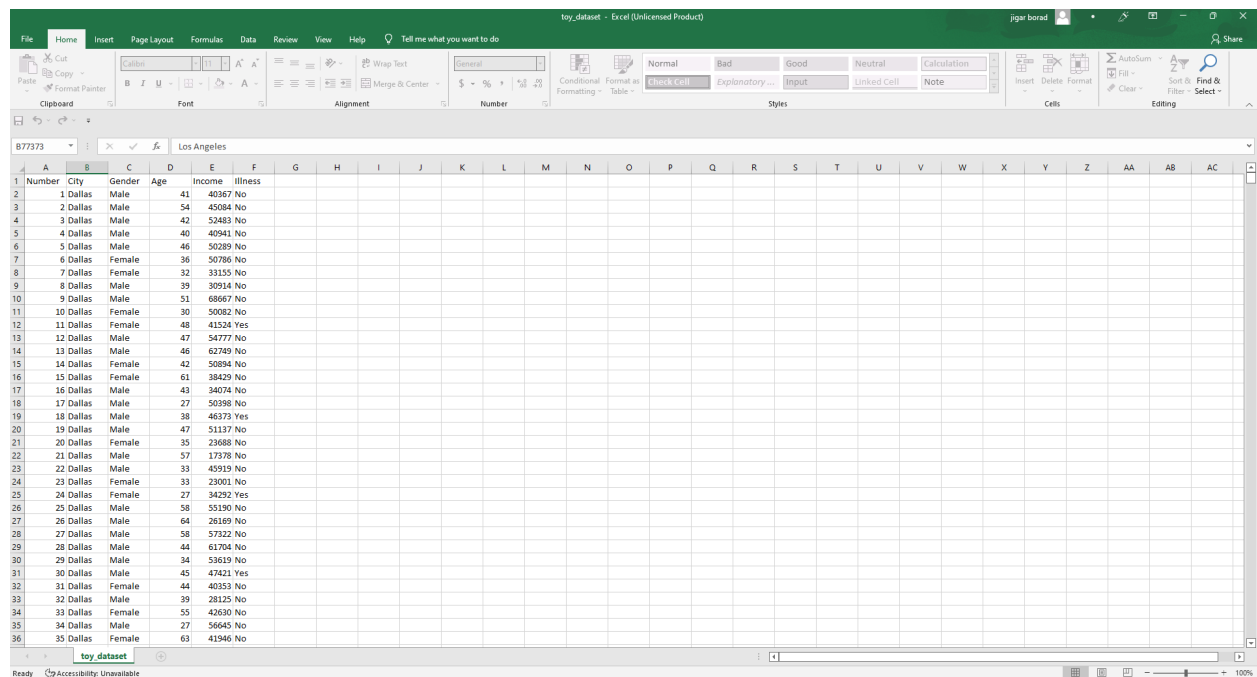


Name : Jigar Borad  
Batch Code: LISUM10  
Submission Date 25/06/2022  
Submitted To: Data Glacier

## Subject : Machine Learning Model Deployment On Flask

### Step 1 : Selecting Toy DataSet from kaggle



Number	City	Gender	Age	Income	Illness
1	Dallas	Male	41	40367	No
2	Dallas	Male	54	45084	No
3	Dallas	Male	42	53483	No
4	Dallas	Male	40	40941	No
5	Dallas	Male	46	50289	No
6	Dallas	Female	36	50786	No
7	Dallas	Female	32	33155	No
8	Dallas	Male	39	39514	No
9	Dallas	Male	51	68667	No
10	Dallas	Female	30	50082	No
11	Dallas	Female	48	41524	Yes
12	Dallas	Male	47	54777	No
13	Dallas	Male	46	62749	No
14	Dallas	Female	42	50894	No
15	Dallas	Female	61	38429	No
16	Dallas	Male	43	34074	No
17	Dallas	Male	27	50398	No
18	Dallas	Male	38	46373	Yes
19	Dallas	Male	47	51137	No
20	Dallas	Female	35	23688	No
21	Dallas	Male	57	17378	No
22	Dallas	Male	33	45919	No
23	Dallas	Female	33	23001	No
24	Dallas	Female	27	34292	Yes
25	Dallas	Male	58	53190	No
26	Dallas	Male	64	26169	No
27	Dallas	Male	58	57322	No
28	Dallas	Male	44	61704	No
29	Dallas	Male	34	53619	No
30	Dallas	Male	45	47421	Yes
31	Dallas	Female	44	40353	No
32	Dallas	Male	39	28125	No
33	Dallas	Female	55	42630	No
34	Dallas	Male	27	56645	No
35	Dallas	Female	63	41946	No

