Data Intake Report

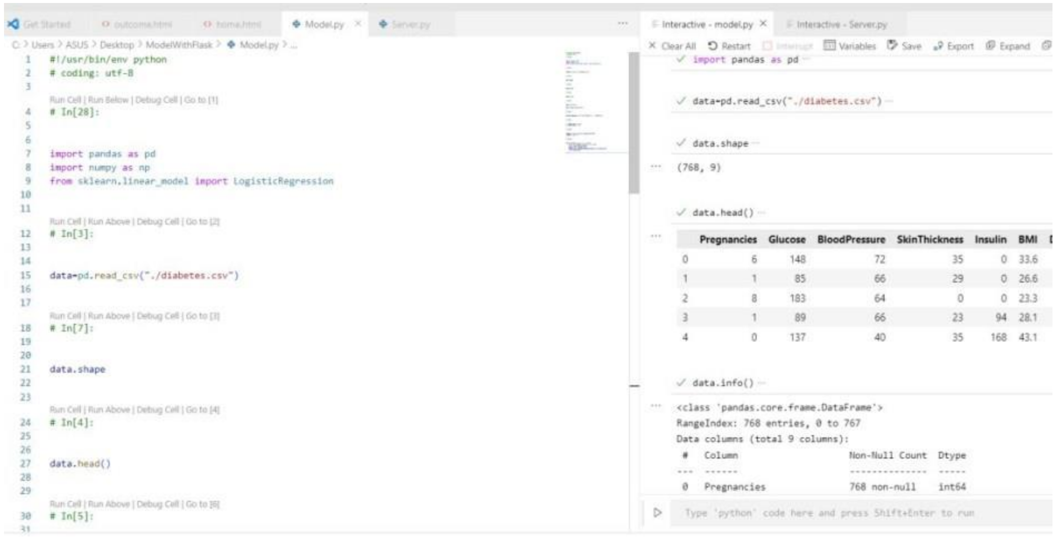
Name: Ana Lilliam Recio Garcia
Report date: 09/28/2022
Internship Batch: LISUM25
Version:<1.0>
Data intake by: Ana Lilliam Recio Garcia
Data intake reviewer:
Data storage location: https://github.com/AnaRecio/DeploymentFlask

Tabular data details:

Total number of observations	768
Total number of files	1
Total number of features	9
Base format of the file	.csv
Size of the data	24KB

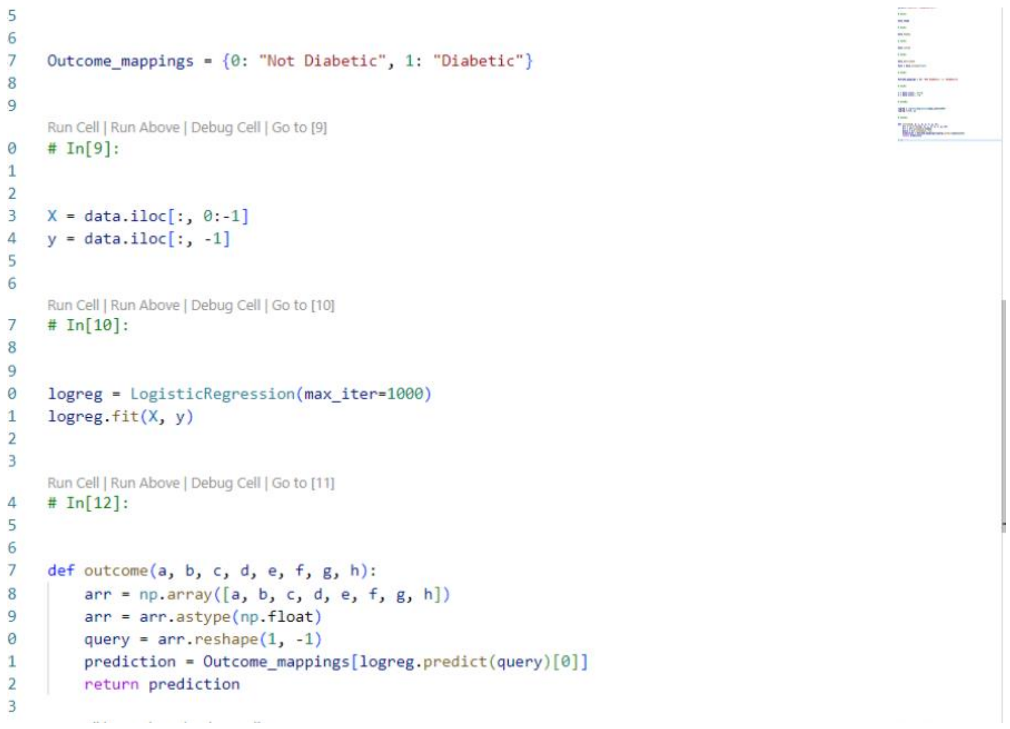
Proposed Approach:

I have created from the file diabetes.csv a model using Logistic Regression, to evaluate the dependent variable which is Outcome (last column) in the dataset, from the independent variables which are the other 7 variables.



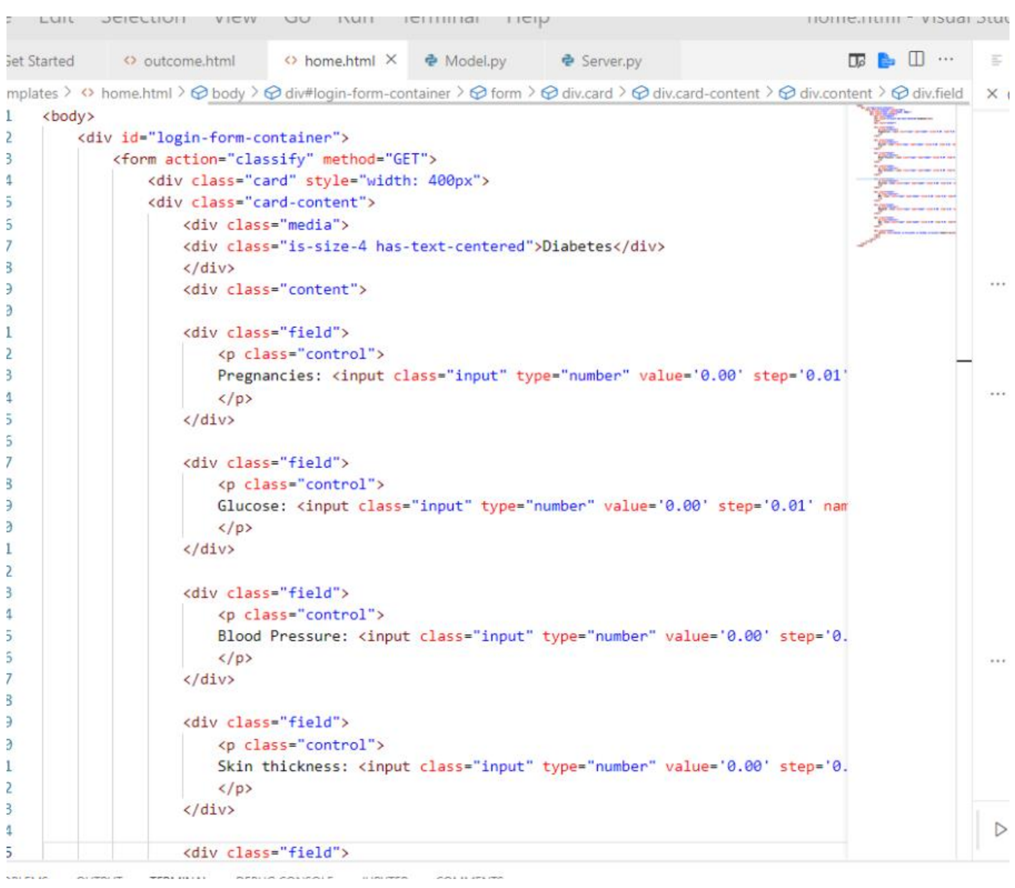
```python
#!/usr/bin/env python
# coding: utf-8

# In[28]:


import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression


# In[3]:


data=pd.read_csv("./diabetes.csv")


# In[7]:


data.shape


# In[4]:


data.head()


# In[5]:
```

```
import pandas as pd

data=pd.read_csv("./diabetes.csv")

data.shape
(768, 9)

data.head()
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

data.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
```



