A Mini- Project Report

on

"Diabetes Prediction Tool"

Submitted to the

Pune Institute of Computer Technology, Pune

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

by

Vedant Puranik	43152
Sahil Naphade	43141

Under the guidance of

Prof. R. R. Chhajed

Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

2020-2021

CERTIFICATE

This is to certify that the project report entitled

DIABETES PREDICTION TOOL

Submitted by

Vedant Puranik	43152
Sahil Naphade	43141

is a bonafide work carried out by them under the supervision of Prof. R. R.Chhajed and it is approved for the partial fulfillment of the requirement of **Computer Laboratory -X** for the award of the Degree of Bachelor of Engineering (Information Technology).

Prof. R. R. Chhajed
Lab Teacher
Department of Information Technology

Dr. A. M. Bagade
Head of Department
Department of Information Technology

Place: PICT, Pune

Date: 29/05/2021

ACKNOWLEDGEMENT

We thank everyone who have helped and provided valuable suggestions for successfully creating a wonderful project.

We are very grateful to our guide, Prof. R. R. Chhajed, Head of Department Dr. A. M. Bagade and our principal Dr. P. T. Kulkarni. They have been very supportive and have ensured that all facilities remained available for smooth progress of the project.

We would like to thank our professor A. C. Karve and Prof. R. R. Chhajed for providing very valuable and timely suggestions and help.

Vedant Puranik
Sahil Naphade

ABSTRACT

The recent covid-19 pandemic has taken its toll on the population across the world. Especially, the elderly and the people with comorbidities like Asthma and Diabetes are especially at risk. Having diabetes also increase the chances of contracting the Mucormycosis, a.k.a. the black fungus. Having such comorbidities also delay the healing procedure of covid-19 infected patients and hence, should not go unknown and untreated.

Therefore, through this project, we wish to allow the general population to gain an insight into their diabetes situation to help them manage the disease better; and effectively help them lead a better lifestyle.

LIST OF FIGURES

Figure Number	Figure Title	Page Number
1	Flow of the system	10
2	Home page screenshot	15
3	Predictor page screenshot	15

LIST OF TABLES

Table Number	Table Title	Page Number
1	Empirical study of the algorithms tested	16
2	Confusion matrix	16

TOPICS

1. Introduction

COVID-19 remains the biggest pandemic the world has ever witnessed. India is at the forefront with a huge number of cases and COVID-related deaths. As per reports, elderly people are highly susceptible to the Corona virus and those who have diabetes are even more prone. In general, people with diabetes are more likely to have more severe symptoms and complications when infected with any virus. Your risk of getting very sick from COVID-19 is likely to be lower if your diabetes is well-managed.

Therefore, finding out if a person has diabetes could be a very important step in understanding how severe would your COVID infection be, if you are ever affected by the virus. Lab methods generally take a while to deliver reports stating your diabetic status.

Hence, to provide a better intuition about the same problem, we propose a diabetes prediction tool. This tool takes 8 different inputs from the user and on the basis of that tells the user whether he or she is possibly diabetic. This tool makes use of Random Forest, which is a supervised machine learning technique for classifying whether the person has diabetes or not. As the prediction is purely based on numbers, this tool cannot be used to obtain a final verdict and in doubt the user is advised to contact a physician.

2. Scope and Objective

Scope of our project is to create a web application with considerable accuracy and speed to predict the condition of diabetes in a patient based on the outcomes of his results on certain parameters. The web application will also provide insights into the data gathered by the exploration of the dataset.

Objective of the project is to educate the populace and provide insights into the probability of one having diabetes and the related conditions. The insights into the data explored will also help in understanding the typical signs of one having diabetes and thus, enabling them to tackle the issue earlier.

3. System architecture and flow

We have considered using the Flask framework for the creation of the website. This mainly resorts to the fact that using the same environment of Python for data model and website ensures high cohesion. Flask is a light-weight micro web framework written in Python, which makes use of additional extensions if needed; to provide extra features like form validations, database abstraction layers, etc. It also contains its own light development server and debugger, support for Unit testing, RESTful service dispatching with support for client-side secure cookies.

We have created a website of two pages. First page of the site is an information page, which shows the insights gained after exploration of the data, and a link to the predictor on the second page. First page is available at decorator (/) which redirects to (/home). Second page contains a form which will accept the values input by the user under various heads, which are the features of our dataset, namely Skin Thickness, Blood pressure, BMI, Diabetes Pedigree function, age, etc. This form submits to the function predictor (decorator /predictor). In this function, the form data (the features input by the user) are extracted.

A pickle dump file (.pkl) was created for the trained model. This file basically contains the pretrained model. The values submitted through the form are scaled with one-hot encoding which is defined in the pickle dump file.

These features are then used to predict the value of the output, which outputs 1 for diabetic and 0 for non-diabetic values.

The final value is shown on the same page, just below the submit button.

