CONTENTS

1. ABSTRACT

Globally, diabetes affects 537 million people, making it the deadliest and the most common non-communicable disease. Many factors can cause a person to get affected by diabetes, like excessive body weight, abnormal cholesterol level, family history, physical inactivity, bad food habit etc. Increased urination is one of the most common symptoms of this disease. People with diabetes for a long time can get several complications like heart disorder, kidney disease, nerve damage, diabetic retinopathy etc. But its risk can be reduced if it is predicted early. In this project, an automatic diabetes prediction system has been developed using a private dataset of female patients and machine learning techniques. The dataset used in this project is Pima Indian diabetes dataset. In this project machine learning classification methods, that is, decision tree, SVM, Random Forest, Logistic Regression, and various techniques are used to determine which algorithm produces the best prediction results.

2. INTRODUCTION

All around there are numerous ceaseless infections that are boundless in evolved and developing nations. One of such sickness is diabetics. Diabetes is a chronic disease that directly affects the pancreas, and the body is incapable of producing insulin. Insulin is mainly responsible for maintaining the blood glucose level. Many factors, such as excessive body weight, physical inactivity, high blood pressure, and abnormal cholesterol level, can cause a person get affected by diabetes. It can cause many complications, but an increase in urination is one of the most common ones. It can damage the skin, nerves, and eyes, and if not treated early, diabetes can cause kidney failure and diabetic retinopathy ocular disease.
Significant numbers of research have been performed to predict diabetes automatically using machine learning and ensemble techniques. Most of these works employed the open-source Pima Indian dataset.

2.1 PROBLEM DESCRIPTION

2.1.1 PROBLEM DEFNITION

The major challenge in predicting diabetes cases is its detection. There are instruments available that can predict diabetes but either they are expensive or are not efficient to calculate the chance of diabetes in humans. Early detection of diabetes can decrease the mortality rate and overall complications. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more patience, time, and expertise. Since we have a good amount of data in today's world, we can use various machine learning algorithms to analyze the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

Cause of diabetes:

Genetic factors are the main cause of diabetics. It is caused by at least two mutant genes in the chromosome 6, The chromosome that effects the response of the body to various antigens. Viral infection may also influence the occurrence of type 1 and type 2 diabetes. Studies have shown that infection with viruses such as rubella, coxsackievirus, mumps, hepatitisB virus, and cytomegalovirus increase the risk of developing diabetes.

Types of diabetic

Type 1: type 1 diabetes means that the immune system is compromised and the cells fail to produce insulin in sufficient amounts. There are no eloquent studies that prove the cause of type 1 diabetes and there are currently no known method of prevention.

Type 2: type 2 diabetes means that the cells produce a low quantity of insulin or the body can't use the insulin correctly. This is the most common type of diabetes, thus affecting 90% of persons diagnosed with diabetes. It is caused by both genetic factors and the manner of living.

Symptoms of Diabetes

- Frequent urination
- Increased thirst
- Tired/sleepiness
- Weight loss
- Blurred vision
- Mood swings
- Confusion and difficulty concentrating
- Frequent infections

2.1.2 PROPOSED SOLUTION

Machine learning techniques have been around us and have been compared and used for analysis for many kinds of data science applications. This project is carried out with the motivation to develop an appropriate computer-based system and decision support that can aid in the early detection of diabetes, in this project we have developed a model which classifies if the patient will have diabetes based on various features (i.e. potential risk factors that can cause diabetes) using LOGISTIC REGRRESION. Hence, the early prognosis of diabetes can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine.

3. METHODODLOGY

HARDWARE SPECIFICATION

Processor : Intel(R) Core(TM) i5-3210M
Memory(RAM) : 4.00 GB

SOFTWARE SPECIFICATION

Operating system : Windows 10
Language use : Python
Front end: HTML,CSS
IDE : Visual studio code
Framework : Django framework
Software type : Web Application

4. DATASETS

In this project, Pima Indian Diabetes Dataset has been used. The dataset was collected among the Pima Indian female population near Phoenix, Arizona. This particular dataset has been widely used in machine learning experiments and is currently available through the UCI repository of standard datasets. This population has been studied continuously by the National Institute of Diabetes, Digestive, and Kidney. UCI repository contains 768 instances of observations and total 9 attributes with no missing values reported. Datasets contain 8 particular variables which were considered high-risk factors for the occurrence of diabetes and 1 target variable containing '1' for diabetic and '0' for non-diabetic patients. The 8 feature variables along with the target variable are shown in the following table.
Link of the Dataset: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Table

No	Attributes	Description	Type
1	Pregnancies	Number of times pregnant	Numeric
2	Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test Numeric	Numeric
3	Blood Pressure	Diastolic blood pressure	Numeric
4	Skin Thickness	Triceps skinfold thickness	Numeric
5	Insulin	2-hour serum insulin	Numeric
6	BMI	Body mass index	Numeric
7	Diabetes Pedigree Function	Diabetes pedigree function	Numeric
8	Age	Patient's ag	Numeric
9	Outcome	Target variable (1 if diabetic, else 0)	Binary

5. METHODS AND ALGORITHMS USED

Decision tree: Decision tree is a basic classification method. It is supervised learning method. Decision tree used when response variable is categorical. Decision tree has tree like structure based model which describes classification process based on input feature. Input variables are any types like graph, text, discrete, continuous etc.

Random forest: Random forest is a machine learning system that averages the predictions of several decision trees. As a result, the random forest can be considered an ensemble learning model. In this research, we have applied random forest with estimators = 400, minimum samples leaf = 5, and 'Gini' impurity metrics utilizing hyper parameter tuning.

Support vector machine: SVM performs supervised classification by choosing the best hyper plane . In this study, we experimented with various SVM kernels in the training set. Finally, we discovered the SVM with a linear kernel, parameters *C* = 10 and gamma = 1, produces the best results in this dataset.

Logistic regression: Logistic regression can be used to predict a binary class. To predict the outcome, it fits an 'S' shaped function . The hyperparameter optimization technique obtained the maximum number of iterations for the convergence of the logistic regression model to be 150.

6. IMPLEMENTATION

6.1 Work Process

The different stages of this project work.

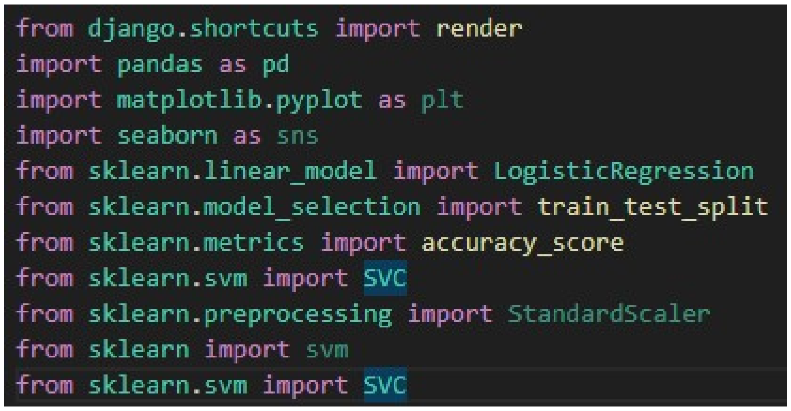
Dataset collection:

The Pima Indian dataset is an open-source dataset that is publicly available for machine learning classification, which has been used in this project work.

It includes understanding the data to study the hidden patterns and trends which helps to predict and evaluating the results. Dataset contains 768 rows and 8 columns i.e., total number of features. Features include Pregnancies, glucose, blood pressure, Skin thickness, Insulin, BMI, Diabetics pedigree function, Age.

Exploring the data

Now we have to set the development environment to build our project. For this project, we are going to build this Diabetes prediction using Machine Learning in visual studio. After downloading the dataset, import the necessary libraries to build the model.



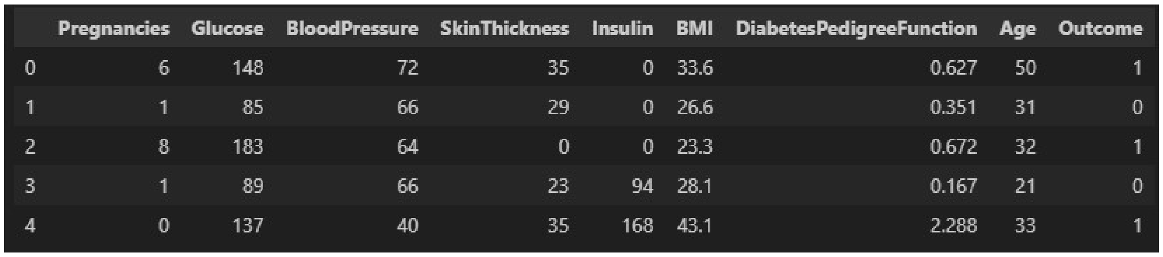
```
from django.shortcuts import render
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.svm import SVC
```

Load the data using the read_csv method in the pandas library. Then the head() method in the pandas library is used to print the rows up to the limit we specify. The default number of rows is five.

