

## Logistic Regression Model to predict Diabetes deployment steps

1. Downloaded Diabetes dataset from [Kaggle](https://www.kaggle.com/datasets/ananandinil/hcare-diabetes).

The screenshot shows the Kaggle interface for the 'Healthcare Diabetes Dataset'. The left sidebar contains navigation links: Home, Competitions, Datasets (selected), Models, Code, Discussions, Learn, More, Your Work, and Viewed. The main content area displays the dataset details for 'Healthcare-Diabetes.csv' (98.57 kB). The 'Data Card' tab is active, showing a description of the file and a table of columns with their respective data types and histograms. The 'Data Explorer' panel on the right shows the dataset name and version.

**Healthcare Diabetes Dataset**

Data Card Code (2) Discussion (1)

Download (27 kB)

**Healthcare-Diabetes.csv** (98.57 kB)

Detail Compact Column 10 of 10 columns

**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). |

**Data Explorer**

Version 1 (98.57 kB)

Healthcare-Diabetes.csv

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

The screenshot shows a Visual Studio Code window with a Jupyter notebook titled 'model.ipynb'. The notebook is open to the 'Code' view. The Explorer sidebar on the left shows the project structure, including a 'Week-4' folder with subfolders 'static' and 'templates', and files 'app.py', 'Healthcare-Diabetes.csv', 'model.ipynb', 'model.pkl', and 'requirements.txt'. The notebook content includes a title 'This notebook contains our Diabetes dataset quick exploration and our model diagnosis', a description of the goal to deploy a Logistic Regression model, and two code cells. The first cell imports pandas, numpy, matplotlib, and seaborn. The second cell reads the 'Healthcare-Diabetes.csv' file and displays its information, showing 2768 entries and 9 columns: Pregnancies, Glucose, BloodPressure, SkinThickness, and Insulin.

File Edit Selection View Go Run Terminal Help • model.ipynb - Untitled (Workspace) - Visual Studio Code

EXPLORER

UNTITLED (WORKSPACE)

Week-4

- static
  - css
    - style.css
  - images
- templates
  - index.html
- app.py
- Healthcare-Diabetes.csv
- model.ipynb
- model.pkl
- requirements.txt

OUTLINE

TIMELINE

MYSQL

model.ipynb

app.py

Healthcare-Diabetes.csv

model.pkl

requirements.txt

Cell 10 of 24

base (Python 3.10.9)

Week-4 > model.ipynb > This notebook contains our Diabetes dataset quick exploration and our model diagnosis

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs | Variables Outline ...

## 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.

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

[28] ✓ 0.5s Python

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

[17] ✓ 0.0s Python

```
<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

```
from sklearn.model_selection import train_test_split
```

[47] ✓ 0.0s Python

```
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)
```

[48] ✓ 0.0s Python

```
from sklearn.linear_model import LogisticRegression
```

[49] ✓ 0.0s Python

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

[ ] Python

