**Data preprocessing**

|  |  |
| --- | --- |
| Date | 11 October 2023 |
| Team id | proj-212176-Team-2 |
| Project Name | AI based Diabetes Prediction System |
| Maximum mark |  |

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining tasks.

**Program :**

**Import the necessary libraries:**

Numpy,pandas,sklearn ,matplotlib.pyplot

**Explaination:**

* Numpy :(import numpy as np) a library for mathematcal operatons and handling arrays.
* pandas :(import pandas as pd) a library for data manipulaton and analysis.
* Matplotlib.pyplot: (import as plt) a library for creatng visualiiaton.
* sklearn ( preproccesing and evaluate model )

**code:**

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler , Normalizer

from sklearn.compose import make\_column\_transformer, make\_column\_selector from sklearn.model\_selection import train\_test\_split

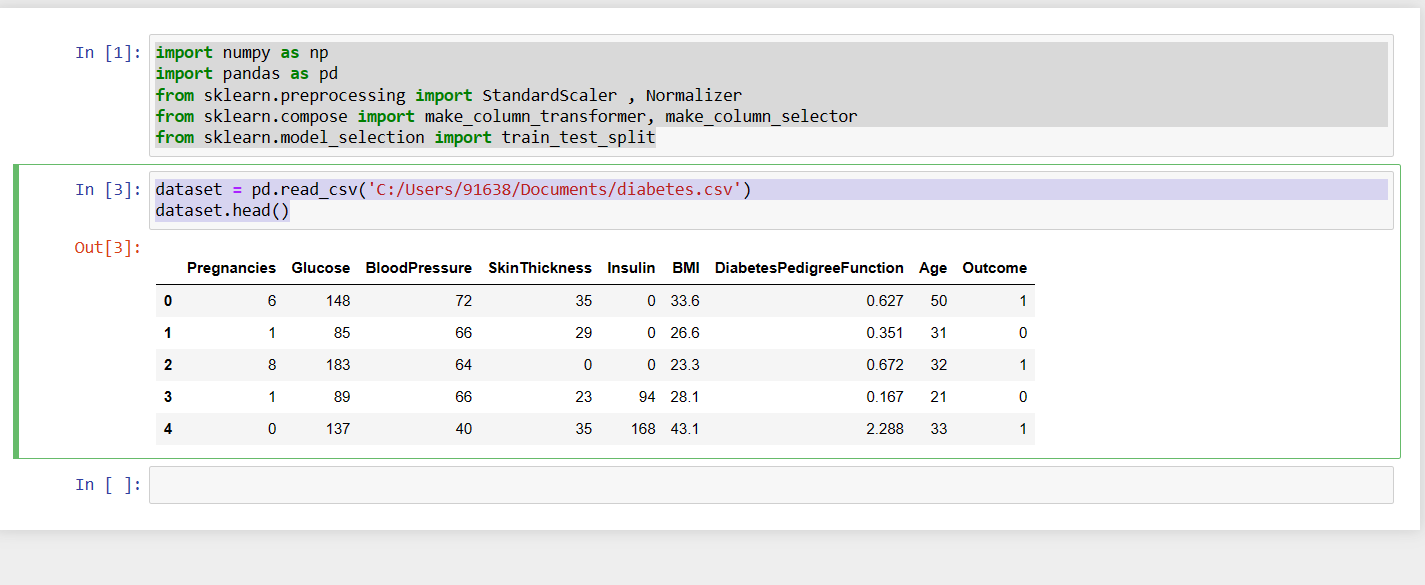
**Import the dataset**

dataset = pd.read\_csv('C:/Users/91638/Documents/diabetes.csv')