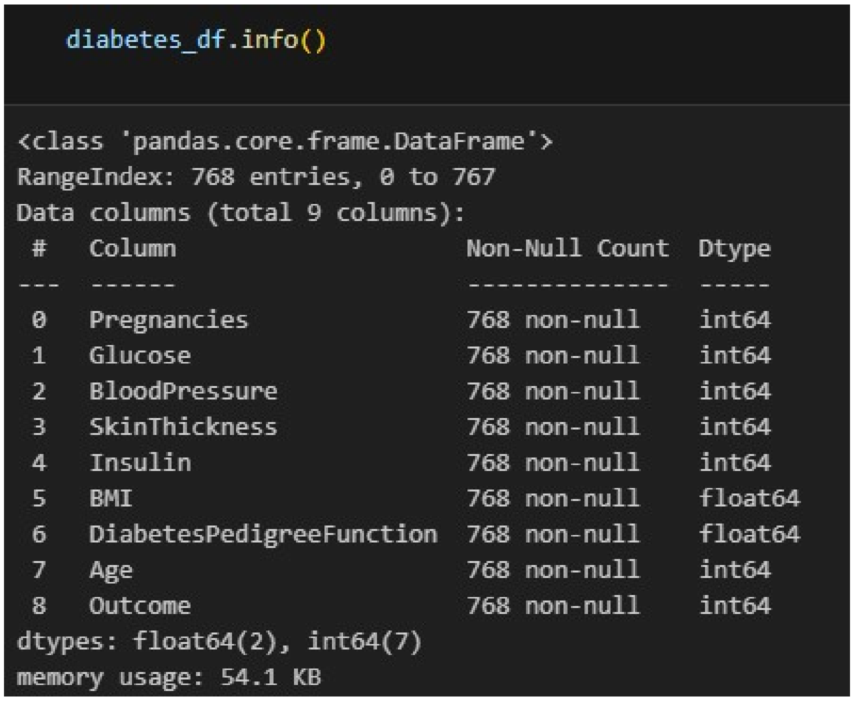
```
data = pd.read_csv(r'C:\Users\Sony\Desktop\mini
project\DiabeticsPrediction\DiabeticsPrediction\diabetes.csv')
```

```
diabetes_df.head()
```

output

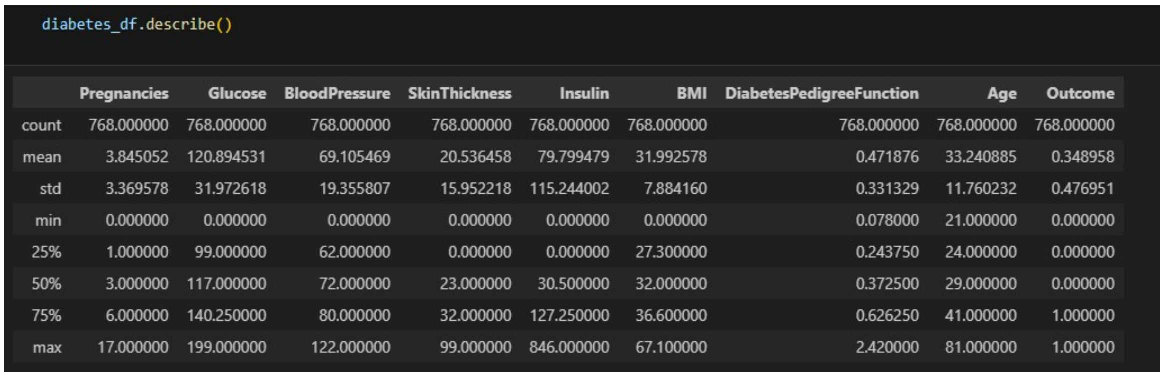


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



```
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

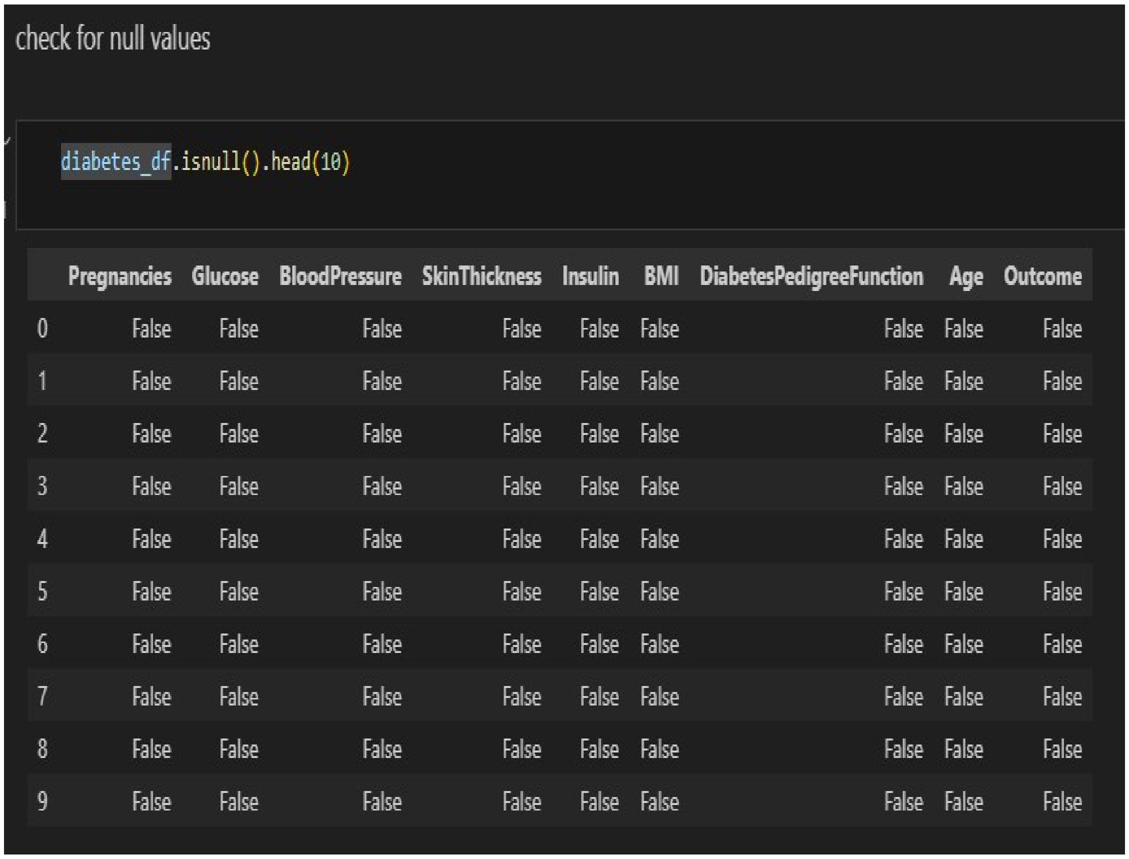


```
diabetes_df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

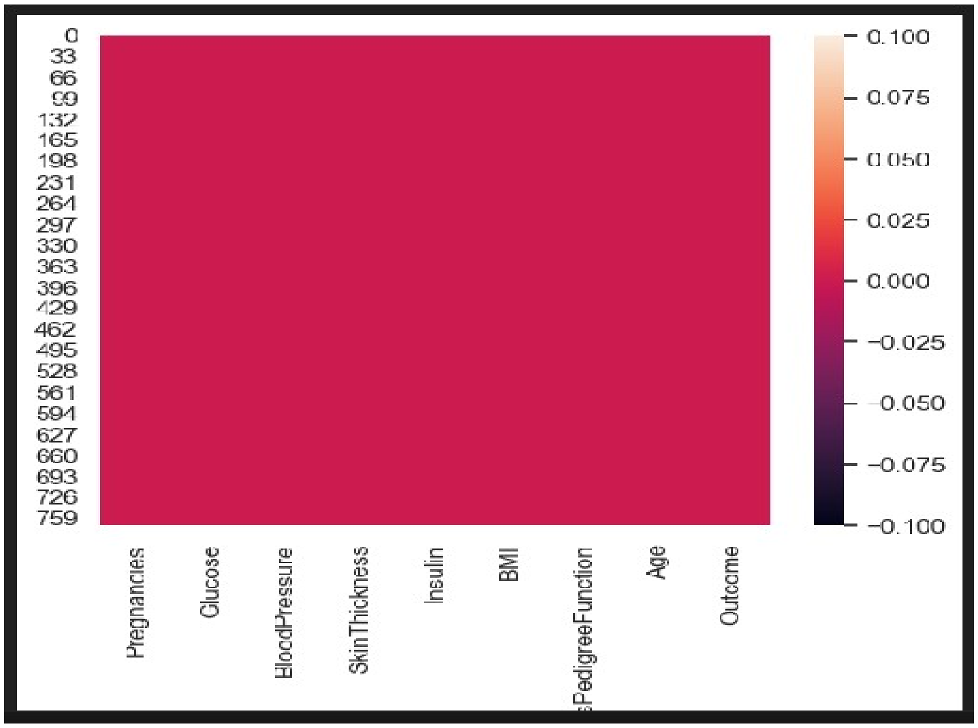
Data Pre-processing:
This phase of model handles inconsistent data in order to get more accurate and precise results. Pre-processed to remove the necessary discrepancies from the dataset, for example, replacing null instances with mean values, dealing with imbalanced class issues etc.

