IMPORTING RELEVANT LIBRARIES

We import necessary libraries including pandas for data handling, scikit-learn modules for machine learning tasks.

In [20]:

```
import pandas as pd
import numpy as np
import seaborn as sn
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Loading Dataset and Inspecting

We load the Pima Indians Diabetes Datase for inspecting into a pandas DataFrame

In [21]:

```
df = pd.read_csv("C:/Users/Sanayak/Desktop/diabetes.csv")
df.head()
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

In [22]:

```
df.tail()
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

In [23]:

```
df.nunique()
```

Out[23]:

```
Pregnancies                  17
Glucose                     136
BloodPressure                47
SkinThickness                51
Insulin                     186
BMI                         248
DiabetesPedigreeFunction    517
Age                          52
Outcome                       2
dtype: int64
```

In [24]:

```
df.describe()
```

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dia
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

DATA CLEANSING AND DATA MANIPULATION

Our Dataset has zero(0) values for in columns 'Glucose','Bloodpressure','Skinthickness','Insulin' and 'Body Mass Index(BMI)'. We replace the zero(0) values with the median.

```
In [31]: df["BloodPressure"] = df["BloodPressure"].replace(0,72)
```

```
In [30]: df["Glucose"] = df["Glucose"].replace(0,117)
```

```
In [33]: df["SkinThickness"] = df["SkinThickness"].replace(0,23)
```

```
In [34]: df["Insulin"] = df["Insulin"].replace(0,30)
```

```
In [35]: df["BMI"] = df["BMI"].replace(0,32)
```

```
In [36]: df.describe()
```

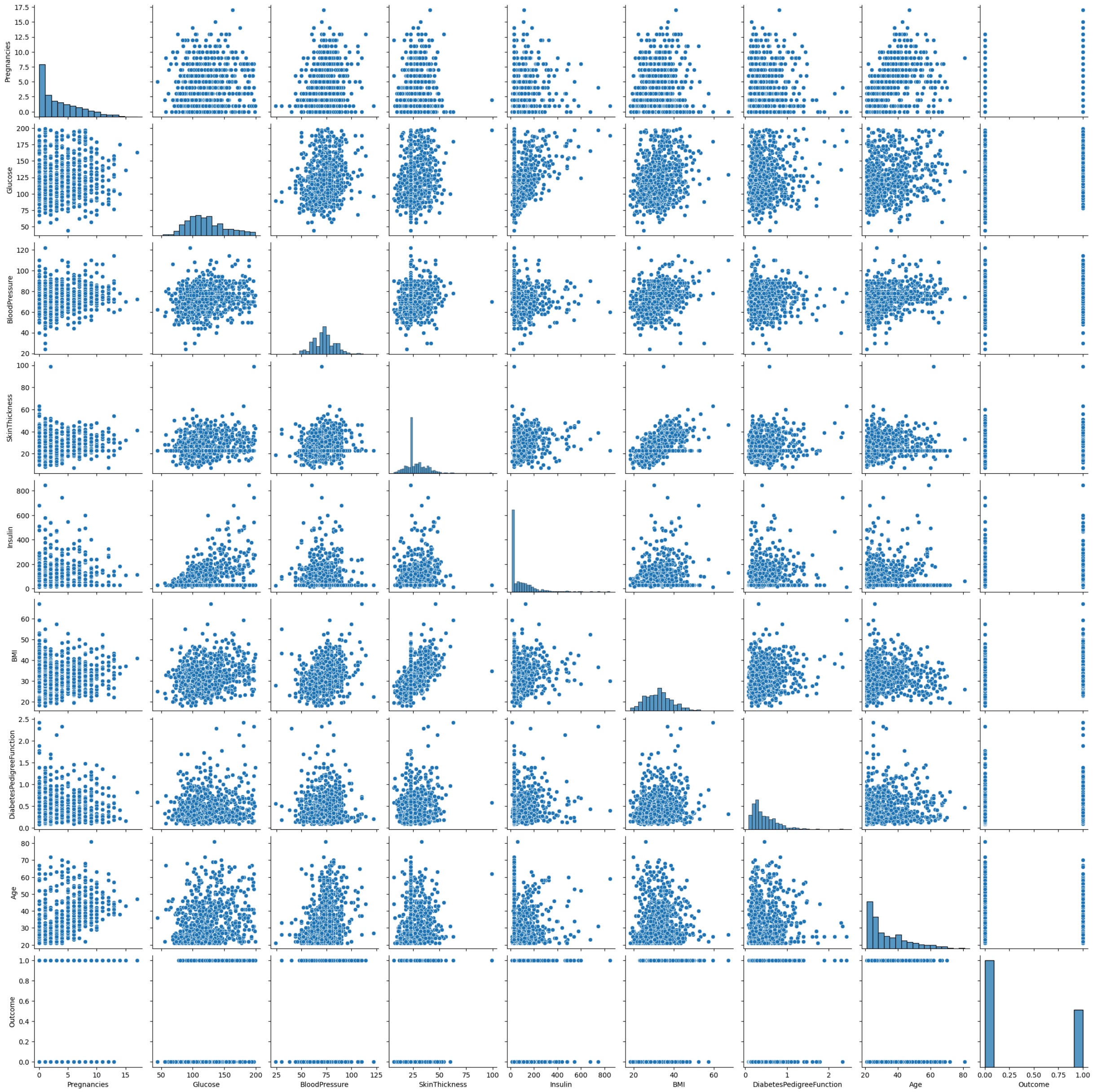
Out[36]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dial
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.656250	72.386719	27.334635	94.408854	32.450911	
std	3.369578	30.438286	12.096642	9.229014	105.695978	6.875366	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.750000	64.000000	23.000000	30.000000	27.500000	
50%	3.000000	117.000000	72.000000	23.000000	31.000000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

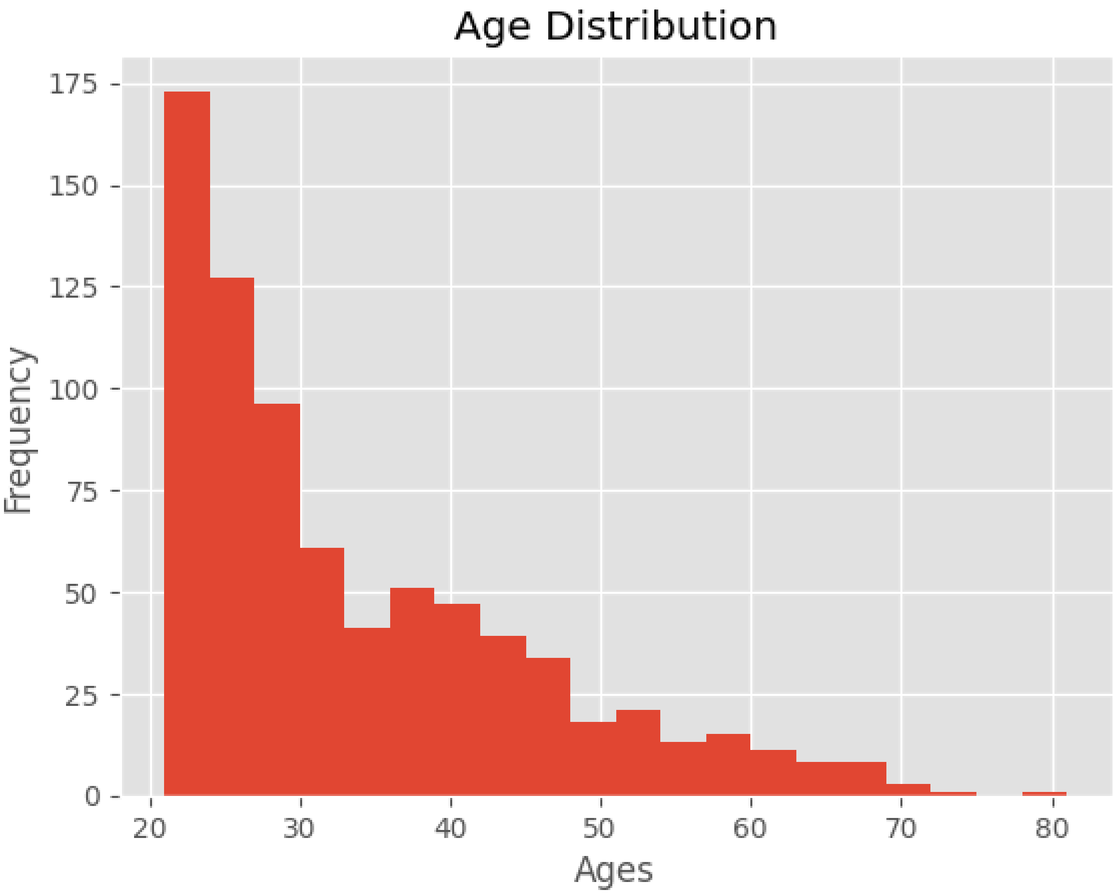
DATA VISUALISATION