Toy Dataset

## Step 2: Model Building and Model Saving

(1)

The notebook interface shows the first two steps of the process. The file explorer on the left lists 'sample\_data' and 'toy\_dataset.csv'. The code cell [1] imports necessary libraries: numpy, pandas, matplotlib.pyplot, sklearn.pipeline, sklearn.compose, sklearn.preprocessing, sklearn.ensemble, and pickle. Cell [2] loads the 'toy\_dataset.csv' file into a pandas DataFrame. Cell [3] prints a detailed review of the dataset, including null values, shape, and a preview of the first five rows.

```
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
import pickle

Importing Dataset

[2] dataset = pd.read_csv('toy_dataset.csv')

[3] print('Data Review')
print('-----')
print(f'Total Null values')
print(dataset.isnull().sum())
print('-----')
print(f'Shape of Dataset')
print(dataset.shape)
print('-----')
dataset.head()

Data Review
-----
Total Null values
number      0
city        0
gender      0
age         0
income      0
illness     0
dtype: int64
-----
Shape of Dataset
(10000, 6)
-----
   number  city  gender  age  income  illness
0        1   Dallas  Male   41  40367.0     No
1        2   Dallas  Male   54  45064.0     No
2        3   Dallas  Male   42  52483.0     No
3        4   Dallas  Male   40  40941.0     No
4        5   Dallas  Male   46  50289.0     No
```

(2)