Missing value identification
If missing values are identified in the dataset. We replaced the missing value with the corresponding mean value.


check for null values

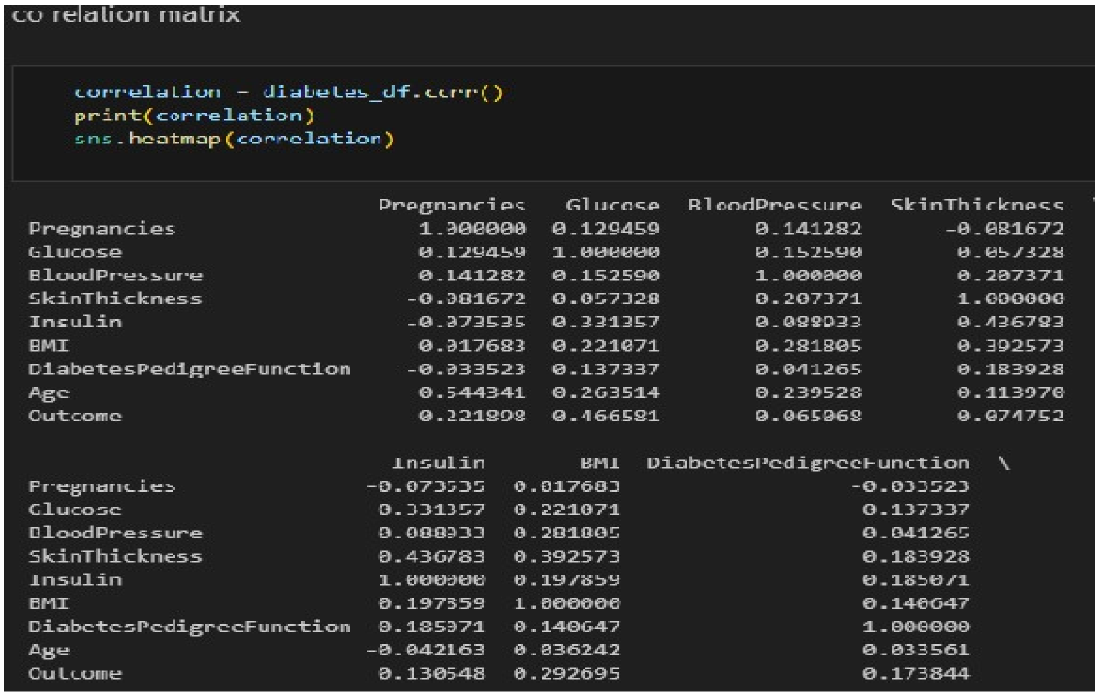
```
diabetes_df.isnull().head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False



Feature selection
Persons correlation method is a popular method to find the most relevant attributes/features. The correlation coefficient is calculated in this method, which correlates with the output and input attributes. The coefficient value remains in the range by between -1 and 1. The value 0.5 and below -0.5 indicates a notable correlation , and the zero values means no correlation.

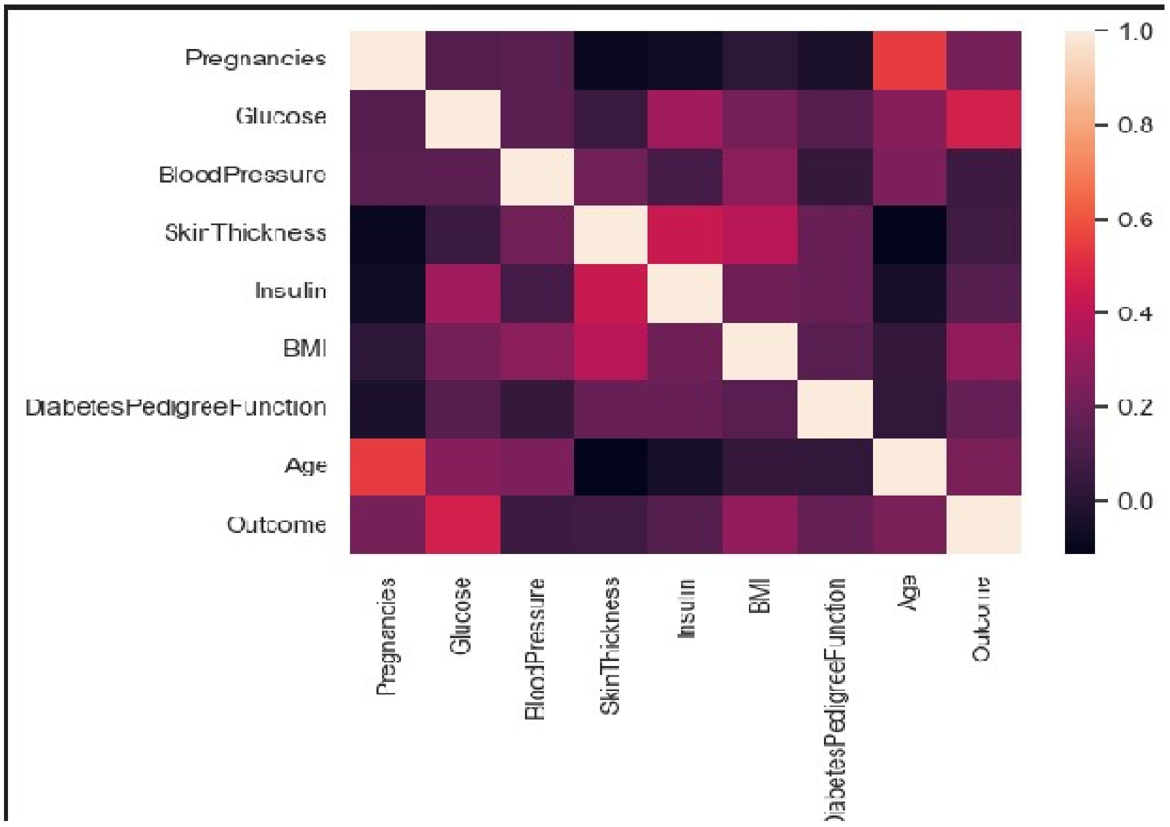
Correlation matrix



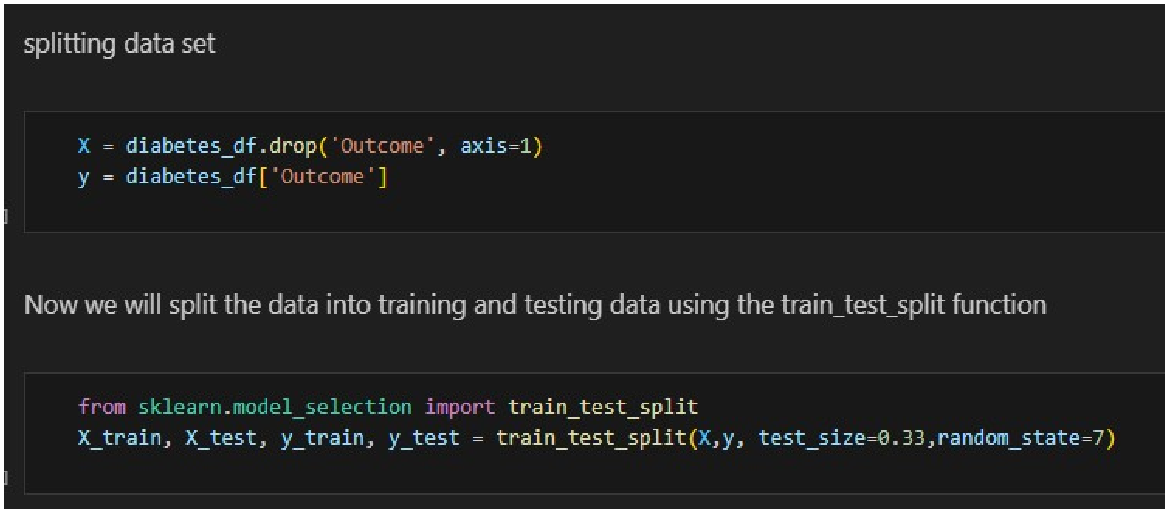
co relation matrix

```
    correlation = diabetes_df.corr()
    print(correlation)
    sns.heatmap(correlation)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528       0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.073535  0.017683                 -0.033523
Glucose                   0.331357  0.221071                  0.137337
BloodPressure             0.088933  0.281805                  0.041265
SkinThickness             0.436783  0.392573                  0.183928
Insulin                   1.000000  0.197859                  0.185071
BMI                       0.197859  1.000000                  0.140647
DiabetesPedigreeFunction  0.185071  0.140647                  1.000000
Age                      -0.042163  0.036242                  0.033561
Outcome                   0.130548  0.292695                  0.173844
```



Splitting of data
Then the dataset was separated into the training set and test set using the holdout validation technique.



splitting data set

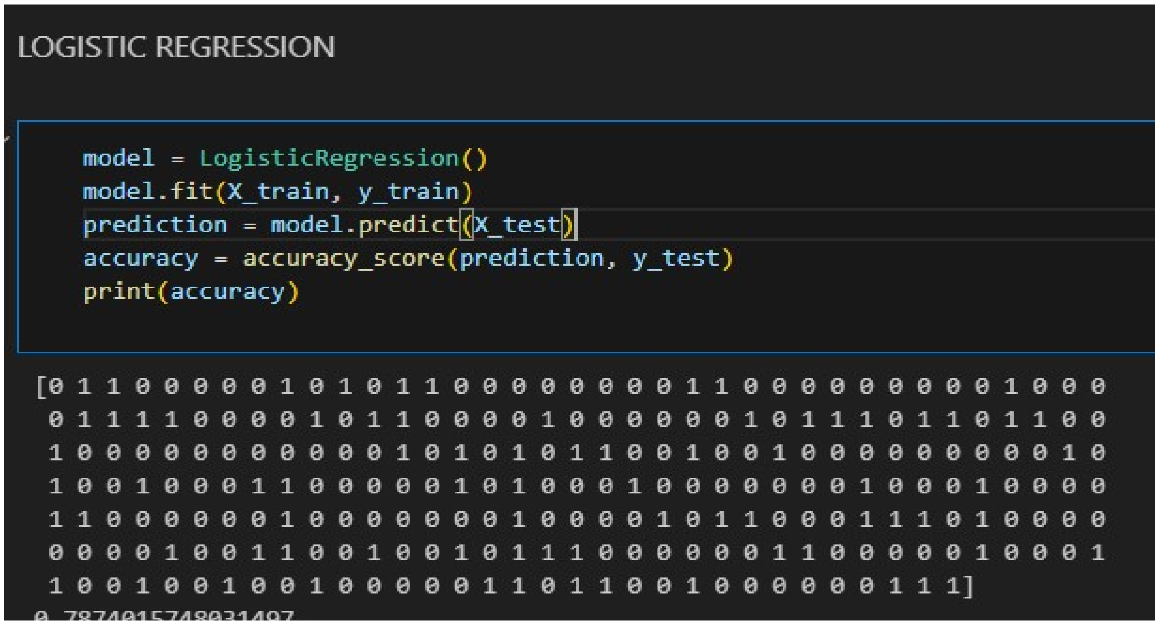
```
X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']
```

Now we will split the data into training and testing data using the train_test_split function

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33,random_state=7)
```