```
predictions = logmodel.predict(x_test)
```

[51] ✓ 0.0s Python

## Diagnosis

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

Cell 10 of 24

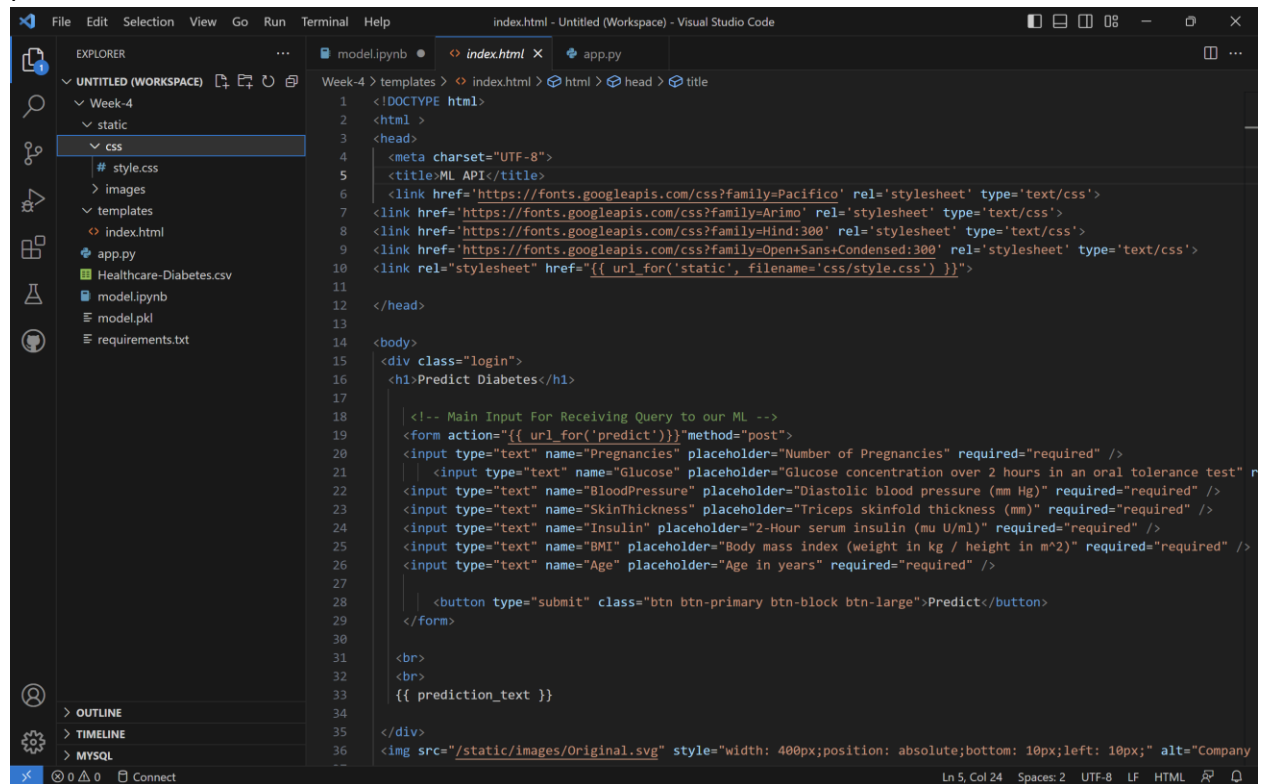
The screenshot shows a Jupyter Notebook titled "model.ipynb" in Visual Studio Code. The notebook is in the "Code" view and contains two cells. The first cell, labeled [52], imports the `classification_report` function from `sklearn.metrics`. The second cell, labeled [53], prints the classification report for the test set. The output of the second cell is a table showing the model's performance metrics.

|              | 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     |

Below the table, the text "Finally, we are going to save our model" is displayed. The third cell, labeled [31], imports the `pickle` module. The fourth cell, labeled [54], saves the model to a file named `model.pkl` using `pickle.dump`.

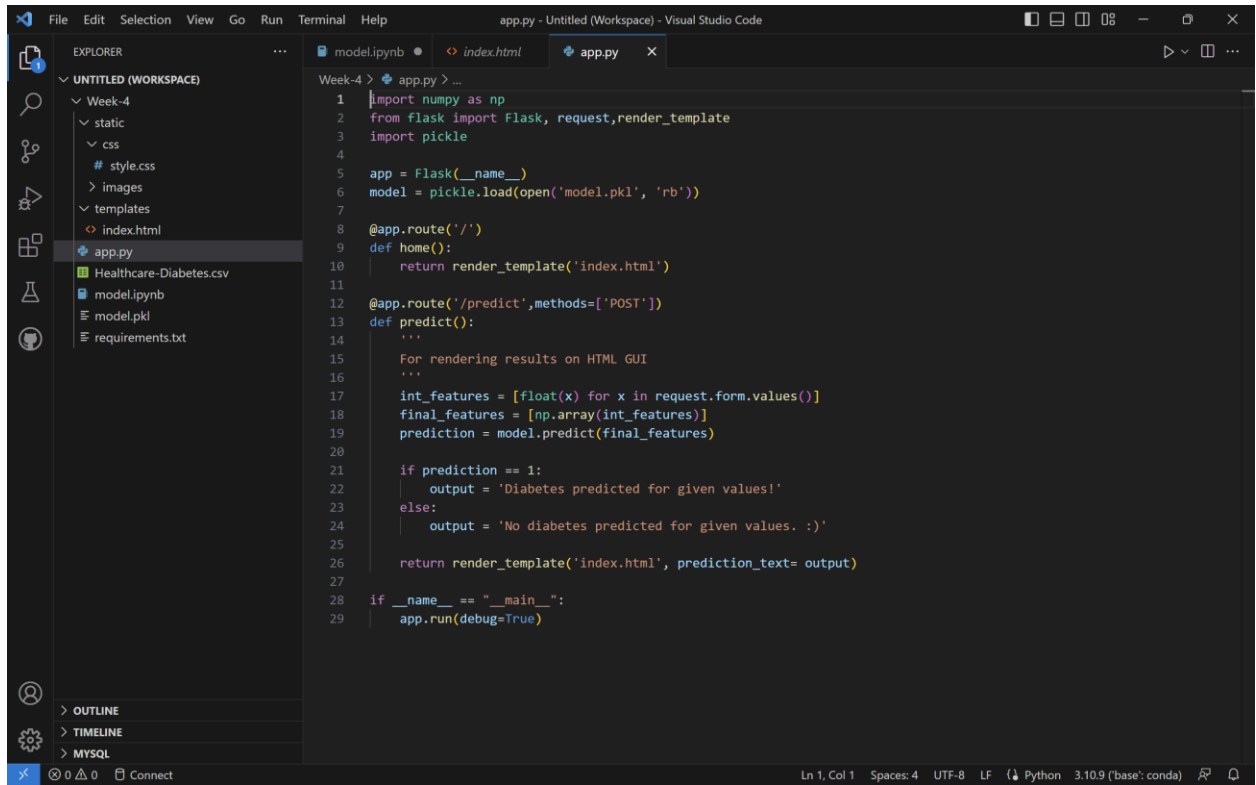
3. As we can see from step 2, the model was saved as `model.pkl`, uploaded alongside this file as well. The next step would be writing our html and css files. Notice the website will take input for each of the variables in our dataset that allows the model to make its

predictions.



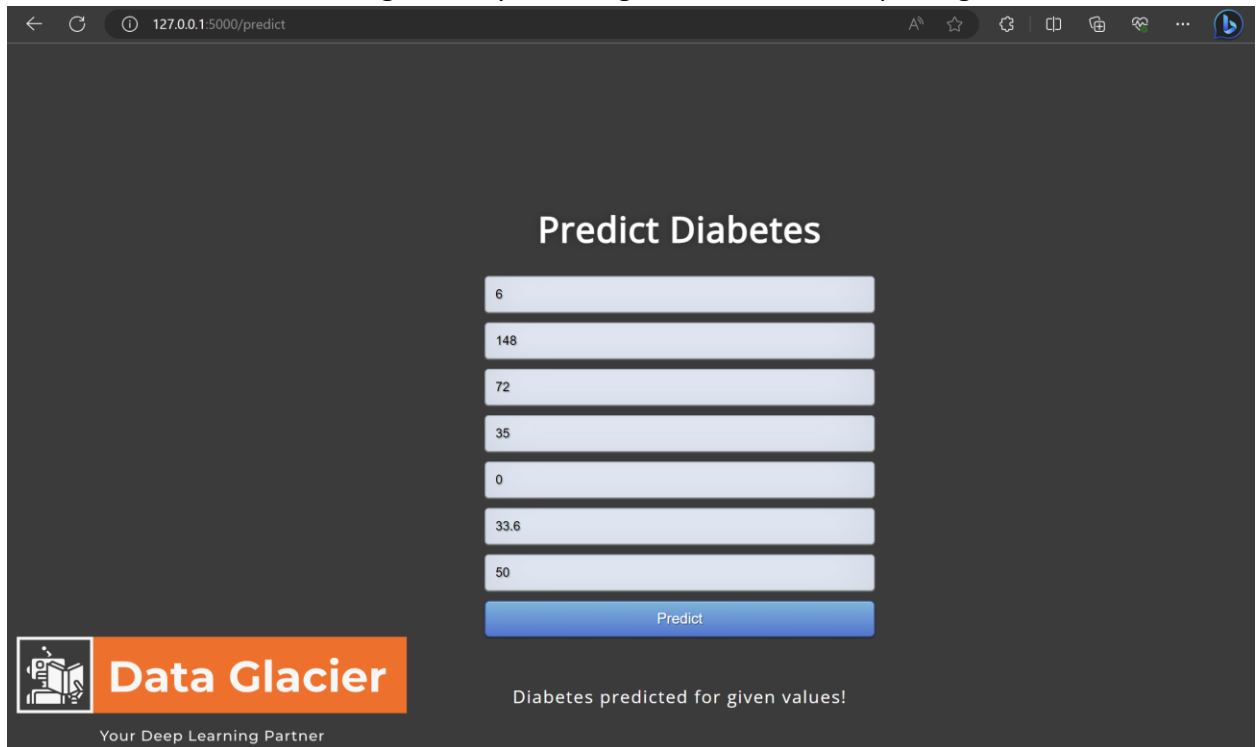