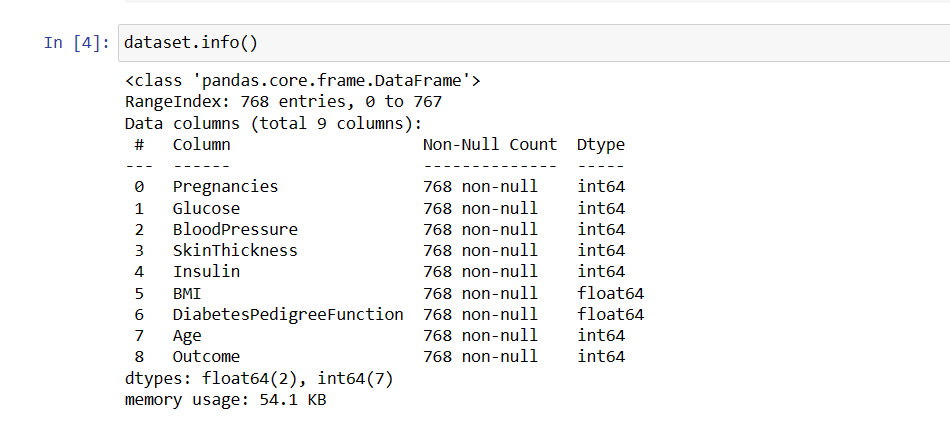
**Data preprocessing**

Data preprocessing is a critical step in building an AI-based diabetes detection model. Properly processed data can significantly impact the performance of your model.

dataset.head()



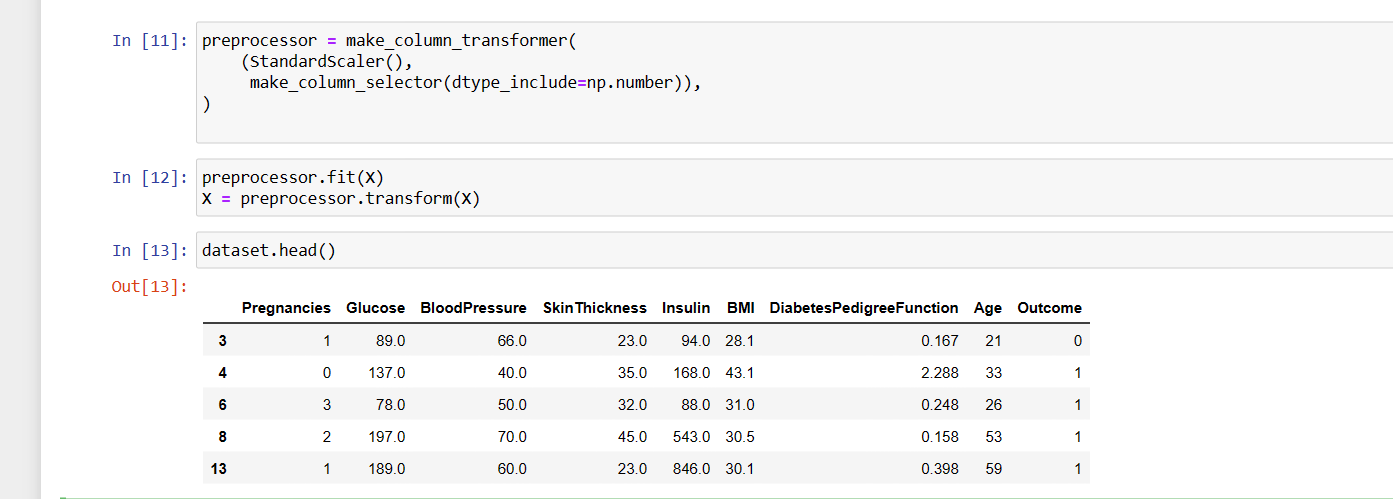
dataset.info()



preprocessor=make\_column\_transformer((StandardScaler(),make\_column\_selector(dtype\_include=np.number)),)

preprocessor.fit(X)

X = preprocessor.transform(X)