Training the model
Different classification algorithms were applied to find the best classification algorithm for this dataset. Finally, the best-performed prediction model is deployed into the proposed website application framework.

a. Logistic Regression
Logistic regression is also a supervised learning classification algorithm. It is used to estimate the probability of a binary response based on one or more predictors. They can be continuous or discrete. Logistic regression used when we want to classify or distinguish some data items into categories. It classify the data in binary form means only in 0 and 1 which refer case to classify patient that is positive or negative for diabetes. Main aim of logistic regression is to best fit which is responsible for describing the relationship between target and predictor variable. Logistic regression is a based on Linear regression model. Logistic regression model uses sigmoid function to predict probability of positive and negative class.



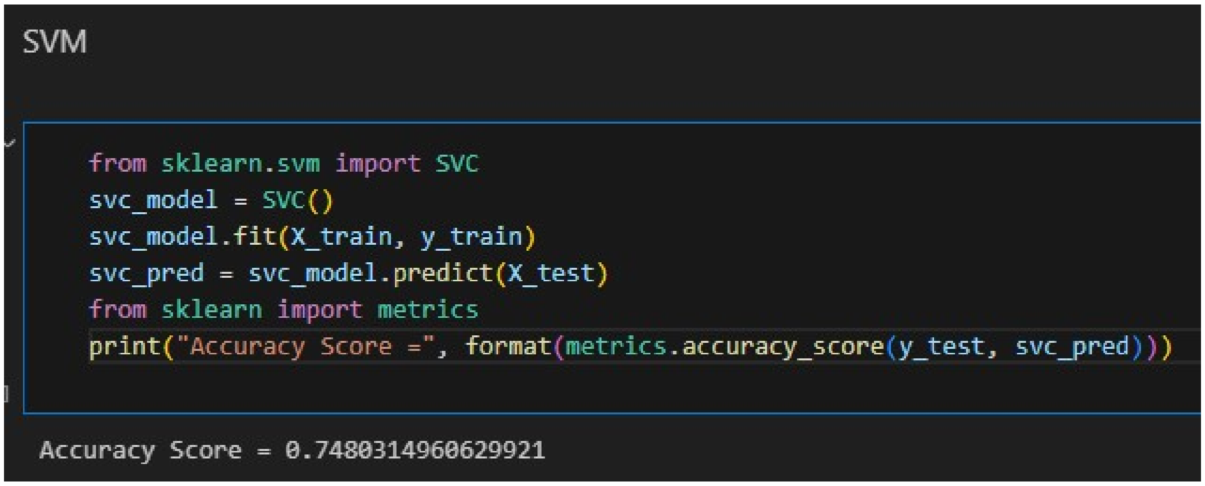
LOGISTIC REGRESSION

```
model = LogisticRegression()
model.fit(X_train, y_train)
prediction = model.predict(X_test)
accuracy = accuracy_score(prediction, y_test)
print(accuracy)
```

```
[0 1 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0
 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0
 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0
 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0
 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1
 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 1 1 1]
0.7874015748031497
```

b. Svm(Super vector machine)

The occurrences of points in area is denoted by the SVM algorithm that are then plotted so that the classes are separated by strong gap. The goal is to determine the maximum-margin hyperplane which provides the greatest parting between the classes. The occurrences which is closest to the maximum-margin hyperplane are called support vectors. The vectors are chosen which are based on the part of the dataset that signifies the training set. Support vectors of two classes enable the creation of two parallel hyperplanes. Therefore, larger the periphery between the two hyperplanes, better will be the generalization error of the classifier. SVMs are implemented in a unique way as compared with other machine learning algorithms.



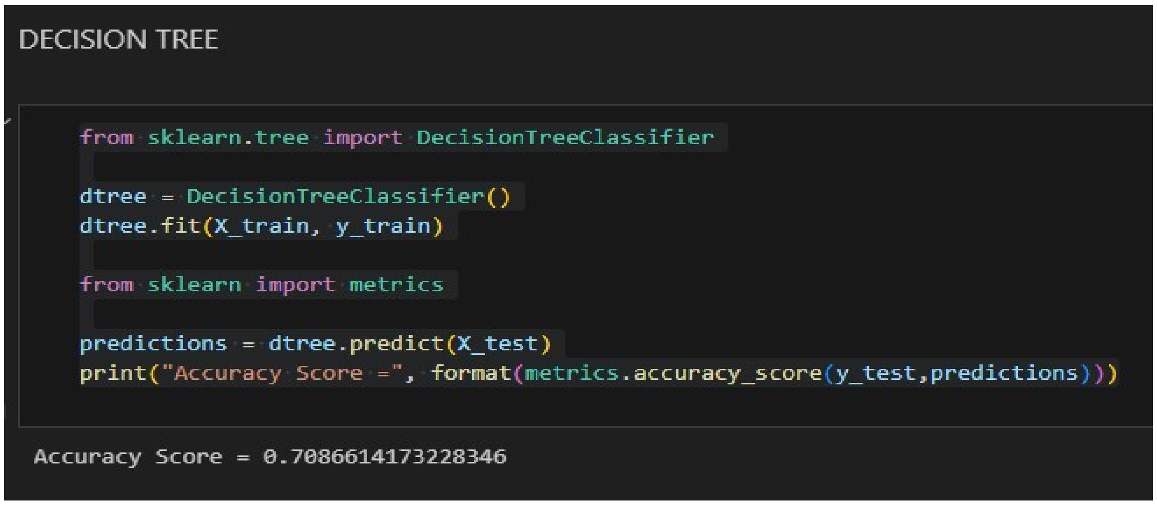
SVM

```
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(X_train, y_train)
svc_pred = svc_model.predict(X_test)
from sklearn import metrics
print("Accuracy Score =", format(metrics.accuracy_score(y_test, svc_pred)))
```

```
Accuracy Score = 0.7480314960629921
```

c. Decision trees

Decision tree is a basic classification method. It is supervised learning method. Decision tree used when response variable is categorical. Decision tree has tree like structure based model which describes classification process based on input feature. Input variables are any types like graph, text, discrete, continuous etc.



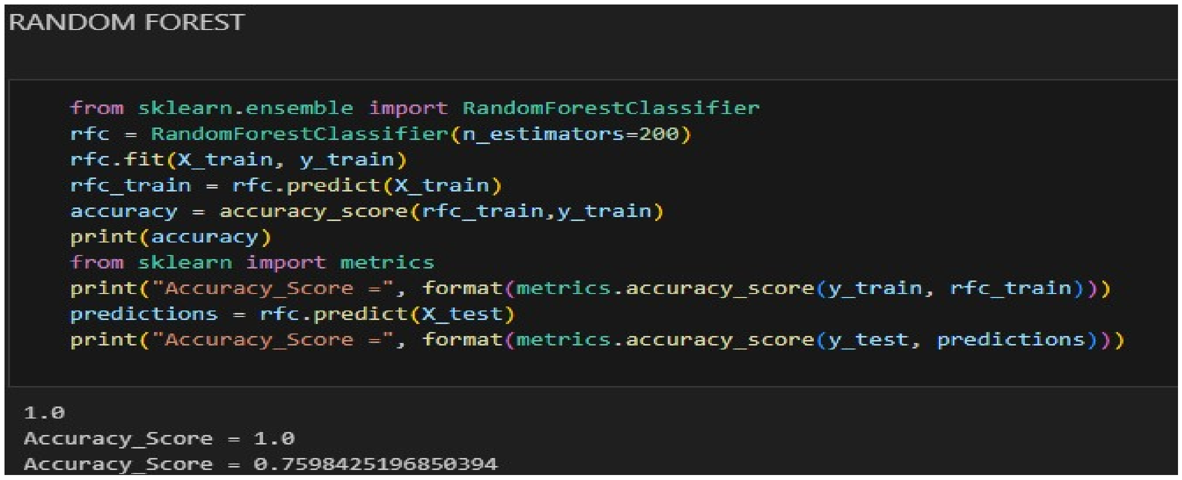
DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

from sklearn import metrics

predictions = dtree.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test,predictions)))
```

```
Accuracy Score = 0.7086614173228346
```

d. Random forest

It is type of ensemble learning method and also used for classification and regression tasks. The accuracy it gives is grater then compared to other models. This method can easily handle large datasets. Random Forest is developed by Leo Bremen. It is popular ensemble Learning Method. Random Forest Improve Performance of Decision Tree by reducing variance. It operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes or classification or mean prediction (regression) of the individual trees.



RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
rfc_train = rfc.predict(X_train)
accuracy = accuracy_score(rfc_train,y_train)
print(accuracy)
from sklearn import metrics
print("Accuracy_Score =", format(metrics.accuracy_score(y_train, rfc_train)))
predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
```

```
1.0
Accuracy_Score = 1.0
Accuracy_Score = 0.7598425196850394
```

Evaluation metrics

The following performance metrics are used to calculate the presentation of various algorithms.