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>ML API</title>
6   <link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet" type="text/css">
7   <link href="https://fonts.googleapis.com/css?family=Arimo" rel="stylesheet" type="text/css">
8   <link href="https://fonts.googleapis.com/css?family=Hind:300" rel="stylesheet" type="text/css">
9   <link href="https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300" rel="stylesheet" type="text/css">
10  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
11 </head>
12
13 <body>
14   <div class="login">
15     <h1>Predict Diabetes</h1>
16
17     <!-- Main Input For Receiving Query to our ML -->
18     <form action="{{ url_for('predict') }}" method="post">
19       <input type="text" name="Pregnancies" placeholder="Number of Pregnancies" required="required" />
20       <input type="text" name="Glucose" placeholder="Glucose concentration over 2 hours in an oral tolerance test" required="required" />
21       <input type="text" name="BloodPressure" placeholder="Diastolic blood pressure (mm Hg)" required="required" />
22       <input type="text" name="SkinThickness" placeholder="Triceps skinfold thickness (mm)" required="required" />
23       <input type="text" name="Insulin" placeholder="2-Hour serum insulin (mu U/ml)" required="required" />
24       <input type="text" name="BMI" placeholder="Body mass index (weight in kg / height in m^2)" required="required" />
25       <input type="text" name="Age" placeholder="Age in years" required="required" />
26
27       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
28     </form>
29
30     <br>
31     <br>
32     {{ prediction_text }}
33
34   </div>
35
36   <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company
```

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



```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14     """
15     For rendering results on HTML GUI
16     """
17     int_features = [float(x) for x in request.form.values()]
18     final_features = [np.array(int_features)]
19     prediction = model.predict(final_features)
20
21     if prediction == 1:
22         output = 'Diabetes predicted for given values!'
23     else:
24         output = 'No diabetes predicted for given values. :)'
25
26     return render_template('index.html', prediction_text= output)
27
28 if __name__ == "__main__":
29     app.run(debug=True)
```

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




127.0.0.1:5000/predict

## Predict Diabetes

6  
148  
72  
35  
0  
33.6  
50

Predict

 **Data Glacier**  
Your Deep Learning Partner

Diabetes predicted for given values!