DIABETES PREDICTION USING MACHINE LEARNING

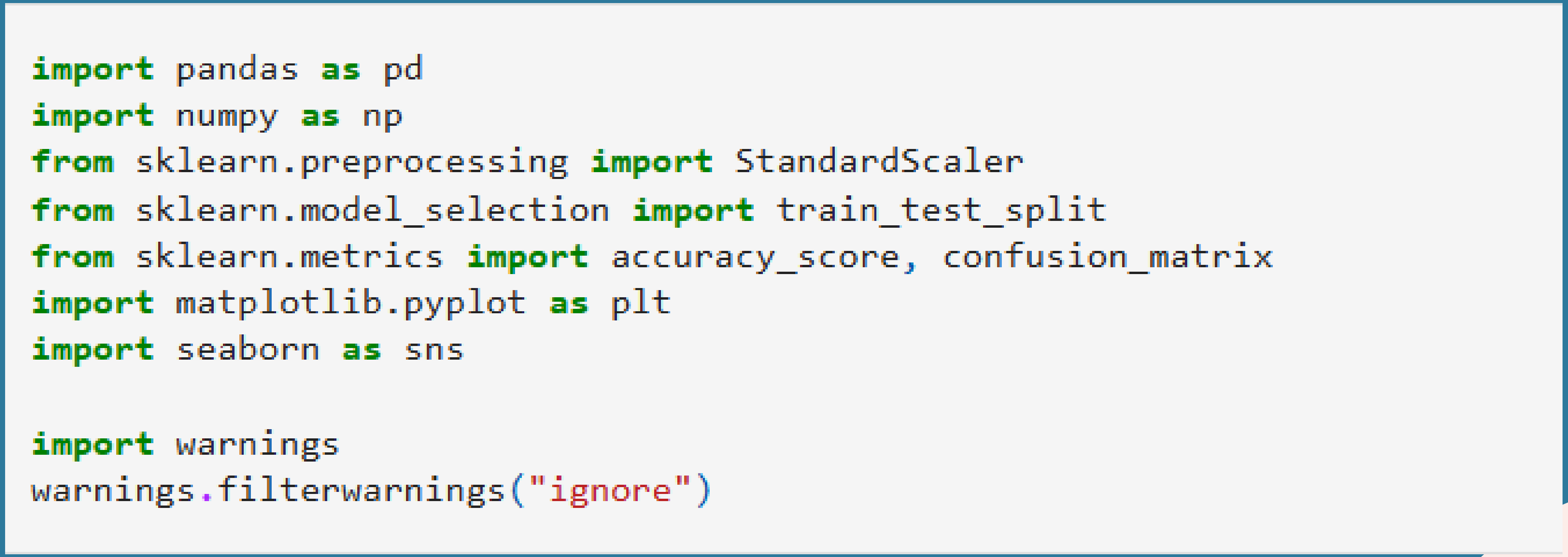
DATASET DESCRIPTION

- **Pregnancies**: This column represents the number of times the individual has been pregnant (if applicable). It's a count variable indicating the gravidity of the individual.
- **Glucose**: This column typically represents the plasma glucose concentration measured in a 2-hour oral glucose tolerance test. It's a crucial indicator for diagnosing diabetes and assessing blood sugar levels.
- **BloodPressure**: This column represents the diastolic blood pressure (mm Hg) of the individual. Diastolic blood pressure is the pressure in the arteries when the heart rests between beats.
- **SkinThickness**: This column represents the thickness of the skinfold measured at the triceps using a caliper (in mm). It's often used as a measure of body fat.
- **Insulin**: This column represents the serum insulin level (mu U/ml) of the individual. Insulin is a hormone produced by the pancreas that regulates blood sugar levels.

DATASET DESCRIPTION

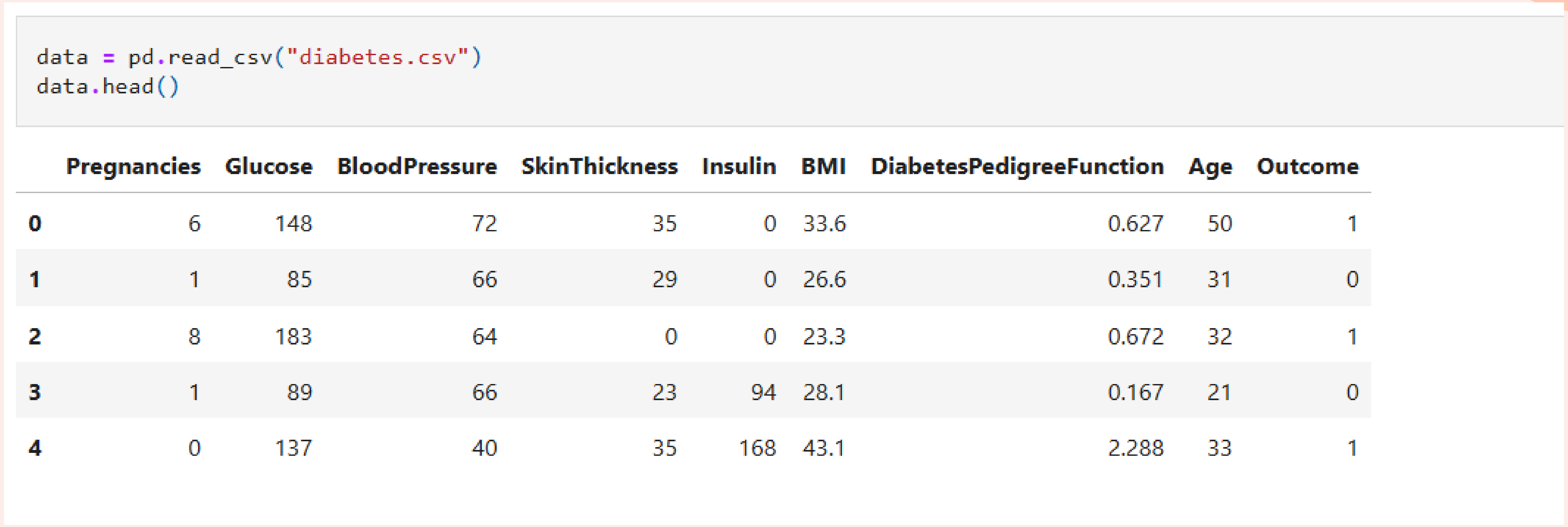
- **BMI (Body Mass Index)**: This column represents the Body Mass Index, which is a measure calculated using an individual's weight (in kg) divided by the square of their height (in meters). It's commonly used as an indicator of body fatness and correlates with the risk of developing obesity-related diseases, including type 2 diabetes.
- **DiabetesPedigreeFunction**: This column represents a function that scores the likelihood of diabetes based on family history. It provides a measure of the genetic influence on diabetes.
- **Age**: This column represents the age of the individual in years.
- **Outcome**: This column typically represents whether an individual has diabetes or not. It's a binary variable where 1 often indicates the presence of diabetes and 0 indicates the absence.

IMPORTING NECESSARY LIBRARIES



```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

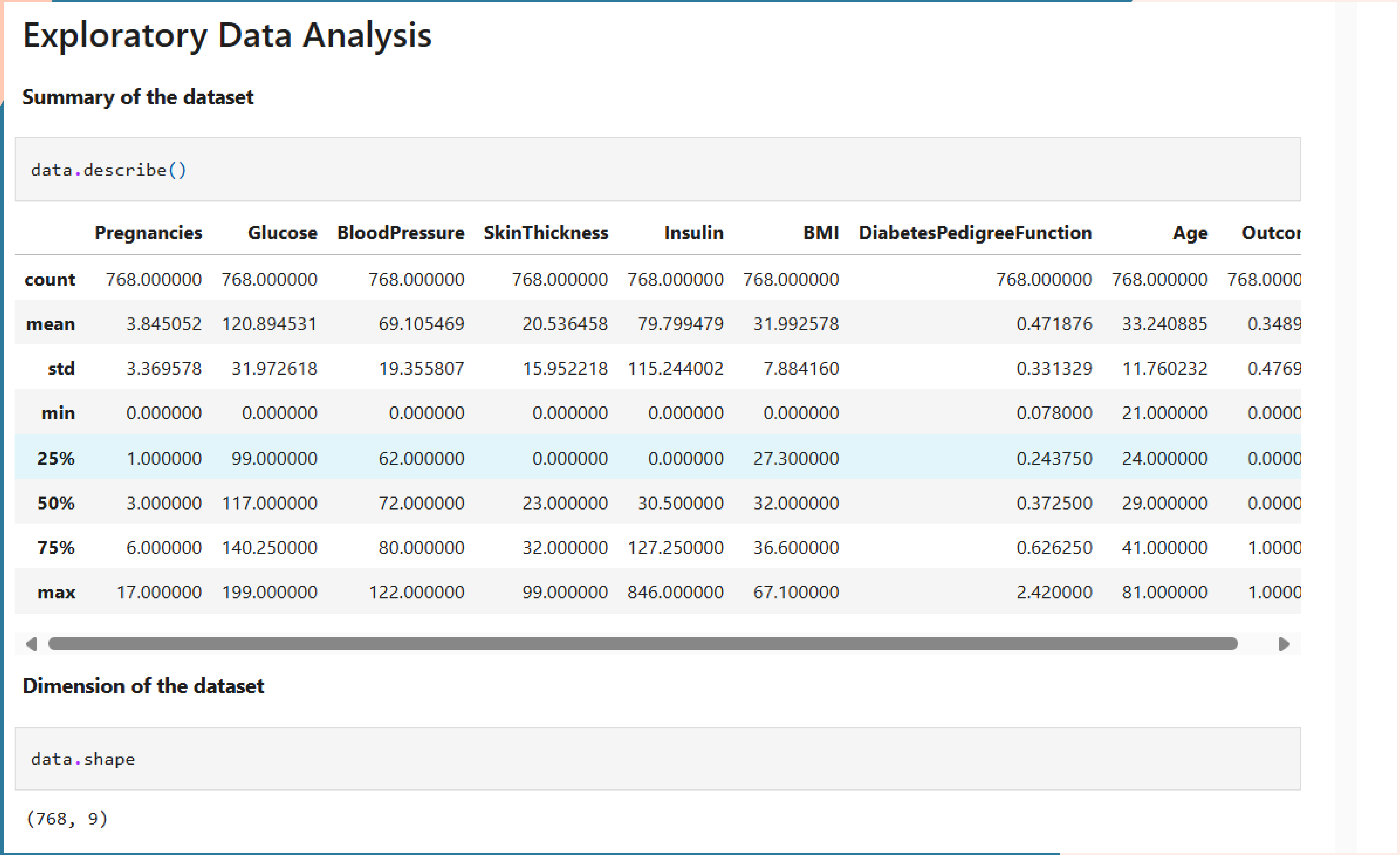
READ THE DATASET