- True positive(TP)-person has disease, and the prediction also has a positive
- True negative(TN)-person not having disease and the prediction also has a negative
- False positive (FP)-person not having disease but the prediction has a positive
- False negative(FN)-person having disease and the prediction also has a positive

Accuracy : we have accuracy matrix to measure the performances of all the models. The ratio of number of correct predictions to the total number of predictions made.
Accuracy=(number of correct predictions)/(total number of predictions made)

Results and discussions

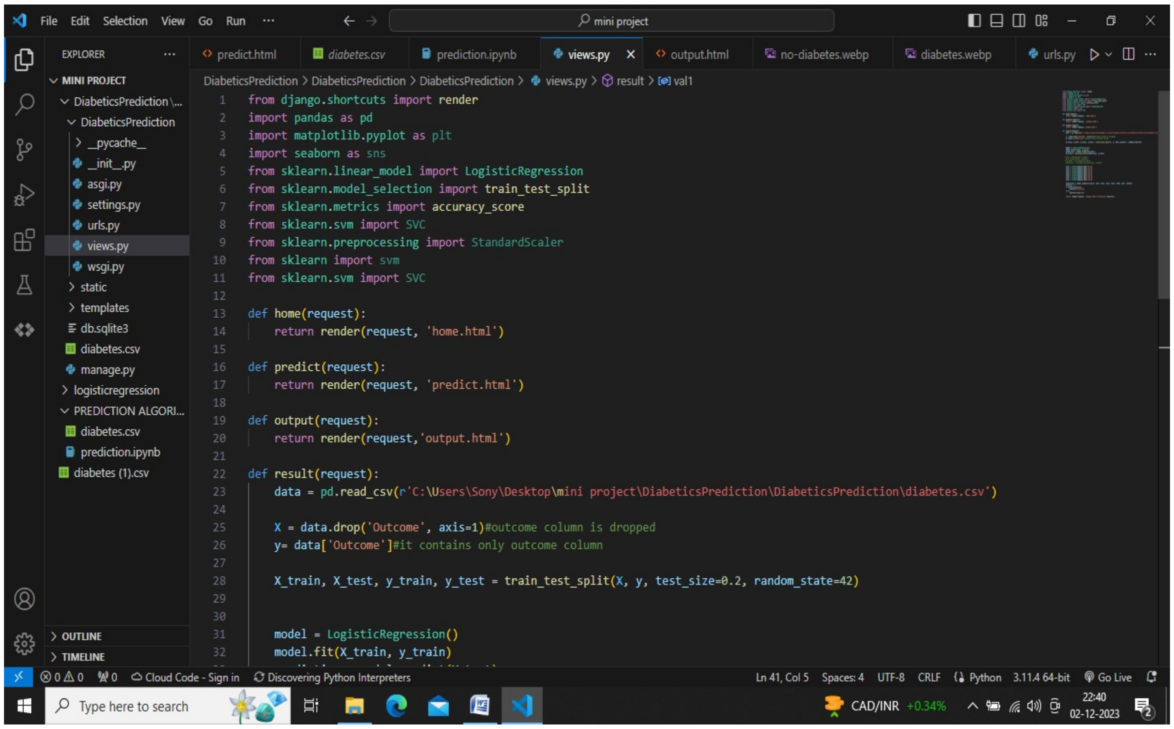
Machine learning classification algorithms developed for prediction of diabetes in earlier stage. We use 70% of data for training and 30% of data for testing. In the ratio of data splitting here we found that Logistic Regression predicted with 78% of accuracy as highest accuracy for the dataset.
Comparisons of all results of all the implemented classifiers are listed in below.

Machine learning algorithms	accuracy
Logistic regression	78%
svm	74%
Random forest	76%
Descion tree	69%

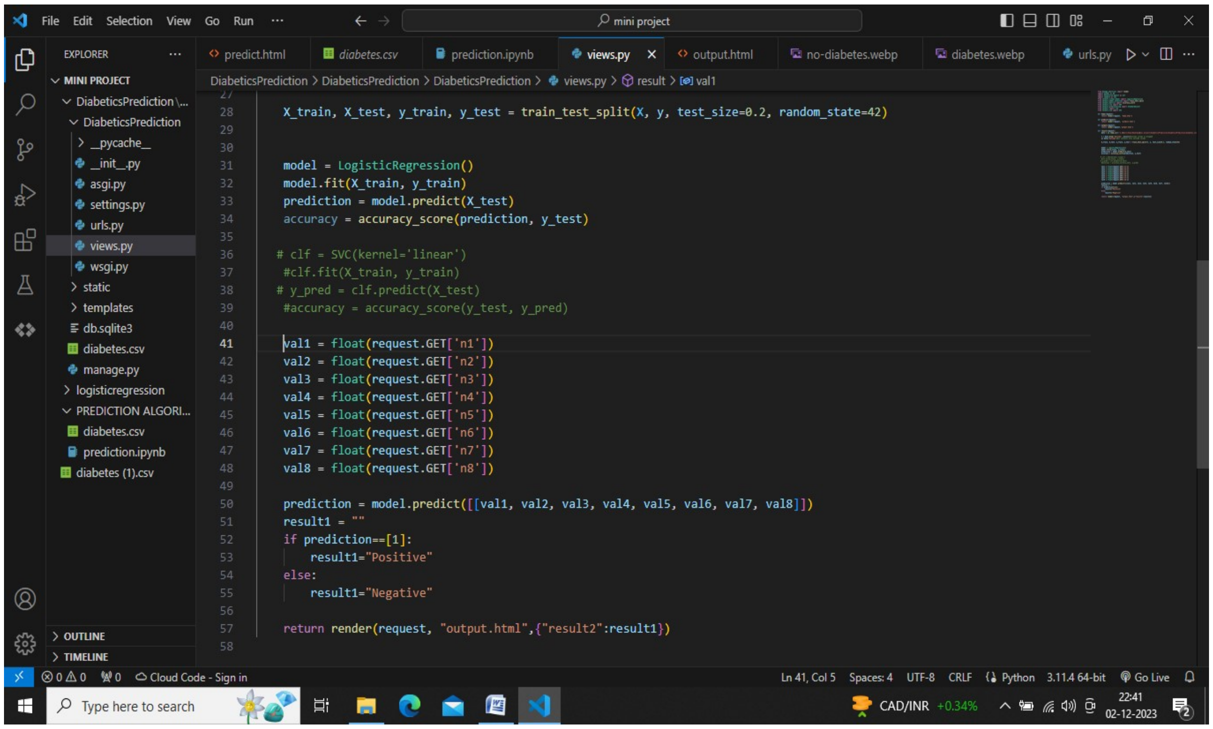
Libraries used:

1. NumPy
2. Matplotlib
3. Seaborn
4. Pandas
5. Sklearn

6.2 PSEUDO CODE



```python
from django.shortcuts import render
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.svm import SVC

def home(request):
    return render(request, 'home.html')

def predict(request):
    return render(request, 'predict.html')

def output(request):
    return render(request,'output.html')

def result(request):
    data = pd.read_csv(r'C:\Users\Sony\Desktop\mini project\DiabeticsPrediction\DiabeticsPrediction\diabetes.csv')

    X = data.drop('Outcome', axis=1)#outcome column is dropped
    y= data['Outcome']#it contains only outcome column

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


    model = LogisticRegression()
    model.fit(X_train, y_train)
```