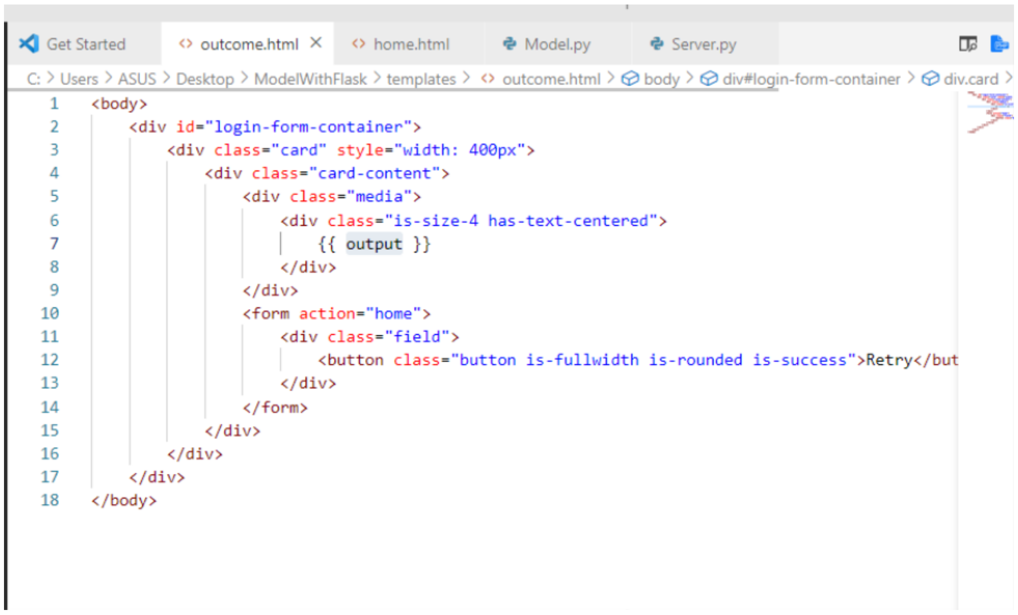
```
Outcome_mappings = {0: "Not Diabetic", 1: "Diabetic"}


Run Cell | Run Above | Debug Cell | Go to [9]
# In[9]:


X = data.iloc[:, 0:-1]
y = data.iloc[:, -1]


Run Cell | Run Above | Debug Cell | Go to [10]
# In[10]:


logreg = LogisticRegression(max_iter=1000)
logreg.fit(X, y)


Run Cell | Run Above | Debug Cell | Go to [11]
# In[12]:


def outcome(a, b, c, d, e, f, g, h):
    arr = np.array([a, b, c, d, e, f, g, h])
    arr = arr.astype(np.float)
    query = arr.reshape(1, -1)
    prediction = Outcome_mappings[logreg.predict(query)[0]]
    return prediction

```

I've created a simple form in HTLM so the user can input on the variables and have an outcome