The notebook continues with the model building and saving process. Cell [4] separates the features (X) and target variable (y). Cell [5] encodes the categorical variable 'illness' into integers using a LabelEncoder. Cell [6] splits the data into training and testing sets using train\_test\_split. Cell [7] creates a ColumnTransformer to handle the categorical encoding and trains a RandomForestClassifier. Cell [8] creates a pipeline combining the transformer and the classifier. Cell [9] fits the pipeline to the training data and predicts on the test data. Cell [10] calculates and prints the accuracy score of the model. Cell [11] saves the trained pipeline as a pickle file named 'illnesstrainer.pkl'.

```
[4] X = dataset.iloc[:,1:-1].values
y = dataset.iloc[:,1].values

Encoding y values from categorical to integers

[5] le = LabelEncoder()
y = le.fit_transform(y)

Splitting data into Train and test

[6] from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)

encoding X_train from categorical to integer and training the model with random forest classifier

[7] ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [6,1])], remainder='passthrough')
classifier=RandomForestClassifier(n_estimators=100)

making pipeline for running onehotencoding and training

[8] pipeline = make_pipeline(ct, classifier)

predicting data to see accuracy

[9] pipeline.fit(X_train,y_train)
y_pred = pipeline.predict(X_test)

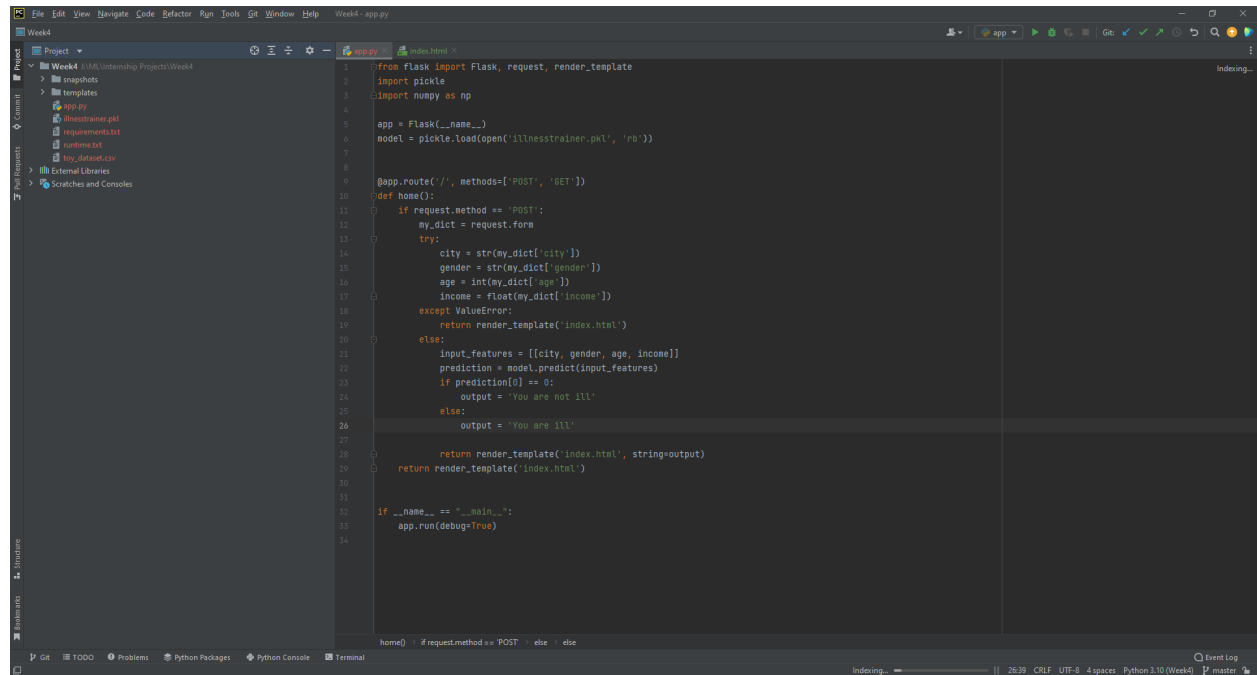
[10] from sklearn.metrics import accuracy_score
score=accuracy_score(y_test,y_pred)
print(f'accuracy is {("{:.2f}")}.format(score*100)} %')

accuracy is 86.91 %

saving model

[11] pickle_out = open('illnesstrainer.pkl',"wb")
pickle.dump(pipeline, pickle_out)
pickle_out.close()
```