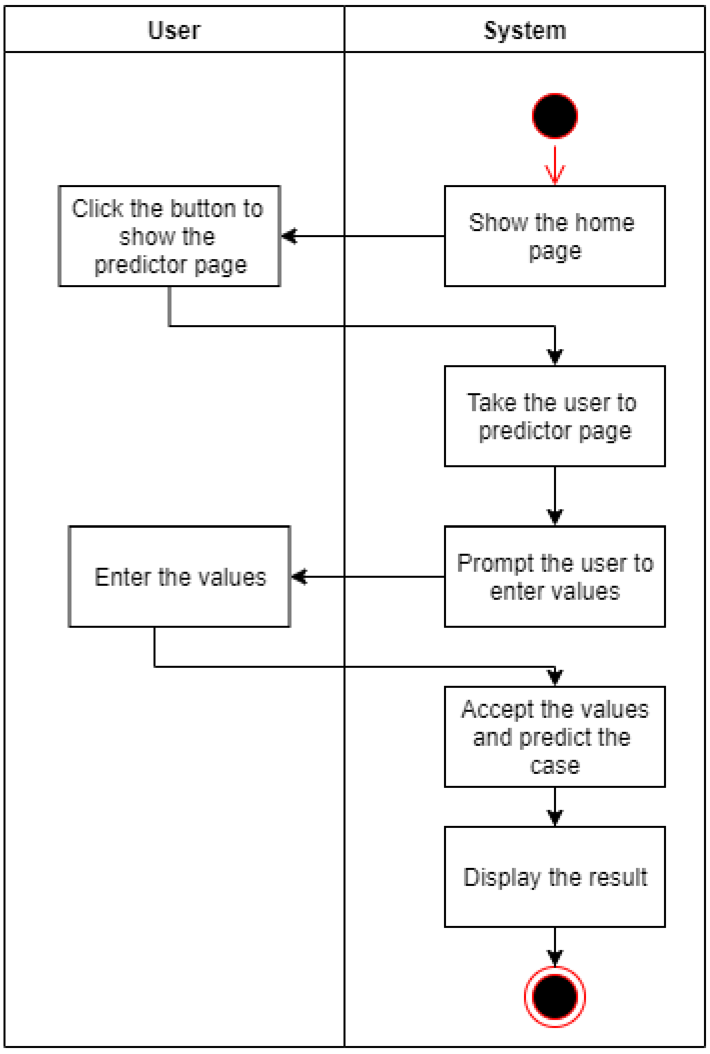
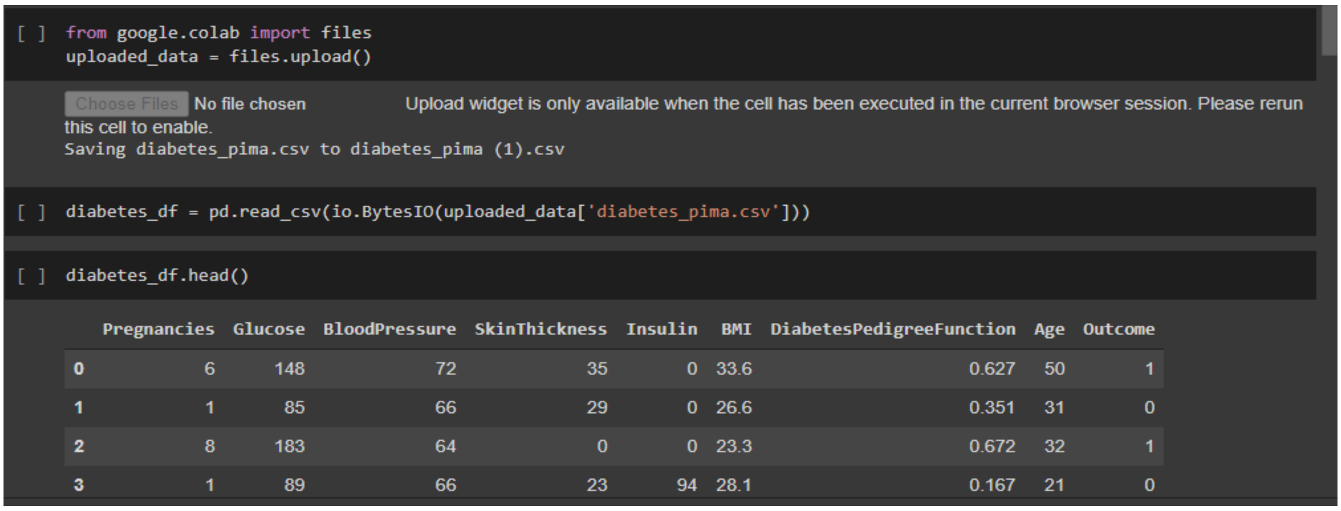

User
System
Show the home page
Click the button to show the predictor page
Take the user to predictor page
Prompt the user to enter values
Enter the values
Accept the values and predict the case
Display the result

Figure 1: Flow of the system

4. Code and snapshots

Code for data model:

4.1 Reading data

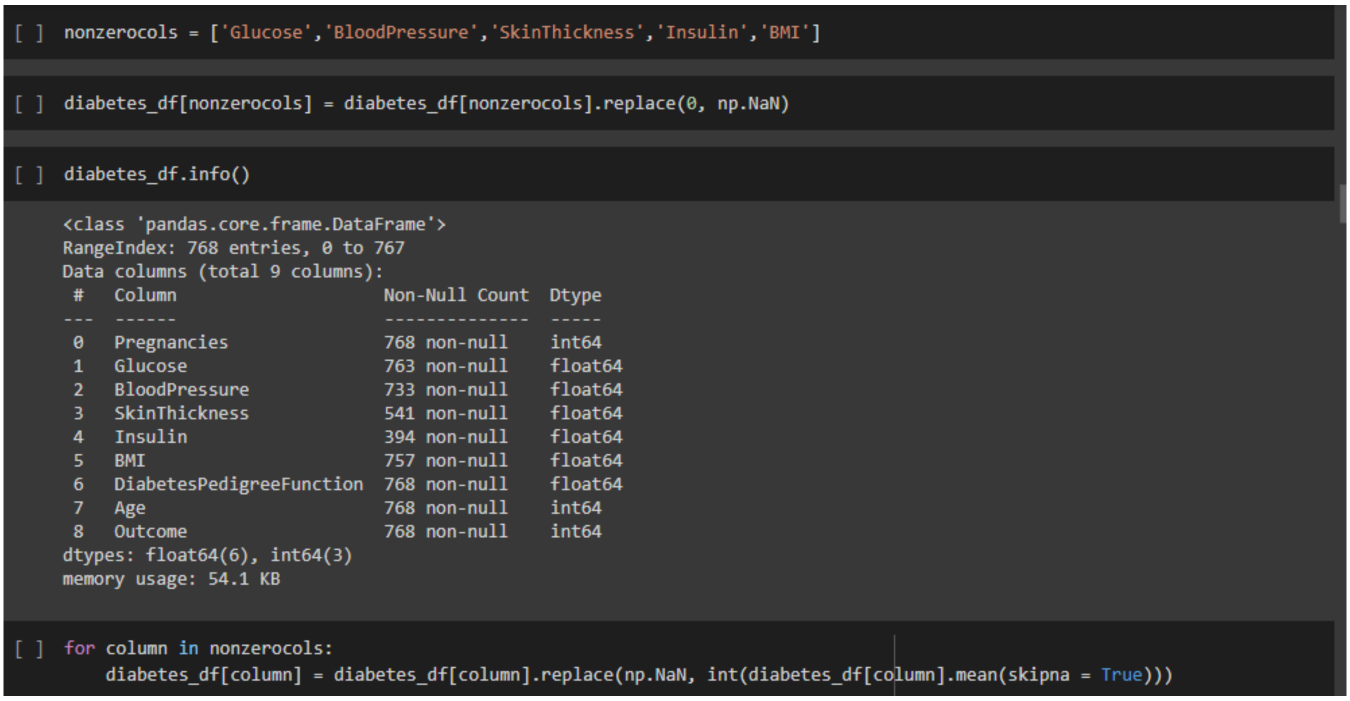


```
[ ] from google.colab import files
    uploaded_data = files.upload()

    Choose Files No file chosen     Upload widget is only available when the cell has been executed in the current browser session. Please rerun
    this cell to enable.
