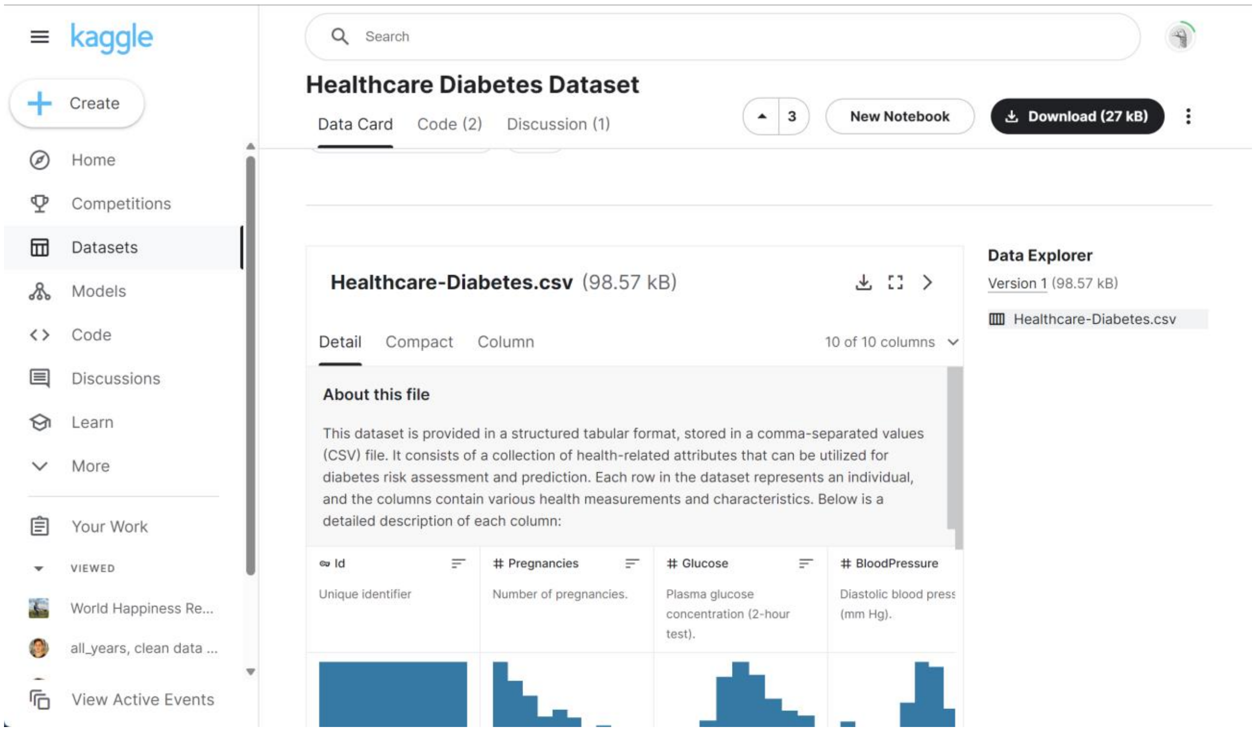
Ian Paulino 08/28/2023
LISUM24

# Logistic Regression Model to predict Diabetes deployment steps

1. Downloaded Diabetes dataset from Kaggle.



Healthcare Diabetes Dataset

Healthcare-Diabetes.csv (98.57 kB)