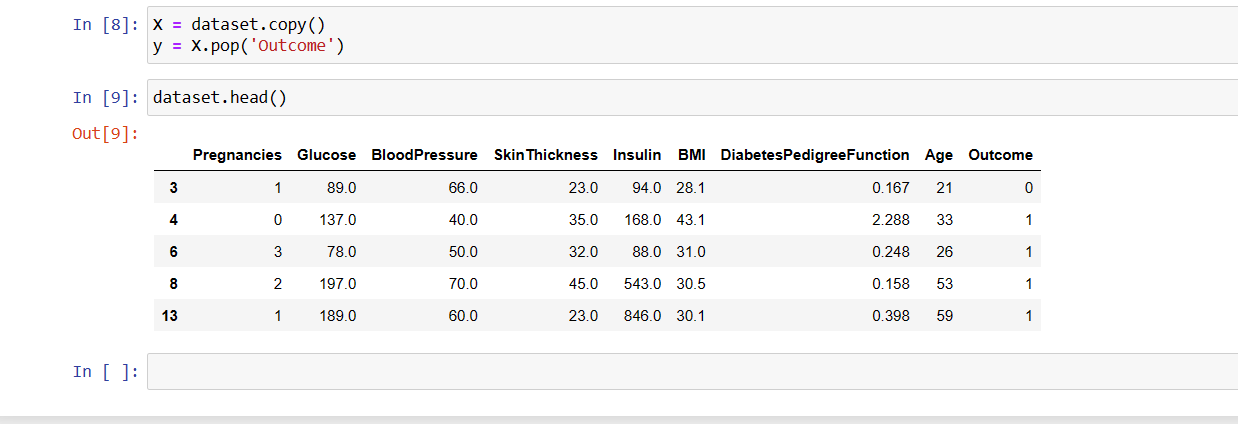


**Data Cleaning**:

* Handle missing values: Identify and handle missing data. You can either impute missing values or remove rows/columns with missing data depending on the extent of missingness.
* Outlier detection and treatment: Identify and deal with outliers in your data. Outliers can negatively impact model performance.

X = dataset.copy()

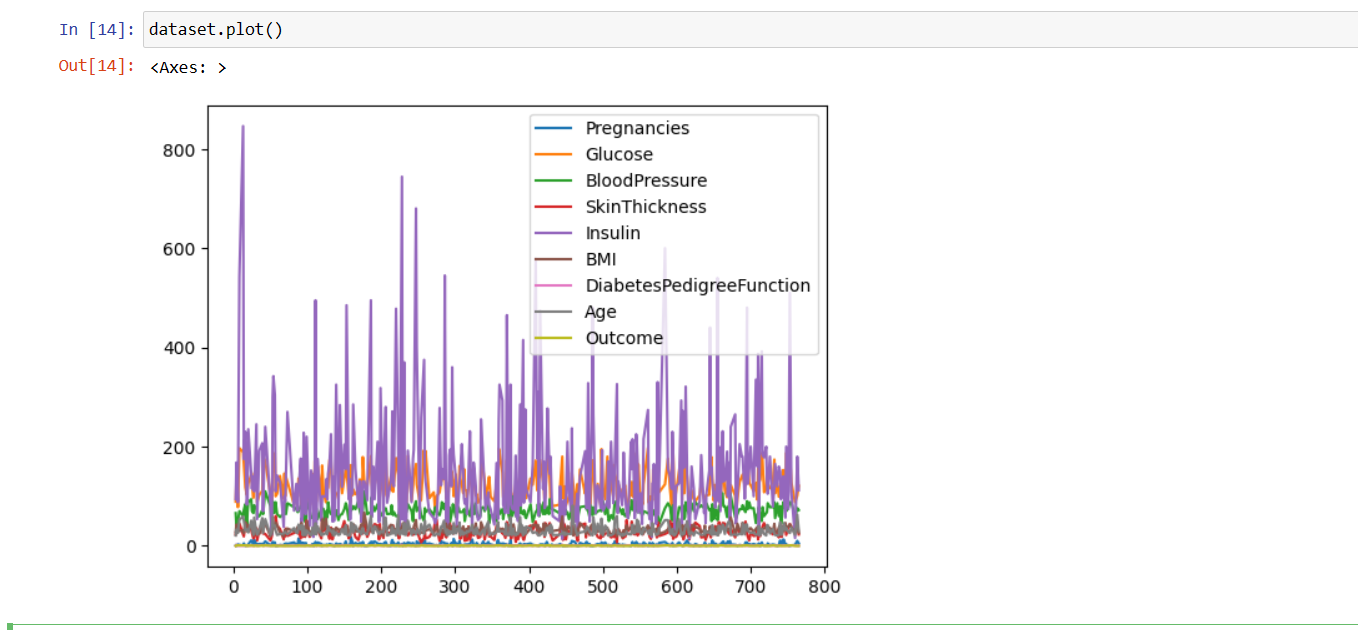
y = X.pop('Outcome')



This code demonstrates the basic steps for data cleaning, such as handling missing values, removing duplicates, and optionally addressing outliers, data types, column names, and reindexing. Adjust these steps as necessary based on your dataset's characteristics and the specific data quality issues you encounter.

**Data visualization:**

Data visualization is an important step in understanding your dataset when working on an AI-based diabetes detection project. You can use libraries like Matplotlib and Seaborn in Python to create various types of visualizations.



