Data Glacier

Your Deep Learning Partner

Week 4: ML model Deployment on Flask

Name: Ashish Sasanapuri

Batch Code: LISUM26

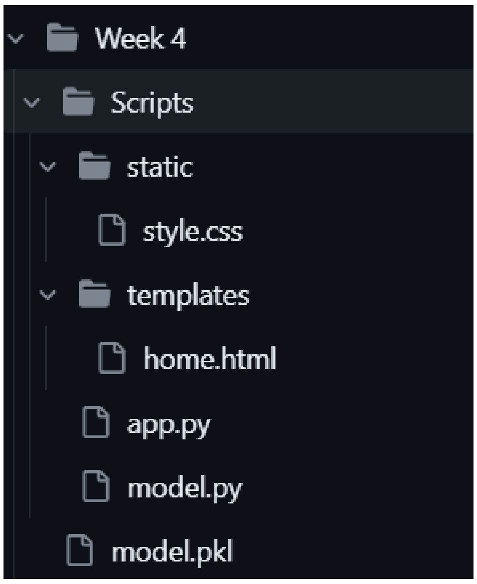
Submission Date: 27/10/2023

Submitted To: GitHub

1. Introduction

The aim of this project is to build a machine learning model and deploy it on Flask. For this purpose, we selected a diabetes dataset which is provided by Sklearn and used it to train a Linear Regression model for prediction of diabetes progression after 1 year of baseline.

The folder structure of the project goes as displayed below:



```
Week 4
  Scripts
    static
      style.css
    templates
      home.html
    app.py
    model.py
  model.pkl
```

The project files, in order, of this project include:

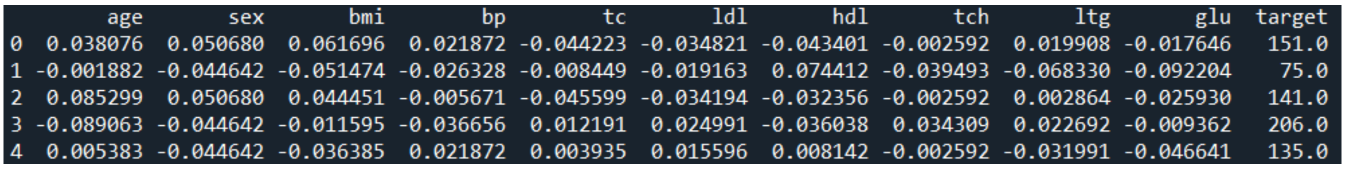
- **model.py** – Predicts the progression of diabetes and saves the model as model.pkl.
- **app.py** – Contains APIs to fetch data input by user on webpage and calculate the predicted result.
- **home.html** – Displays the UI for the user to input the values of each feature.
- **style.css** – Stylesheet for the webpage.

2. Explaining the files

This section will describe **model.py**, **app.py**, and **home.html** file and their outcomes in detail.

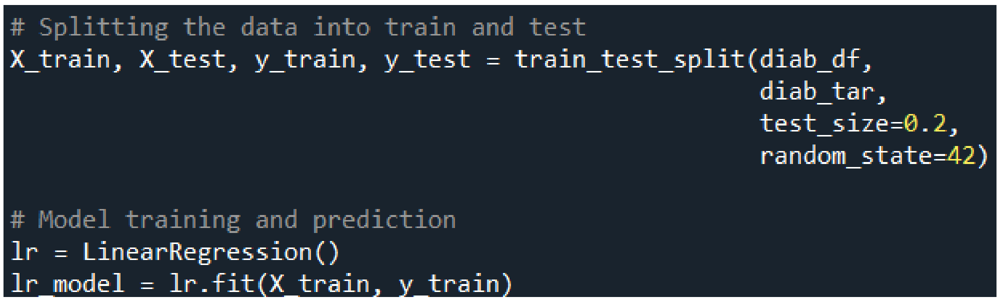
a. Model.py

The dataset acquired from Sklearn contains 10 features and 1 target. The features include Age, Sex, Body Mass Index (BMI), Blood Pressure (BP), Total Serum Cholesterol (TC), Low-Density Lipoproteins (LDL), High-Density Lipoproteins (HDL), Total Cholesterol (TCH), Possibly log of serum triglycerides level (LTG), and Blood Sugar level (GLU). The target feature is a measure of the disease progression 1 year after the baseline. The values have already been mean centred and scaled by the standard deviation times of square root of number samples i.e., 442. Hence, the values lie between 1 and -1.



```
        age       sex       bmi        bp        tc       ldl       hdl       tch       ltg       glu  target
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401 -0.002592  0.019908 -0.017646   151.0
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412 -0.039493 -0.068330 -0.092204    75.0
2  0.085299  0.050680  0.044451 -0.005671 -0.045599 -0.034194 -0.032356 -0.002592  0.002864 -0.025930   141.0
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038  0.034309  0.022692 -0.009362   206.0
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142 -0.002592 -0.031991 -0.046641   135.0
```