    Saving diabetes_pima.csv to diabetes_pima (1).csv

[ ] diabetes_df = pd.read_csv(io.BytesIO(uploaded_data['diabetes_pima.csv']))

[ ] diabetes_df.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

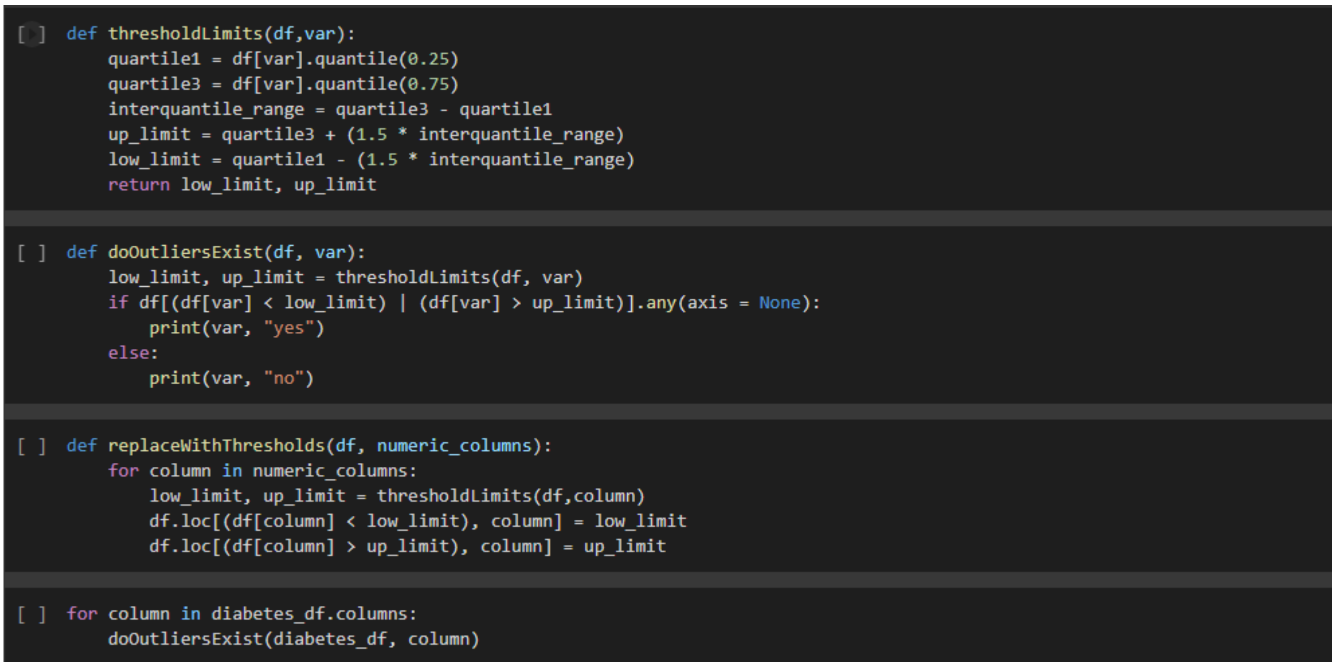
4.2 Preprocessing

4.2.1 Replacing 0's in columns with average value

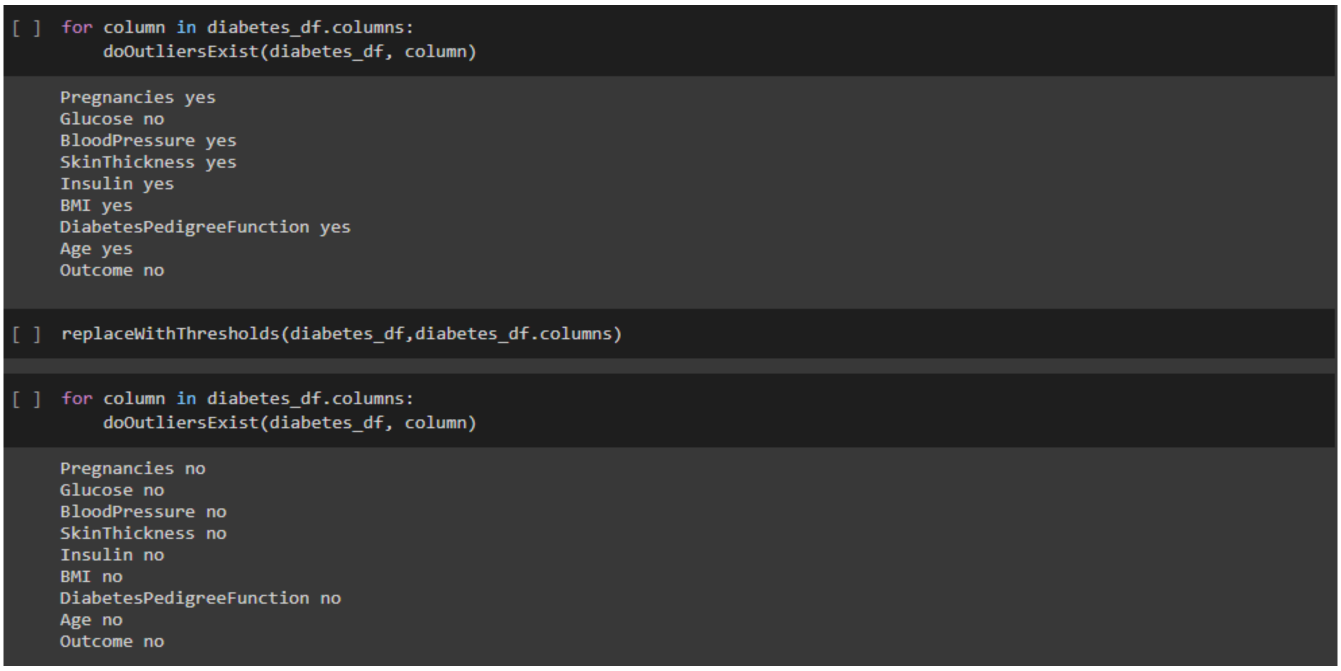