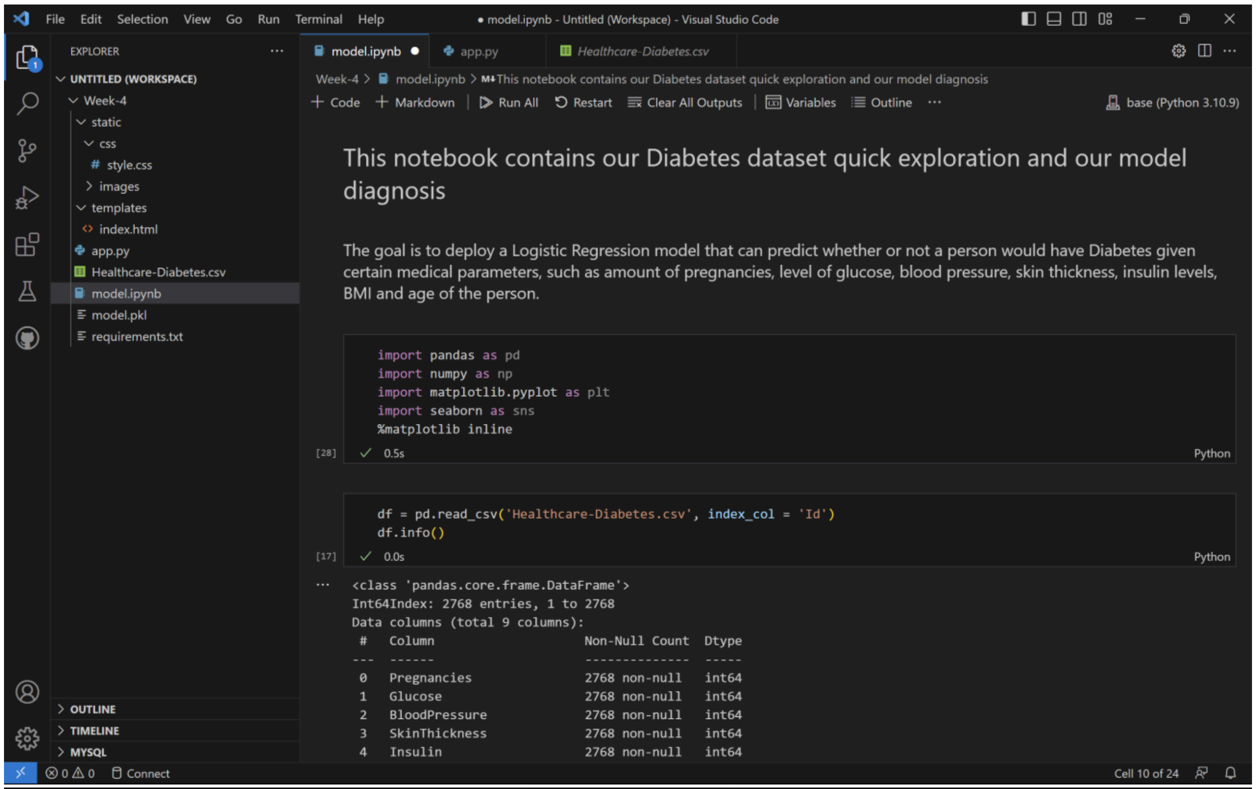
About this file

This dataset is provided in a structured tabular format, stored in a comma-separated values (CSV) file. It consists of a collection of health-related attributes that can be utilized for diabetes risk assessment and prediction. Each row in the dataset represents an individual, and the columns contain various health measurements and characteristics. Below is a detailed description of each column:

| Id | Pregnancies | Glucose | BloodPressure |
|---|---|---|---|
| Unique identifier | Number of pregnancies. | Plasma glucose concentration (2-hour test). | Diastolic blood press (mm Hg). |

