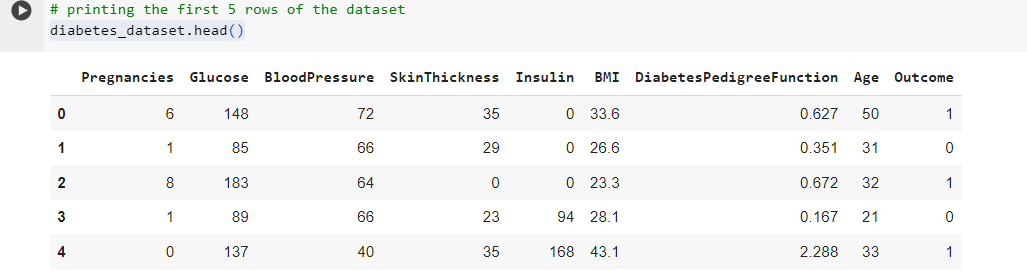
**RESULTS**

The pd.read\_csv() function is being called from the pandas library. This function is used to read data from a CSV (Comma Separated Values) file. The argument 'content/diabetes.csv' specifies the path to the CSV file that you want to read. Further the Data Frame returned by pd.read\_csv() is assigned to a variable named diabetes\_dataset.

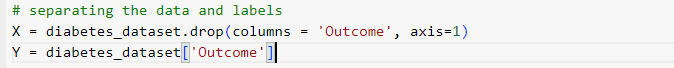


diabetes\_dataset.head() is used to display the first few rows of the DataFrame diabetes\_dataset.

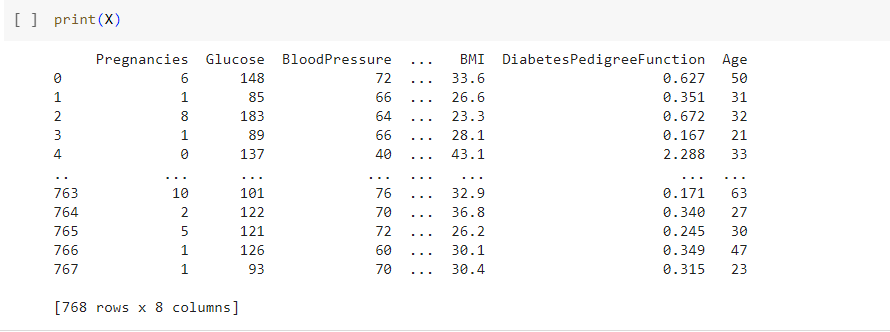


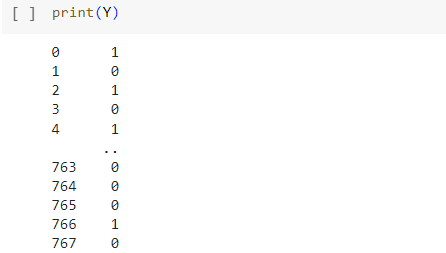
The code diabetes\_dataset.describe() provides summary statistics for each numerical column in the DataFrame diabetes\_dataset. These statistics include measures like count, mean, standard deviation, minimum, maximum, and various percentiles (25th, 50th, and 75th).

The dataset is divided into two parts, The class label ‘Outcome’ is stored in variable Y. while the data of remaining attributes is stored in variable X.



diabetes\_dataset.drop(columns='Outcome', axis=1): This part selects all columns from the diabetes\_dataset DataFrame except the 'Outcome' column.



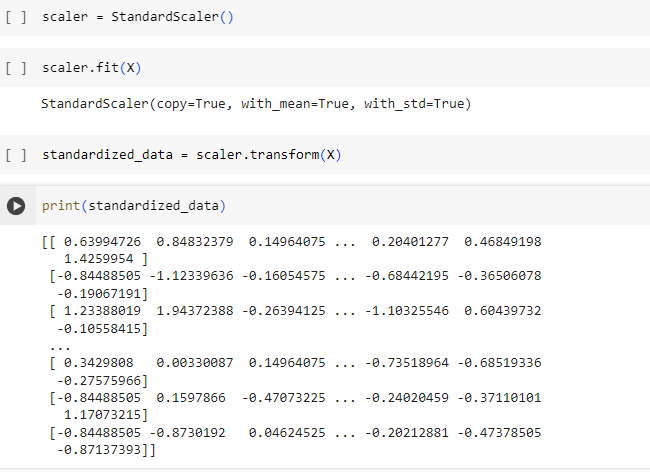


**Data Standardization**

scaler = StandardScaler(): This creates an instance of the StandardScaler class from scikit-learn. StandardScaler is used for standardizing features by removing the mean and scaling to unit variance.

scaler.fit(X): This method fits the scaler to the input features (X). It computes the mean and standard deviation for each feature in X. This step is necessary to learn the parameters needed for standardization.

standardized\_data = scaler.transform(X): This line transforms the input features (X) using the parameters learned during the fitting step. It applies the standardization formula to each feature, resulting in standardized data where each feature has a mean of 0 and a standard deviation of 1.



The standardized data is kept in X variable.

**TRAINING DATA**

train\_test\_split(X, Y, test\_size=0.2, stratify=Y, random\_state=2): This function splits the input features (X) and the target variable (Y) into four subsets:

X\_train: This contains the input features for training the model.

X\_test: This contains the input features for testing the model.

Y\_train: This contains the target variable for training the model.

Y\_test: This contains the target variable for testing the model.

test\_size=0.2: This parameter specifies the proportion of the dataset to include in the test split.stratify=Y: This parameter ensures that the class distribution in the target variable (Y) is preserved in the train-test split.random\_state=2: This parameter sets the random seed for reproducibility. It ensures that the data split is deterministic, meaning that if you run the code multiple times with the same random\_state, you'll get the same split each time.



**Training the Model**

An classifier is created which is an instance of SVC object for classification tasks.SVC is a type of support vector machine (SVM) classifier. During the object creation the function takes a parameter named ‘kernel’,this parameter specifies the type of kernel used by the SVM.In this case, the linear kernel is chosen. A linear kernel creates decision boundaries that are linear hyper planes in the input space. This means the model will try to find the best linear separation between the classes.

The model is trained by training data (ie X-Train and Y-Train) . After training ,the model has learned to distinguish between the classes represented by the features in X\_train, based on the corresponding labels in Y\_train.

**Prediction System**

The user is prompted to enter various attributes related to diabetes, such as the number of pregnancies, glucose level, blood pressure, skin thickness, insulin level, BMI (Body Mass Index), diabetes pedigree function (dpf), and age.The user inputs are stored in variables such as pregnacies, Glucose, BloodPressure, etc.These variables are then combined into a tuple named input\_data.The input\_data tuple is converted into a NumPy array (input\_data\_as\_numpy\_array) and reshaped into a format suitable for making predictions (input\_data\_reshaped).The input data is standardized using the scaler object previously fitted on the training data. The transform() method is applied to input\_data\_reshaped to ensure that the input data is scaled in the same way as the training data.The standardized input data (std\_data) is passed to the predict() method of the trained classifier (classifier) to obtain the prediction.The prediction is printed out, and based on the prediction, a message indicating whether the person is predicted to be gestational diabetic or not is displayed.