```
data = pd.read_csv("diabetes.csv")
data.head()
```

	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

EDA



Exploratory Data Analysis

Summary of the dataset

```
data.describe()
```

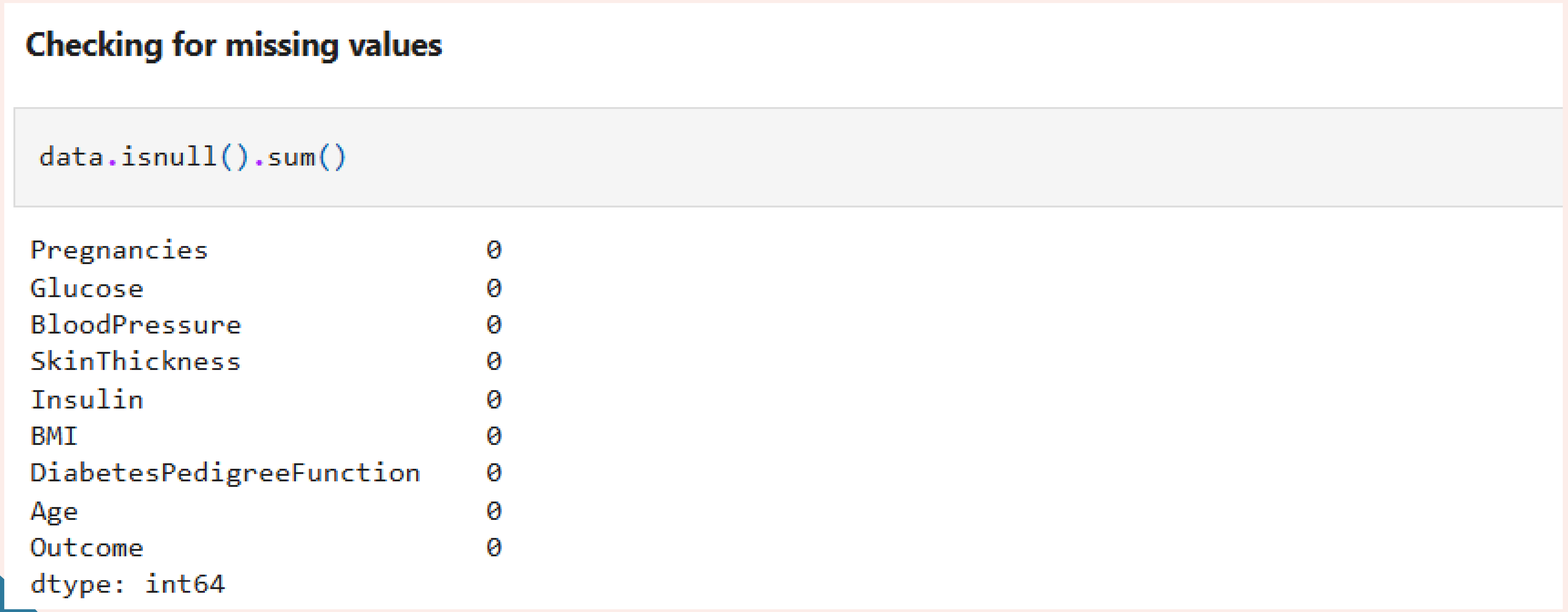
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcor
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.0000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.3489
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.4769
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.0000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.0000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.0000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.0000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.0000

Dimension of the dataset

```
data.shape
```

```
(768, 9)
```

We can see there few data for columns Glucose , Insulin, skin thickenss, BMI and Blood Pressure which have value as 0. That's not possible,right? you can do a quick search to see that one cannot have 0 values for these. Let's deal with that. we can either remove such data or simply replace it with their respective mean values. Let's do the latter.



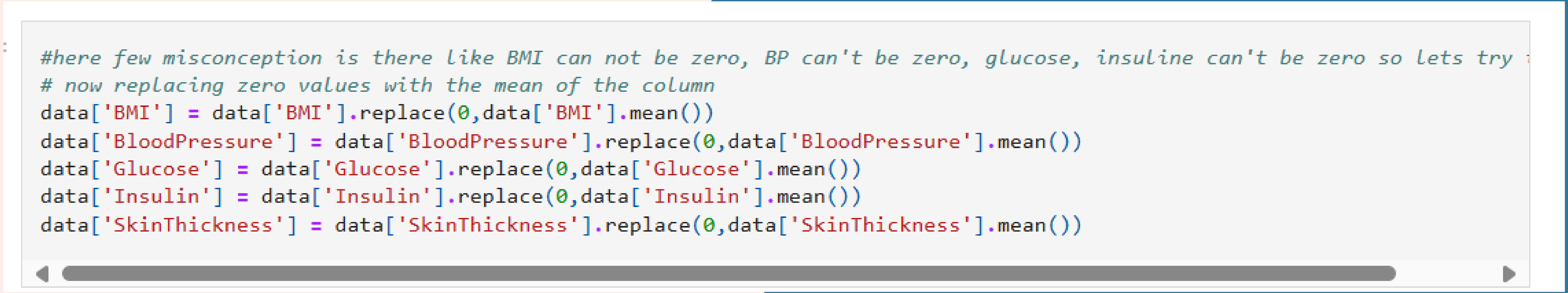
Checking for missing values