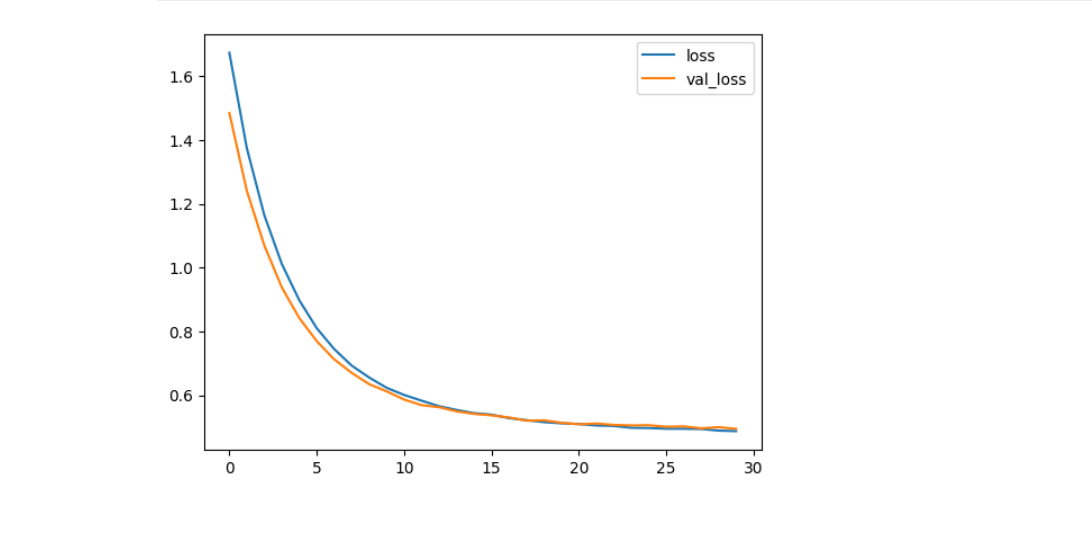
This code provides examples of various data visualization techniques:

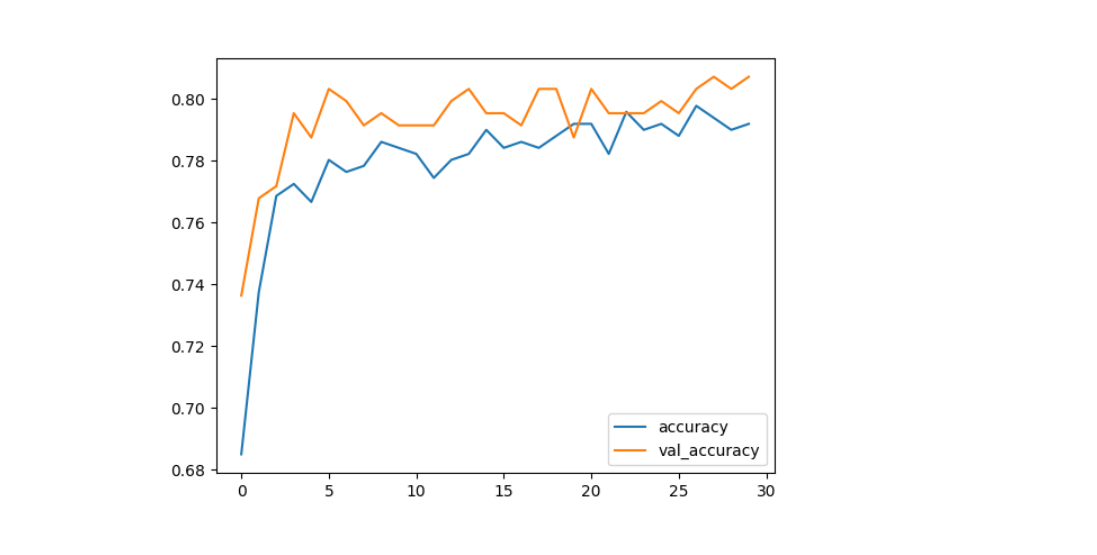
1. Displaying the first few rows of the dataset to get an overview.
2. Generating summary statistics for numerical features.
3. Creating histograms to visualize the distribution of numerical features.
4. Generating boxplots to identify potential outliers.
5. Creating a pairplot to visualize relationships between features, with hue indicating the outcome class.
6. Creating a correlation heatmap to visualize feature correlations.