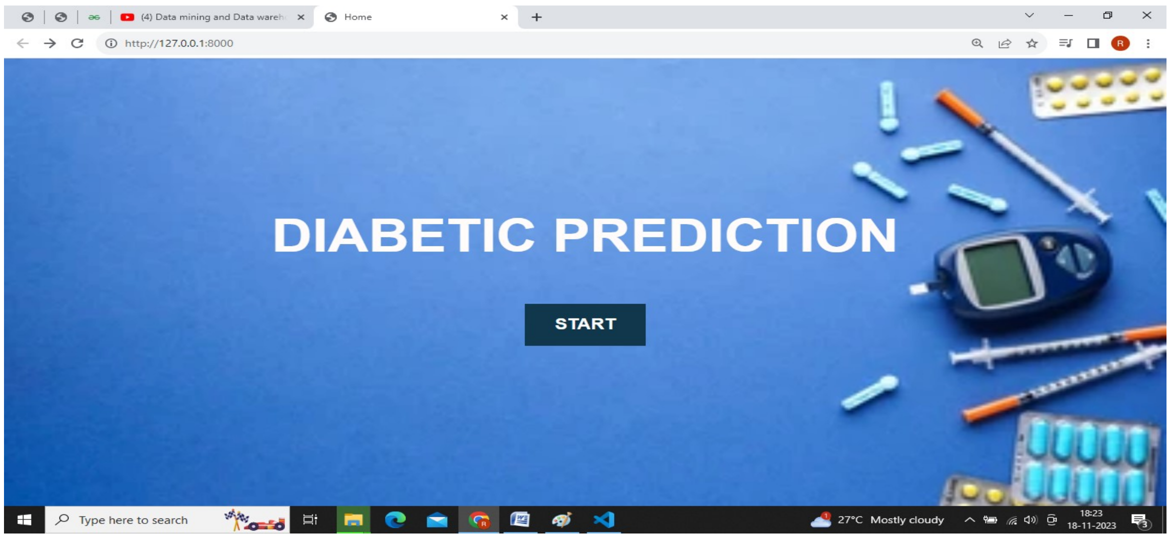


```python
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


    model = LogisticRegression()
    model.fit(X_train, y_train)
    prediction = model.predict(X_test)
    accuracy = accuracy_score(prediction, y_test)

   # clf = SVC(kernel='linear')
    #clf.fit(X_train, y_train)
   # y_pred = clf.predict(X_test)
    #accuracy = accuracy_score(y_test, y_pred)

    val1 = float(request.GET['n1'])
    val2 = float(request.GET['n2'])
    val3 = float(request.GET['n3'])
    val4 = float(request.GET['n4'])
    val5 = float(request.GET['n5'])
    val6 = float(request.GET['n6'])
    val7 = float(request.GET['n7'])
    val8 = float(request.GET['n8'])

    prediction = model.predict([[val1, val2, val3, val4, val5, val6, val7, val8]])
    result1 = ""
    if prediction==[1]:
        result1="Positive"
    else:
        result1="Negative"

    return render(request, "output.html",{"result2":result1})
```

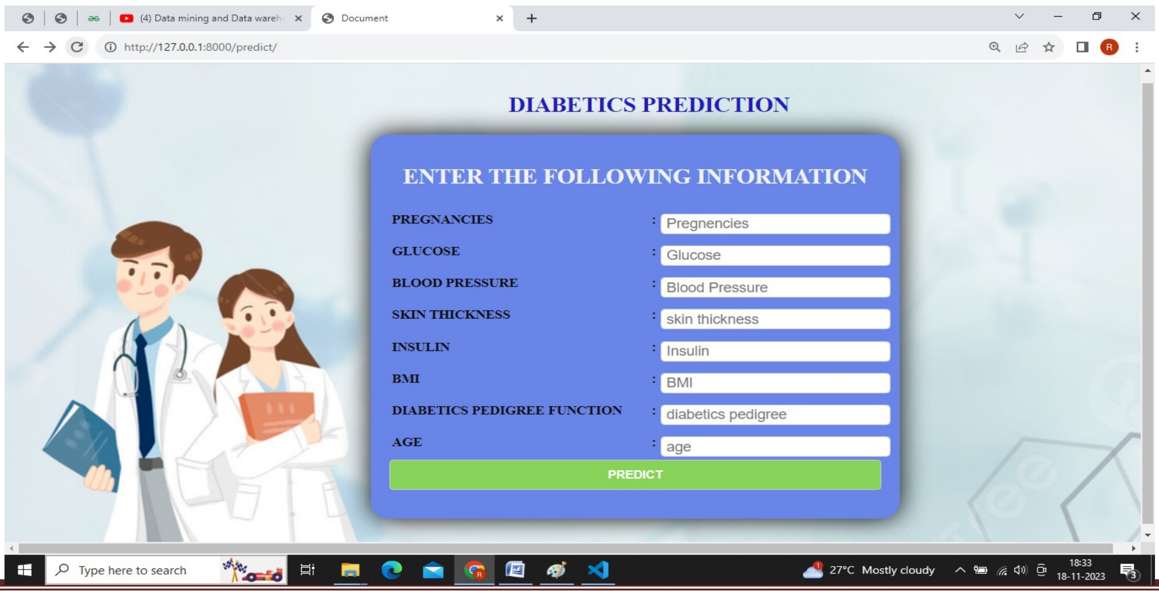
6.3 USER INTERFACE

The last part of the project is the creation of user interface for the model. The user interface is used to enter unseen data for the model to read and then make a prediction. The user interface is using django framework, HTML, and CSS.

HOME PAGE

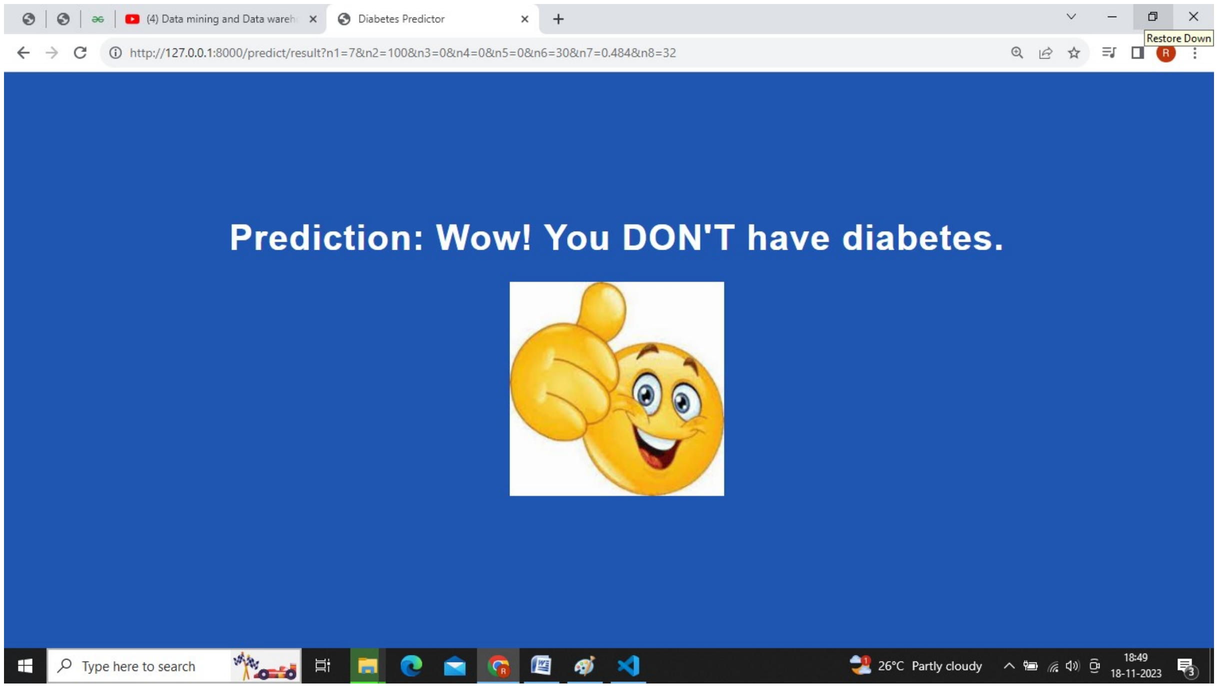


PREDICTION PAGE


http://127.0.0.1:8000/predict/
DIABETICS PREDICTION
ENTER THE FOLLOWING INFORMATION
PREDICT

PREGNANCIES	:	Pregnencies
GLUCOSE	:	Glucose
BLOOD PRESSURE	:	Blood Pressure
SKIN THICKNESS	:	skin thickness
INSULIN	:	Insulin
BMI	:	BMI
DIABETICS PEDIGREE FUNCTION	:	diabetics pedigree
AGE	:	age

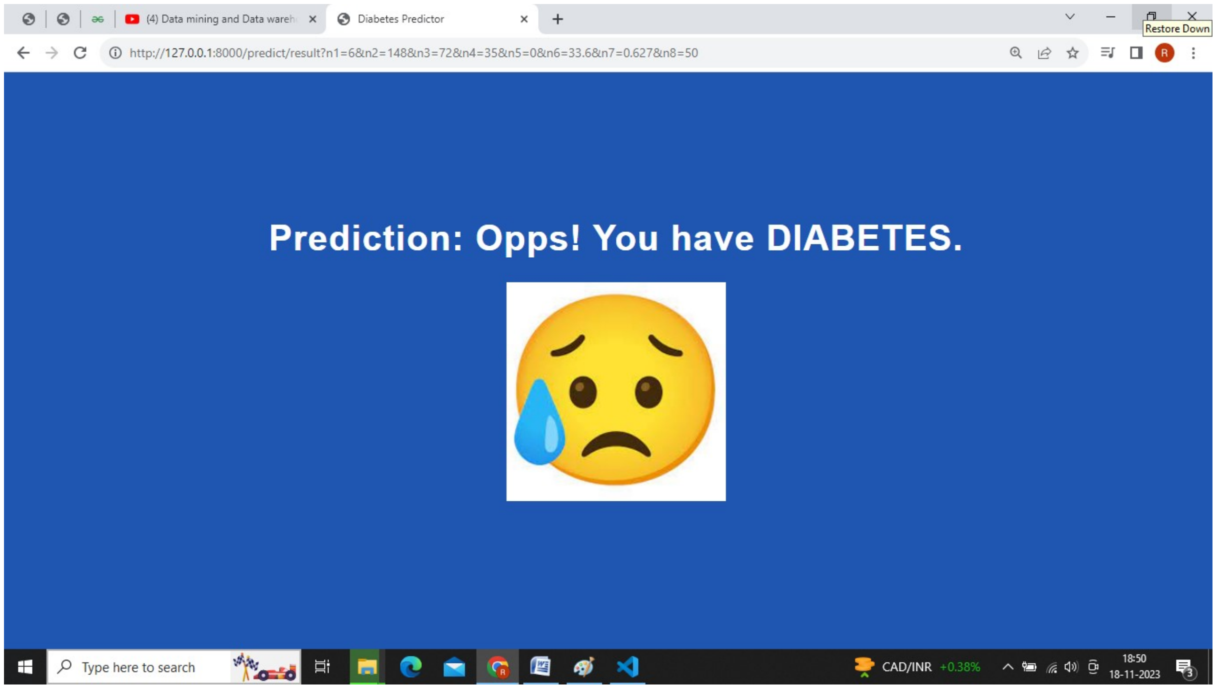
If patient don't have diabetes



http://127.0.0.1:8000/predict/result?n1=7&n2=100&n3=0&n4=0&n5=0&n6=30&n7=0.484&n8=32

Prediction: Wow! You DON'T have diabetes.