```
data.isnull().sum()
```

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

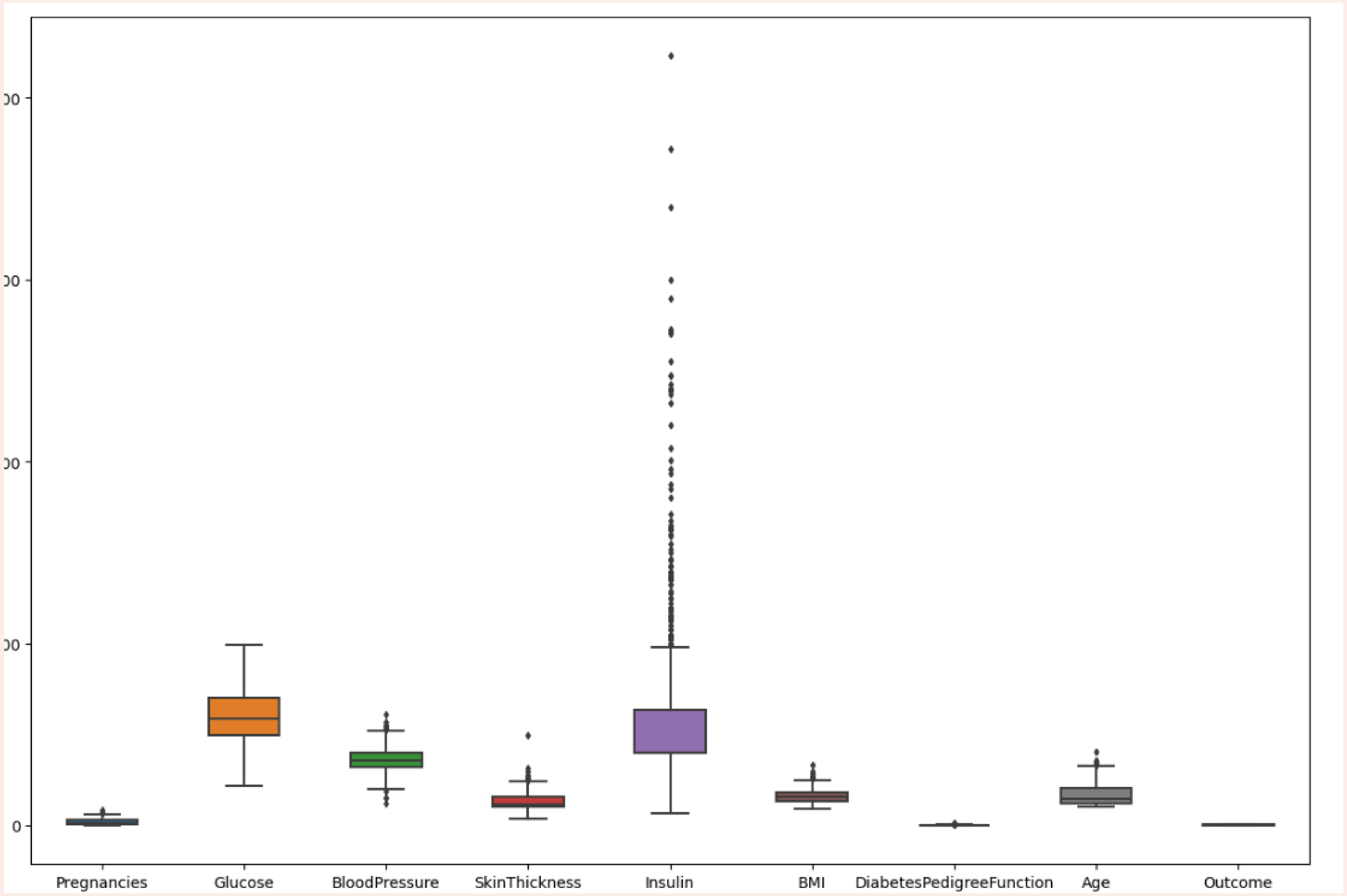
There is no missing values in are dataset

Now replacing zero values with the mean of the column



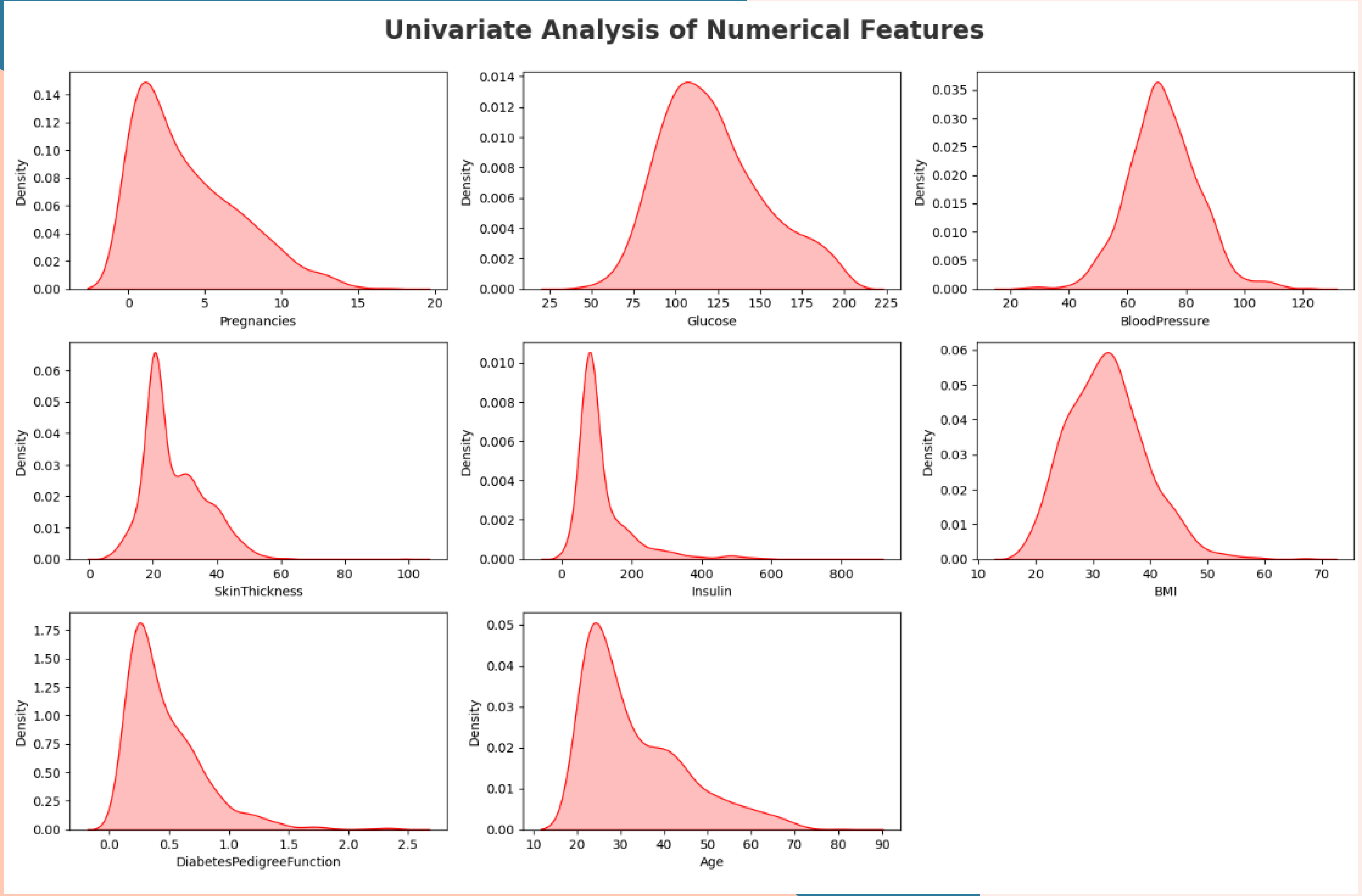
```
#here few misconception is there like BMI can not be zero, BP can't be zero, glucose, insuline can't be zero so lets try
# now replacing zero values with the mean of the column
data['BMI'] = data['BMI'].replace(0,data['BMI'].mean())
data['BloodPressure'] = data['BloodPressure'].replace(0,data['BloodPressure'].mean())
data['Glucose'] = data['Glucose'].replace(0,data['Glucose'].mean())
data['Insulin'] = data['Insulin'].replace(0,data['Insulin'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0,data['SkinThickness'].mean())
```

Now we have dealt with the 0 values and data looks better. But, there still are outliers present in some columns.lets visualize it

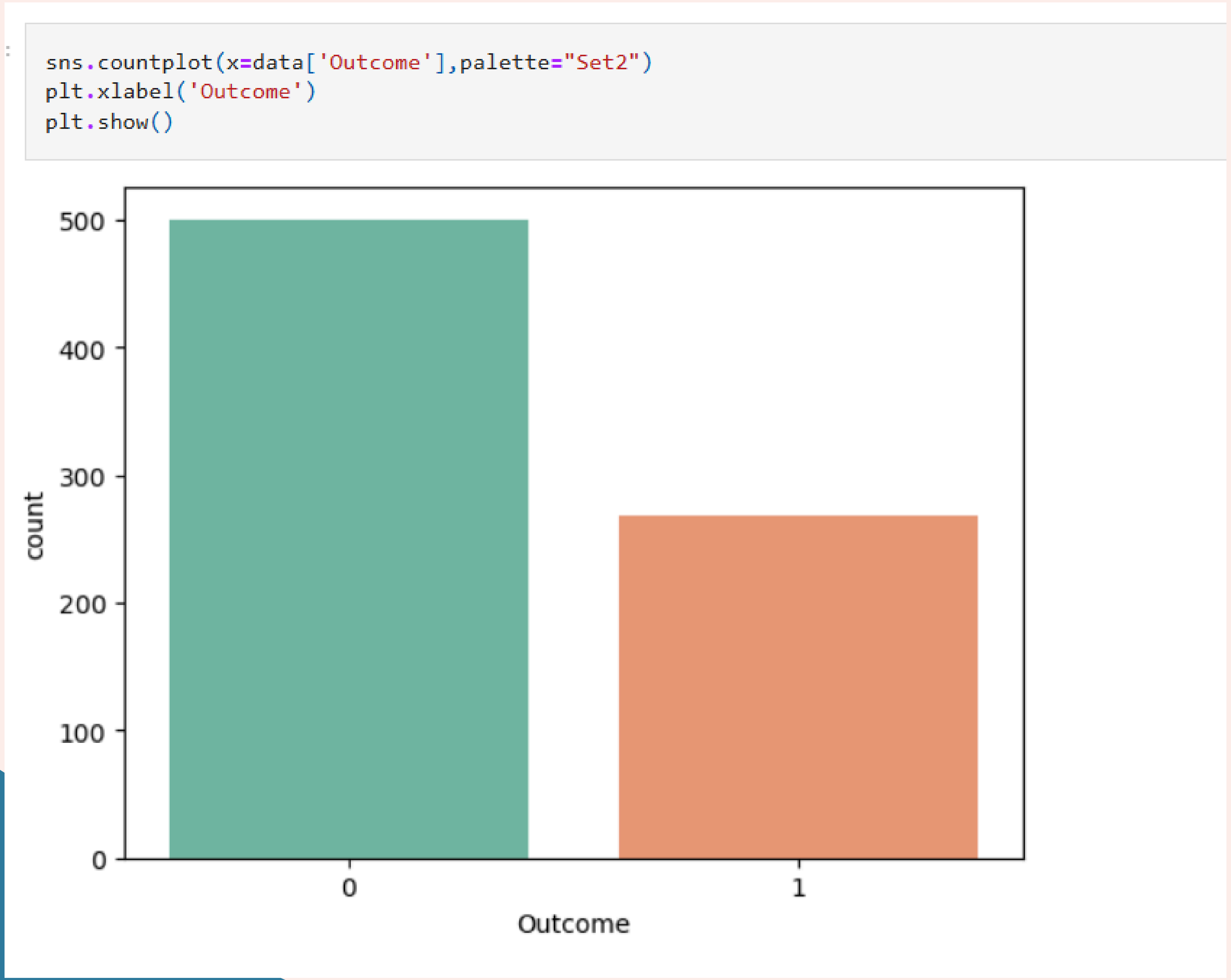


The dataset exhibits the presence of outliers in the insulin variable, with a maximum value exceeding 800, while the overall range for insulin values predominantly lies between 0 and 200.

UNIVARIATE ANALYSIS OF NUMERICAL FEATURES



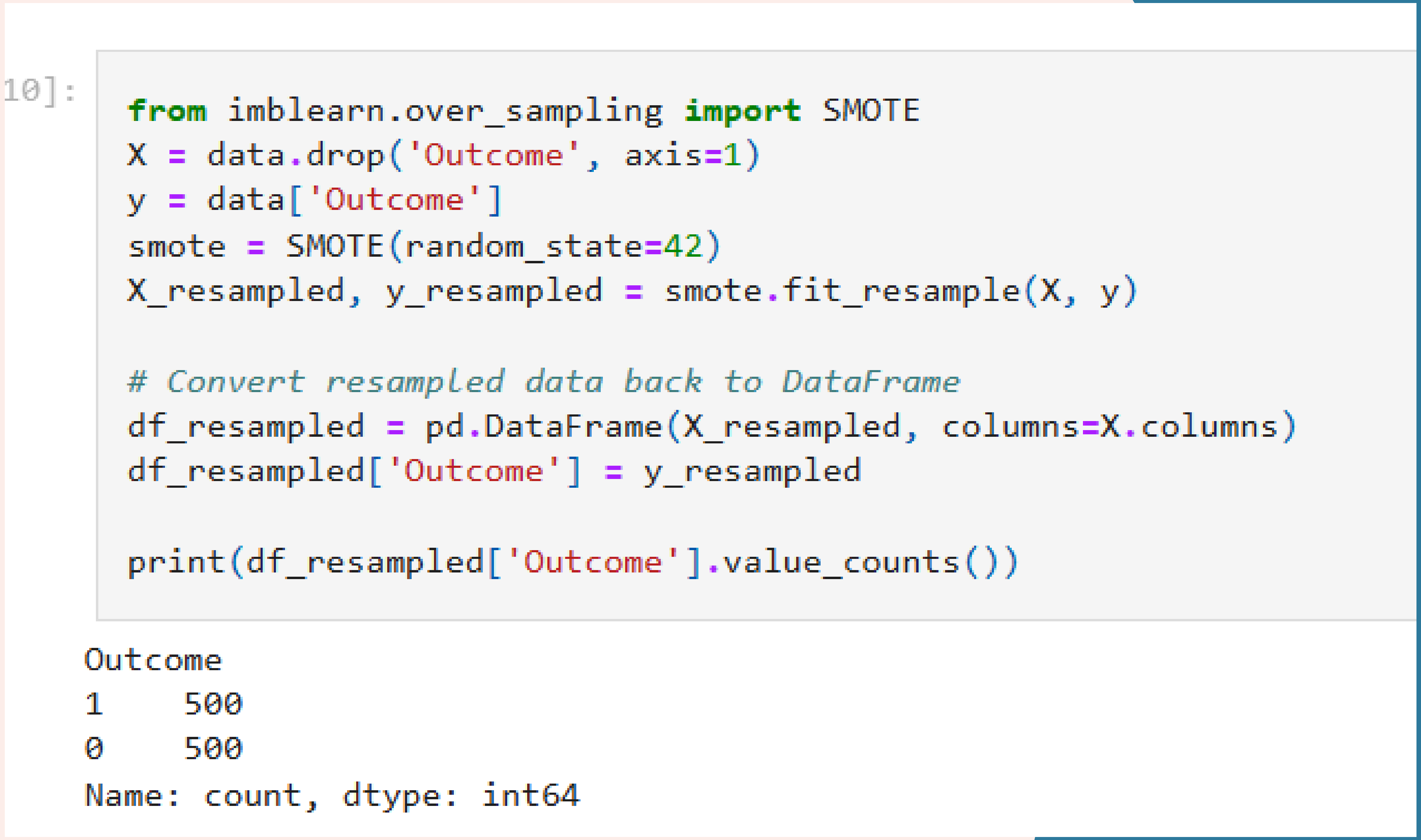
The dataset demonstrates distinct distribution patterns among its variables. Pregnancies, SkinThickness, Insulin, DiabetesPedigreeFunction, and Age show right-skewed distributions, indicating a concentration of lower values with a long tail on the right. Conversely, Glucose, BloodPressure, and BMI exhibit normal distributions, characterized by their bell-shaped curves and symmetry around the mean.