2. Wrote ipynb file with general exploration of data and our model, uploaded alongside this pdf document.

Ian Paulino 08/28/2023
LISUM24



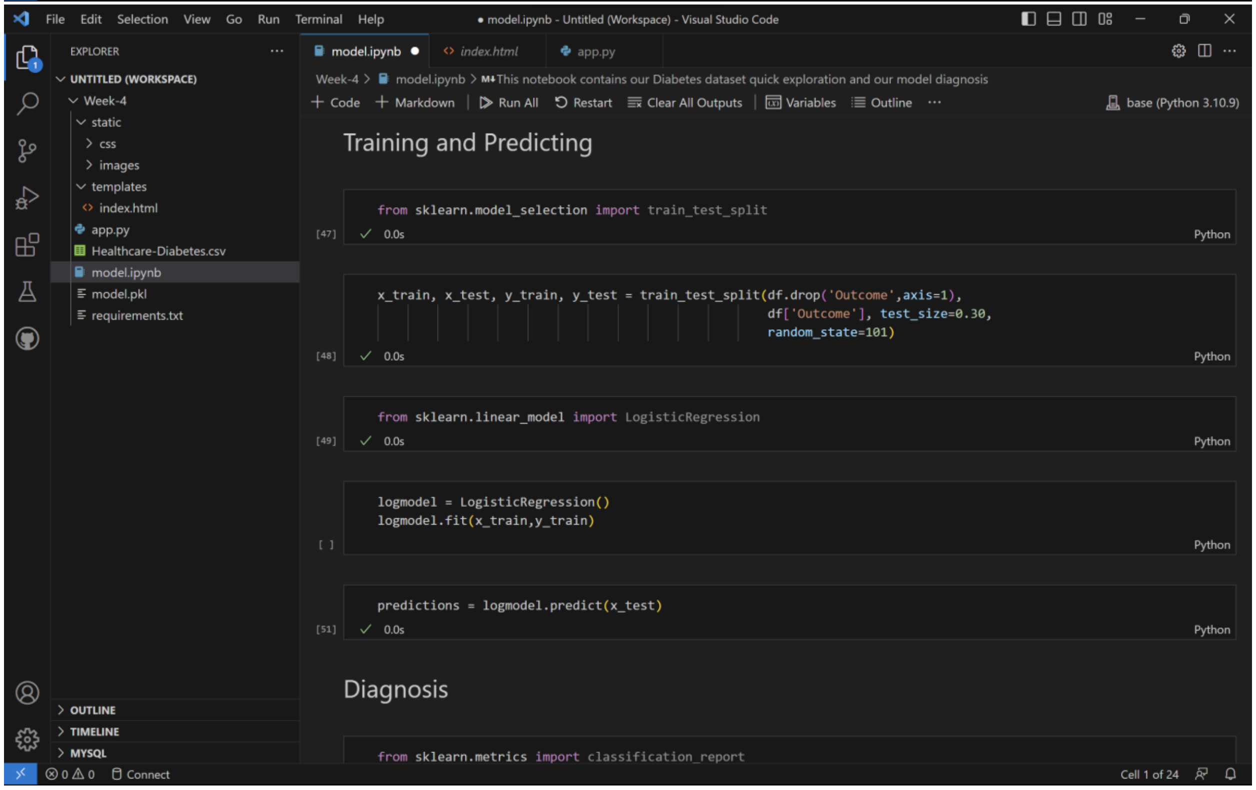
This notebook contains our Diabetes dataset quick exploration and our model diagnosis

The goal is to deploy a Logistic Regression model that can predict whether or not a person would have Diabetes given certain medical parameters, such as amount of pregnancies, level of glucose, blood pressure, skin thickness, insulin levels, BMI and age of the person.

```
[28]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
[17]
df = pd.read_csv('Healthcare-Diabetes.csv', index_col = 'Id')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2768 entries, 1 to 2768
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               2768 non-null   int64
 1   Glucose                   2768 non-null   int64
 2   BloodPressure             2768 non-null   int64
 3   SkinThickness             2768 non-null   int64
 4   Insulin                   2768 non-null   int64
```



Training and Predicting

```
[47]
from sklearn.model_selection import train_test_split
```

```
[48]
x_train, x_test, y_train, y_test = train_test_split(df.drop('Outcome',axis=1),
                                                    df['Outcome'], test_size=0.30,
                                                    random_state=101)
```

```
[49]
from sklearn.linear_model import LogisticRegression
```

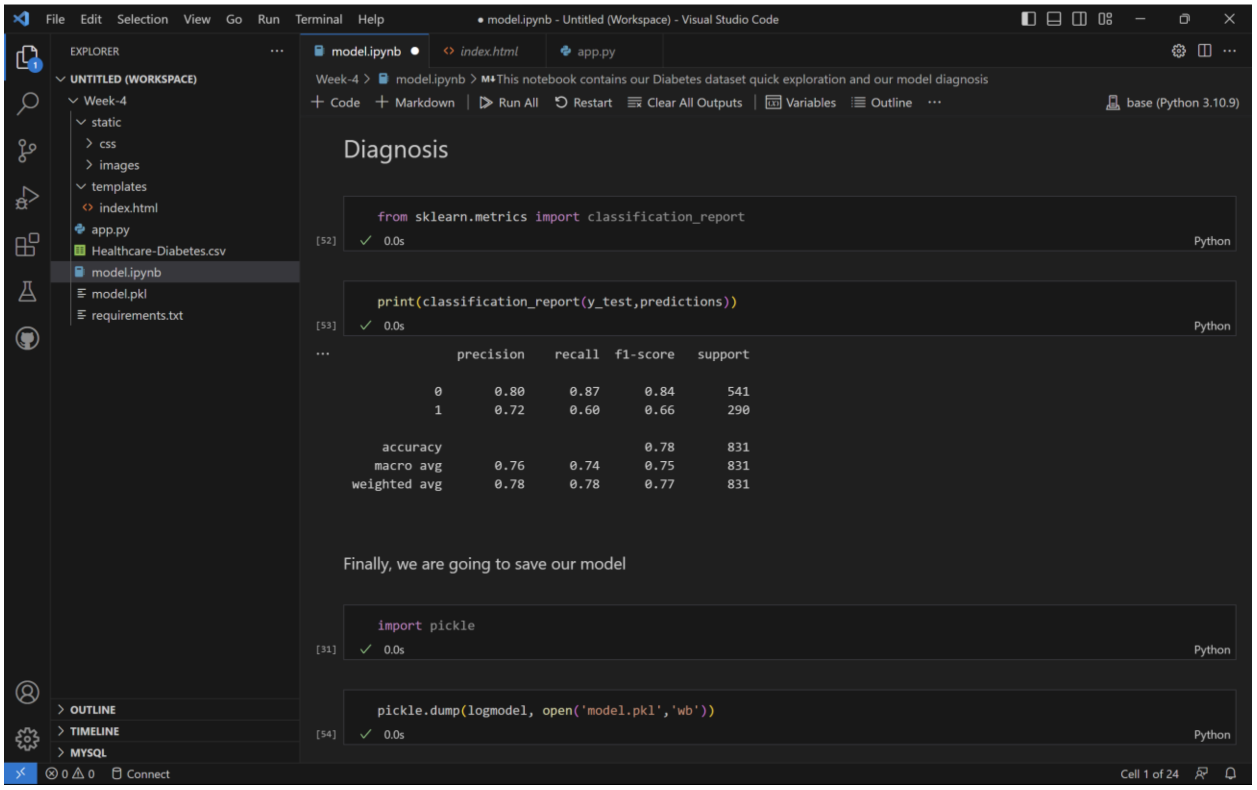
```
[ ]
logmodel = LogisticRegression()
logmodel.fit(x_train,y_train)
```

```
[51]
predictions = logmodel.predict(x_test)
```