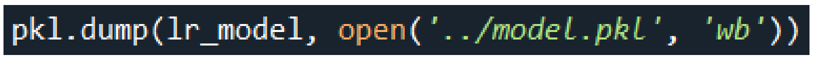
We split the dataset into train – to train the model on and test – to test the model on, and use Linear Regression for prediction.



```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(diab_df,
                                                    diab_tar,
                                                    test_size=0.2,
                                                    random_state=42)

# Model training and prediction
lr = LinearRegression()
lr_model = lr.fit(X_train, y_train)
```

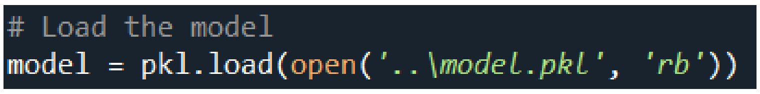
The trained model is then saved using **Pickle** library. This model can be later loaded in **app.py** file for prediction of user inputs.



```
pkl.dump(lr_model, open('../model.pkl', 'wb'))
```

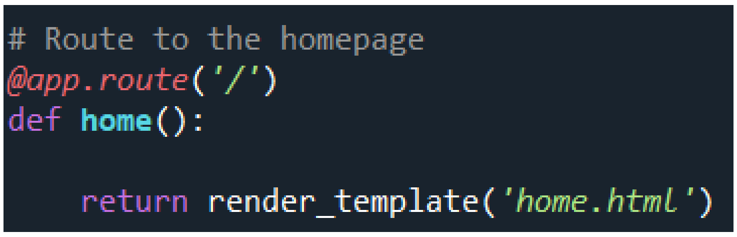
b. App.py

This is the most important python file which uses **Flask** for model deployment and creating the APIs. The model is loaded in this file for prediction of the user inputs.



```
# Load the model
model = pkl.load(open('..\model.pkl', 'rb'))
```

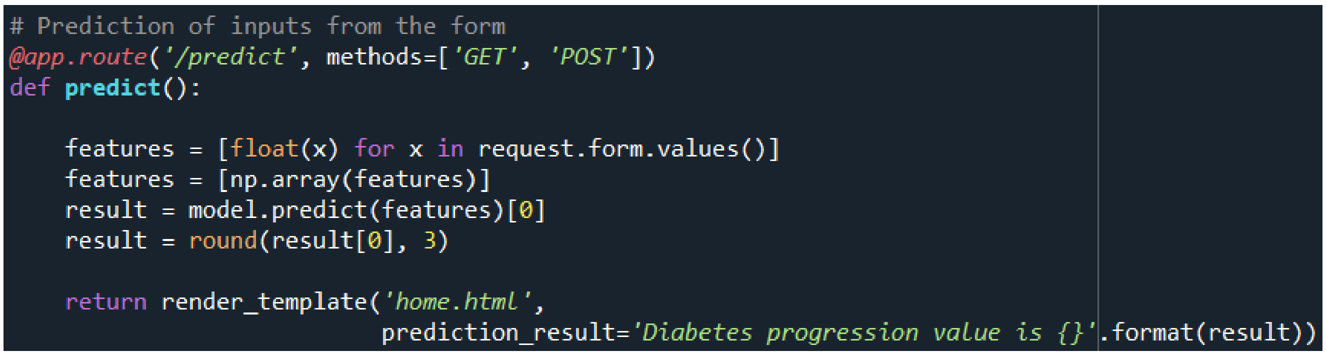
We import flask libraries such as **Flask**, **request** and **render_template**. The below code will direct to the homepage of the web application.



```
# Route to the homepage
@app.route('/')
def home():

    return render_template('home.html')
```

The next code below is triggered when the user has input all the required features on the form appearing on the homepage and submits the form. As described in the code, the form values are fetched via **request.form.values()** function and store the floated values as an array since the model takes the input features as an array. The loaded model predicts the input values and stores the value as **result**. We round the resulting value up to 3 decimal points. The **render_template()** then populates the result on the homepage under the submit button.