```
[ ] nonzerocols = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']

[ ] diabetes_df[nonzerocols] = diabetes_df[nonzerocols].replace(0, np.NaN)

[ ] 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                   763 non-null    float64
     2   BloodPressure             733 non-null    float64
     3   SkinThickness             541 non-null    float64
     4   Insulin                   394 non-null    float64
     5   BMI                       757 non-null    float64
     6   DiabetesPedigreeFunction  768 non-null    float64
     7   Age                       768 non-null    int64
     8   Outcome                   768 non-null    int64
    dtypes: float64(6), int64(3)
    memory usage: 54.1 KB

[ ] for column in nonzerocols:
        diabetes_df[column] = diabetes_df[column].replace(np.NaN, int(diabetes_df[column].mean(skipna = True)))
```

4.2.2 Detecting and removing outliers

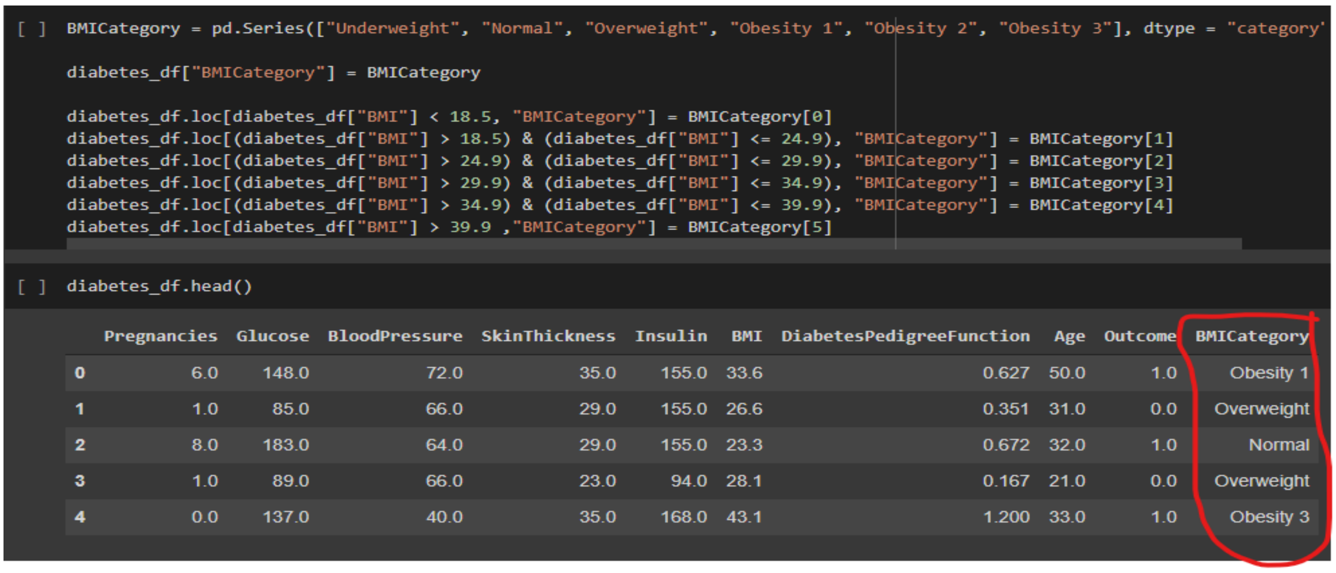


```
[ ] def thresholdLimits(df,var):
        quartile1 = df[var].quantile(0.25)
        quartile3 = df[var].quantile(0.75)
        interquantile_range = quartile3 - quartile1
        up_limit = quartile3 + (1.5 * interquantile_range)
        low_limit = quartile1 - (1.5 * interquantile_range)
        return low_limit, up_limit

[ ] def doOutliersExist(df, var):
        low_limit, up_limit = thresholdLimits(df, var)
        if df[(df[var] < low_limit) | (df[var] > up_limit)].any(axis = None):
            print(var, "yes")
        else:
            print(var, "no")

[ ] def replaceWithThresholds(df, numeric_columns):
        for column in numeric_columns:
            low_limit, up_limit = thresholdLimits(df,column)
            df.loc[(df[column] < low_limit), column] = low_limit
            df.loc[(df[column] > up_limit), column] = up_limit

[ ] for column in diabetes_df.columns:
        doOutliersExist(diabetes_df, column)
```



```
[ ] for column in diabetes_df.columns:
        doOutliersExist(diabetes_df, column)

    Pregnancies yes
    Glucose no
    BloodPressure yes
    SkinThickness yes
    Insulin yes
    BMI yes
    DiabetesPedigreeFunction yes
    Age yes
    Outcome no

[ ] replaceWithThresholds(diabetes_df,diabetes_df.columns)

[ ] for column in diabetes_df.columns:
        doOutliersExist(diabetes_df, column)

    Pregnancies no
    Glucose no
    BloodPressure no
    SkinThickness no
    Insulin no
    BMI no
    DiabetesPedigreeFunction no
    Age no
    Outcome no
```

4.2.3 Converting BMI into a categorical column

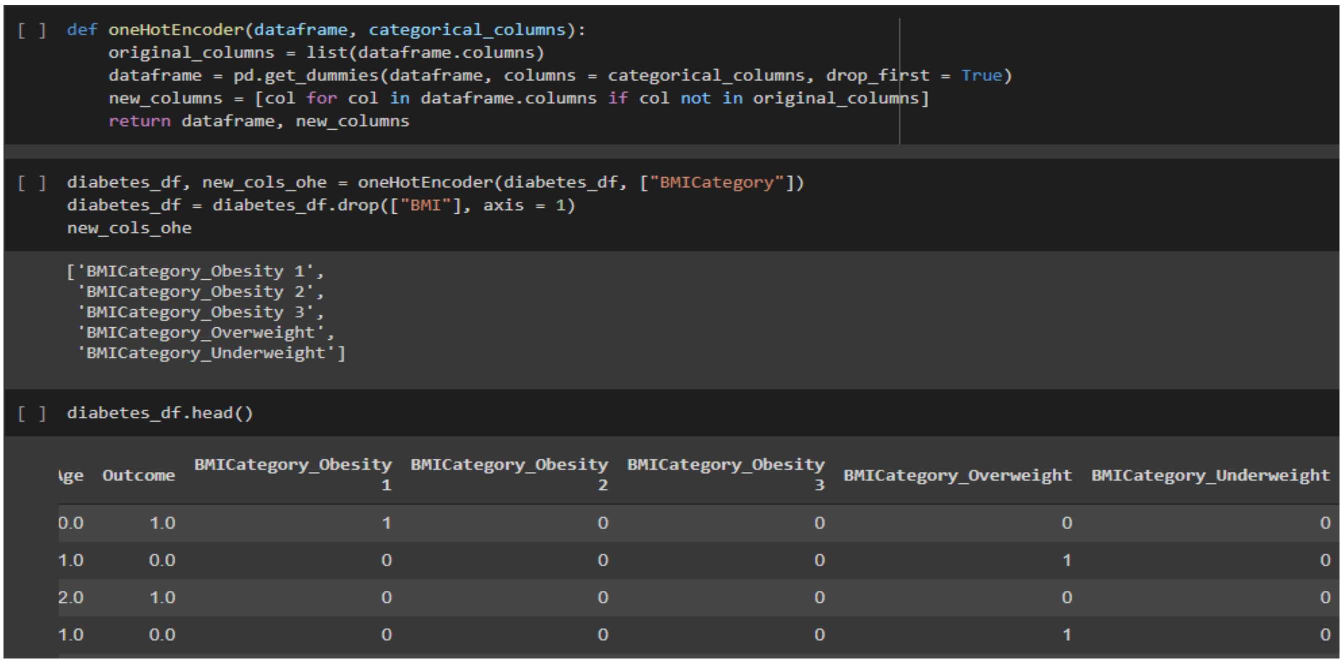


