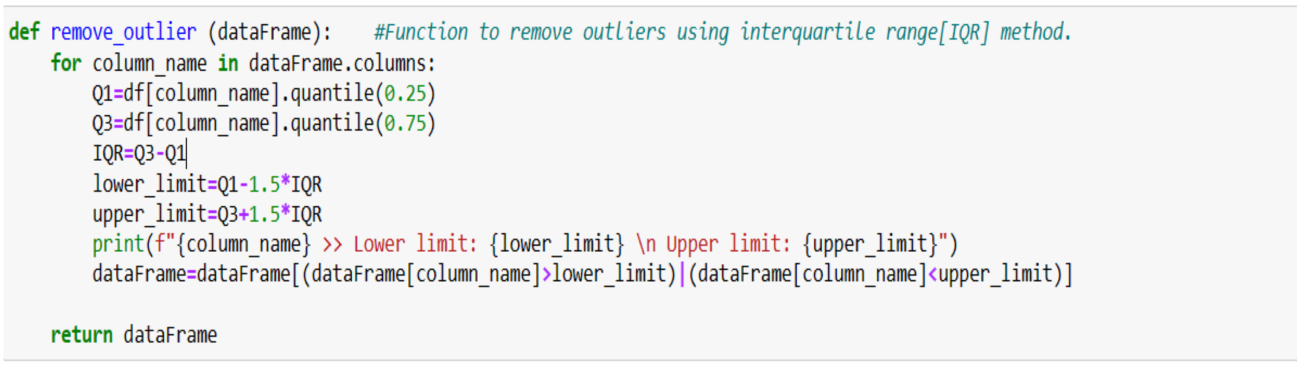
```
df
```

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35.000000	79.799479	33.6	0.627	50	1
1	1	85	66	29.000000	79.799479	26.6	0.351	31	0
2	8	183	64	20.536458	79.799479	23.3	0.672	32	1
3	1	89	66	23.000000	94.000000	28.1	0.167	21	0
4	0	137	40	35.000000	168.000000	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48.000000	180.000000	32.9	0.171	63	0
764	2	122	70	27.000000	79.799479	36.8	0.340	27	0
765	5	121	72	23.000000	112.000000	26.2	0.245	30	0
766	1	126	60	20.536458	79.799479	30.1	0.349	47	1
767	1	93	70	31.000000	79.799479	30.4	0.315	23	0

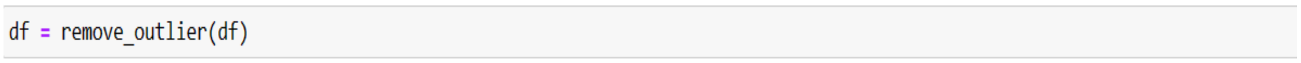
768 rows × 9 columns

Outlier Removal



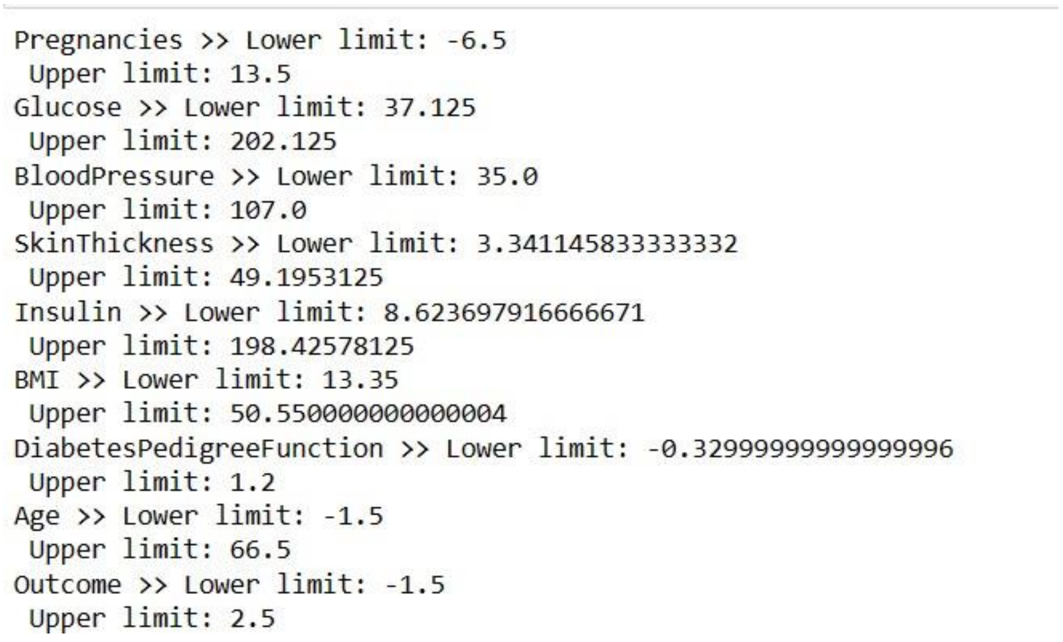
```
def remove_outlier (dataFrame):    #Function to remove outliers using interquartile range[IQR] method.
    for column_name in dataFrame.columns:
        Q1=df[column_name].quantile(0.25)
        Q3=df[column_name].quantile(0.75)
        IQR=Q3-Q1
        lower_limit=Q1-1.5*IQR
        upper_limit=Q3+1.5*IQR
        print(f"{column_name} >> Lower limit: {lower_limit} \n Upper limit: {upper_limit}")
        dataFrame=dataFrame[(dataFrame[column_name]>lower_limit)|(dataFrame[column_name]<upper_limit)]

    return dataFrame
```