If patient have diabetes

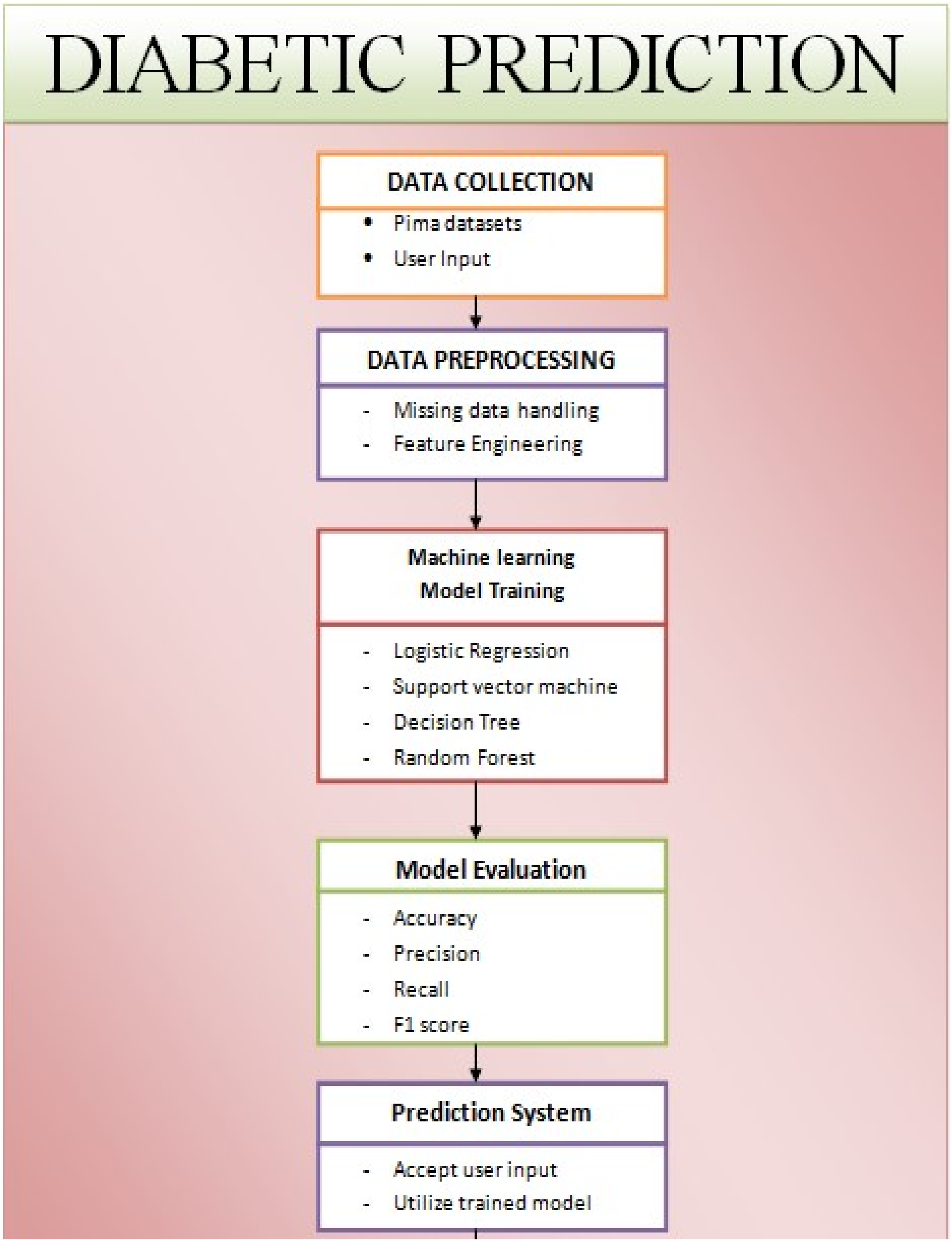


http://127.0.0.1:8000/predict/result?n1=6&n2=148&n3=72&n4=35&n5=0&n6=33.6&n7=0.627&n8=50

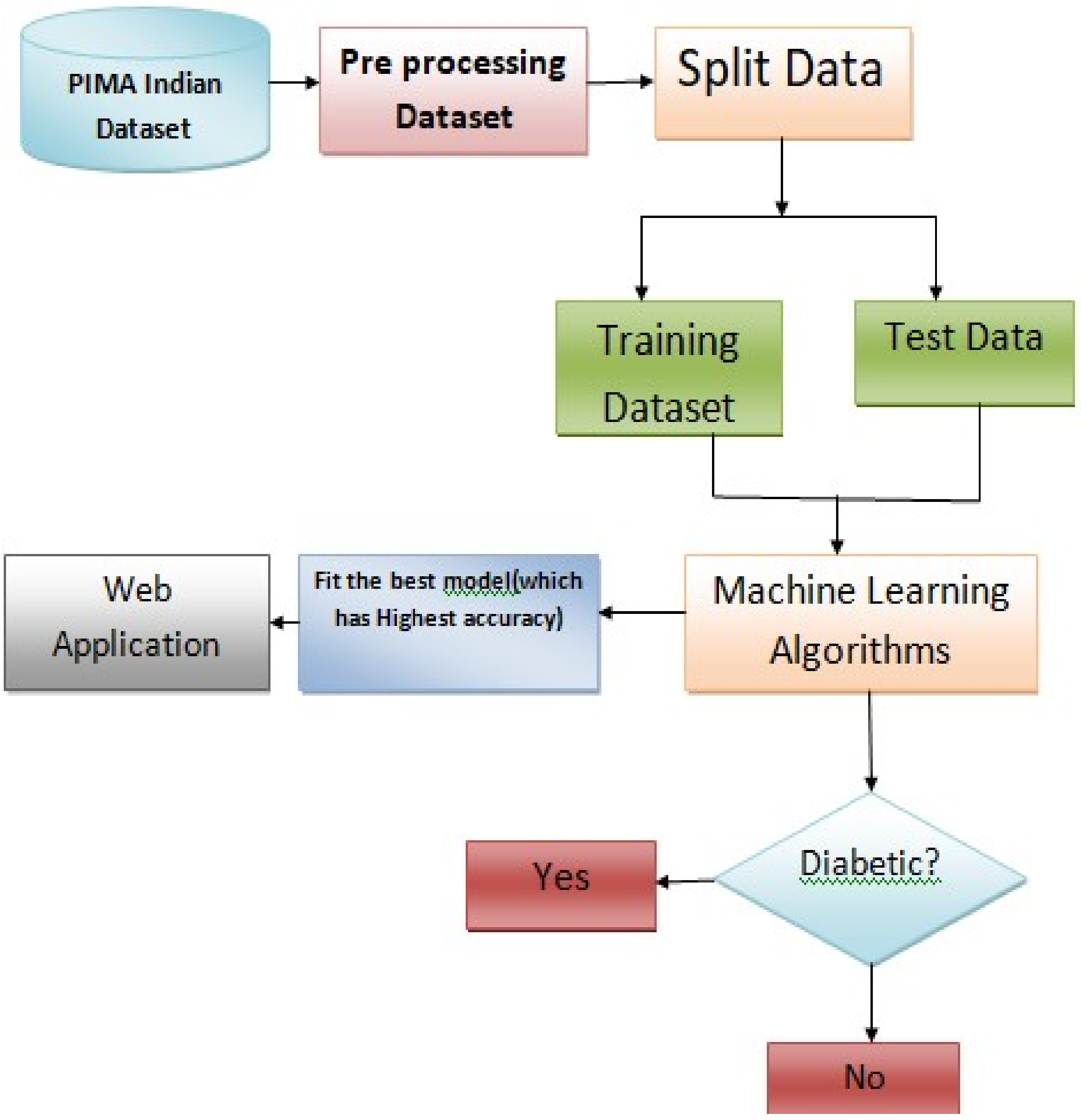
Prediction: Opps! You have DIABETES.