```
[ ] BMICategory = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category

    diabetes_df["BMICategory"] = BMICategory

    diabetes_df.loc[diabetes_df["BMI"] < 18.5, "BMICategory"] = BMICategory[0]
    diabetes_df.loc[(diabetes_df["BMI"] > 18.5) & (diabetes_df["BMI"] <= 24.9), "BMICategory"] = BMICategory[1]
    diabetes_df.loc[(diabetes_df["BMI"] > 24.9) & (diabetes_df["BMI"] <= 29.9), "BMICategory"] = BMICategory[2]
    diabetes_df.loc[(diabetes_df["BMI"] > 29.9) & (diabetes_df["BMI"] <= 34.9), "BMICategory"] = BMICategory[3]
    diabetes_df.loc[(diabetes_df["BMI"] > 34.9) & (diabetes_df["BMI"] <= 39.9), "BMICategory"] = BMICategory[4]
    diabetes_df.loc[diabetes_df["BMI"] > 39.9 ,"BMICategory"] = BMICategory[5]
```

```
[ ] diabetes_df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction   Age  Outcome  BMICategory
0          6.0    148.0           72.0           35.0    155.0  33.6                     0.627  50.0      1.0    Obesity 1
1          1.0     85.0           66.0           29.0    155.0  26.6                     0.351  31.0      0.0   Overweight
2          8.0    183.0           64.0           29.0    155.0  23.3                     0.672  32.0      1.0       Normal
3          1.0     89.0           66.0           23.0     94.0  28.1                     0.167  21.0      0.0   Overweight
4          0.0    137.0           40.0           35.0    168.0  43.1                     1.200  33.0      1.0    Obesity 3
```

4.2.4 One-hot encoding BMI categories



```
[ ] def oneHotEncoder(dataframe, categorical_columns):
        original_columns = list(dataframe.columns)
        dataframe = pd.get_dummies(dataframe, columns = categorical_columns, drop_first = True)
        new_columns = [col for col in dataframe.columns if col not in original_columns]
        return dataframe, new_columns
```

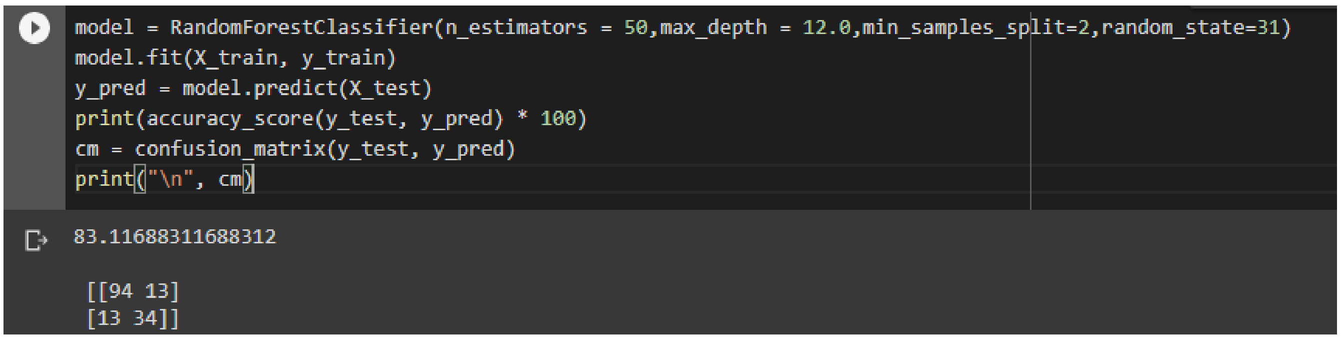
```
[ ] diabetes_df, new_cols_ohe = oneHotEncoder(diabetes_df, ["BMICategory"])
    diabetes_df = diabetes_df.drop(["BMI"], axis = 1)
    new_cols_ohe
```

```
    ['BMICategory_Obesity 1',
     'BMICategory_Obesity 2',
     'BMICategory_Obesity 3',
     'BMICategory_Overweight',
     'BMICategory_Underweight']
```

```
[ ] diabetes_df.head()
```

```
ge  Outcome  BMICategory_Obesity  BMICategory_Obesity  BMICategory_Obesity  BMICategory_Overweight  BMICategory_Underweight
                               1                    2                    3
0.0     1.0                    1                    0                    0                       0                        0
1.0     0.0                    0                    0                    0                       1                        0
2.0     1.0                    0                    0                    0                       0                        0
1.0     0.0                    0                    0                    0                       1                        0
```

4.3 Hyperparameter-tuned Model Training and Result



```
model = RandomForestClassifier(n_estimators = 50,max_depth = 12.0,min_samples_split=2,random_state=31)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred) * 100)
cm = confusion_matrix(y_test, y_pred)
print("\n", cm)
```

```
83.11688311688312

 [[94 13]
 [13 34]]
```

BACKEND

4.4 Code for extracting the values through the form:

Note: This is the code inside the predictor function.