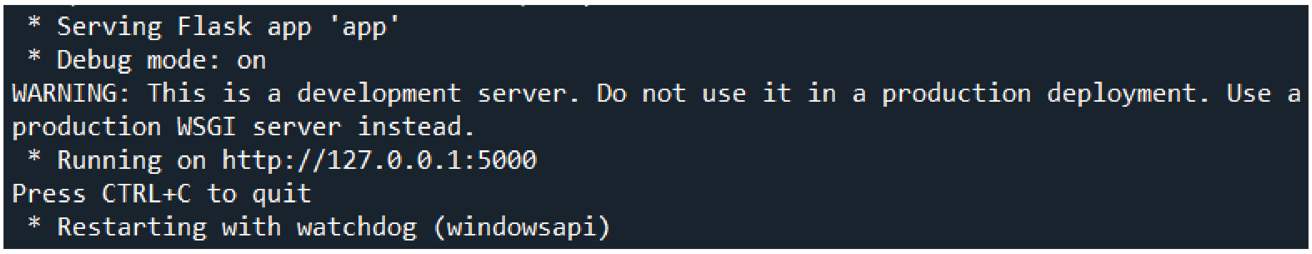
```
# Prediction of inputs from the form
@app.route('/predict', methods=['GET', 'POST'])
def predict():

    features = [float(x) for x in request.form.values()]
    features = [np.array(features)]
    result = model.predict(features)[0]
    result = round(result[0], 3)

    return render_template('home.html',
                           prediction_result='Diabetes progression value is {}'.format(result))
```