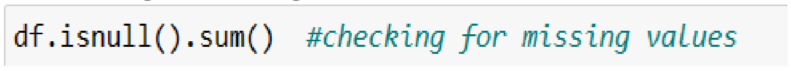
```
df = remove_outlier(df)
```

Output:



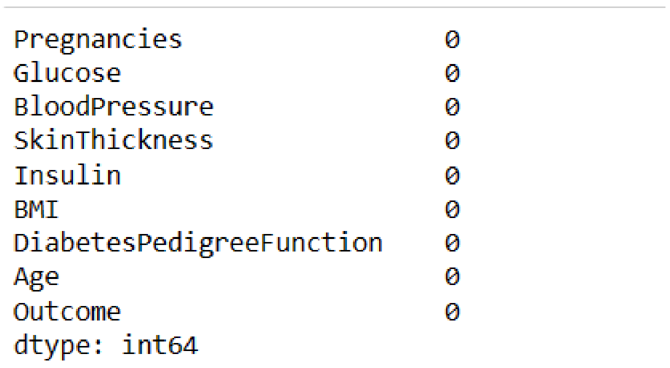
```
Pregnancies >> Lower limit: -6.5
 Upper limit: 13.5
Glucose >> Lower limit: 37.125
 Upper limit: 202.125
BloodPressure >> Lower limit: 35.0
 Upper limit: 107.0
SkinThickness >> Lower limit: 3.341145833333332
 Upper limit: 49.1953125
Insulin >> Lower limit: 8.623697916666671
 Upper limit: 198.42578125
BMI >> Lower limit: 13.35
 Upper limit: 50.550000000000004
DiabetesPedigreeFunction >> Lower limit: -0.32999999999999996
 Upper limit: 1.2
Age >> Lower limit: -1.5
 Upper limit: 66.5
Outcome >> Lower limit: -1.5
 Upper limit: 2.5
```

Handling Missing Values:



```
df.isnull().sum()  #checking for missing values
```

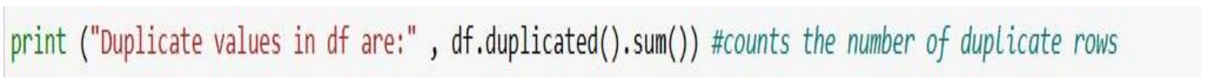
Output:



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

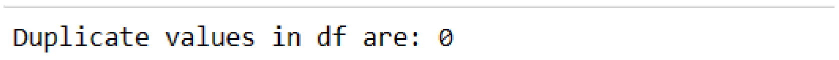
Observation: There are 8 features and 768 entires, all non-null. All of the features are numerical and the target is categorical.

Checking For Duplicates values:



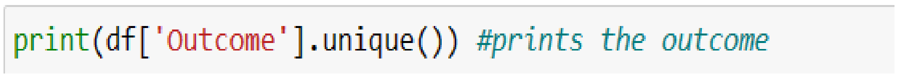
```
print ("Duplicate values in df are:" , df.duplicated().sum()) #counts the number of duplicate rows
```

Output:



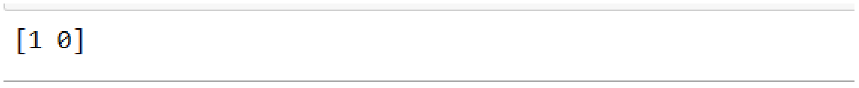
```
Duplicate values in df are: 0
```

Exploring Categorical Variables:



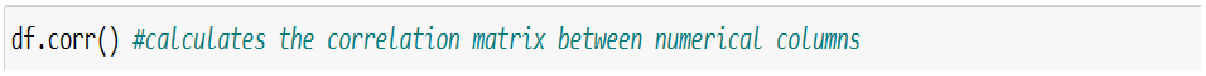
```
print(df['Outcome'].unique()) #prints the outcome
```

Output:



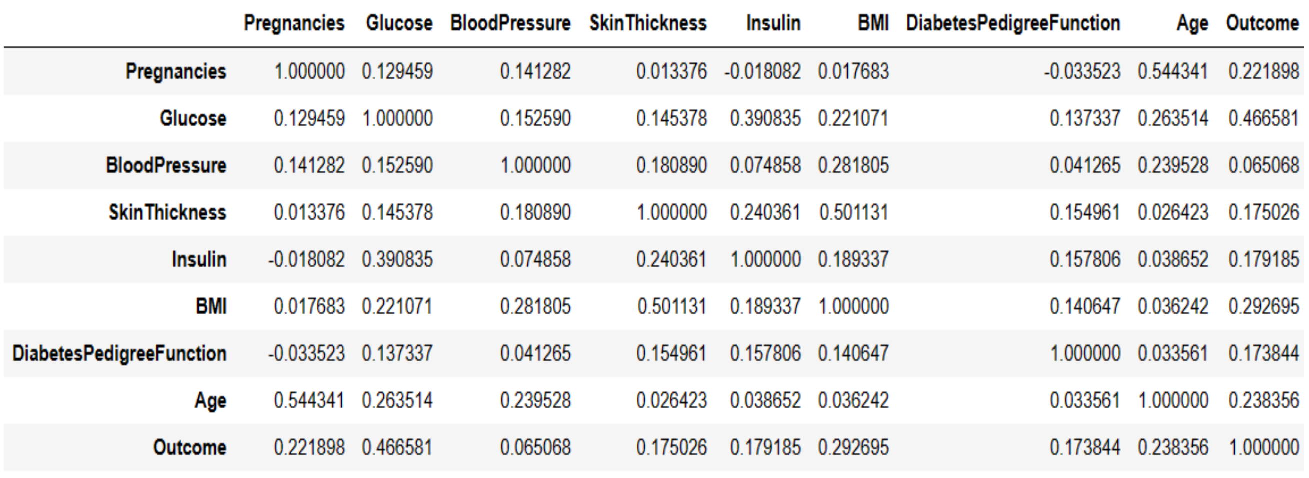
```
[1 0]
```

4:Correlation Analysis:



```
df.corr() #calculates the correlation matrix between numerical columns
```

Output:

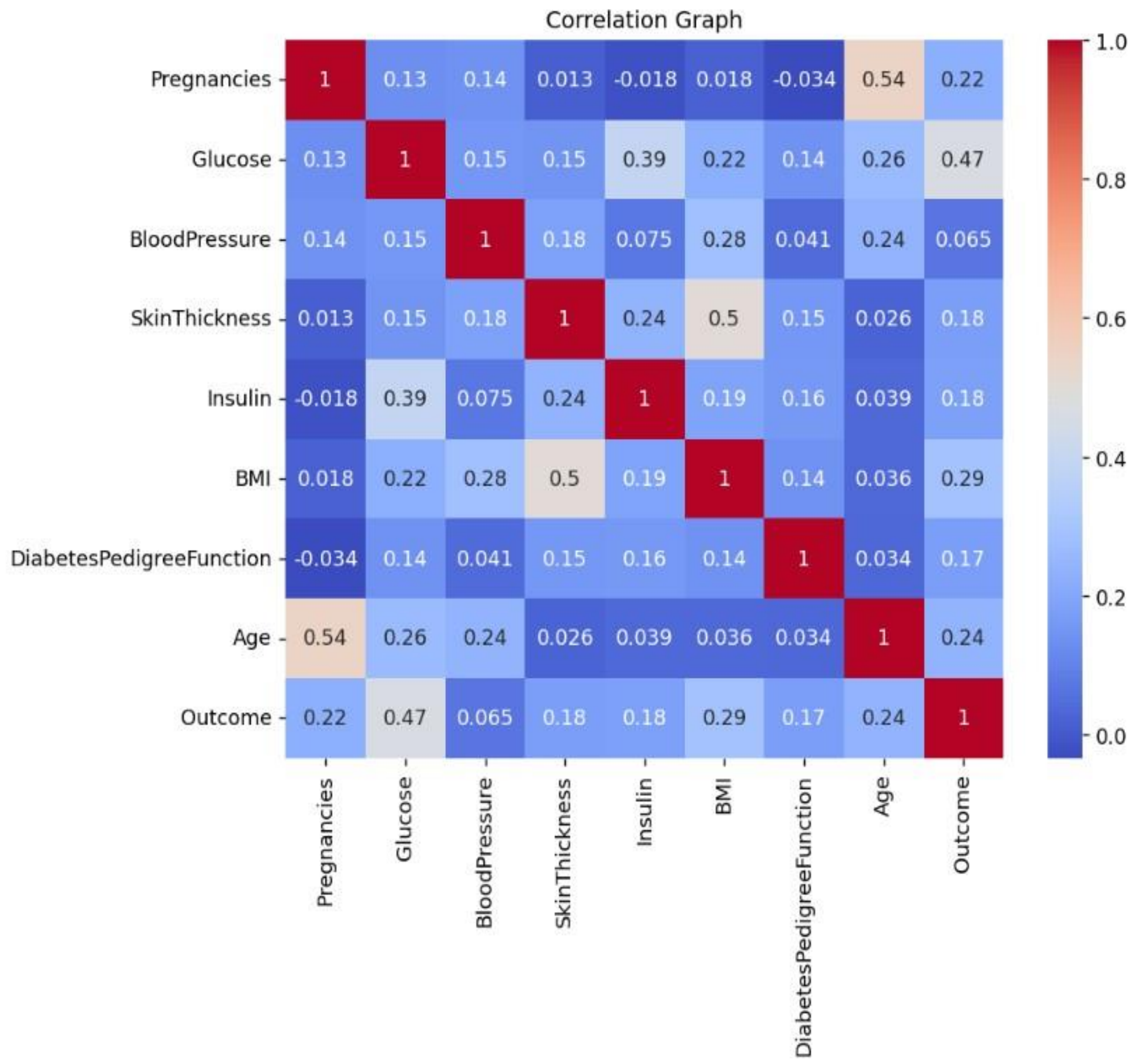


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	0.013376	-0.018082	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.145378	0.390835	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.180890	0.074858	0.281805	0.041265	0.239528	0.065068
SkinThickness	0.013376	0.145378	0.180890	1.000000	0.240361	0.501131	0.154961	0.026423	0.175026
Insulin	-0.018082	0.390835	0.074858	0.240361	1.000000	0.189337	0.157806	0.038652	0.179185
BMI	0.017683	0.221071	0.281805	0.501131	0.189337	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.154961	0.157806	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	0.026423	0.038652	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.175026	0.179185	0.292695	0.173844	0.238356	1.000000

```python
plt.figure (figsize = [8,6],  dpi = 130 )  #create figure for visualization
plt.title ("Correlation Graph" , fontsize = 11 )
sns.heatmap (df.corr(), annot = True , cmap="coolwarm" )
#creates heatmap to visualize correlation with numerical values annotated.
```

Output:

```
<AxesSubplot:title={'center':'Correlation Graph'}>
```



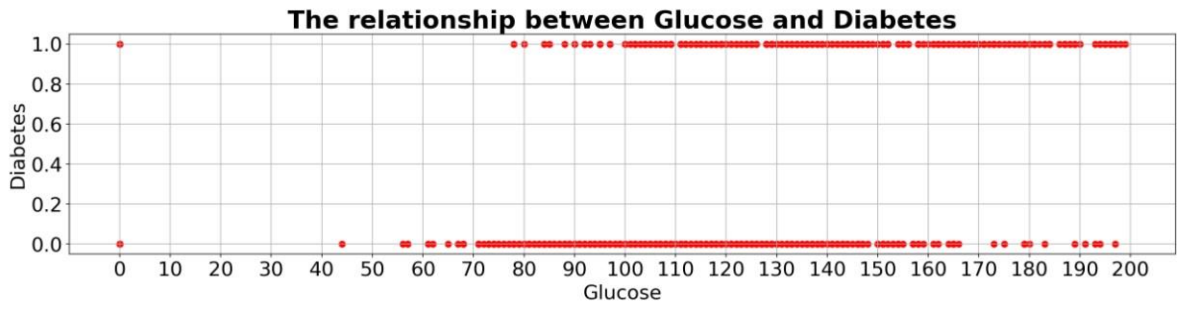
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.14	0.013	-0.018	0.018	-0.034	0.54	0.22
Glucose	0.13	1	0.15	0.15	0.39	0.22	0.14	0.26	0.47
BloodPressure	0.14	0.15	1	0.18	0.075	0.28	0.041	0.24	0.065
SkinThickness	0.013	0.15	0.18	1	0.24	0.5	0.15	0.026	0.18
Insulin	-0.018	0.39	0.075	0.24	1	0.19	0.16	0.039	0.18
BMI	0.018	0.22	0.28	0.5	0.19	1	0.14	0.036	0.29
DiabetesPedigreeFunction	-0.034	0.14	0.041	0.15	0.16	0.14	1	0.034	0.17
Age	0.54	0.26	0.24	0.026	0.039	0.036	0.034	1	0.24
Outcome	0.22	0.47	0.065	0.18	0.18	0.29	0.17	0.24	1

5:Data Visualization:

The relationship between Glucose and Diabetes:

```
plt.figure(figsize = [20, 4] , dpi = 150)
plt.scatter (df["Glucose"] , df["Outcome"] , color = "red")
# creates scatter plots to visualize relationships between glucose and the Diabetes.
plt.title ("The relationship between Glucose and Diabetes" , weight='bold', fontsize = 25)
plt.xticks (range (0 , 205 , 10), fontsize = 20)
plt.yticks (fontsize = 20)
plt.xlabel ('Glucose', fontsize = 20 )
plt.ylabel ('Diabetes' , fontsize = 20)
plt.grid ()
plt.show ()
```

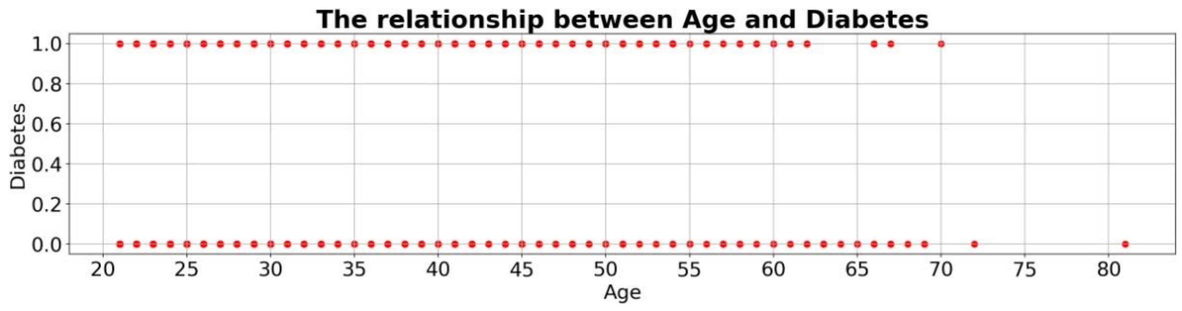
Output:



The relationship between Age and Diabetes:

```
plt.figure(figsize = [20, 4] , dpi = 150)
plt.scatter (df["Age"] , df["Outcome"] , color = "red")
# creates scatter plots to visualize relationships between age and the Diabetes.
plt.title ("The relationship between Age and Diabetes" , weight='bold', fontsize = 25)
plt.xticks (range (20 , 85 , 5), fontsize = 20)
plt.yticks (fontsize = 20)
plt.xlabel ('Age', fontsize = 20 )
plt.ylabel ('Diabetes' , fontsize = 20)
plt.grid ()
plt.show ()
```

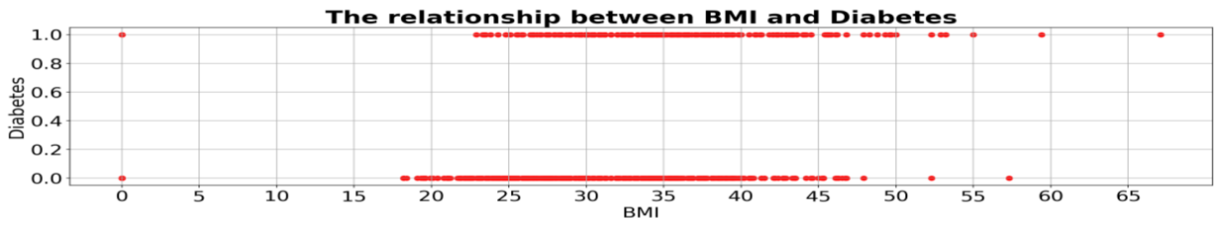
Output:



The relationship between BMI and DIABETES:

```
plt.figure(figsize = [20, 4] , dpi = 150)
plt.scatter (df["BMI"] , df["Outcome"] , color = "red")
# creates scatter plots to visualize relationships between BMI and the Diabetes.
plt.title ("The relationship between BMI and Diabetes" , weight = 'bold', fontsize = 25)
plt.xticks (range (0 , 70 , 5) , fontsize=20)
plt.yticks (fontsize = 20)
plt.xlabel ('BMI', fontsize = 20 )
plt.ylabel ('Diabetes' , fontsize = 20)
plt.grid ()
plt.show ()
```

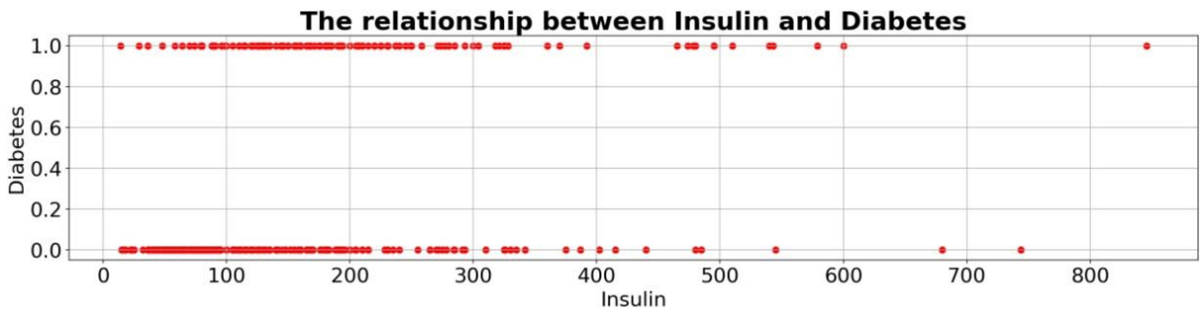
Output:



The relationship between Insulin and Diabetes:

```
plt.figure(figsize = [20, 4] , dpi = 150)
plt.scatter (df["Insulin"] , df["Outcome"] , color = "red")
# creates scatter plots to visualize relationships between Insulin and the Diabetes.
plt.title ("The relationship between Insulin and Diabetes" , weight = 'bold', fontsize = 25)
plt.xticks (range (0 , 900 , 100) , fontsize = 20)
plt.yticks (fontsize = 20)
plt.xlabel ('Insulin', fontsize = 20 )
plt.ylabel ('Diabetes' , fontsize = 20)
plt.grid ()
plt.show ()
```