```
if request.method == "POST" and form.validate():
    pregnancies = form.pregnancies.data
    glucose = form.glucose.data
    bp = form.bp.data
    st = form.st.data
    insulin = form.insulin.data
    bmi = form.bmi.data
    pedigreeFunc = form.diabetesPedigree.data
    age = form.bmi.data
```

4.5 Code for loading the pickle file and predict the output:

```
scalerLoad = pickle.load(open("scaler.pkl", "rb"))

modelLoad = pickle.load(open("randomForestModel.sav", "rb"))
featureList = np.array([pregnancies, glucose, bp, st, insulin, pedigreeFunc, age]).reshape(1, -1)

featureList = scalerLoad.transform(featureList)

finalFeatures = np.concatenate((featureList, BMIVar), axis = 1)
value = modelLoad.predict(finalFeatures)
print("Value = {}".format(value))
```

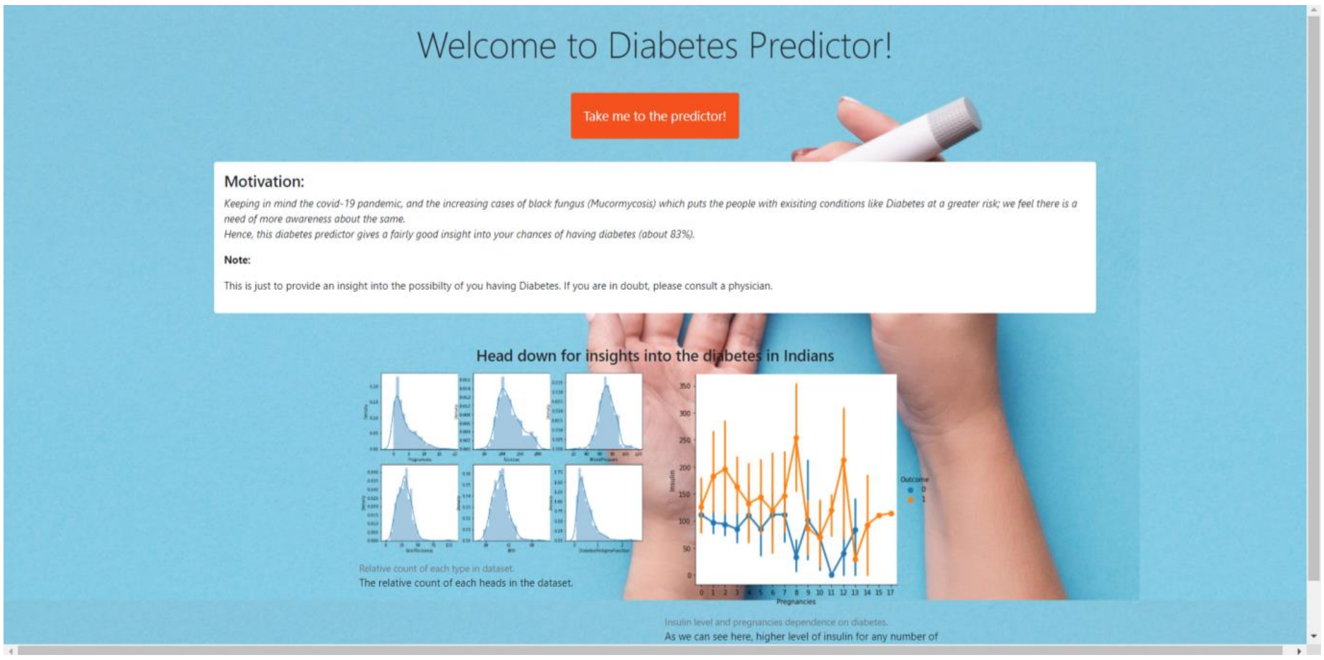
The modelLoad is the loaded model from the pickle file; whereas scalerLoad variable is the scalar defined and trained for the dataset using sklearn.

Finally, the numpy array of features; finalFeatures; is used to predict the outcome from the given data.

Snapshots of the project:

4.6 UI

4.6.1 Homepage



Welcome to Diabetes Predictor!

Take me to the predictor!

Motivation:

Keeping in mind the covid-19 pandemic, and the increasing cases of black fungus (Mucormycosis) which puts the people with exisiting conditions like Diabetes at a greater risk; we feel there is a need of more awareness about the same.
Hence, this diabetes predictor gives a fairly good insight into your chances of having diabetes (about 83%).

Note:

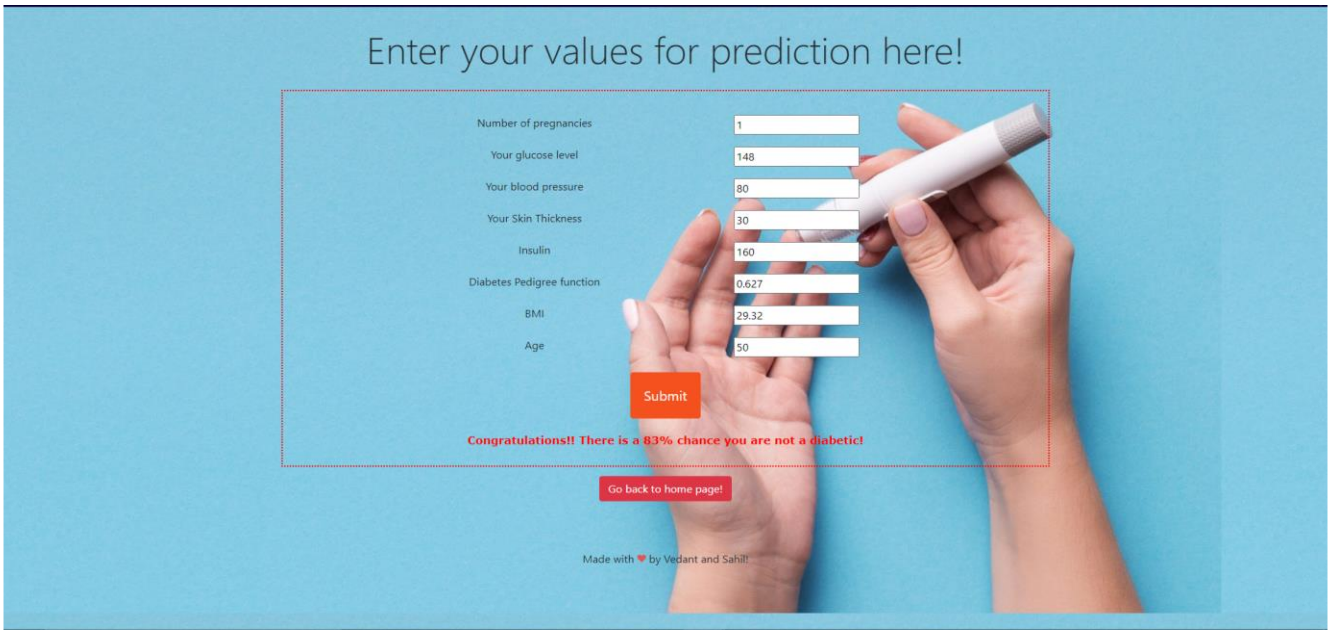
This is just to provide an insight into the possibilty of you having Diabetes. If you are in doubt, please consult a physician.

Head down for insights into the diabetes in Indians

Relative count of each type in dataset.
The relative count of each heads in the dataset.

Insulin level and pregnancies dependence on diabetes.
As we can see here, higher level of insulin for any number of

4.6.2 Predictor page



Enter your values for prediction here!

Number of pregnancies	1
Your glucose level	148
Your blood pressure	80
Your Skin Thickness	30
Insulin	160
Diabetes Pedigree function	0.627
BMI	29.32
Age	50

Submit

Congratulations!! There is a 83% chance you are not a diabetic!

Go back to home page!

Made with ❤ by Vedant and Sahil!