### Step 3: Building Flask App



```
1 from flask import Flask, request, render_template
2 import pickle
3 import numpy as np
4
5 app = Flask(__name__)
6 model = pickle.load(open('illnesstrainer.pkl', 'rb'))
7
8
9 @app.route('/', methods=['POST', 'GET'])
10 def home():
11     if request.method == 'POST':
12         my_dict = request.form
13         try:
14             city = str(my_dict['city'])
15             gender = str(my_dict['gender'])
16             age = int(my_dict['age'])
17             income = float(my_dict['income'])
18         except ValueError:
19             return render_template('index.html')
20         else:
21             input_features = [[city, gender, age, income]]
22             prediction = model.predict(input_features)
23             if prediction[0] == 0:
24                 output = 'You are not ill'
25             else:
26                 output = 'You are ill'
27
28             return render_template('index.html', string=output)
29     return render_template('index.html')
30
31
32 if __name__ == '__main__':
33     app.run(debug=True)
34
```

### Step 4: Building HTML file

(1)

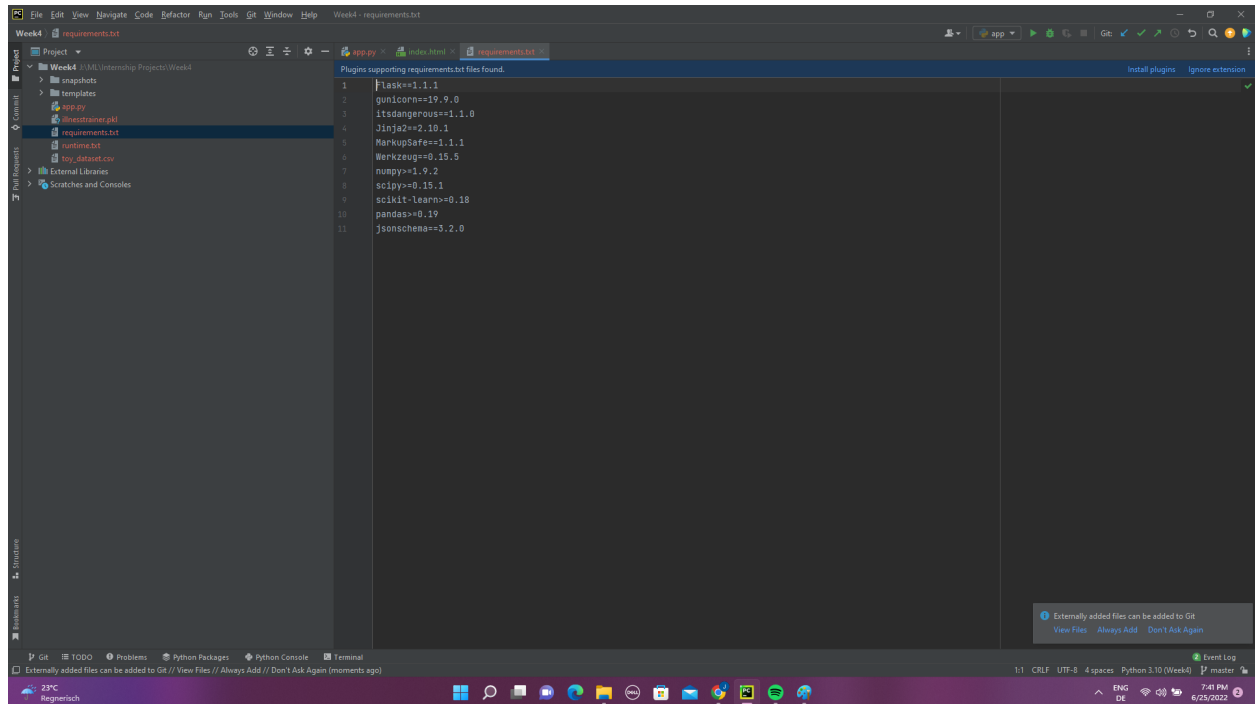
The screenshot shows a VS Code editor with the following components:

- File Explorer (Left):** Displays the project structure for 'Week4'. The 'index.html' file is selected under the 'templates' directory.
- Code Editor (Center):** Shows the content of 'index.html'. The code is a single HTML file that includes Bootstrap CSS and contains a form for an illness tracker. The form has four input fields: 'city', 'gender', 'age', and 'income', each with a label and a placeholder. A 'Predict' button is at the bottom. The form is styled with Bootstrap classes like 'container', 'text-center', 'alert-info', and 'form-group'.
- Output Console (Bottom):** Shows the command prompt output, indicating that the application is running successfully on port 5000. The output includes the Flask application's startup message and the URL to access the application.

(2)

![Screenshot of a VS Code editor showing an HTML file named index.html. The file contains a form with input fields for city, gender, age, and income, a predict button, and a feedback message area. The form is styled with Bootstrap classes. The code is as follows: <pre><!-- index.html -->
<!-- jQuery and Bootstrap Bundle -->
<script src=](https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js)