```
sns.countplot(x=data['Outcome'],palette="Set2")
plt.xlabel('Outcome')
plt.show()
```

In this dataset, there are 500 non-diabetic cases compared to 280 diabetic cases. This imbalance can potentially affect the performance of machine learning models, as they may become biased towards the majority class (non-diabetic) and underperform in predicting the minority class (diabetic)

HANDLING IMBALANCED DATASET

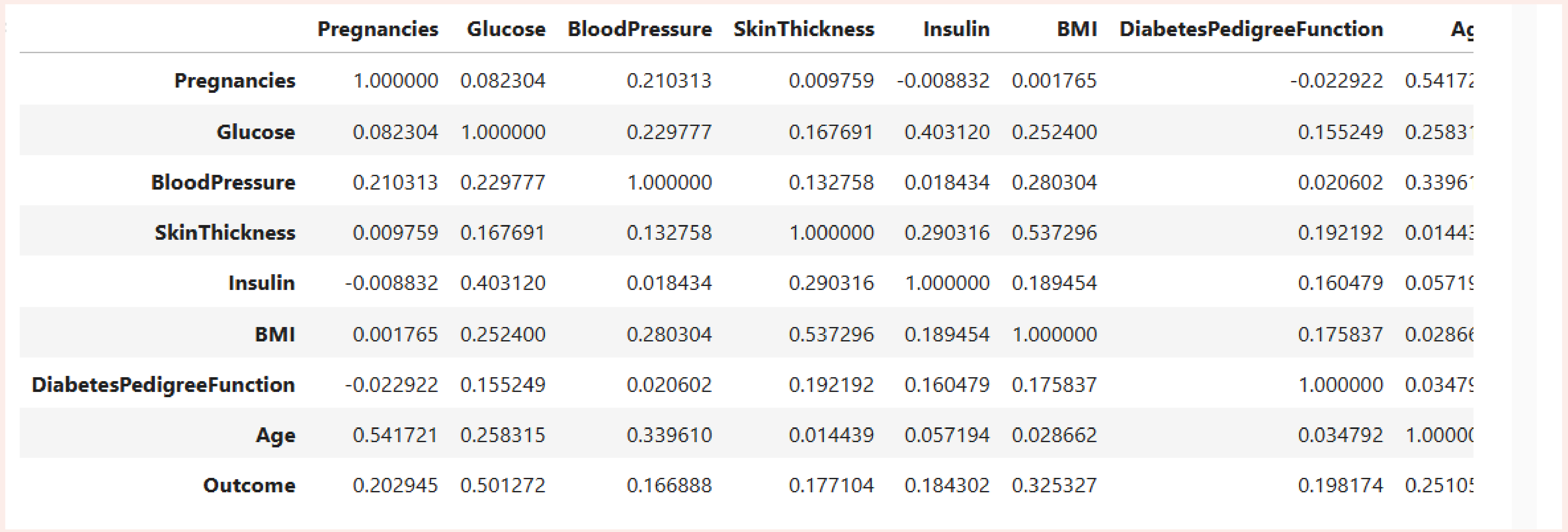


```
10]:
```

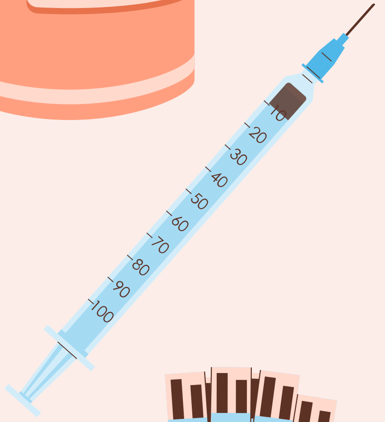
```python
from imblearn.over_sampling import SMOTE
X = data.drop('Outcome', axis=1)
y = data['Outcome']
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Convert resampled data back to DataFrame
df_resampled = pd.DataFrame(X_resampled, columns=X.columns)
df_resampled['Outcome'] = y_resampled

