IBM – NAAN MUDHALVAN➔ARTIFICIAL INTELLIGENCE

PHASE – 3

Project – 2: AI – Based Diabetes Prediction System

Content : Development Part -1

In this part you will begin building your project by loading and preprocessing the dataset.
In this phase begin developing the diabetes prediction system by preparing the data and selecting relevant features.

About Dataset :

Context :

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Content :

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

Sources:

(a) Original owners : National Institute of Diabetes and Digestive and Kidney Diseases

(b) Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu)
Research Center, RMI Group Leader
Applied Physics Laboratory
The Johns Hopkins University
Johns Hopkins Road
Laurel, MD 20707
(301) 953-6231

(c) Date received : 9 May 1990

Number of Instances*: 768*

Number of Attributes*: 8 plus class*

For Each Attribute*: (all numeric-valued)*

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Missing Attribute Values*: Yes*

Class Distribution: *(class value 1 is interpreted as "tested positive for* diabetes")

Diabetes Prediction using Logistic Regression Algorithm in Machine Learning

Diabetes Prediction:

The dataset comprises crucial health-related features such as 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', and 'Age'. The main objective was to predict the 'Outcome' label, which signifies the likelihood of diabetes.

About the Data:

Data Overview: This is a diabetes.csv data

Import Required Libraries:

```
# Ignore warning messages to prevent them from being displayed during code
execution
import warnings
warnings.filterwarnings('ignore')


import numpy as np    # Importing the NumPy library for linear algebra
operations
import pandas as pd   # Importing the Pandas library for data processing and
CSV file handling


import os
for dirname, _, filenames in os.walk('/diabetes.csv/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
(C:Users/Sakthi/Downloads/archive.zip/diabetes.csv)
```

```
import seaborn as sns                  # Importing the Seaborn library
for statistical data visualization
import matplotlib.pyplot as plt        # Importing the Matplotlib
library for creating plots and visualizations
import plotly.express as px            # Importing the Plotly Express
library for interactive visualizations
```

Exploratory Data Analysis:

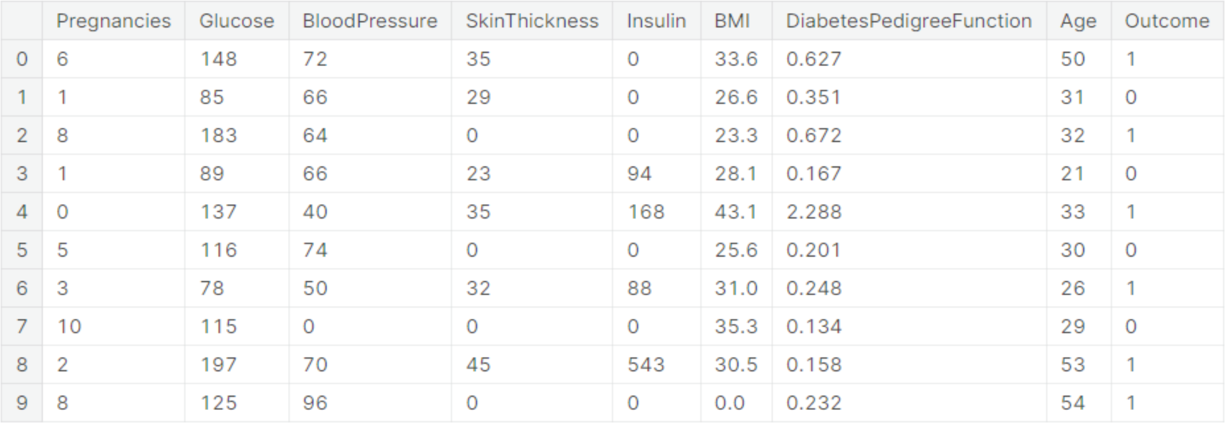
Load and Prepare Data:

```
df=pd.read_csv('C:Users/Sakthi/Downloads/archive.zip/diabetes.csv')
```