## Step 5: Creating Requirement.txt file for deploying Flask app

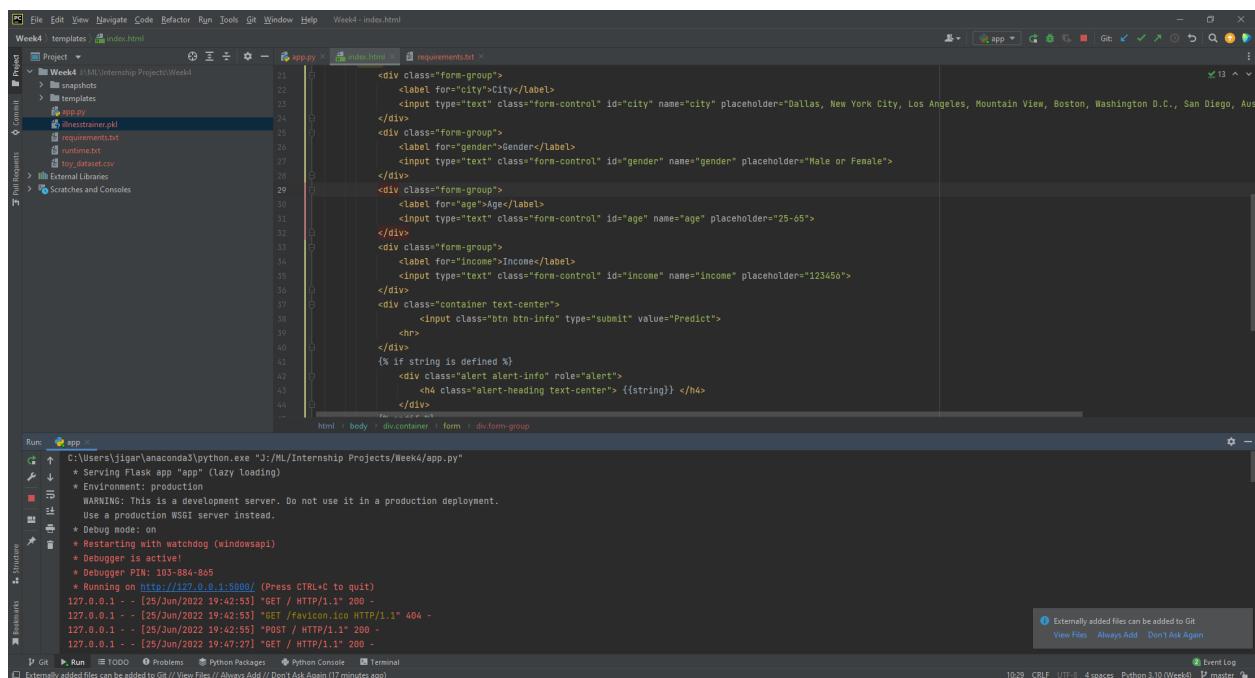


The screenshot shows the Visual Studio Code editor with the file `requirements.txt` open. The file contains the following dependencies:

```
1 flask==1.1.1
2 gunicorn==19.9.0
3 itsdangerous==1.1.0
4 jinja2==2.10.1
5 MarkupSafe==1.1.1
6 Werkzeug==0.15.5
7 numpy==1.9.2
8 scipy==0.15.1
9 scikit-learn==0.18
10 pandas==0.19
11 jssonschema==3.2.0
```

The interface also shows a sidebar with the project structure, including `Week4`, `requirements.txt`, and `requirements.txt`. The bottom status bar indicates the file encoding is UTF-8 and the line ending is CRLF.

## Step 6: Running Flask App



The screenshot shows the Visual Studio Code editor with the file `index.html` open. The file contains the following HTML code:

```
21 <div class="form-group">
22   <label for="city">City</label>
23   <input type="text" class="form-control" id="city" name="city" placeholder="Dallas, New York City, Los Angeles, Mountain View, Boston, Washington D.C., San Diego, Austin">
24 </div>
25 <div class="form-group">
26   <label for="gender">Gender</label>
27   <input type="text" class="form-control" id="gender" name="gender" placeholder="Male or Female">
28 </div>
29 <div class="form-group">
30   <label for="age">Age</label>
31   <input type="text" class="form-control" id="age" name="age" placeholder="25-65">
32 </div>
33 <div class="form-group">
34   <label for="income">Income</label>
35   <input type="text" class="form-control" id="income" name="income" placeholder="123456">
36 </div>
37 <div class="container text-center">
38   <input class="btn btn-info" type="submit" value="Predict">
39 </div>
40 </div>
41 {% if string is defined %}
42 <div class="alert alert-info" role="alert">
43   <h4 class="alert-heading text-center"> {{string}} </h4>
44 </div>
45 </div>
```