print(df_resampled['Outcome'].value_counts())
```

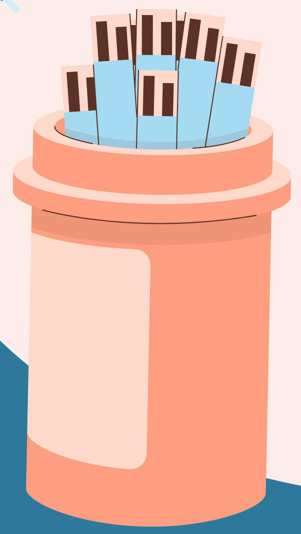
```
Outcome
1    500
0    500
Name: count, dtype: int64
```

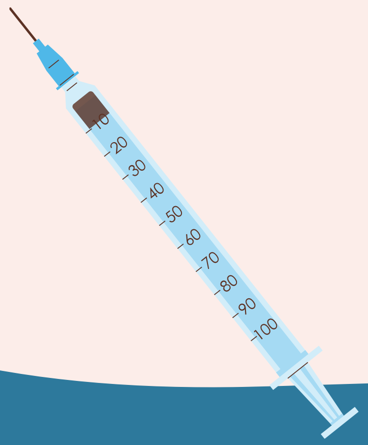
CORRELATION



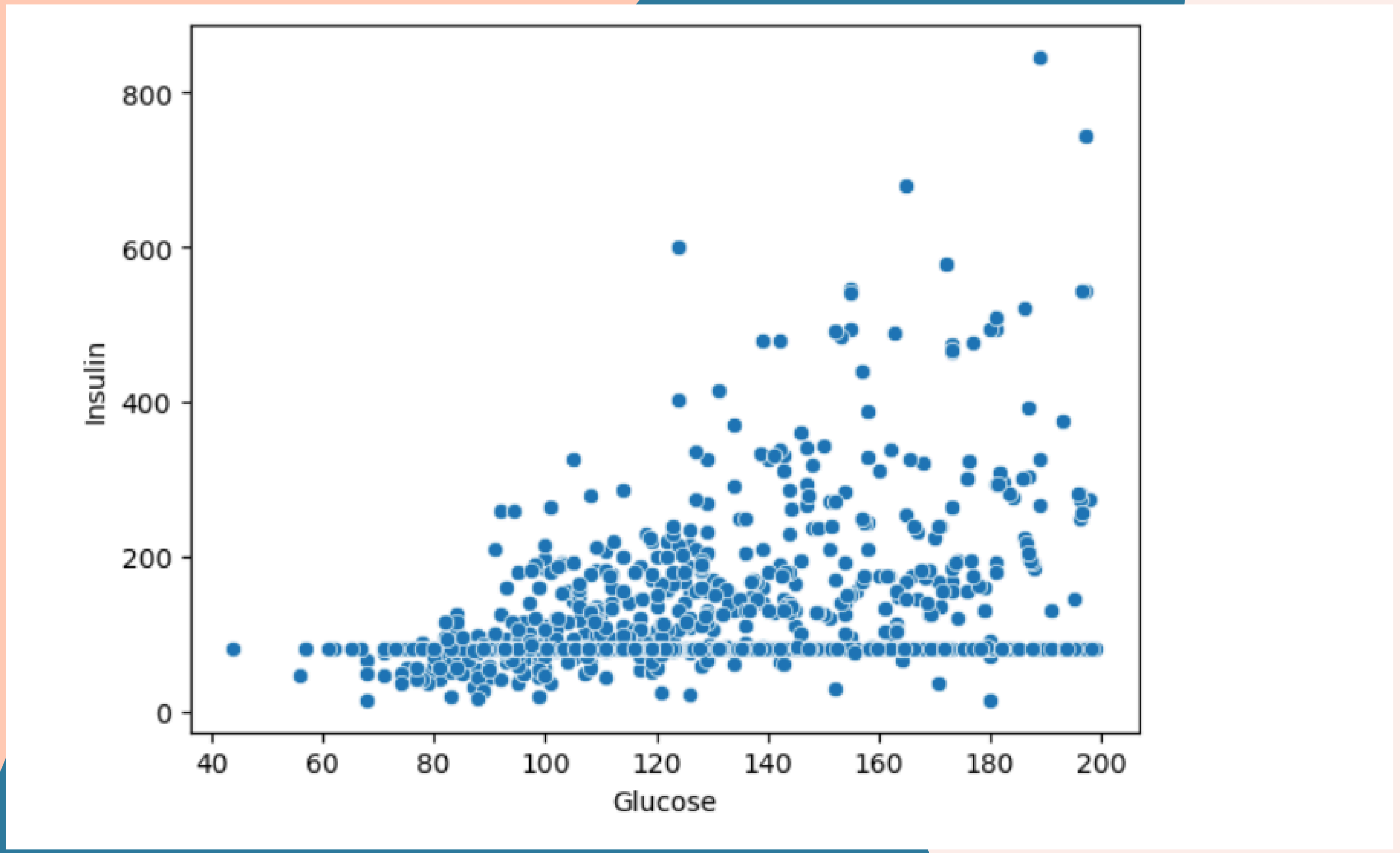
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
Pregnancies	1.000000	0.082304	0.210313	0.009759	-0.008832	0.001765	-0.022922	0.5417
Glucose	0.082304	1.000000	0.229777	0.167691	0.403120	0.252400	0.155249	0.2583
BloodPressure	0.210313	0.229777	1.000000	0.132758	0.018434	0.280304	0.020602	0.3396
SkinThickness	0.009759	0.167691	0.132758	1.000000	0.290316	0.537296	0.192192	0.0144
Insulin	-0.008832	0.403120	0.018434	0.290316	1.000000	0.189454	0.160479	0.0571
BMI	0.001765	0.252400	0.280304	0.537296	0.189454	1.000000	0.175837	0.0286
DiabetesPedigreeFunction	-0.022922	0.155249	0.020602	0.192192	0.160479	0.175837	1.000000	0.0347
Age	0.541721	0.258315	0.339610	0.014439	0.057194	0.028662	0.034792	1.0000
Outcome	0.202945	0.501272	0.166888	0.177104	0.184302	0.325327	0.198174	0.2510



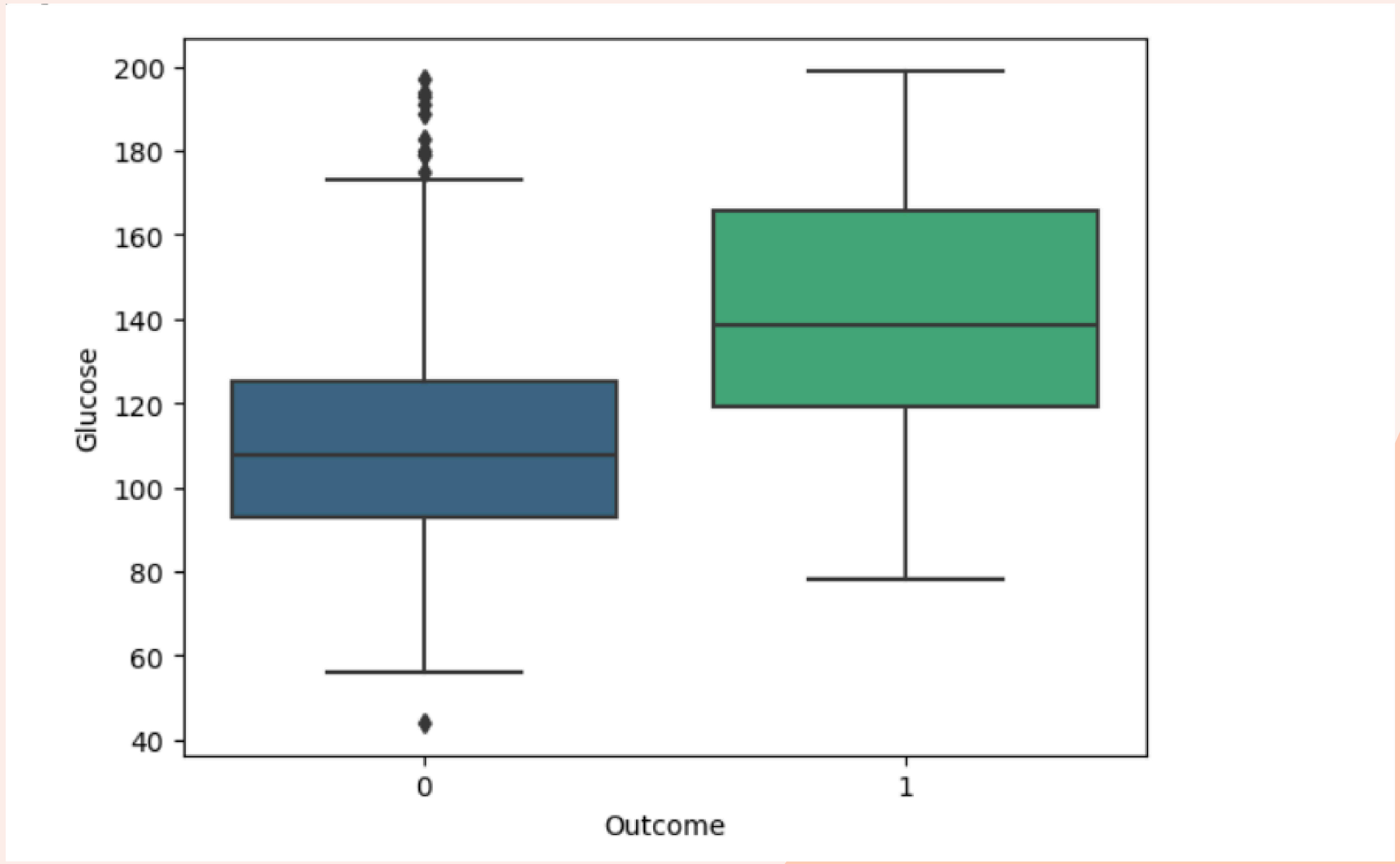




The dataset reveals a 0.53 correlation between BMI and SkinThickness, indicating a moderate positive relationship. Additionally, there is a 0.54 correlation between Age and Pregnancies, suggesting a similar moderate positive association between these variables.

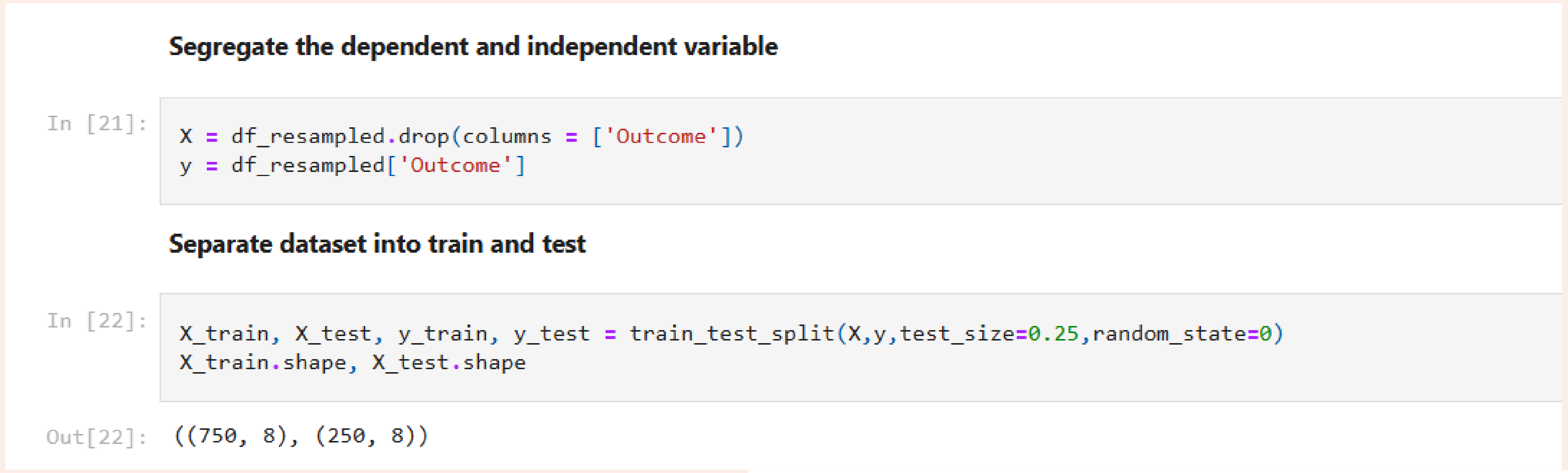


The data demonstrates a predominantly positive correlation between glucose and insulin, indicating that insulin tends to increase as glucose levels rise.



Diabetic individuals commonly present with glucose levels ranging from 80 to 200 mg/dL, which contrasts with the narrower range observed in non-diabetic individuals.

SPLITTING THE DATA



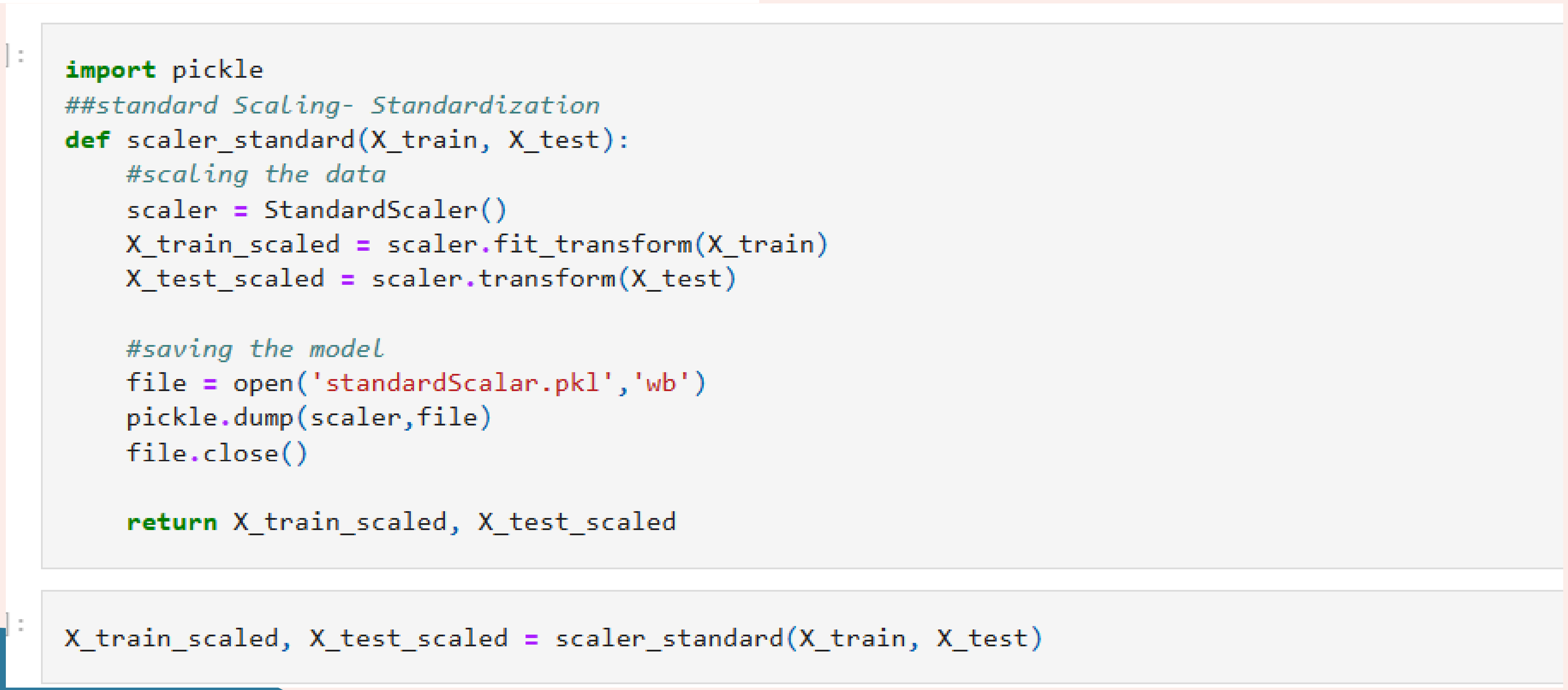
Segregate the dependent and independent variable