history\_df = pd.DataFrame(history.history)

history\_df.loc[:, ['loss','val\_loss']].plot();

history\_df.loc[:, ['accuracy','val\_accuracy']].plot();





from sklearn.metrics import confusion\_matrix

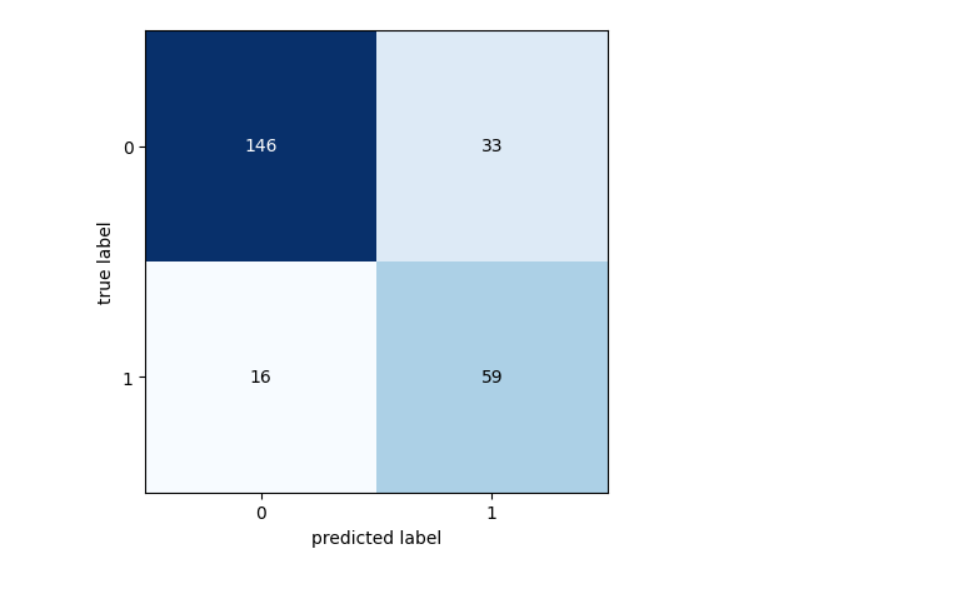
import matplotlib.pyplot as plt

cm = confusion\_matrix(y\_\_predict, y\_\_real)

from mlxtend.plotting import plot\_confusion\_matrix

fig, ax = plot\_confusion\_matrix(conf\_mat=cm)

plt.show()

**Data Analysis:**

Data analysis is a crucial step in developing an AI-based diabetes detection model. Through data analysis, you can gain insights into the dataset, understand the relationships between features, and make informed decisions about feature selection, preprocessing, and model development. Below are some key steps and code examples for data analysis in Python using popular libraries like Pandas, NumPy, and Matplotlib.

dataset.rename(columns={'DiabetesPedigreeFunction': 'DPF'}, inplace= True)

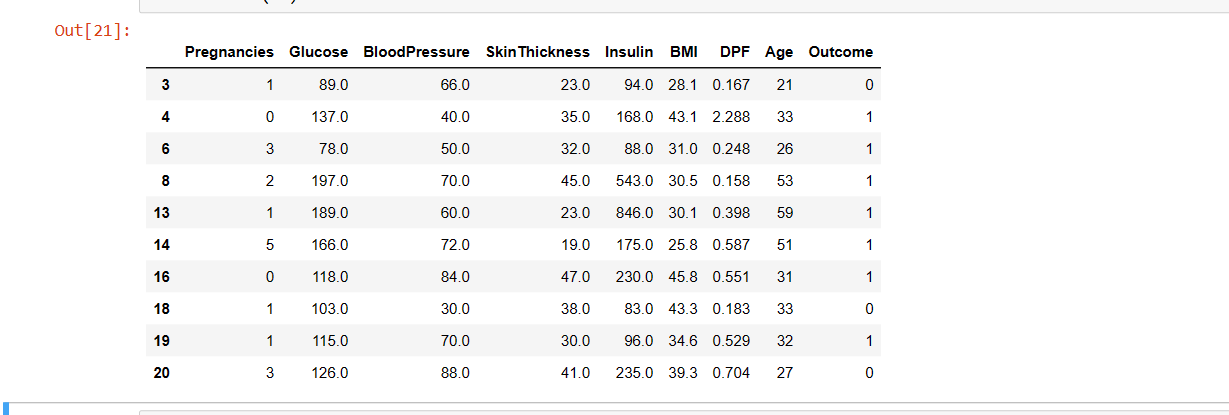
to\_nan = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin']

to\_nan.append(['BMI', 'DPF', 'Age'])

for i in range(len(to\_nan)):