Diagnosis

```
from sklearn.metrics import classification_report
```

Ian Paulino 08/28/2023
LISUM24



## Diagnosis

```
[52] from sklearn.metrics import classification_report
```

```
[53] print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.80      0.87      0.84       541
           1       0.72      0.60      0.66       290

    accuracy                           0.78       831
   macro avg       0.76      0.74      0.75       831
weighted avg       0.78      0.78      0.77       831
```

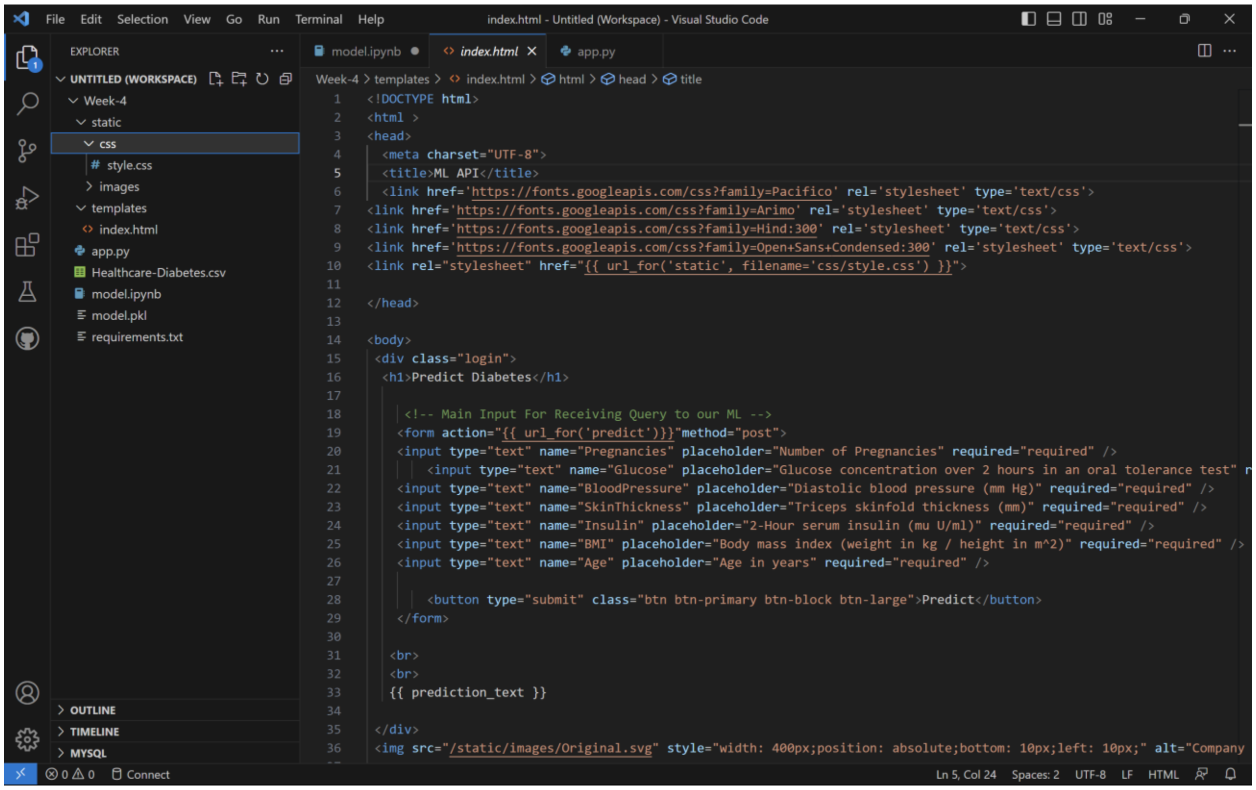
Finally, we are going to save our model

```
[31] import pickle
```

```
[54] pickle.dump(logmodel, open('model.pkl','wb'))
```

