AI based Diabetes Prediction System

PHASE 4: DEVELOPMENT PART – II

(Building, Testing and Evaluating the model)

Importing necessary libraries, data exploration and preprocessing are done in last phase

The developing and training the dataset for the model is implemented in this phase

The complete program for the diabetes prediction system is documented.

Importing libraries:

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

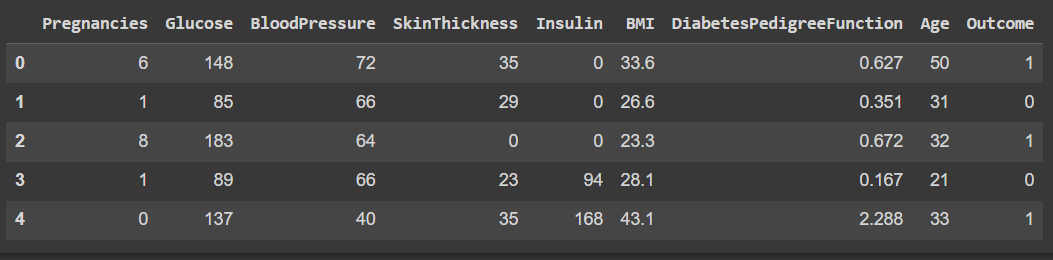
from sklearn import svm

from sklearn.metrics import accuracy\_score

Loading the dataset to the dataframe using the pandas library:

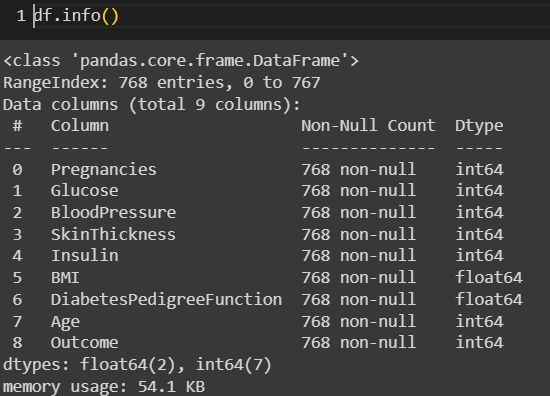
df=pd.read\_csv("/content/diabetes.csv") #loading the dataset

df.head()

DATA EXPLORATION:

Basic information about the dataset:

df.info()

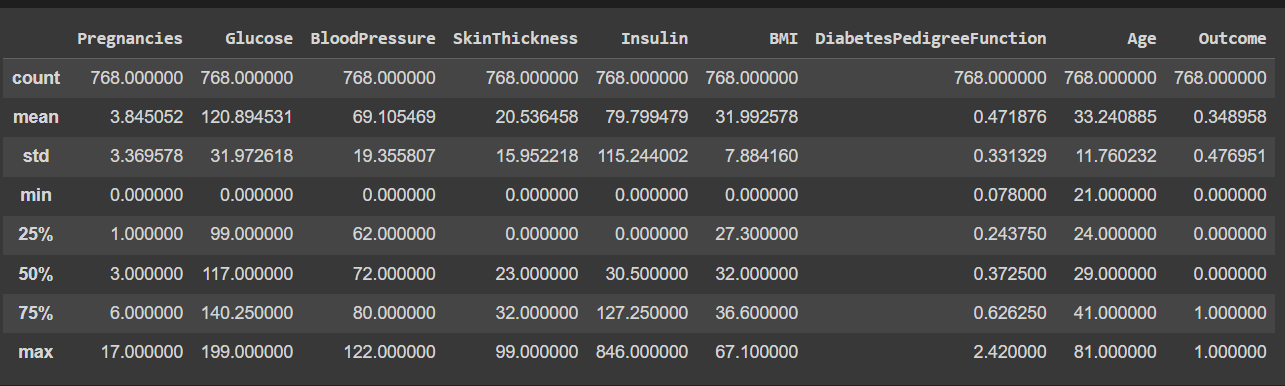
 Getting dimensions of the dataset:

df.shape



Stastistical measures of data:

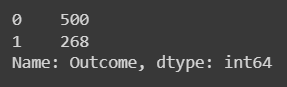
df.describe()



Analysing outcome:

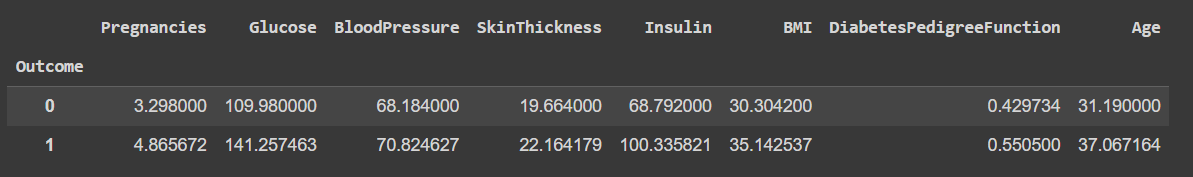
df['Outcome'].value\_counts()

# 0--> non-diabetic people

# 1--> diabetic people

Checking the description of the ‘Outcome’:

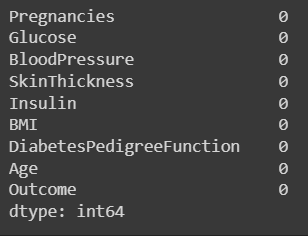
df.groupby('Outcome').mean()



Checking missing values:

missing\_values = df.isnull().sum()

print(missing\_values)



Filling missing values:

mean\_fill = df.fillna(df.mean())

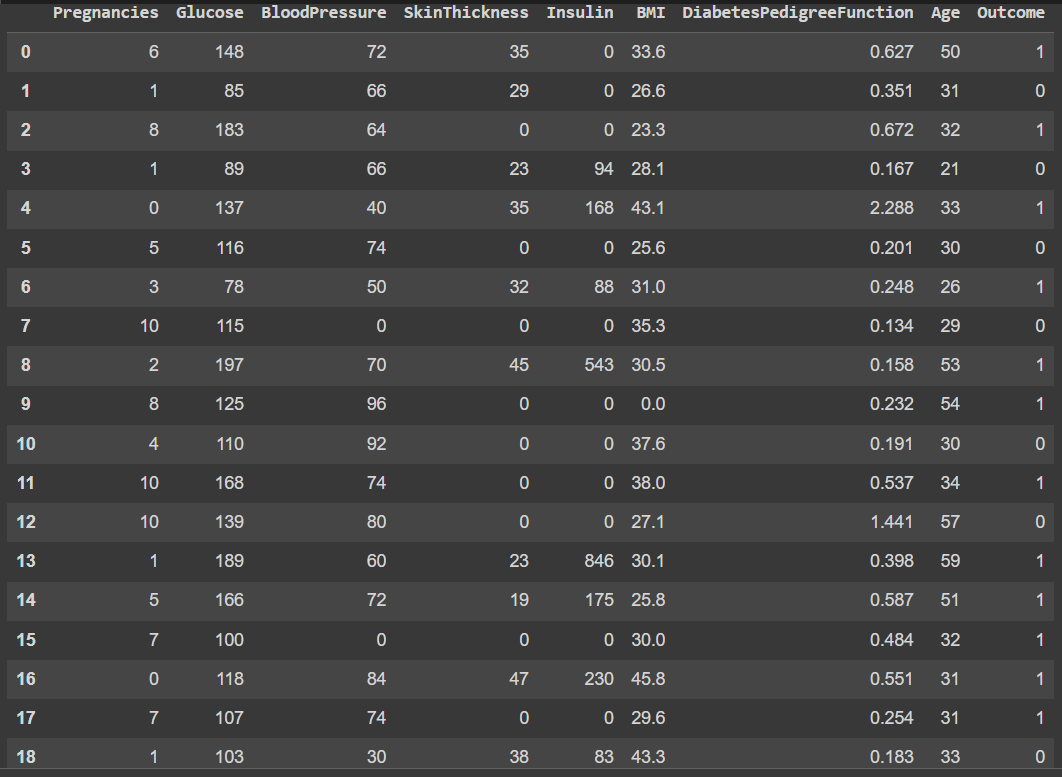
df.fillna(mean\_fill, inplace=True)

Checking duplicate values and deleting them:

duplicate\_values = df.duplicated().sum()

df.drop\_duplicates(inplace=True)

df.head(20)

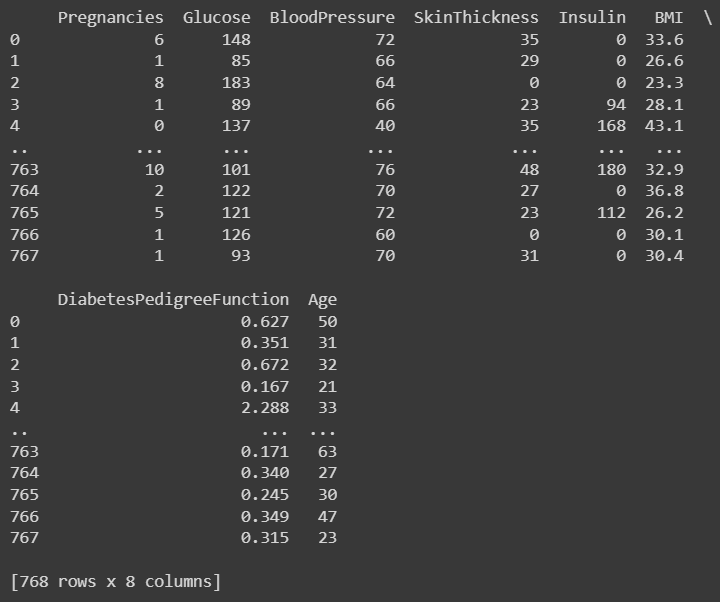


Seperating the data and labels:

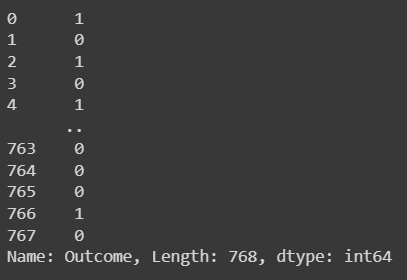
X = df.drop(columns='Outcome', axis=1)

Y = df['Outcome']

print(X)



print(Y)



Data Standardization:

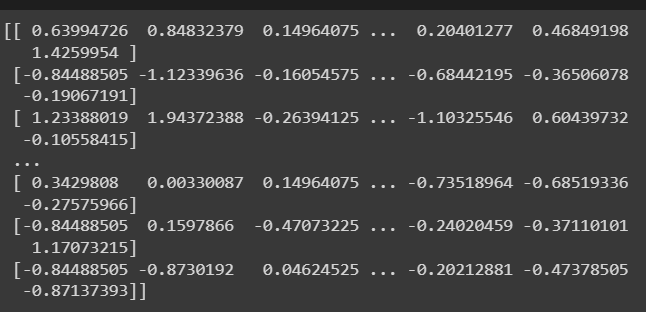
Data standardization, also known as data scaling or data normalization, is a crucial preprocessing step in machine learning. It involves transforming the features (variables) in our dataset into a standard format or scale to ensure that the machine learning algorithms can work effectively and make meaningful predictions.

scale = StandardScaler()

scale.fit(X)

sd\_data = scale.transform(X)

print(sd\_data)

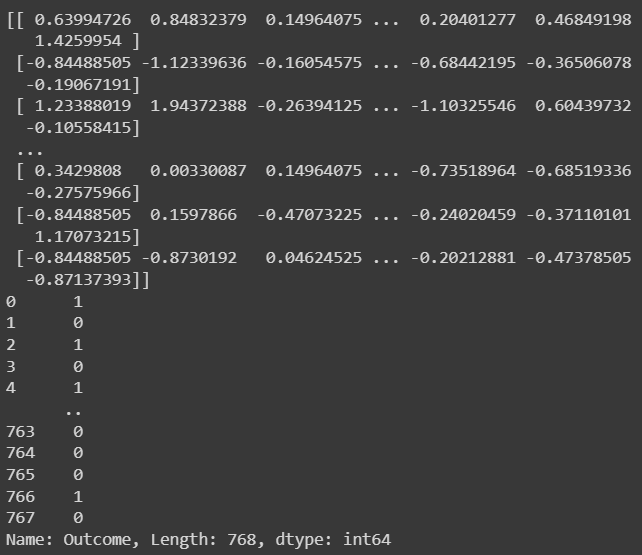


X = sd\_data

Y= df['Outcome']

print(X)

print(Y)



BUILDING THE MODEL

Splitting data for training:

The splitting of the dataset using the train\_test\_split will split the dataset into 4 parts

The train – X which is the input for the model to train

The train – Y which is the input for the model to train

Y,X-test which are used to test the model and increasing the efficiency.

20% of the dataset are used for testing and stratify will split the data according to their columns

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y, test\_size=0.2, stratify = Y, random\_state=2)

Dimensions of the train and test dataset:

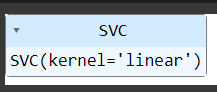
print(X.shape, X\_train.shape, Y\_test.shape)

Choosing the training model and loading it to the classifier:

A classifier is a type of algorithm used to categorize or label objects into different classes or categories.

classifier = svm.SVC(kernel= 'linear')

classifier.fit(X\_train,Y\_train)



Accuracy score on the given training data

The accuracy score is a commonly used metric to evaluate the performance of a classification model. It measures the proportion of correctly predicted instances (samples or observations) out of the total number of instances.

X\_train\_prediction = classifier.predict(X\_train)

training\_model\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print(training\_model\_accuracy)



Accuracy score on the test data:

X\_test\_prediction = classifier.predict(X\_test)

test\_data\_accuracy= accuracy\_score(X\_test\_prediction, Y\_test)

print(test\_data\_accuracy)



An accuracy score of greater than 70 percentage is considered as a good machine learning model and hereby we get 77% on the test data and 78% on the training data which proves that this model fit for this diabetes prediction.

TESTING THE MODEL

Testing with custom input:

input\_data = (10, 125, 70, 26, 115, 31.1, 0.205, 41)

Transformed to numpy array

data\_array = np.asarray(input\_data)

Reshape array for one instance

reshaped\_data\_input = data\_array.reshape(1,-1)

Standardization

Since we use standardized input for training we need to use the standardized input for testing the model

std\_input\_data= scale.transform(reshaped\_data\_input)

print(std\_input\_data)



prediction =  classifier.predict(std\_input\_data)

print(prediction)

0 🡪 Person is not Diabetic

1 🡪 Person is Diabetic

EVALUATING THE MODEL

if(prediction[0] == 0):

  print("The person is not diabetic")

else:

  print("The person is diabetic")



The linear regression based machine learning model is best fit for the given dataset and gives us 78 percentage accuracy. And also passed the custom input.

Conclusion:

Thus the building, testing and evaluating of the model is done in this phase and the model is ready to be used.