```
In [21]:  X = df_resampled.drop(columns = ['Outcome'])
          y = df_resampled['Outcome']
```

Separate dataset into train and test

```
In [22]:  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
          X_train.shape, X_test.shape

Out[22]:  ((750, 8), (250, 8))
```

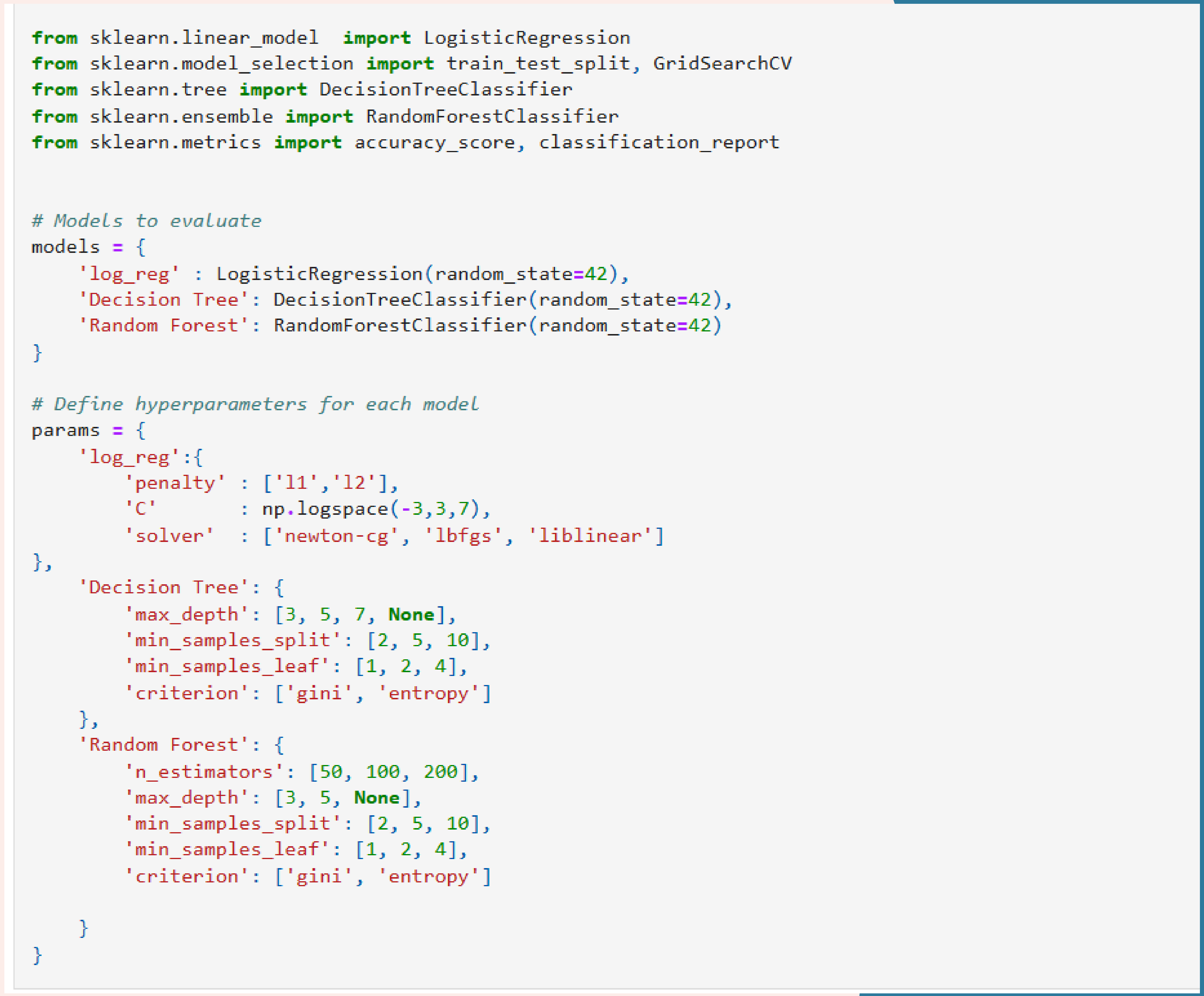
FEATURE SCALING



```
]:  import pickle
    ##standard Scaling- Standardization
    def scaler_standard(X_train, X_test):
        #scaling the data
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        #saving the model
        file = open('standardScalar.pkl','wb')
        pickle.dump(scaler,file)
        file.close()

        return X_train_scaled, X_test_scaled
```

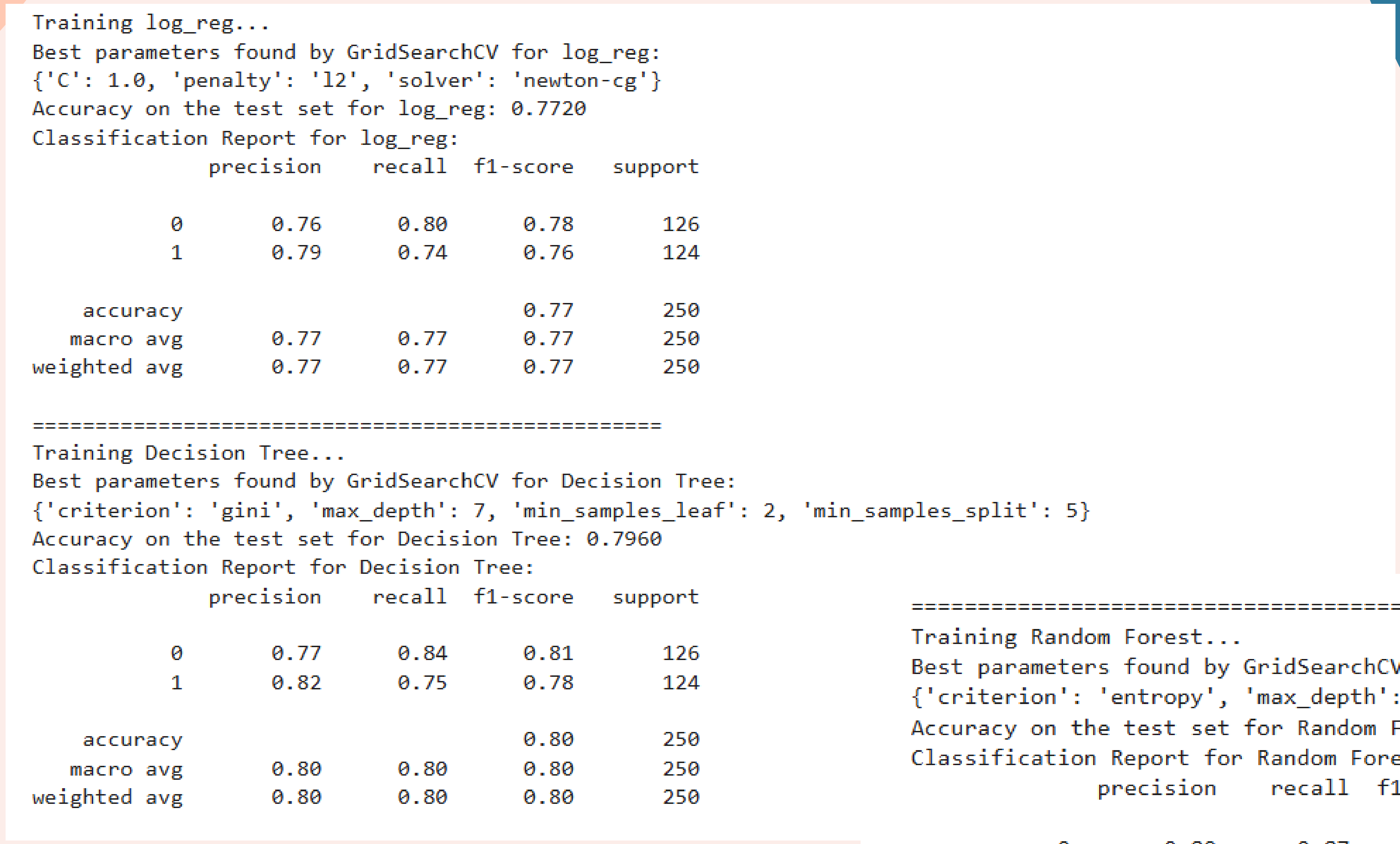
```
]:  X_train_scaled, X_test_scaled = scaler_standard(X_train, X_test)
```

MODEL BUILDING

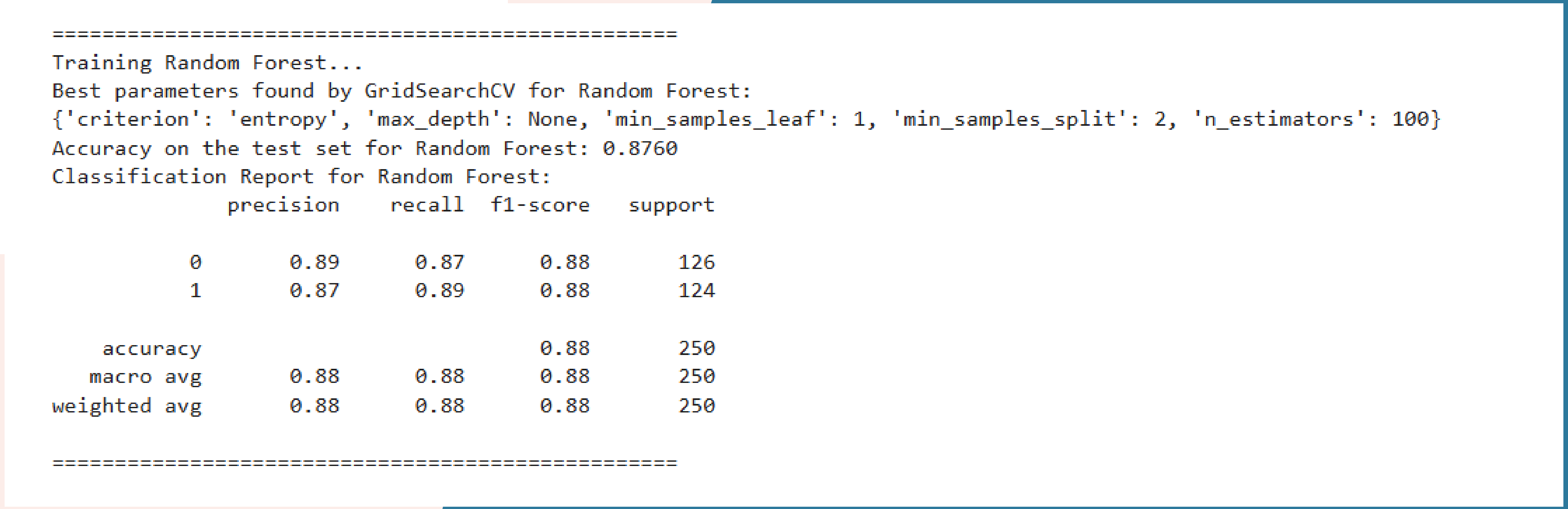