UnderStanding the Variables:

```
df.head(10)
```

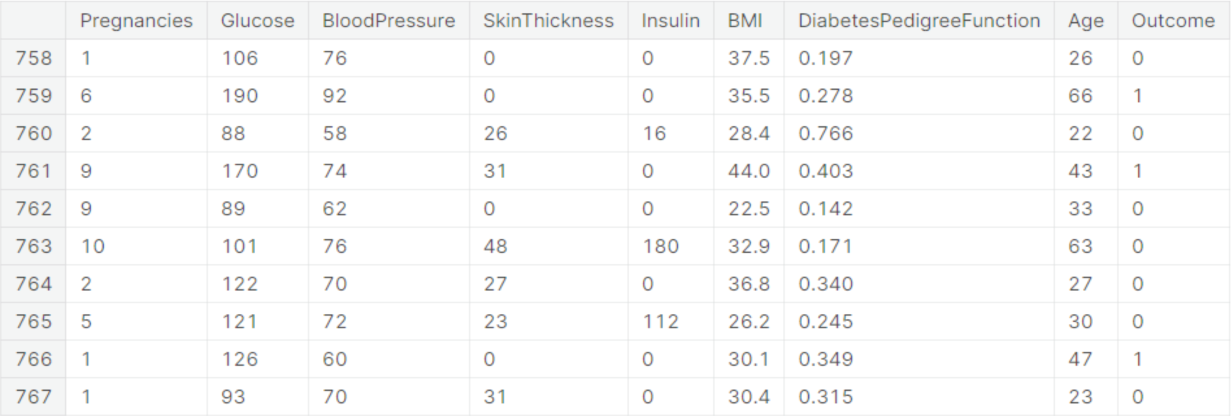
Output:



	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

```
df.tail(10)
```

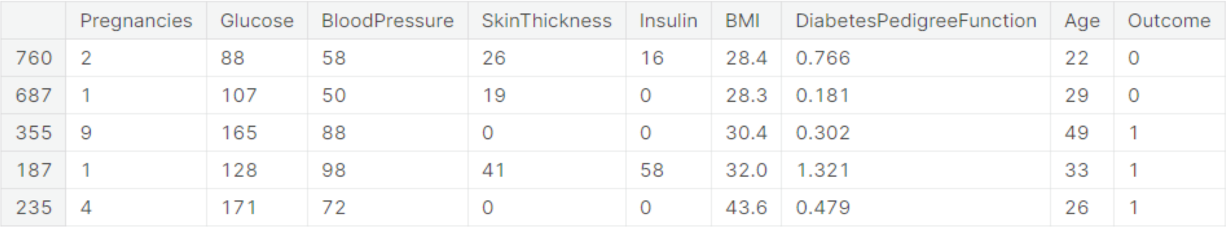
Output:



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
758	1	106	76	0	0	37.5	0.197	26	0
759	6	190	92	0	0	35.5	0.278	66	1
760	2	88	58	26	16	28.4	0.766	22	0
761	9	170	74	31	0	44.0	0.403	43	1
762	9	89	62	0	0	22.5	0.142	33	0
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

```
df.sample(5)
```