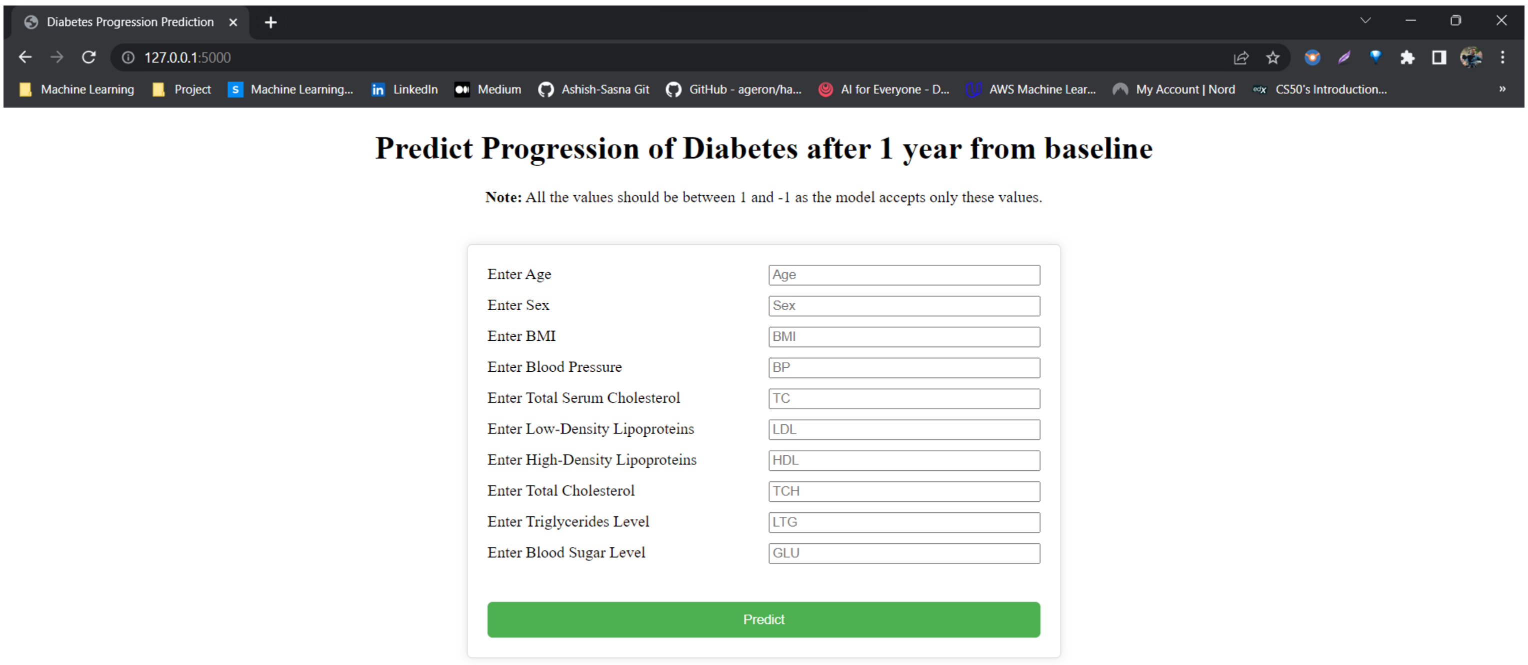
c. Home.html

This section describes the homepage of the web application. When we run the **app.py** python file, the console will generate the local url – 'http://127.0.0.1:5000/' for the web page.



```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```

We can open the homepage by accessing this URL in a browser.

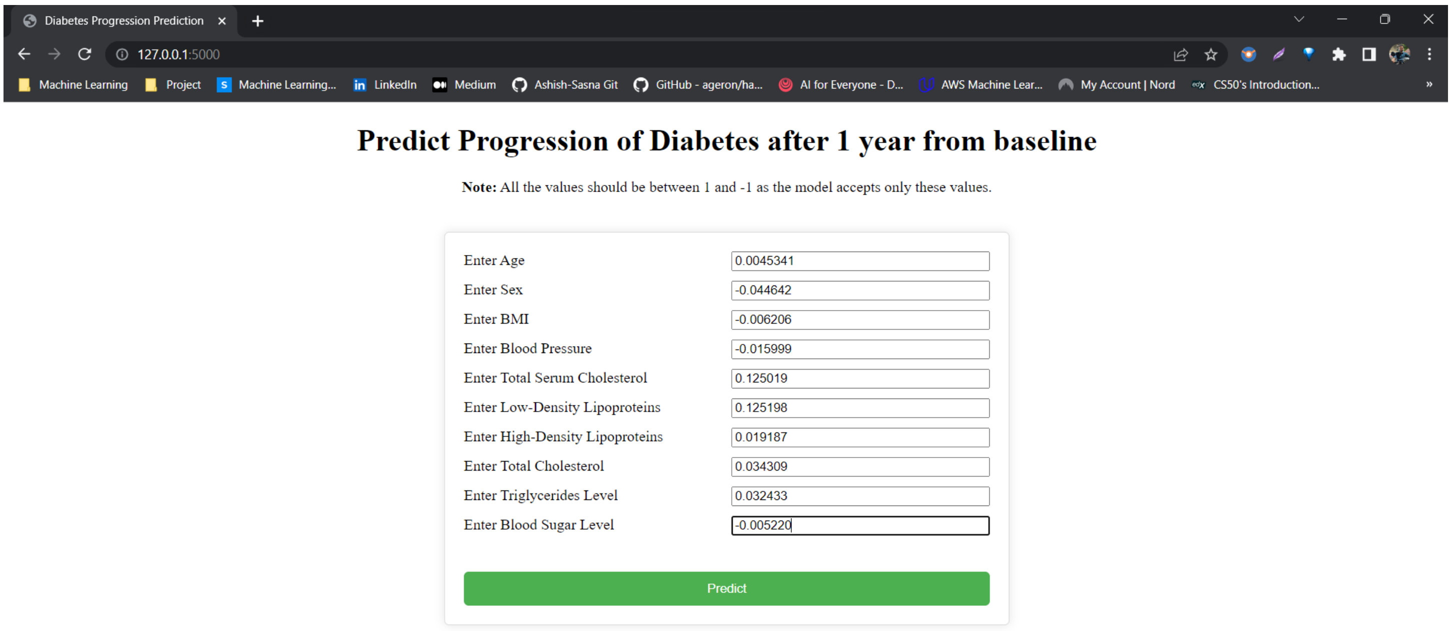


Predict Progression of Diabetes after 1 year from baseline

Note: All the values should be between 1 and -1 as the model accepts only these values.

Enter Age	Age
Enter Sex	Sex
Enter BMI	BMI
Enter Blood Pressure	BP
Enter Total Serum Cholesterol	TC
Enter Low-Density Lipoproteins	LDL
Enter High-Density Lipoproteins	HDL
Enter Total Cholesterol	TCH
Enter Triglycerides Level	LTG
Enter Blood Sugar Level	GLU

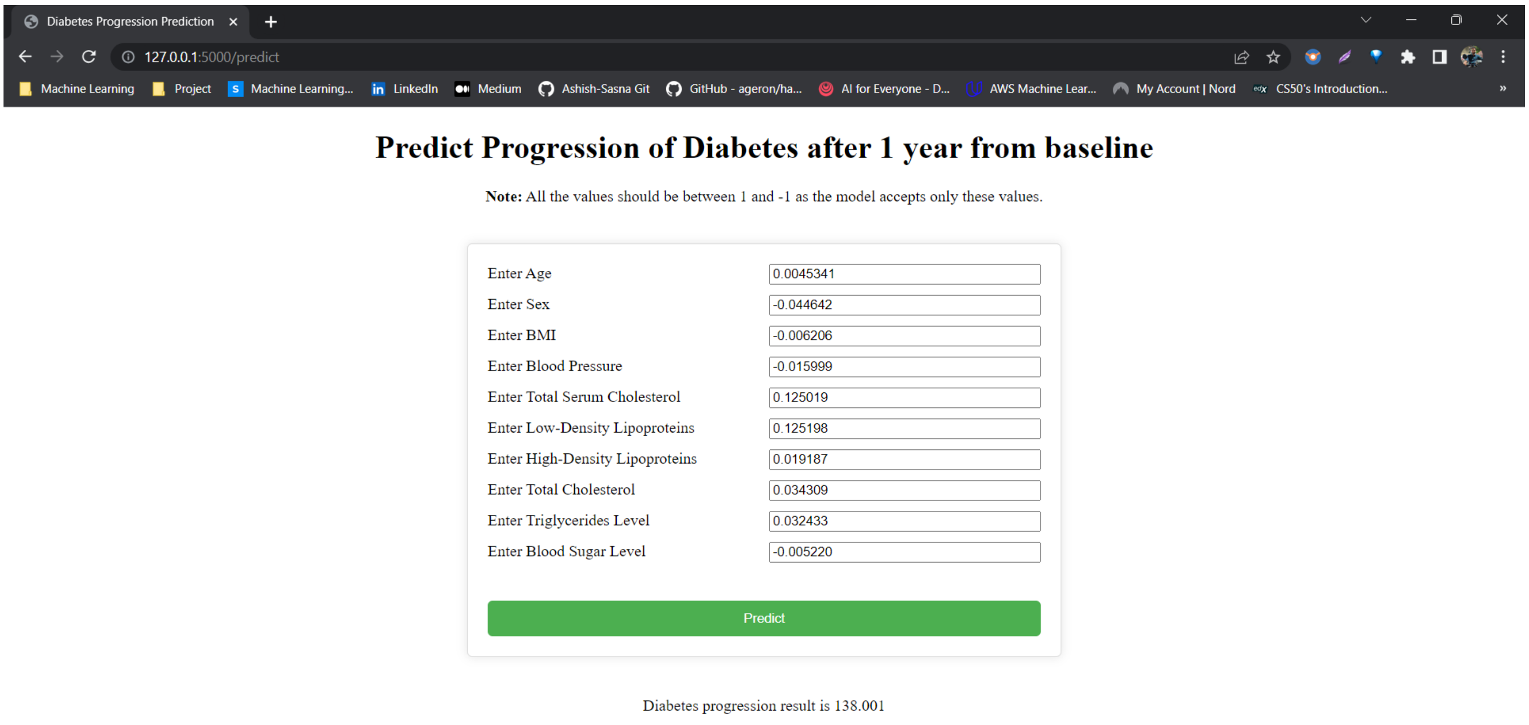
The webpage shows the form that the users can access to enter the values as displayed. Once the field are filled, the user can submit the form using **Predict** button at the bottom. This will trigger the predict API in the **app.py** file and calculate the results.