```
from sklearn.linear_model  import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report


# Models to evaluate
models = {
    'log_reg' : LogisticRegression(random_state=42),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42)
}

# Define hyperparameters for each model
params = {
    'log_reg':{
        'penalty' : ['l1','l2'],
        'C'       : np.logspace(-3,3,7),
        'solver'  : ['newton-cg', 'lbfgs', 'liblinear']
},
    'Decision Tree': {
        'max_depth': [3, 5, 7, None],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'criterion': ['gini', 'entropy']
    },
    'Random Forest': {
        'n_estimators': [50, 100, 200],
        'max_depth': [3, 5, None],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'criterion': ['gini', 'entropy']

    }
}
```

MODEL EVALUATION

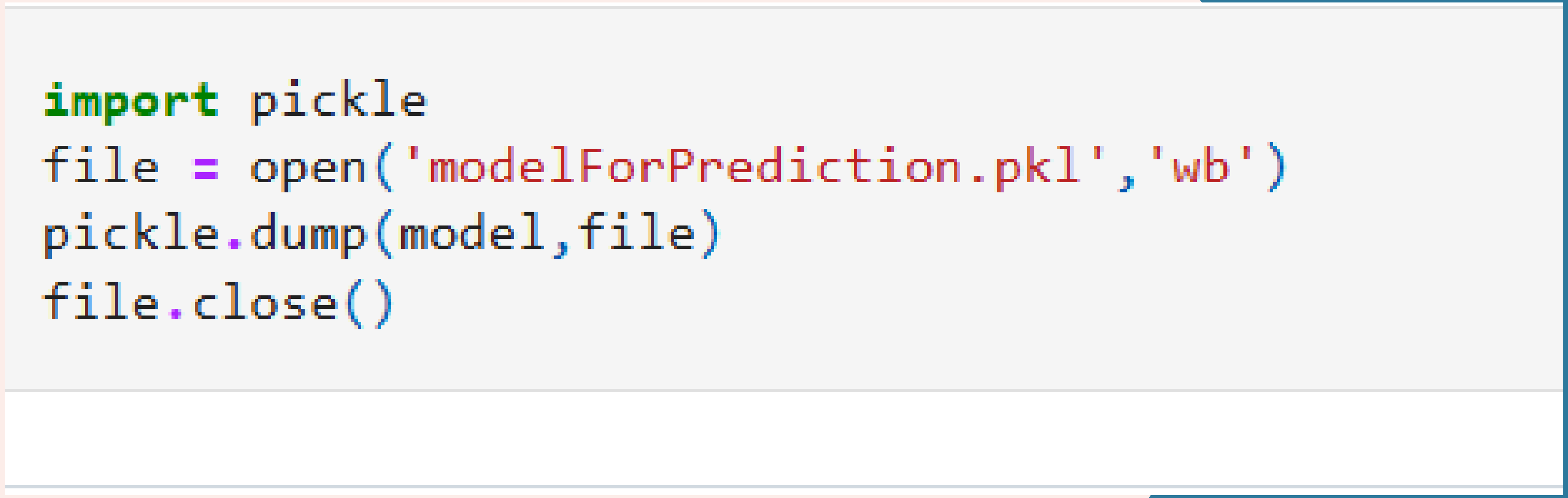


```
Training log_reg...
Best parameters found by GridSearchCV for log_reg:
{'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy on the test set for log_reg: 0.7720
Classification Report for log_reg:
              precision    recall  f1-score   support

           0       0.76      0.80      0.78       126
           1       0.79      0.74      0.76       124

    accuracy                           0.77       250
   macro avg       0.77      0.77      0.77       250
weighted avg       0.77      0.77      0.77       250

==================================================
Training Decision Tree...
Best parameters found by GridSearchCV for Decision Tree:
{'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 2, 'min_samples_split': 5}
Accuracy on the test set for Decision Tree: 0.7960
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.77      0.84      0.81       126
           1       0.82      0.75      0.78       124

    accuracy                           0.80       250
   macro avg       0.80      0.80      0.80       250
weighted avg       0.80      0.80      0.80       250
```



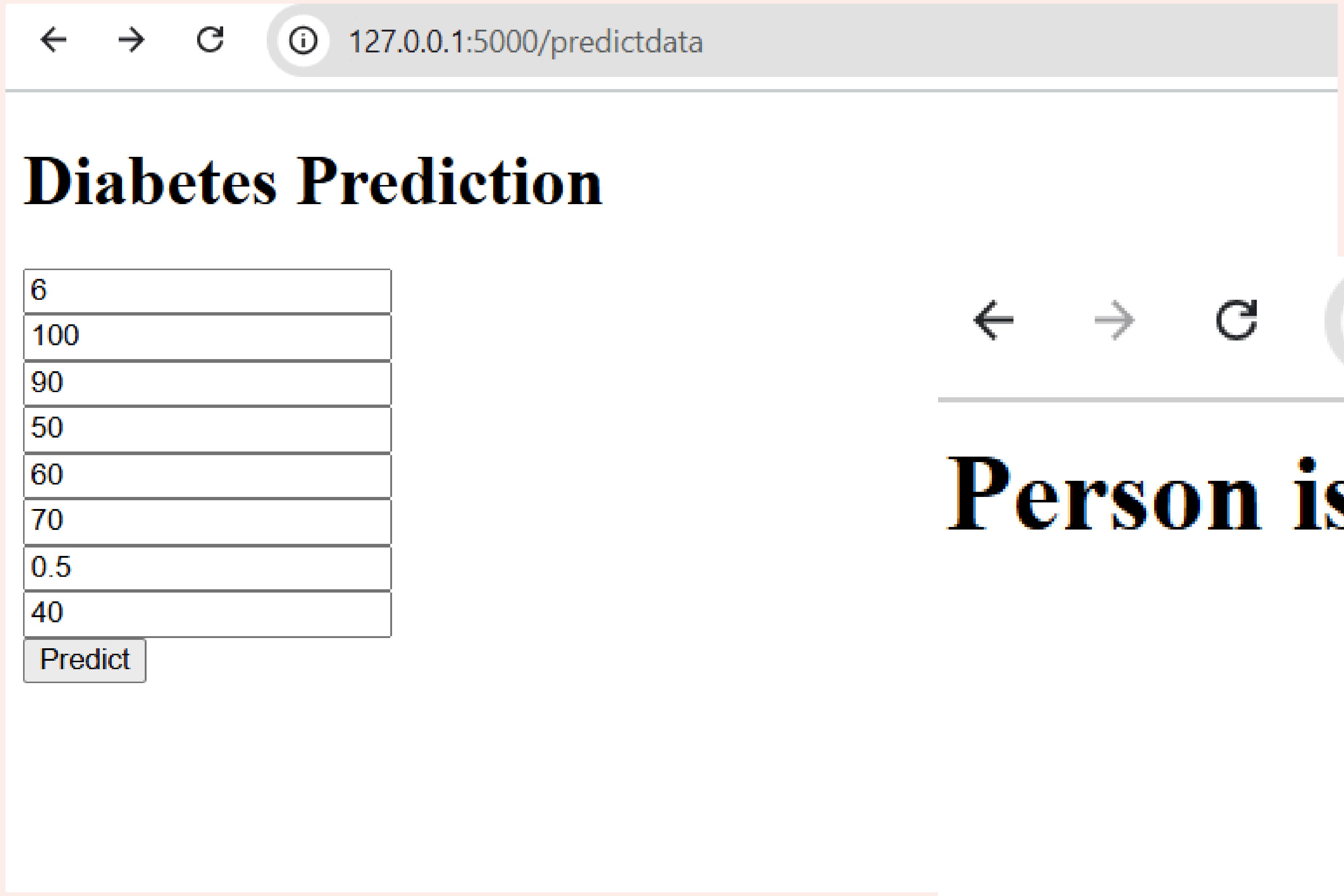
```
==================================================
Training Random Forest...
Best parameters found by GridSearchCV for Random Forest:
{'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Accuracy on the test set for Random Forest: 0.8760
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88       126
           1       0.87      0.89      0.88       124

    accuracy                           0.88       250
   macro avg       0.88      0.88      0.88       250
weighted avg       0.88      0.88      0.88       250