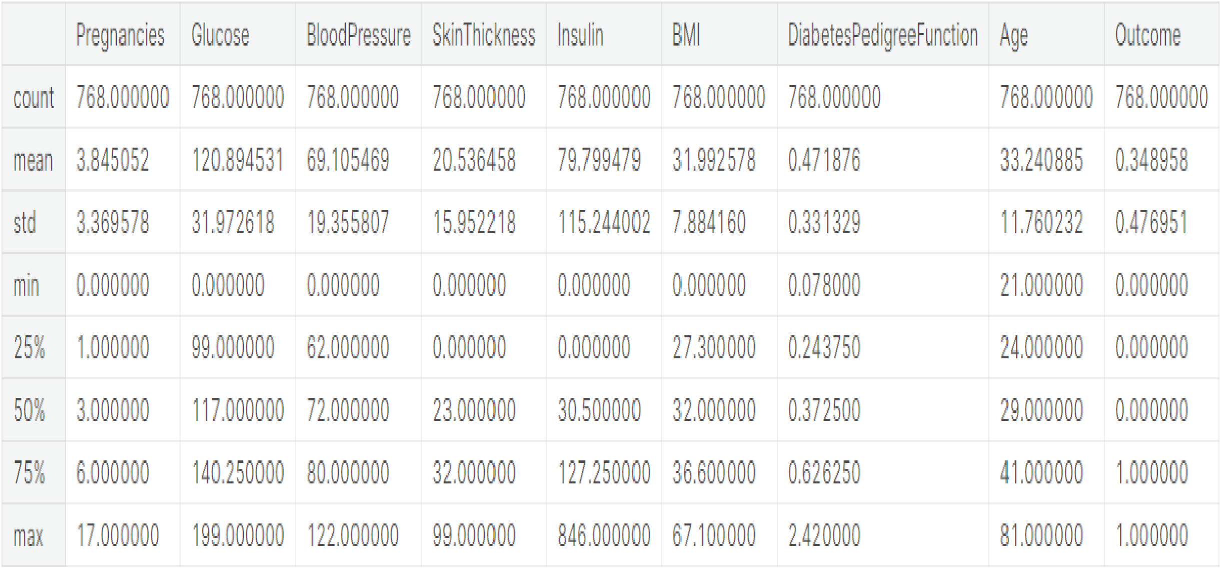
Output:



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
760	2	88	58	26	16	28.4	0.766	22	0
687	1	107	50	19	0	28.3	0.181	29	0
355	9	165	88	0	0	30.4	0.302	49	1
187	1	128	98	41	58	32.0	1.321	33	1
235	4	171	72	0	0	43.6	0.479	26	1

```
df.describe()
```

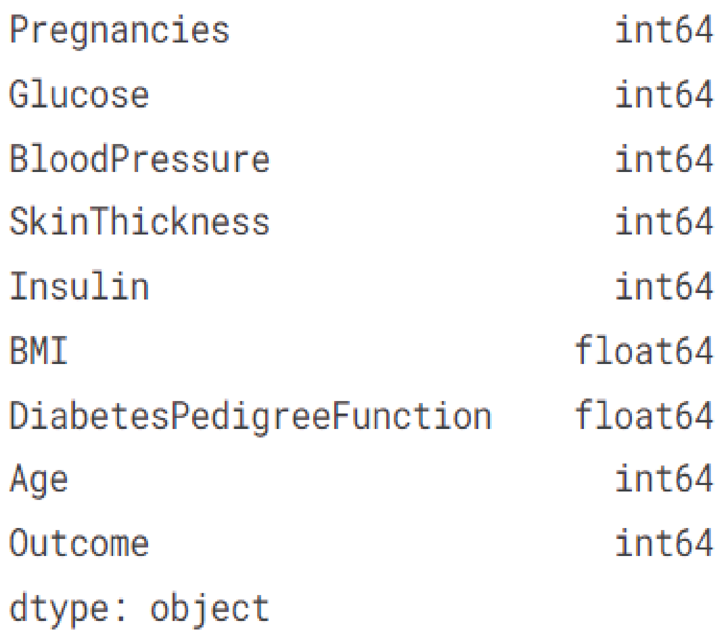
Output:



	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.dtypes
```

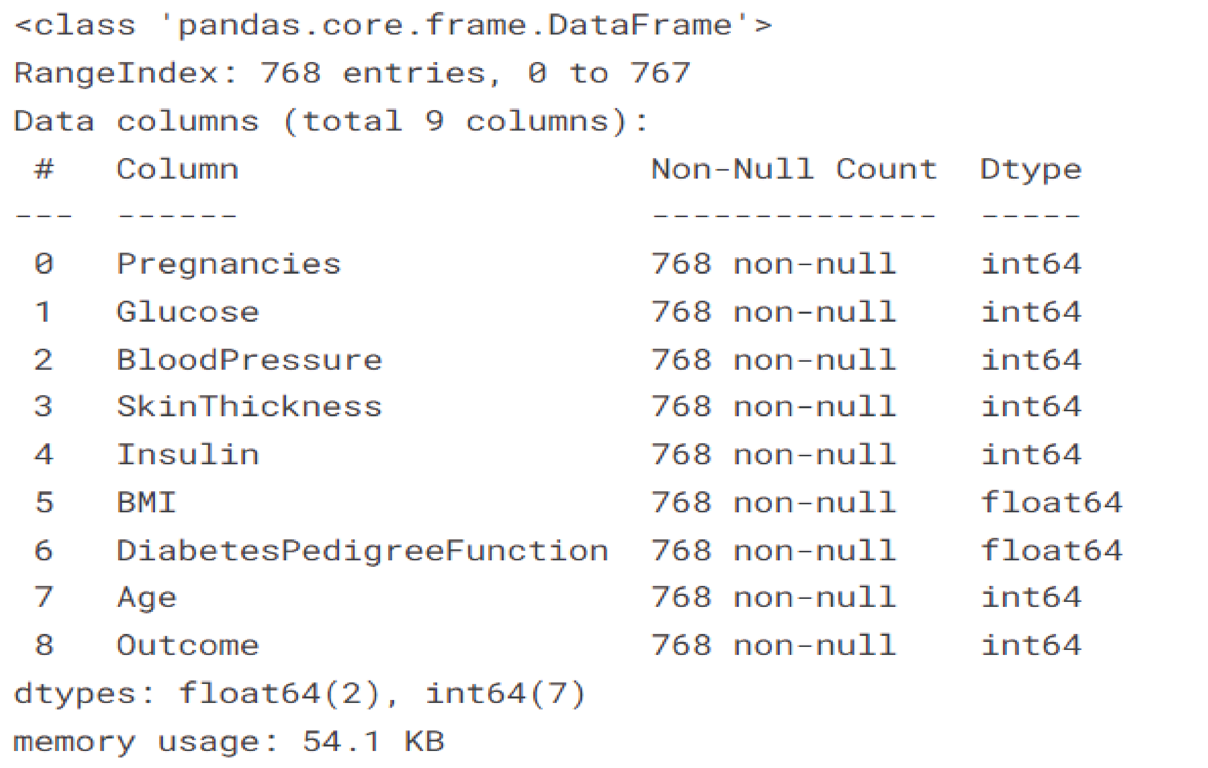
Output:



```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

```
df.info()
```

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.size
```

Output:

```
6912
```

```
df.shape
```

Output:

```
(768, 9)
```

Data Cleaning:

```
df.shape
```

Output:

```
(768, 9)
```

```
df=df.drop_duplicates()
df.shape
```

Output:

```
(768, 9)
```

Check null Values:

```
df.isnull().sum()
```

Output:

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

There is no Missing Values present in the Data

```
df.columns
```

Output:

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

Check the number of Zero Values in Dataset:

```
print("No. of Zero Values in Glucose ", df[df['Glucose']==0].shape[0])
```

Output:

```
No. of Zero Values in Glucose  5
```

```
print("No. of Zero Values in Blood Pressure ", df[df['BloodPressure']==0].
shape[0])
```

Output:

```
No. of Zero Values in Blood Pressure  35
```

```
print("No. of Zero Values in SkinThickness ", df[df['SkinThickness']==0].
shape[0])
```

Output:

```
No. of Zero Values in SkinThickness  227
```

```
print("No. of Zero Values in Insulin ", df[df['Insulin']==0].shape[0])
```

Output:

```
No. of Zero Values in Insulin  374
```

```
print("No. of Zero Values in BMI ", df[df['BMI']==0].shape[0])
```

Output:

```
No. of Zero Values in BMI  11
```

Replace zeroes with mean of that Columns:

```
df['Glucose']=df['Glucose'].replace(0, df['Glucose'].mean())
print('No of zero Values in Glucose ', df[df['Glucose']==0].shape[0])
```

Output:

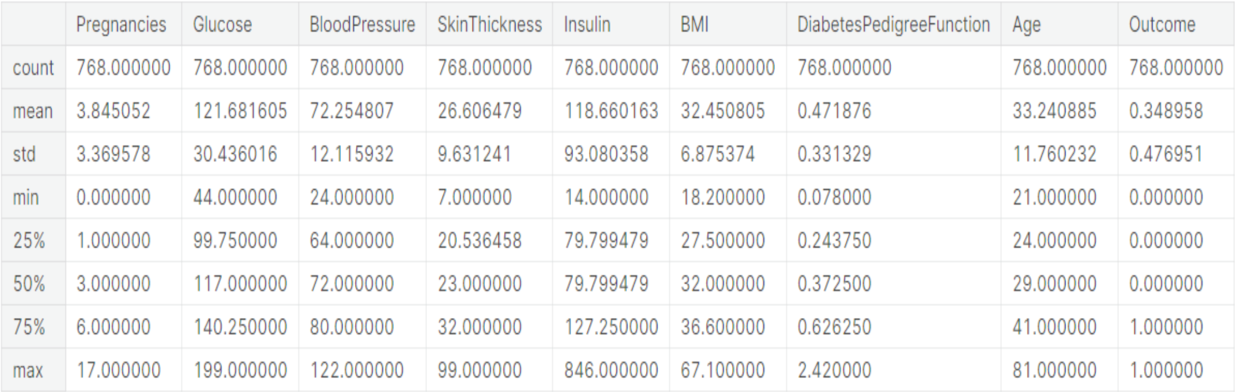
```
No of zero Values in Glucose  0
```

```
df['BloodPressure']=df['BloodPressure'].replace(0, df['BloodPressure'].mean
())
df['SkinThickness']=df['SkinThickness'].replace(0, df['SkinThickness'].mean
())
df['Insulin']=df['Insulin'].replace(0, df['Insulin'].mean())
df['BMI']=df['BMI'].replace(0, df['BMI'].mean())
```

Validate the Zero Values:

```
df.describe()
```

Output:



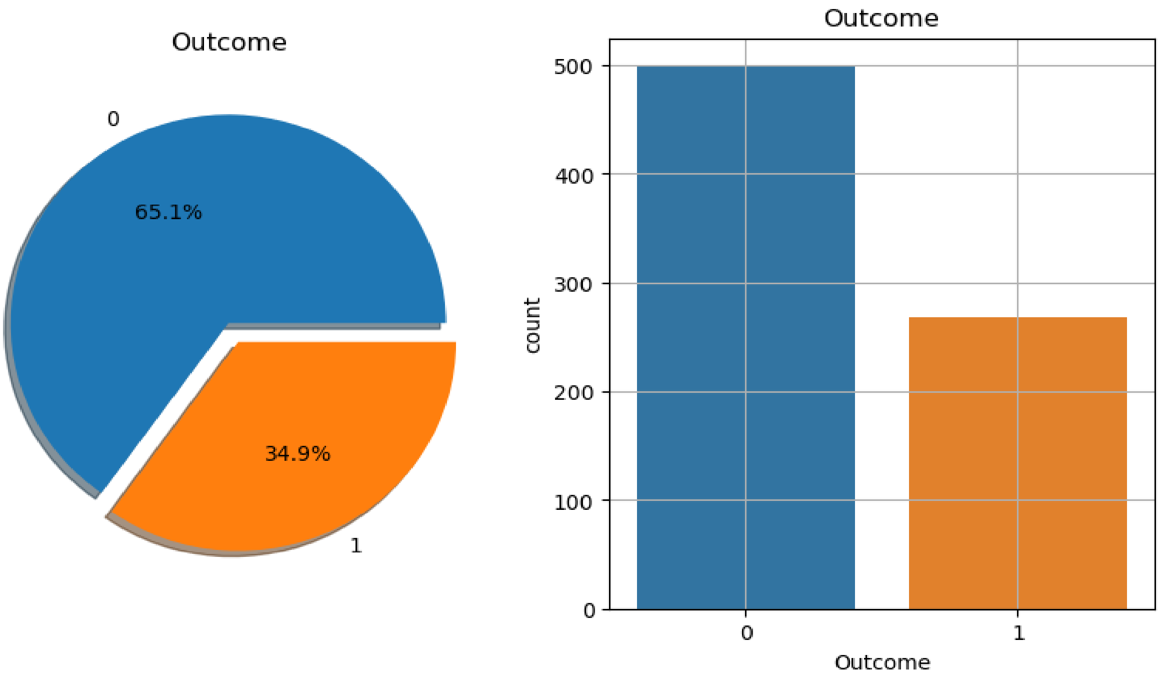
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.681605	72.254807	26.606479	118.660163	32.450805	0.471876	33.240885	0.348958
std	3.369578	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	79.799479	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

Data Visualization:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'df' is your DataFrame containing the dataset
# If you haven't imported your dataset yet, import it here

# Create subplots
f, ax = plt.subplots(1, 2, figsize=(10, 5))

# Pie chart for Outcome distribution
df['Outcome'].value_counts().plot.pie(explode=[0, 0.1], autopct='%1.1f%%', ax=ax[0], shadow=True)
ax[0].set_title('Outcome')
ax[0].set_ylabel(' ')

# Count plot for Outcome distribution
sns.countplot(x='Outcome', data=df, ax=ax[1])  # Use 'x' instead of 'Outcome'
ax[1].set_title('Outcome')
# Display class distribution
N, P = df['Outcome'].value_counts()
print('Negative (0):', N)
print('Positive (1):', P)

# Adding grid and showing plots
plt.grid()
plt.show()
```