3. As we can see from step 2, the model was saved as model.pkl, uploaded alongside this file as well. The next step would be writing our html and css files. Notice the website will take input for each of the variables in our dataset that allows the model to make its

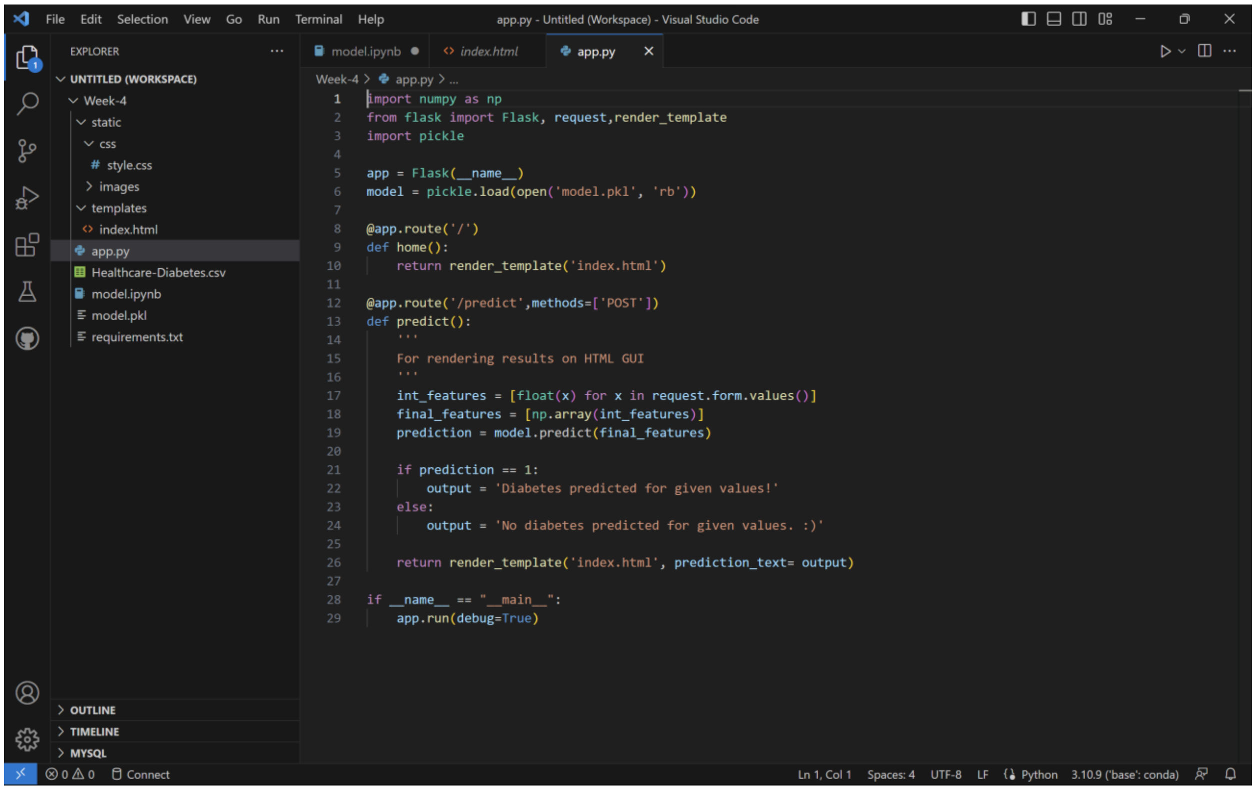
predictions.



```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
  <h1>Predict Diabetes</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
    <input type="text" name="Pregnancies" placeholder="Number of Pregnancies" required="required" />
        <input type="text" name="Glucose" placeholder="Glucose concentration over 2 hours in an oral tolerance test" r
    <input type="text" name="BloodPressure" placeholder="Diastolic blood pressure (mm Hg)" required="required" />
    <input type="text" name="SkinThickness" placeholder="Triceps skinfold thickness (mm)" required="required" />
    <input type="text" name="Insulin" placeholder="2-Hour serum insulin (mu U/ml)" required="required" />
    <input type="text" name="BMI" placeholder="Body mass index (weight in kg / height in m^2)" required="required" />
    <input type="text" name="Age" placeholder="Age in years" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>
 <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company
```