Predict Progression of Diabetes after 1 year from baseline

Note: All the values should be between 1 and -1 as the model accepts only these values.

Enter Age	0.0045341
Enter Sex	-0.044642
Enter BMI	-0.006206
Enter Blood Pressure	-0.015999
Enter Total Serum Cholesterol	0.125019
Enter Low-Density Lipoproteins	0.125198
Enter High-Density Lipoproteins	0.019187
Enter Total Cholesterol	0.034309
Enter Triglycerides Level	0.032433
Enter Blood Sugar Level	-0.005220



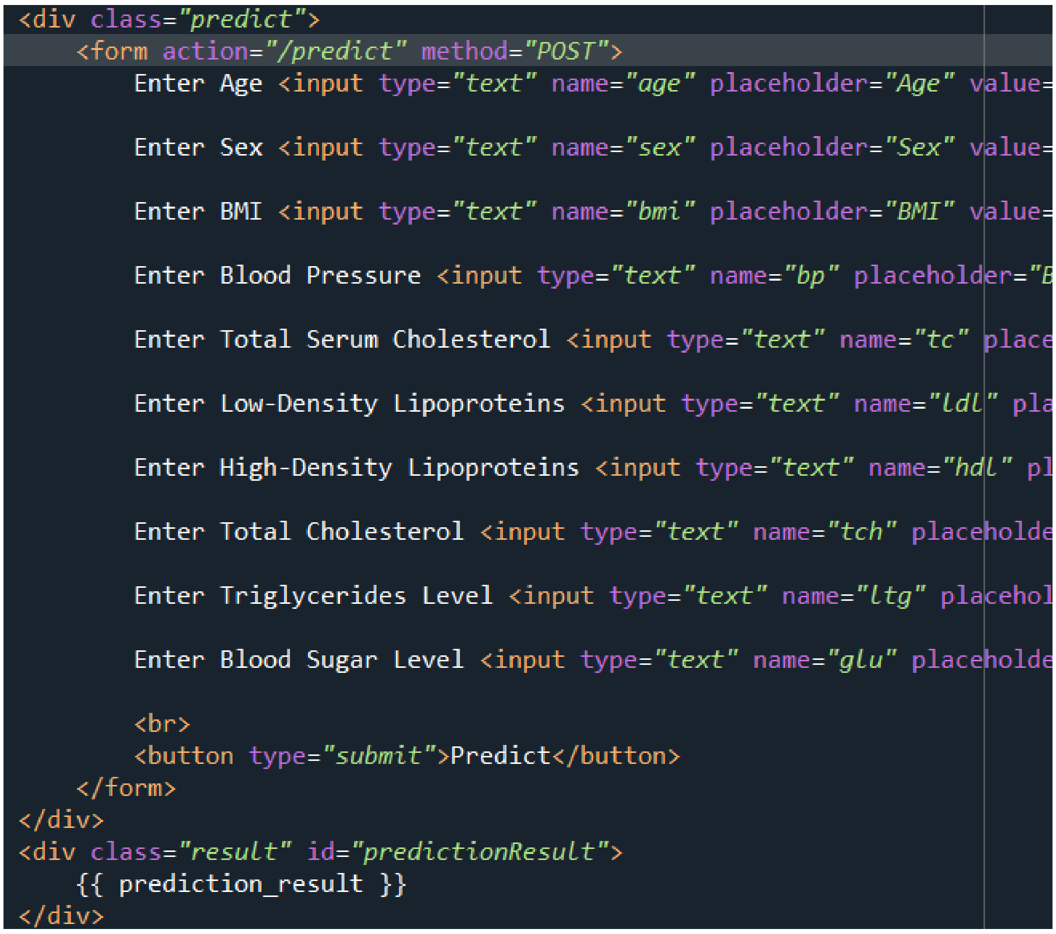
Predict Progression of Diabetes after 1 year from baseline

Note: All the values should be between 1 and -1 as the model accepts only these values.

Enter Age	0.0045341
Enter Sex	-0.044642
Enter BMI	-0.006206
Enter Blood Pressure	-0.015999
Enter Total Serum Cholesterol	0.125019
Enter Low-Density Lipoproteins	0.125198
Enter High-Density Lipoproteins	0.019187
Enter Total Cholesterol	0.034309
Enter Triglycerides Level	0.032433
Enter Blood Sugar Level	-0.005220

Diabetes progression result is 138.001

As you can see, after hitting the **Predict** button, the result is displayed below. This can be explained by the html code in below figure. The **input tag** takes the input from the user for all the 10 features. After submitting the form, the **/predict** API will calculate the result and render the result at **{{prediction_result}}** section.



```
<div class="predict">
    <form action="/predict" method="POST">
        Enter Age <input type="text" name="age" placeholder="Age" value=

        Enter Sex <input type="text" name="sex" placeholder="Sex" value=

        Enter BMI <input type="text" name="bmi" placeholder="BMI" value=

        Enter Blood Pressure <input type="text" name="bp" placeholder="B

        Enter Total Serum Cholesterol <input type="text" name="tc" place

        Enter Low-Density Lipoproteins <input type="text" name="ldl" pla

        Enter High-Density Lipoproteins <input type="text" name="hdl" pl

        Enter Total Cholesterol <input type="text" name="tch" placeholde

        Enter Triglycerides Level <input type="text" name="ltg" placehol

        Enter Blood Sugar Level <input type="text" name="glu" placeholde

        <br>
        <button type="submit">Predict</button>
    </form>
</div>
<div class="result" id="predictionResult">
    {{ prediction_result }}
</div>
```