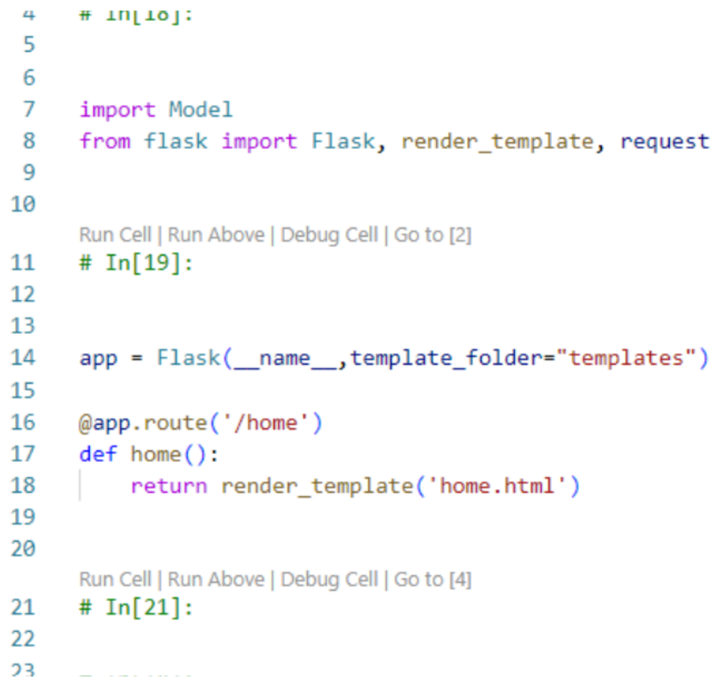
Get Started
outcome.html
home.html
Model.py
Server.py
mplates > home.html > body > div#login-form-container > form > div.card > div.card-content > div.content > div.field

```
<body>
    <div id="login-form-container">
        <form action="classify" method="GET">
            <div class="card" style="width: 400px">
            <div class="card-content">
                <div class="media">
                <div class="is-size-4 has-text-centered">Diabetes</div>
                </div>
                <div class="content">

                <div class="field">
                    <p class="control">
                    Pregnancies: <input class="input" type="number" value='0.00' step='0.01'
                    </p>
                </div>

                <div class="field">
                    <p class="control">
                    Glucose: <input class="input" type="number" value='0.00' step='0.01' nam
                    </p>
                </div>

                <div class="field">
                    <p class="control">
                    Blood Pressure: <input class="input" type="number" value='0.00' step='0.
                    </p>
                </div>

                <div class="field">
                    <p class="control">
                    Skin thickness: <input class="input" type="number" value='0.00' step='0.
                    </p>
                </div>

                <div class="field">
```

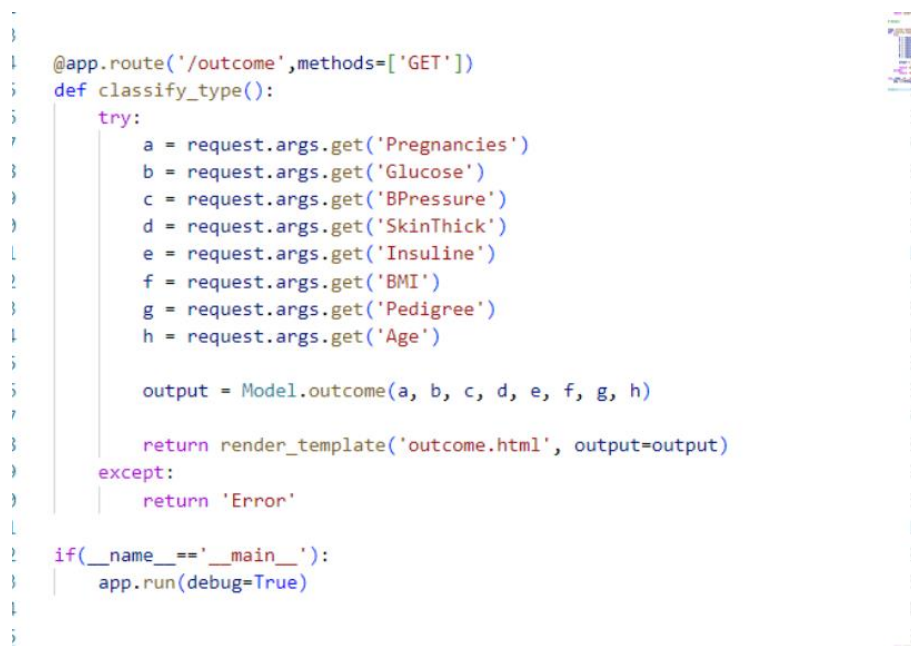

Get Started
outcome.html
home.html
Model.py
Server.py
C: > Users > ASUS > Desktop > ModelWithFlask > templates > outcome.html > body > div#login-form-container > div.card >

```
<body>
    <div id="login-form-container">
        <div class="card" style="width: 400px">
            <div class="card-content">
                <div class="media">
                    <div class="is-size-4 has-text-centered">
                        {{ output }}
                    </div>
                </div>
                <form action="home">
                    <div class="field">
                        <button class="button is-fullwidth is-rounded is-success">Retry</but
                    </div>
                </form>
            </div>
        </div>
    </div>
</body>
```