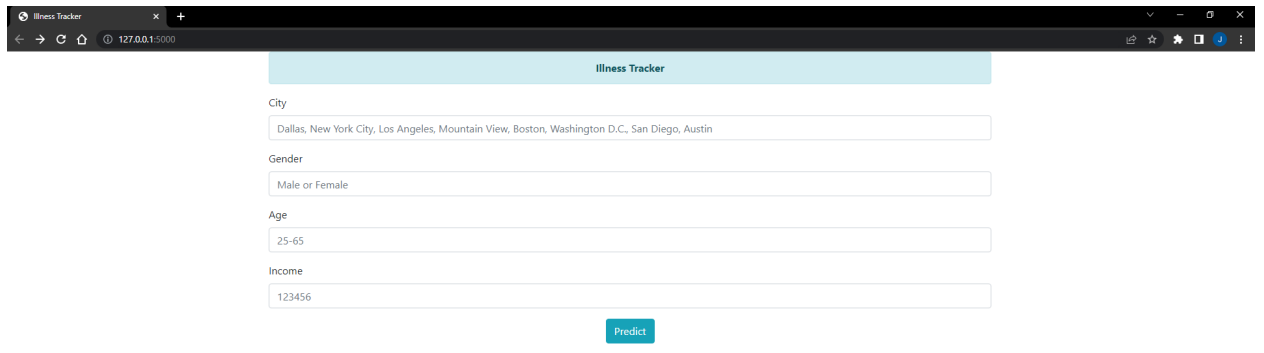
The terminal output shows the following messages:

```
Run app
C:\Users\jigar\anaconda3\python.exe "J:\ML\Internship Projects\Week4\app.py"
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 103-884-805
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [25/Jun/2022 19:42:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [25/Jun/2022 19:42:53] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [25/Jun/2022 19:42:55] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [25/Jun/2022 19:47:27] "GET / HTTP/1.1" 200 -
```

The interface also shows a sidebar with the project structure, including `Week4`, `requirements.txt`, and `requirements.txt`. The bottom status bar indicates the file encoding is UTF-8 and the line ending is CRLF.

## Step 7: Deployed Flask App

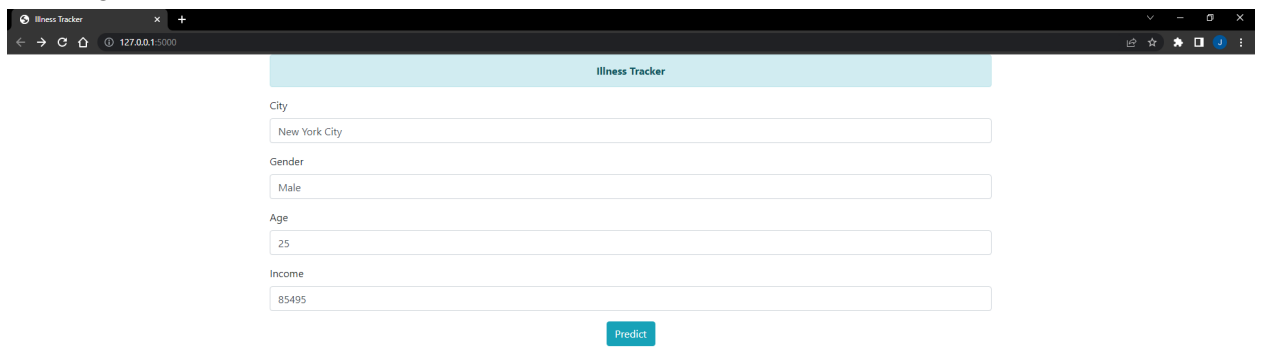
### (1) Home Page



A screenshot of a web browser displaying the 'Illness Tracker' application. The browser's address bar shows the URL '127.0.0.1:5000'. The application has a light blue header with the title 'Illness Tracker'. Below the header, there are four input fields for user information: 'City' (with a dropdown menu showing 'Dallas, New York City, Los Angeles, Mountain View, Boston, Washington D.C., San Diego, Austin'), 'Gender' (with a dropdown menu showing 'Male or Female'), 'Age' (with a dropdown menu showing '25-65'), and 'Income' (with a text input showing '123456'). A teal 'Predict' button is located below the input fields. The background of the page is a light gray gradient.

---

### (2) Entering Data



A screenshot of the same 'Illness Tracker' application, but with data entered into the input fields. The 'City' dropdown is set to 'New York City', the 'Gender' dropdown is set to 'Male', the 'Age' dropdown is set to '25', and the 'Income' text input is set to '85495'. The 'Predict' button remains visible below the input fields. The browser's address bar still shows '127.0.0.1:5000'.

(3) Predicted Result

Illness Tracker

City

Dallas, New York City, Los Angeles, Mountain View, Boston, Washington D.C., San Diego, Austin

Gender

Male or Female

Age

25-65

Income

123456

Predict

You are not ill