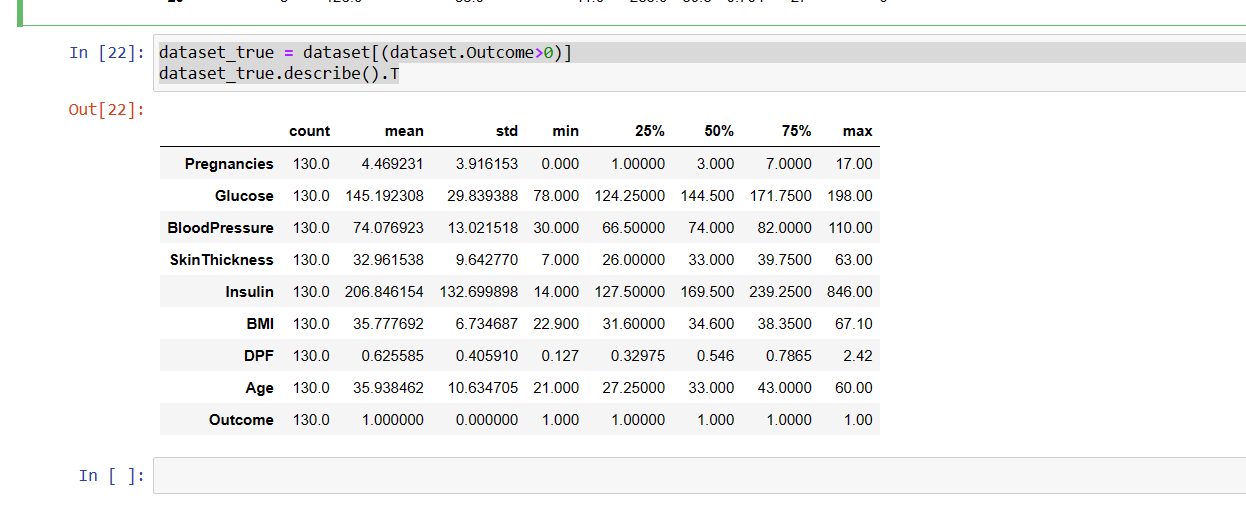
dataset[to\_nan[i]] = dataset[to\_nan[i]].replace(0, np.nan)

dataset.head(10)



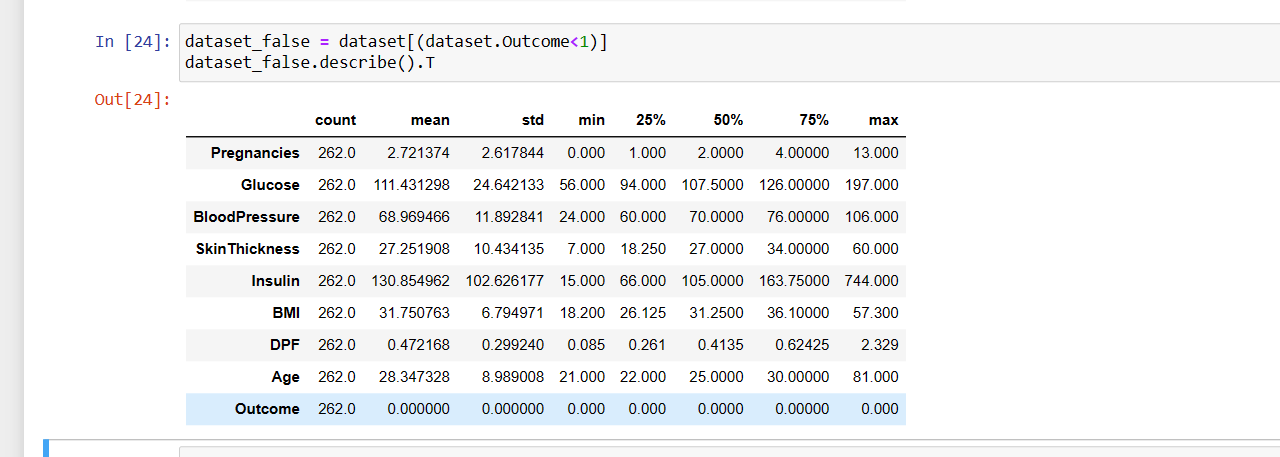
dataset\_true = dataset[(dataset.Outcome>0)]

dataset\_true.describe().T



dataset\_false = dataset[(dataset.Outcome<1)]

dataset\_false.describe().T



dataset.describe().T