Output:

```
Negative (0): 500
Positive (1): 268
```

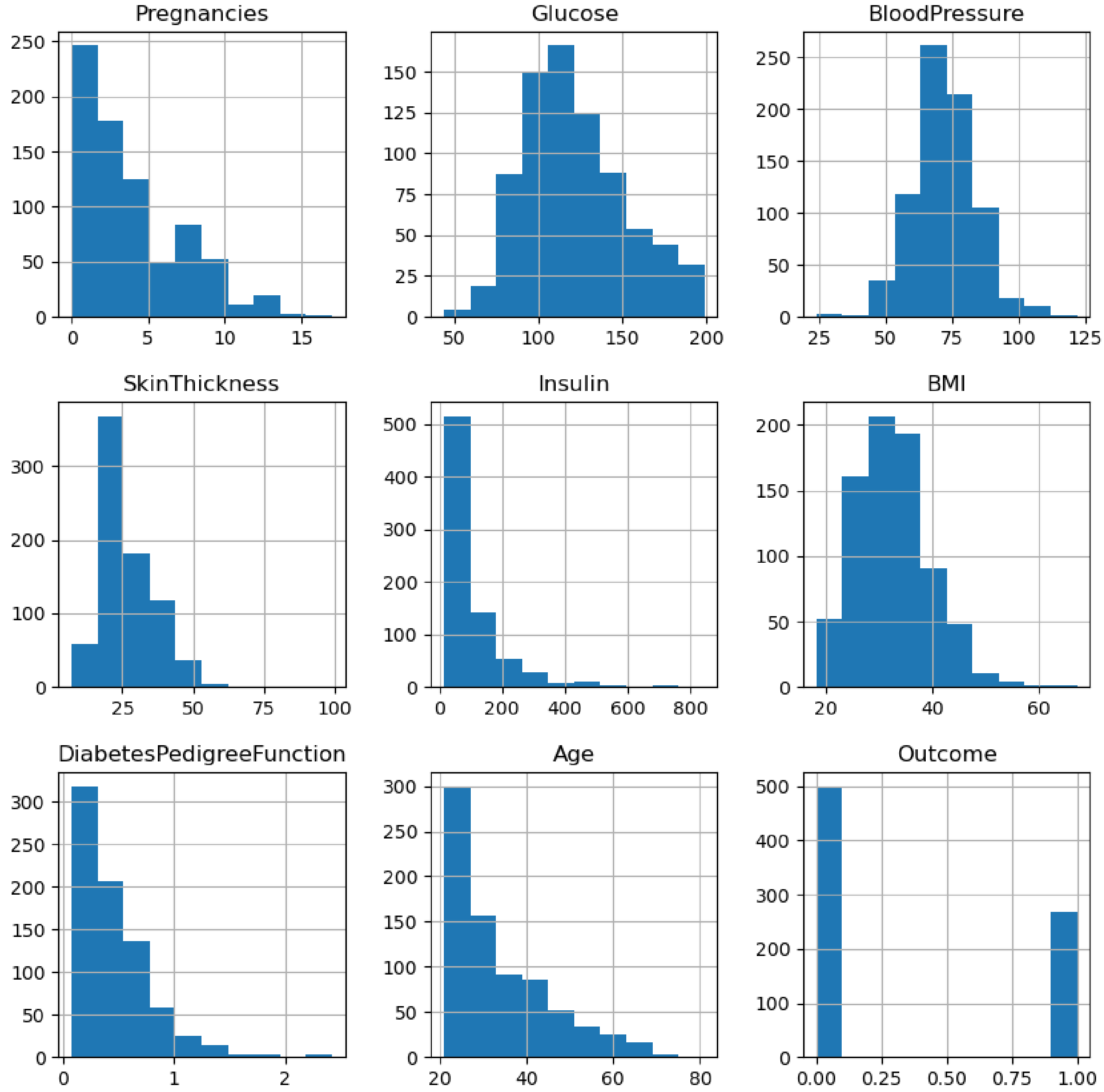


- *1 Represent --> Diabetes Positive*
- *0 Represent --> Daibetes Negative*

Histograms:

```
df.hist(bins=10, figsize=(10, 10))
plt.show()
```