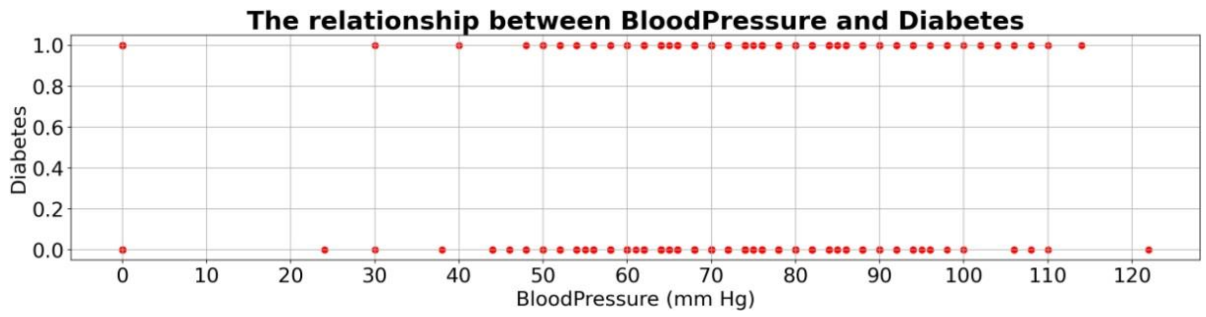
Output:



The relationship between BloodPressure and Diabetes:

```
plt.figure(figsize = [20, 4] , dpi = 150)
plt.scatter (df["BloodPressure"] , df["Outcome"] , color = "red")
# creates scatter plots to visualize relationships between BP and the Diabetes.
plt.title ("The relationship between BloodPressure and Diabetes" , weight = 'bold', fontsize = 25)
plt.xticks (range (0 , 125 , 10) , fontsize = 20)
plt.yticks (fontsize = 20)
plt.xlabel ('BloodPressure (mm Hg)', fontsize = 20)
plt.ylabel ('Diabetes' , fontsize = 20)
plt.grid ()
plt.show ()
```

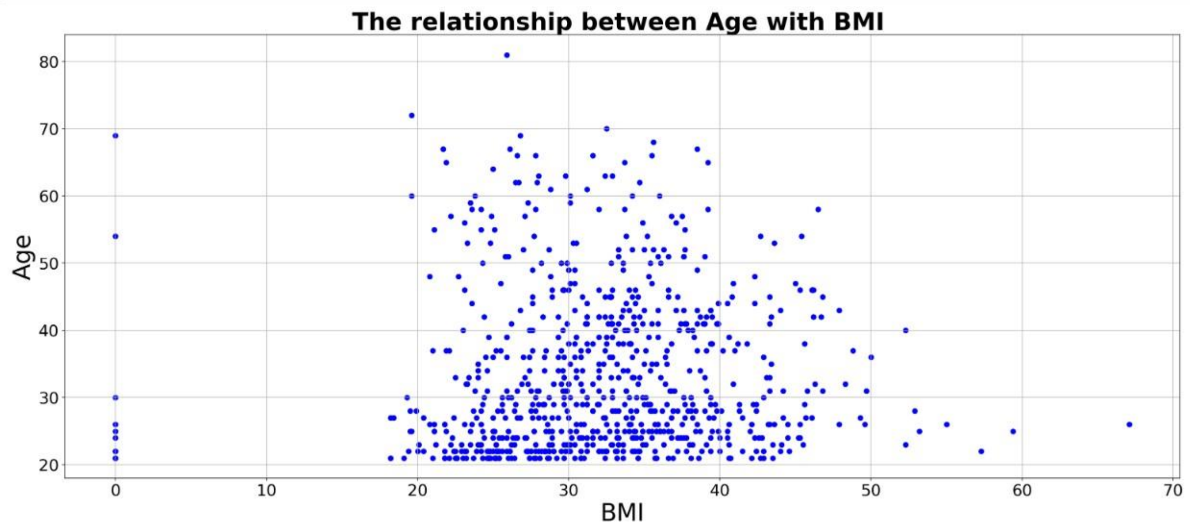
Output:



The relationship between Age and BMI:

```
plt.figure(figsize = [25, 10] , dpi = 150)
plt.scatter (df["BMI"] , df["Age"] , color = "blue")
# creates scatter plots to visualize relationships between Age and the BMI.
plt.title ("The relationship between Age with BMI " , weight = 'bold', fontsize = 30)
plt.xticks (range (0 , 80 , 10) , fontsize = 20)
plt.yticks (range (20 , 90 , 10) , fontsize = 20)
plt.xlabel ('BMI', fontsize = 30 )
plt.ylabel ('Age' , fontsize = 30)
plt.grid ()
plt.show ()
```

Output:



Feature Selection:

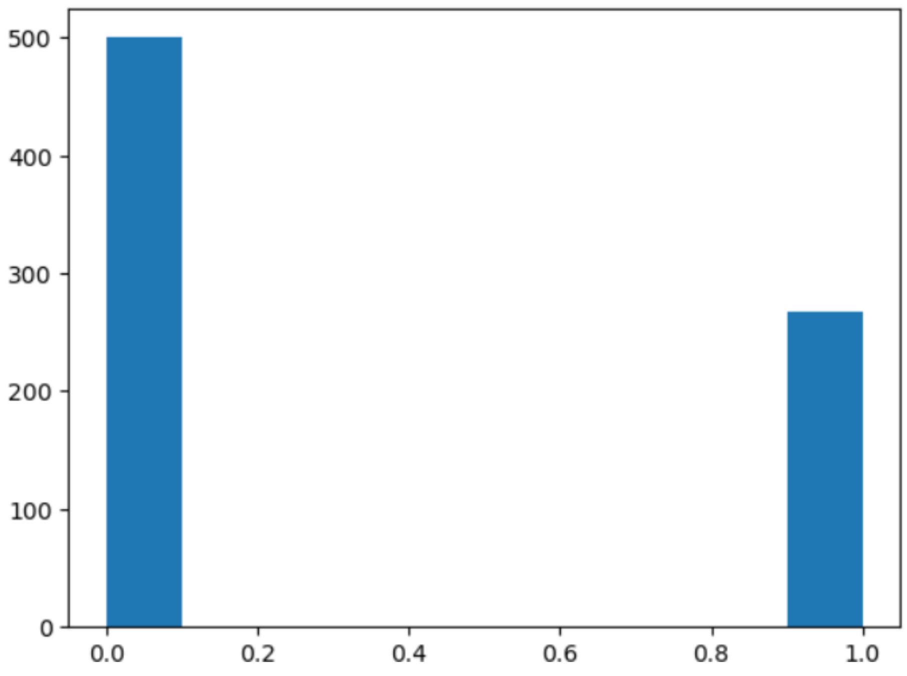
```
x = pd.DataFrame (data , columns = ["Pregnancies" , "Glucose" , "BloodPressure" , "SkinThickness" , "Insulin" , "BMI"
                                   , "DiabetesPedigreeFunction" , "Age"]) # Features(independent variables)
y = data.Outcome # Target variables(dependent variables)
```

Among the 768 people, 268 people suffer from diabetes and 500 people do not have diabetes.

```
plt.hist (y)
```

Output:

```
(array([500.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 268.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



6:Data Splitting:

Training = 75, Testing = 25

```
X_train , X_test , y_train , y_test = train_test_split (X , y , test_size = 0.25 , random_state = 0)
```

7: Model Building and Evaluation[Logistic Regression]:

```
logreg = LogisticRegression (solver = "liblinear") #Creates a logistic regression model
logreg.fit (X_train , y_train)        # Fitting a  model(Trains the model on the training set).
y_pred = logreg.predict(X_test)          # Predicted class labels from test features(makes preditions on the training set.)
y_predicted_proba = logreg.predict_proba(X_test) #Predicted probabilities from test features(makes prediction on testing set.)
```

```
print ("Accuracy: " , metrics.accuracy_score (y_test , y_pred))  #calculate accuracy of predictions.
```

Output:

```
Accuracy:  0.8072916666666666
```

Confusion matrix:

```
confusion_matrix (y , logreg.predict (X)) #creates a confussion matrix to visualize true vs predicted outcomes.
```

```
array([[447,  53],
       [117, 151]], dtype=int64)
```

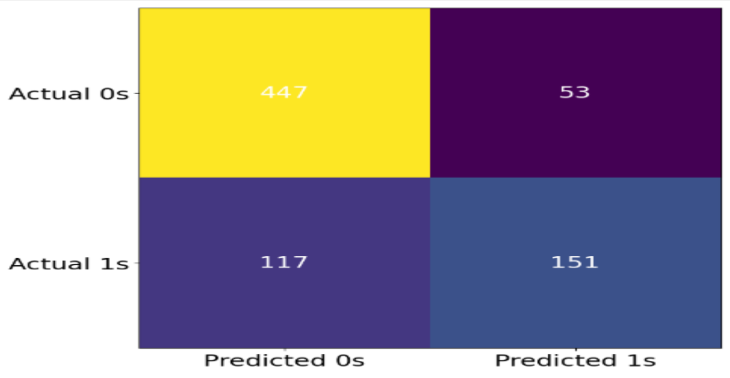
```
cm = confusion_matrix (y , logreg.predict(X))

fig , ax = plt.subplots (figsize = (8,8))
ax.imshow (cm)
ax.grid (False)
ax.xaxis.set (ticks = (0 , 1) , ticklabels = ("Predicted 0s" , "Predicted 1s"))
ax.yaxis.set (ticks = (0 , 1) , ticklabels = ("Actual 0s" , "Actual 1s"))

ax.tick_params(axis='both', which='major', labelsize=20)
#ax.tick_params(axis='both', which='minor', labelsize=20)
#plt.xticks(fontsize=14, rotation=90)

ax.set_ylim (1.5 , -0.5)
for i in range (2):
    for j in range (2):
        ax.text (j , i , cm[i,j] , ha = "center" , va ="center" , color ="white", fontsize = 20 )
plt.show()
```

Output:



Classification Report:

```
print (classification_report (y , logreg.predict (X)))
#Prints detailed classification report with precision,recall.F1-score,etc.
```

Output:

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       500
           1       0.74      0.56      0.64       268

    accuracy                           0.78       768
   macro avg       0.77      0.73      0.74       768
weighted avg       0.77      0.78      0.77       768
```