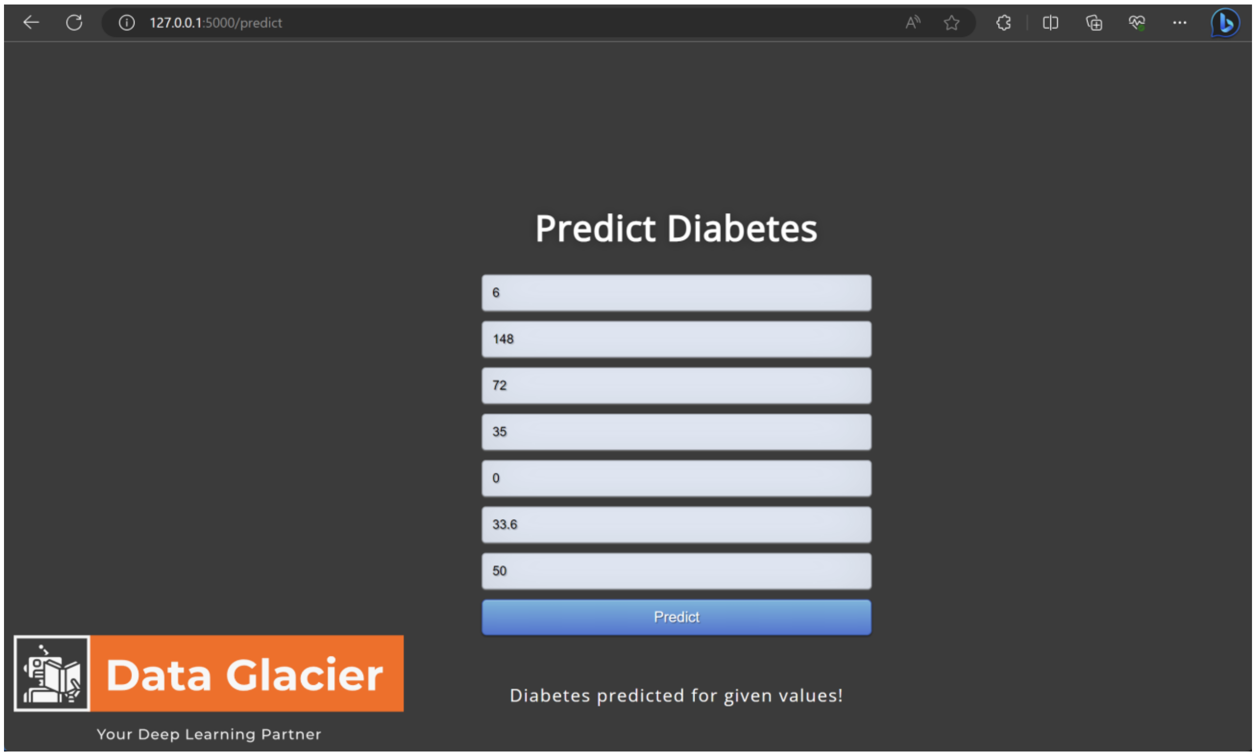
Ian Paulino 08/28/2023
LISUM24

4. Finally, we write one last python file using Flask, which will allow us to deploy our model online.



```python
import numpy as np
from flask import Flask, request,render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    if prediction == 1:
        output = 'Diabetes predicted for given values!'
    else:
        output = 'No diabetes predicted for given values. :)'

    return render_template('index.html', prediction_text= output)

if __name__ == "__main__":
    app.run(debug=True)
```

5. This is the result after running it locally, entering the website and inputting values.



Diabetes predicted for given values!