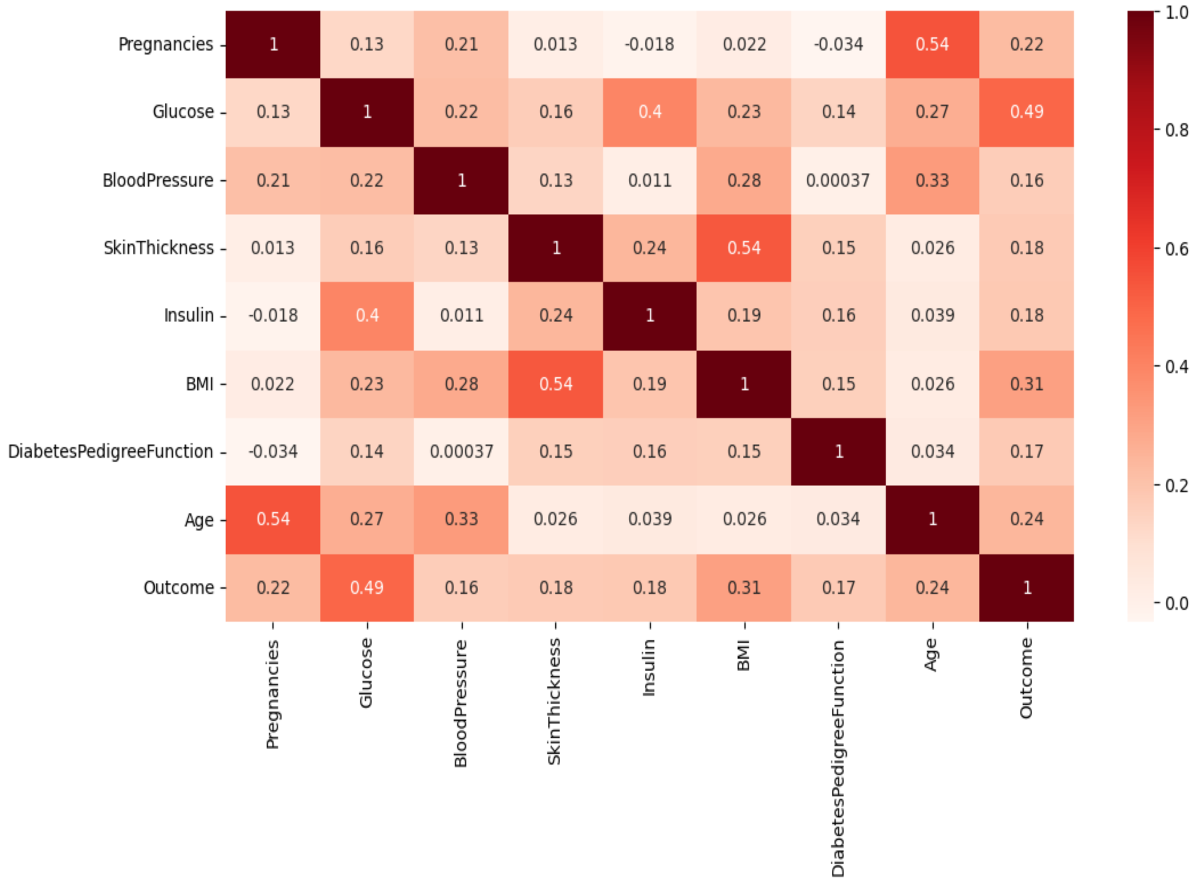
Output:



```
plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(), annot=True, cmap='Reds')
plt.plot()
# Creating a heatmap of the correlation matrix for the columns in the Dat
aFrame data
```

```
[]
```