7. LIST OF FIGURES

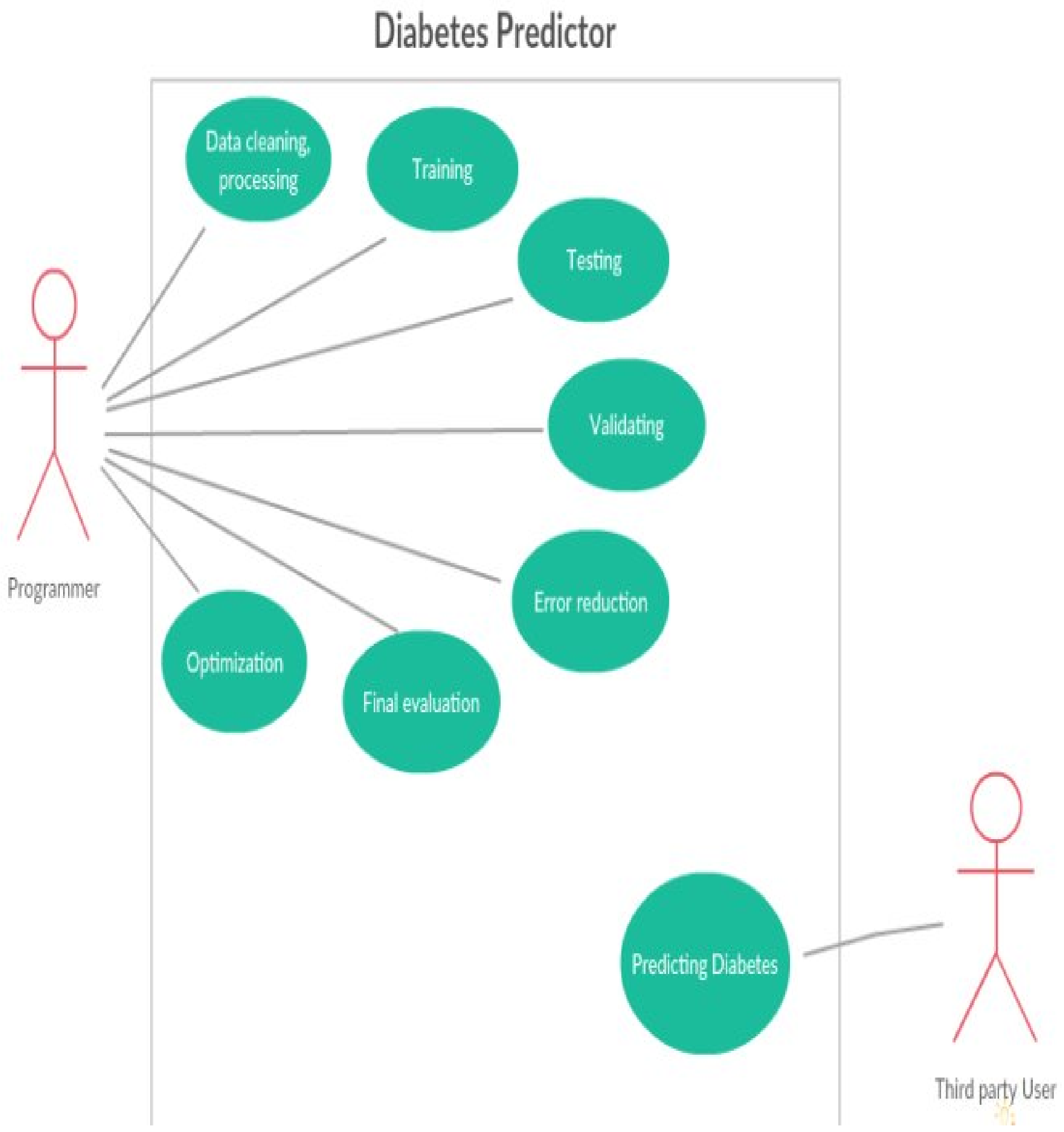
7.1 ARCHITECTURE DIAGRAM



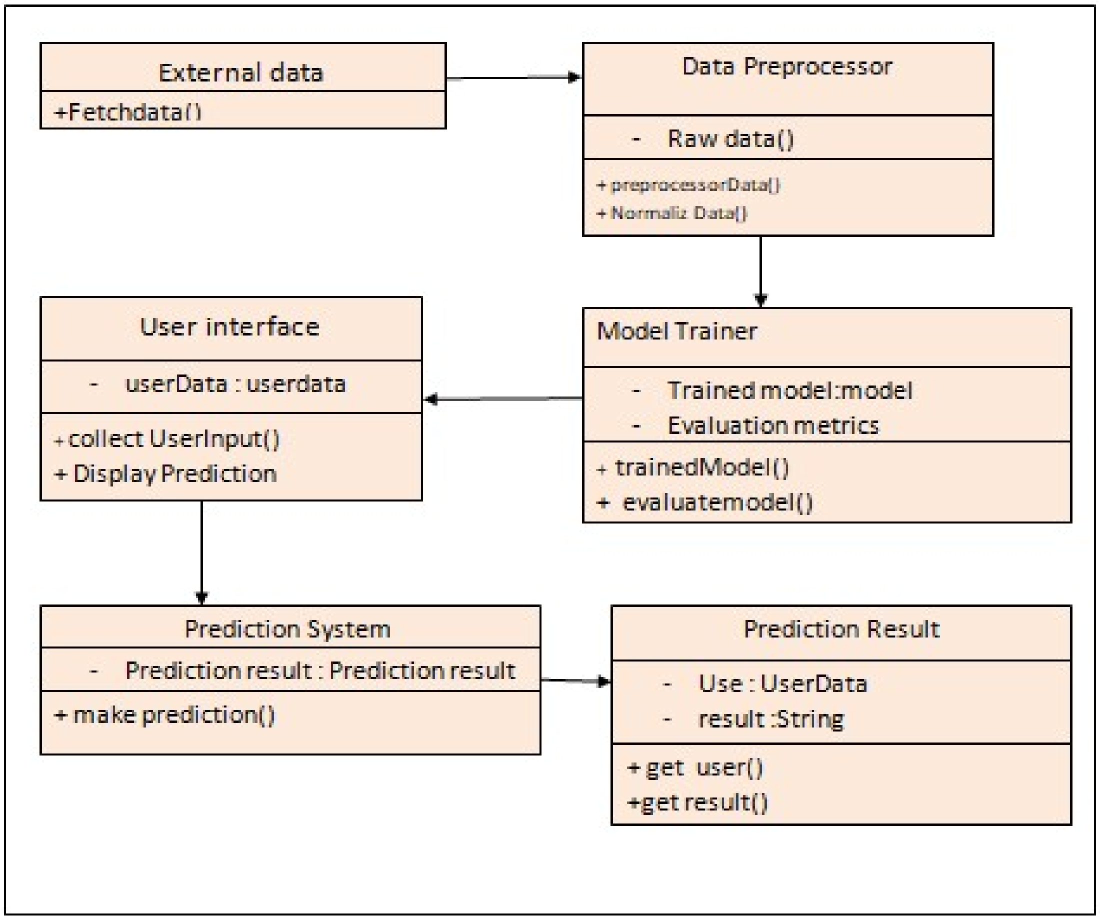
7.2 PROCESS FLOW DIAGRAM



7.3 USE CASE DIAGRAM



7.4 CLASS DIAGRAM



8. TESTING

Test Case Name: Interface Availability

- **Test Input:** Open the Diabetics Prediction interface
- **Expected Result:** The interface should load without errors, displaying the input fields and buttons.
- **Actual Result:** The interface loads correctly.
- **Status:** Pass

Test Case Name: Input Validation

- **Test Input:** Enter invalid characters (e.g., symbols) in the age field.
- **Expected Result:** The system should display an error message indicating invalid input.
- **Actual Result:** The system displays the expected error message.
- **Status:** Pass

Test Case Name: Model Integration

- **Test Input:** Provide valid input for all required fields.
- **Expected Result:** The system should interact with the machine learning model and provide a prediction.
- **Actual Result:** The system successfully integrates with the model and returns a prediction.
- **Status:** Pass

Test Case Name: Handling Missing Data

- **Test Input:** Leave one or more required fields blank.
- **Expected Result:** The system should prompt the user to fill in all required fields.
- **Actual Result:** The system correctly identifies missing data and prompts the user.
- **Status:** Pass

Test Case Name: Response Time

- **Test Input:** Submit a prediction request.
- **Expected Result:** The system should provide a prediction within a reasonable time frame (e.g., within 5 seconds).
- **Actual Result:** The system responds within the expected time frame.
- **Status:** Pass

Test Case Name: User Feedback

- **Test Input:** Submit valid input for prediction.
- **Expected Result:** The system should display a clear and informative prediction result.
- **Actual Result:** The system provides a clear prediction result.
- **Status:** Pass

Test Case Name: Error Handling

- **Test Input:** Introduce a network error during a prediction request.
- **Expected Result:** The system should display a user-friendly error message.
- **Actual Result:** The system presents a user-friendly error message.
- **Status:** Pass

9. CONCLUSION

Diabetes can be a reason for reducing life expectancy and quality. Predicting this chronic disorder earlier can reduce the risk and complications of many diseases in the long run. In this project work, an automatic diabetes prediction system using various machine learning approaches has been proposed. The open-source Pima Indian have been used in this project work. The logistic regression algorithm achieved the best performance with 78% accuracy. Finally, the best-performed logistic regression framework has been deployed into a website application to predict diabetes instantly.
Future scope

10.REFERENCES

1. Mitushi Soni, Dr. Sunita Varma, “Diabetes Prediction using Machine Learning Techniques” International Journal of Engineering Research & Technology (IJERT)
2. https://www.youtube.com/watch?v=E0gaPgiISoU&list=PL-QRwhZe4lltkA6KaWkSYMrl63YaFEeLz
3. https://www.analyticsvidhya.com/blog/2022/01/diabetes-prediction-using-machine-learning
4. Choubey, D.K., Paul, S., Kumar, S., Kumar, S., 2017. Classification of Pima indian diabetes dataset using naive bayes with genetic algorithm as an attribute selection, in: Communication and Computing Systems: Proceedings of the International Conference on Communication and Computing System (ICCCS 2016), pp. 451– 455.