==================================================
```

MODEL SAVING



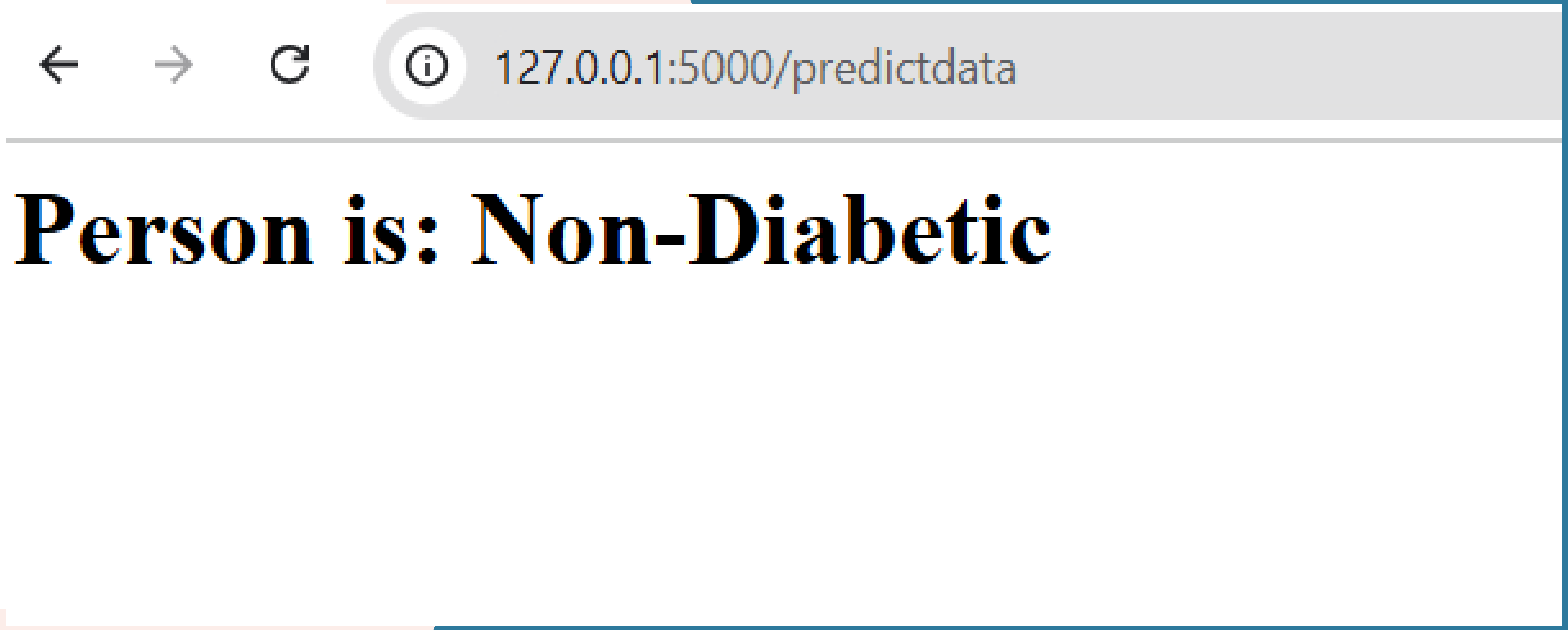
```
import pickle
file = open('modelForPrediction.pkl','wb')
pickle.dump(model,file)
file.close()
```

USER INTERFACE


127.0.0.1:5000/predictdata

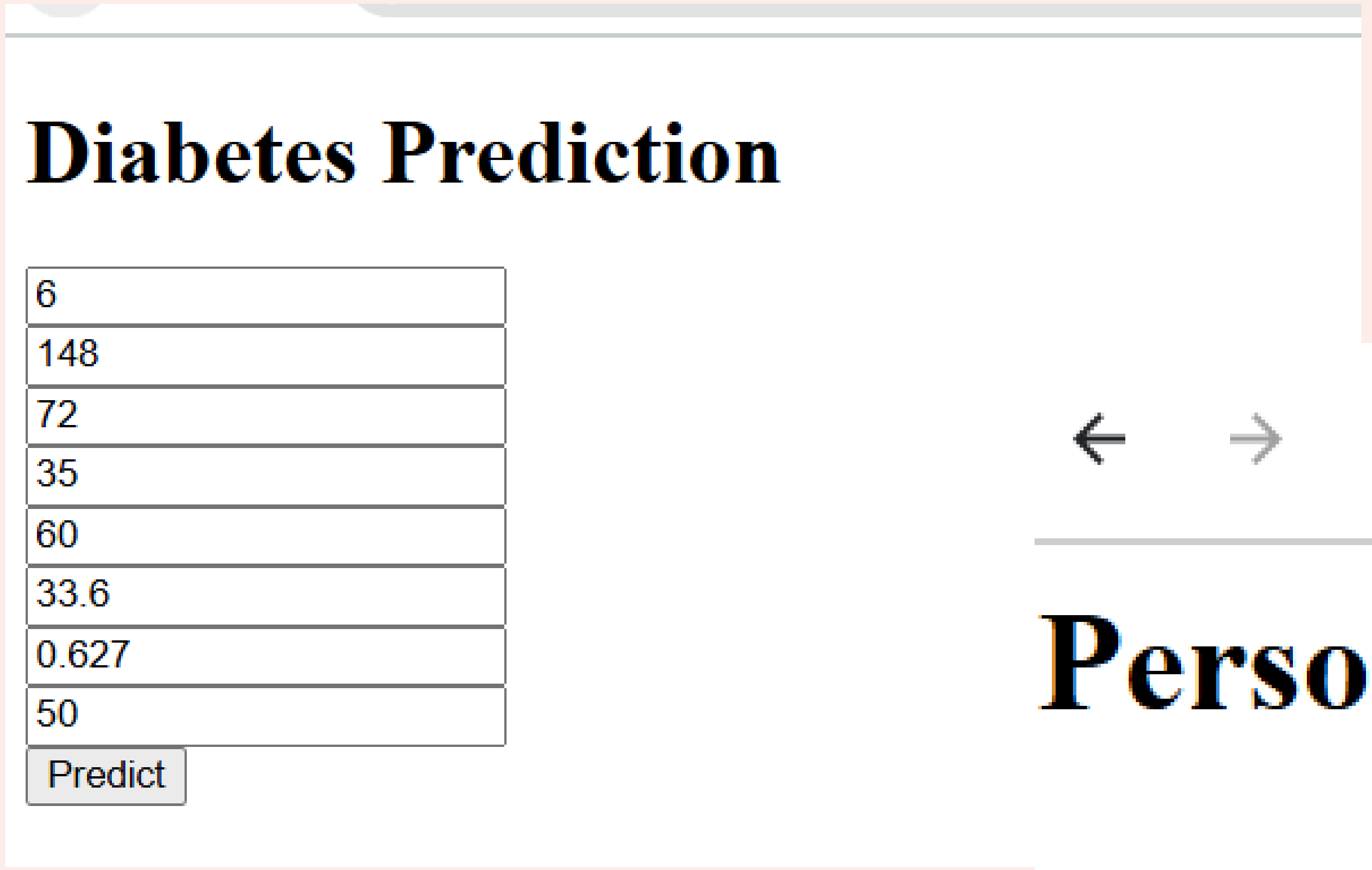
Diabetes Prediction

```
6
100
90
50
60
70
0.5
40
```

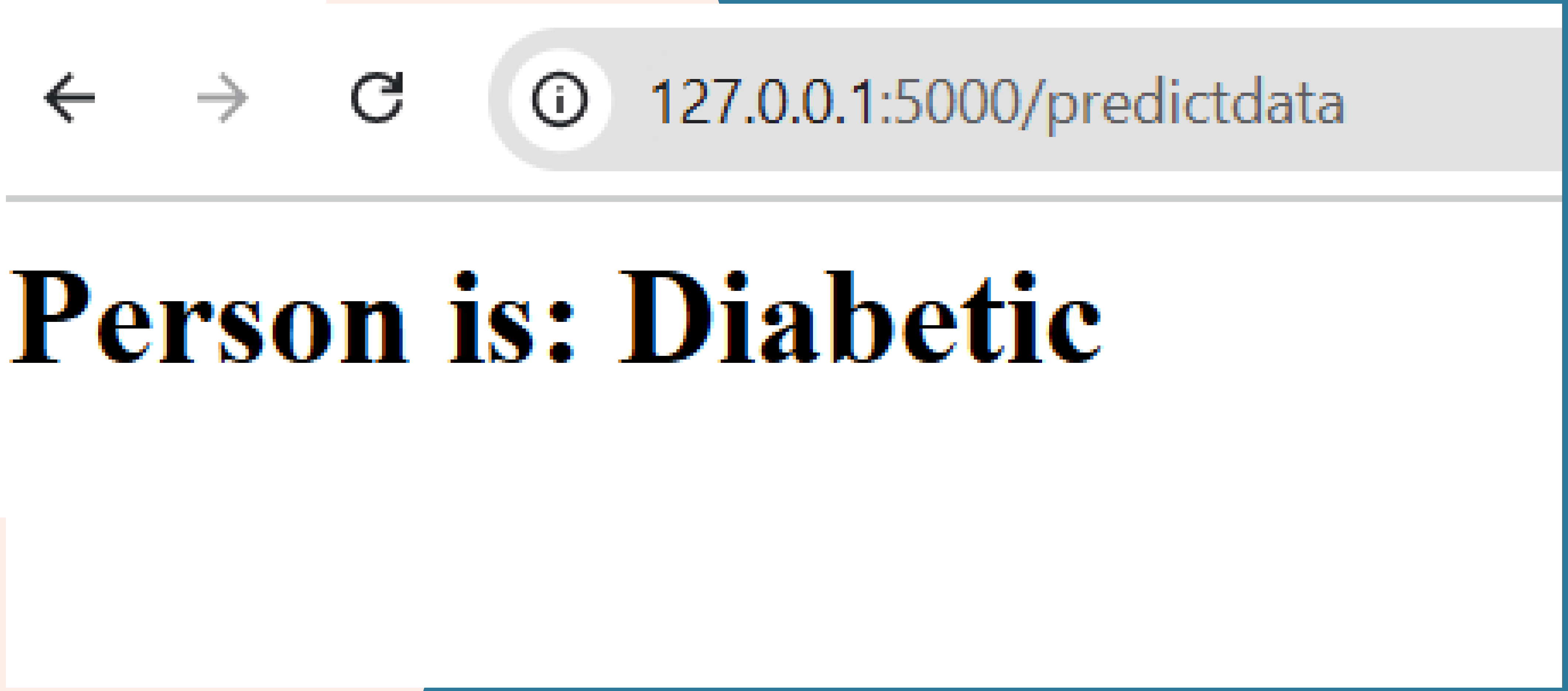

127.0.0.1:5000/predictdata

Person is: Non-Diabetic

USER INTERFACE


Diabetes Prediction

6
148
72
35
60
33.6
0.627
50



Person is: Diabetic

THANK YOU!