Output:



```
mean = df['Outcome'].mean()
# Calculating the mean value of the 'Outcome' column in the DataFrame data
mean
# Displaying the calculated mean value
```

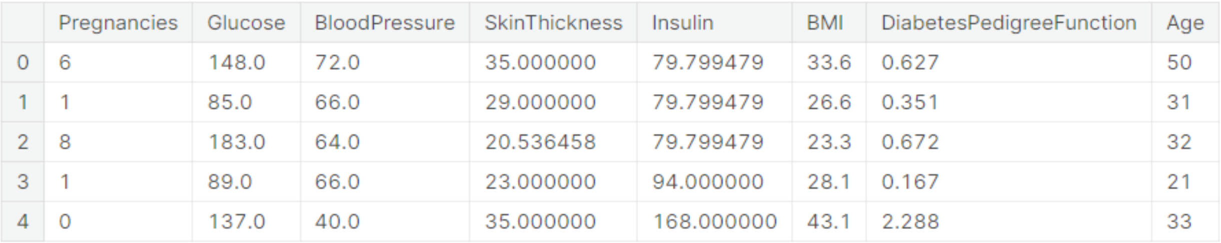
Output:

```
0.3489583333333333
```

Split the DataFrame into X and y:

```
target_name='Outcome'

y=df[target_name]

X= df.drop(target_name, axis=1)


X.head()
```

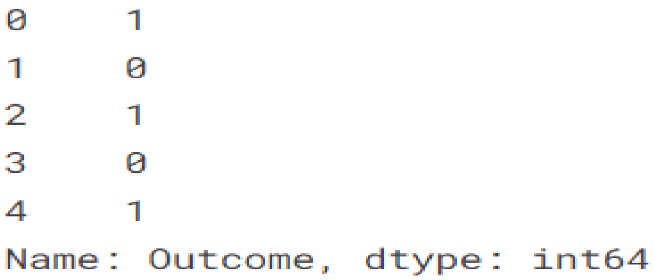
Output:



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33

```
y.head()
```

Output:



```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

Future Scalling:

```
# Standard Scaler:
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
SSX = scaler.transform(X)


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(SSX, y, test_size=0.2,
random_state=7)


X_train.shape, y_train.shape
```

Output:

```
((614, 8), (614,))
```

```
X_test.shape, y_test.shape
```

Output:

```
((154, 8), (154,))
```