8:Additional Analysis and Prediction:

```
model2 = LogisticRegression (solver = "liblinear"  , C = 10.0 , random_state = 0)
#creates another model with adjusted hyperparameters.
model2.fit(X,y)
```

Output:

```
LogisticRegression(C=10.0, random_state=0, solver='liblinear')
```

Creating a New Data Point:

```
df2 = pd.DataFrame ({"Pregnancies" : [0] , "Glucose" :[80] , "BloodPressure" :[72] ,"SkinThickness" : [0] ,
                    "Insulin" : [0] , "BMI" : [23] , "DiabetesPedigreeFunction" : [0.5] ,
                    "Age" : [30] , "Outcome" : [0]})
df2
#create a dataframe with a single row representing a new data point with various features and target values of 0(no diabetes)
```

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0	80	72	0	0	23	0.5	30	0

```
data2 = data.append(df2)
data2 #Appends this new data point to the original dataset.
```

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35.000000	79.799479	33.6	0.627	50	1
1	1	85	66	29.000000	79.799479	26.6	0.351	31	0
2	8	183	64	20.536458	79.799479	23.3	0.672	32	1
3	1	89	66	23.000000	94.000000	28.1	0.167	21	0
4	0	137	40	35.000000	168.000000	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
764	2	122	70	27.000000	79.799479	36.8	0.340	27	0
765	5	121	72	23.000000	112.000000	26.2	0.245	30	0
766	1	126	60	20.536458	79.799479	30.1	0.349	47	1
767	1	93	70	31.000000	79.799479	30.4	0.315	23	0
0	0	80	72	0.000000	0.000000	23.0	0.500	30	0

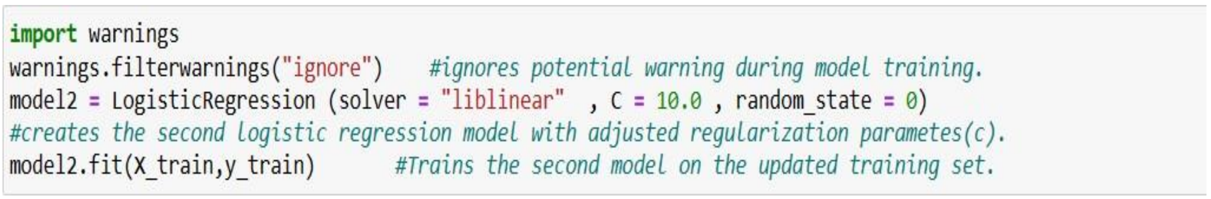
769 rows × 9 columns

Preparing Data for Model 2:

```
X_train = data2 [["Pregnancies" , "Glucose" , "BloodPressure" , "SkinThickness" , "Insulin" , "BMI"
, "DiabetesPedigreeFunction" , "Age"]] [:768]
#selects features for training from the updated dataset,using data up to the 768th row
y_train = data2[["Outcome"]][:768].values.reshape (-1,1)
#selects and reshapes target values for training.
```

```
X_test = data2 [["Pregnancies" , "Glucose" , "BloodPressure" , "SkinThickness" , "Insulin" , "BMI"
                  , "DiabetesPedigreeFunction" , "Age"]] [768:]
#selects features for testing,using the newly added data point(769th row).
```

Training and Prediction with Model 2:

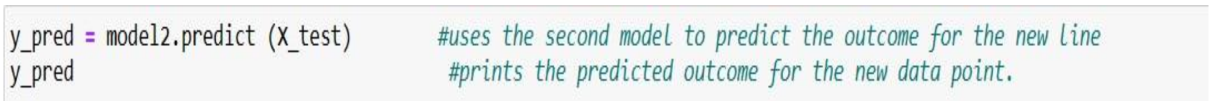


```
import warnings
warnings.filterwarnings("ignore")    #ignores potential warning during model training.
model2 = LogisticRegression (solver = "liblinear"  , C = 10.0 , random_state = 0)
#creates the second logistic regression model with adjusted regularization parametes(c).
model2.fit(X_train,y_train)      #Trains the second model on the updated training set.
```

Output:

```
LogisticRegression(C=10.0, random_state=0, solver='liblinear')
```

Final Prediction:



```
y_pred = model2.predict (X_test)       #uses the second model to predict the outcome for the new line
y_pred                                  #prints the predicted outcome for the new data point.
```

Output:

```
array([0], dtype=int64)
```

Conclusion:

In this project, we used logistic regression, a supervised machine learning algorithm, to predict whether a person has diabetes or not based on various features such as glucose, pregnancy, body mass index, age, and diabetes pedigree function. We used Python and popular libraries such as Pandas, Scikit-Learn, and Matplotlib to perform data analysis, model building, and evaluation.

The results show that logistic regression achieves an accuracy of 80% on the test data. We also identify the most important features that influence the prediction of diabetes using the coefficient values of the logistic regression model. The project demonstrates the potential of using logistic regression to assist in the diagnosis and management of diabetes.

References

1. https://youtu.be/ZhHpqL_qtJI?si=N8kxCweHByKgWz8c
2. https://www.kaggle.com/code/johncubides/diabetes-regresion-logostica
3. http://www.kaggle.com