5. Results

To get an estimate of the highest accuracy achievable on our dataset, we implemented a number of machine learning algorithms for the same. Out of all the algorithms tried, Random Forest performed the best with a test accuracy of 83%. The open-source dataset we used is called the PIMA Indians Diabetes Dataset, which is a standard dataset used for diabetes related problem statements.

Model	**Test Accuracy**
Naive Bayes	33.11%
Decision Tree	70.12%
K-Nearest Neighbors	79.22%
XGBoost	79.8%
SVM	80.5%
Logistic Regression	81.16%
Random Forest	83.11%

Following is an empirical analysis of several supervised machine learning algorithms we tried:

As Random Forest performed the best as compared to others, we used other metrics to get a better idea about the model's classification ability, for which we used confusion matrix.

Confusion Matrix:

	Predicted True	Predicted False
Actually True	94 (TP)	13 (FN)
Actually False	13 (FP)	34 (TN)

Precision = TP/(TP+FP) = 94/(94+13) = 0.878

Recall = TP/(TP+FN) = 94/(94+13) = 0.878

F1-score = 2*precision*recall/(precision + recall) = 0.878

6. Conclusion and Future Scope

Hence, as COVID-19 remains the biggest health problem faced by the world today, knowing whether or not a person has diabetes can help him assess his health and be prepared in advance for steps to be taken, as diabetics face more complications than non-diabetics. To get a numerical prediction, we employed several supervised machine learning algorithms, out of which Random Forest proved to deliver the highest accuracy of 83%. This tool uses traditional machine learning. If more data is available, using neural networks for classification would be a better solution in the future. Additionally, also making an integrated app for obtaining preliminary predictions of several diseases related to kidney, liver, heart, etc. would prove to provide a one-stop solution to a number of problems.

GitHub link for project: https://github.com/ItsNaSa/Diabetes-Predictor

7. References

1. https://machinelearningmastery.com/
2. https://scikit-learn.org/stable/
3. https://www.datacamp.com/community/tutorials/pickle-python-tutorial
4. https://www.kaggle.com/uciml/pima-indians-diabetes-database
5. https://flask.palletsprojects.com/_/downloads/en/1.1.x/pdf/
6. Deploy a machine learning model using flask