Finally, I’ve imported Flask to use as server of the model and made two routes, one to home.html and the other to outcome.html

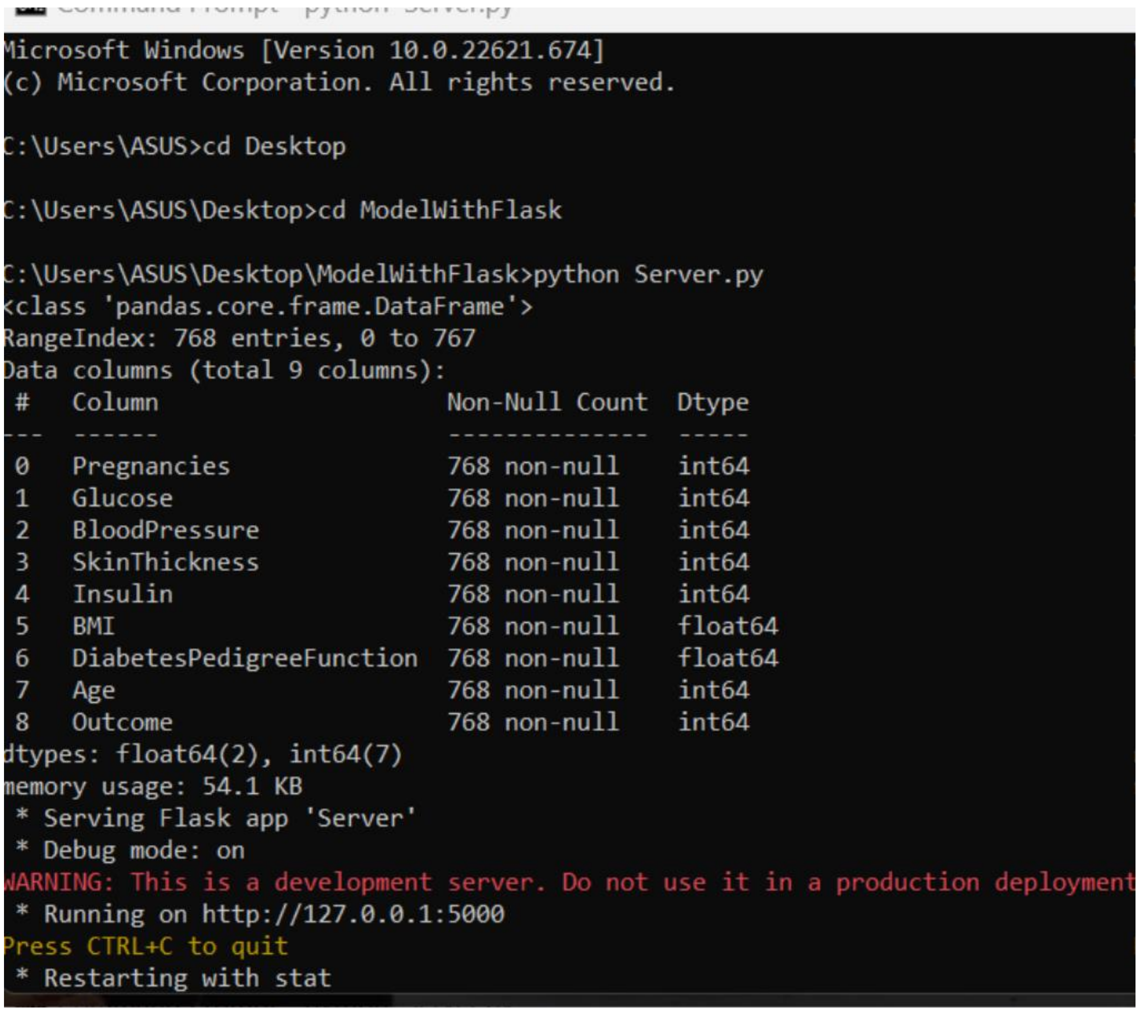

Run Cell | Run Above | Debug Cell | Go to [2]
Run Cell | Run Above | Debug Cell | Go to [4]

```
# In[18]:


import Model
from flask import Flask, render_template, request


# In[19]:


app = Flask(__name__,template_folder="templates")

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


# In[21]:

```

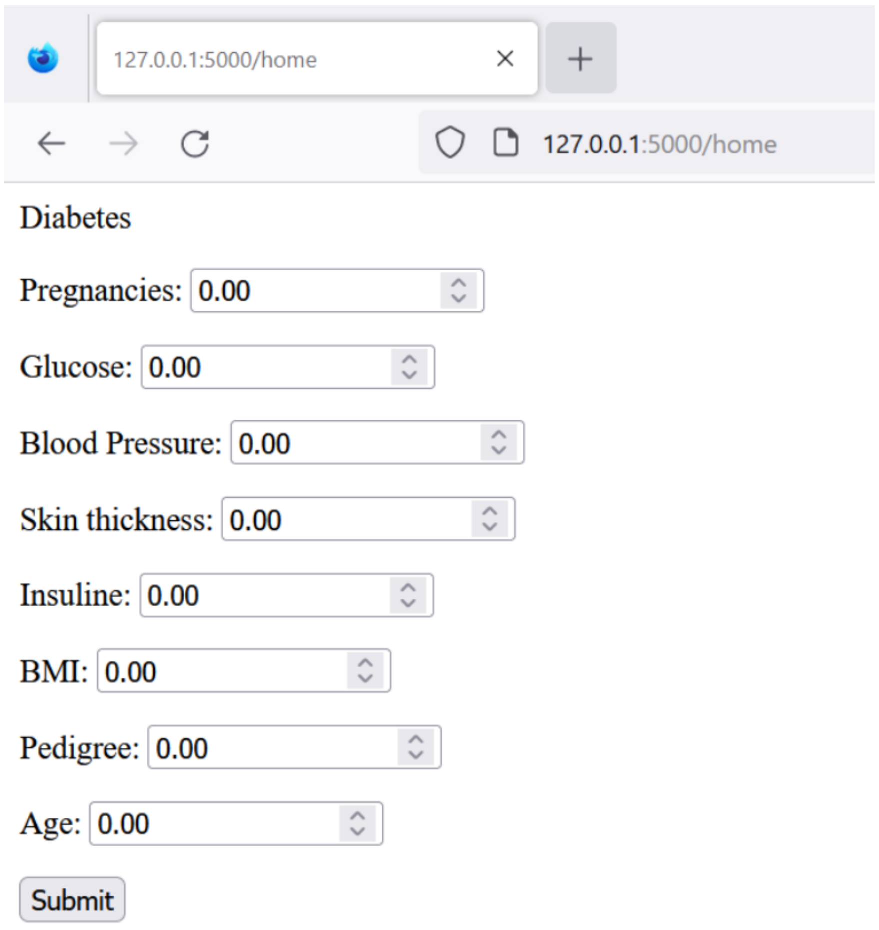


```
@app.route('/outcome',methods=['GET'])
def classify_type():
    try:
        a = request.args.get('Pregnancies')
        b = request.args.get('Glucose')
        c = request.args.get('BPressure')
        d = request.args.get('SkinThick')
        e = request.args.get('Insuline')
        f = request.args.get('BMI')
        g = request.args.get('Pedigree')
        h = request.args.get('Age')

        output = Model.outcome(a, b, c, d, e, f, g, h)

        return render_template('outcome.html', output=output)
    except:
        return 'Error'

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

I ran the server with the console and could see the port where it will run



```
Microsoft Windows [Version 10.0.22621.674]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>cd Desktop

C:\Users\ASUS\Desktop>cd ModelWithFlask

C:\Users\ASUS\Desktop\ModelWithFlask>python Server.py
<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
 * Serving Flask app 'Server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
```



Diabetes

Pregnancies: 0.00

Glucose: 0.00

Blood Pressure: 0.00

Skin thickness: 0.00

Insuline: 0.00

BMI: 0.00

Pedigree: 0.00

Age: 0.00

Submit