```
In [38]: sn.pairplot(df)
```

```
Out[38]: <seaborn.axisgrid.PairGrid at 0x20ccba767e0>
```



```
In [41]: plt.hist(df.Age,bins=20)
         plt.xlabel("Ages")
         plt.ylabel("Frequency")
         plt.title("Age Distribution")
         plt.style.use("fivethirtyeight")
         plt.show()
```



SPLITING DATASET INTO FEATURE AND TARGET VARIABLE

```
In [45]: X = df.drop('Outcome', axis=1)
         y = df['Outcome']
```

SPLITTTING OUR DATASET INTO TRAINING SET AND TESTING SET

```
In [47]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

```
In [ ]: #Standardizing features by removing the mean and scaling to unit variance
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
```

```
In [57]: # Callling Logisticregression setting the maximum iteration to 400
         logreg = LogisticRegression(max_iter=400)
         logreg.fit(X_train, y_train)
         y_pred = logreg.predict(X_test)
         print(y_pred)
```

```
[0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 1 1 1 1
 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0
 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0]
```

```
In [58]: print(confusion_matrix(y_test, y_pred))
         print(classification_report(y_test, y_pred))
```

```
[[82 17]
 [20 35]]
              precision    recall  f1-score   support

           0       0.80      0.83      0.82        99
           1       0.67      0.64      0.65        55

    accuracy                           0.76       154
   macro avg       0.74      0.73      0.74       154
weighted avg       0.76      0.76      0.76       154
```

TRAINED AND PREPARED BY VICTOR INIOBONG Economist | Mathematics Tutor| Data Scientist(In view)