

Name : Jigar Borad  
Batch Code: LISUM10  
Submission Date 04/07/2022  
Submitted To: Data Glacier

## Subject : Machine Learning Model Deployment On Heroku(api and web based)

### Step 1 : Selecting Toy DataSet from kaggle

The screenshot shows an Excel spreadsheet titled 'toy\_dataset' with the following data:

Number	City	Gender	Age	Income	Illness
1	Dallas	Male	41	40367	No
2	Dallas	Male	54	45084	No
3	Dallas	Male	42	53482	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	58894	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	55130	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)

```
IllnessTracker.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 29
+ Code + Text
Importing Libraries
[ ] 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.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
import pickle

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

[ ] 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  45084.0    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)

```
IllnessTracker.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 29
+ Code + Text
Separating X and y
[ ] X = dataset.iloc[:,1:-1].values
y = dataset.iloc[:, -1].values

Encoding y values from categorical to integers
[ ] le = LabelEncoder()
y = le.fit_transform(y)

Splitting dat into Train and test
[ ] 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
[ ] ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0,1])], remainder='passthrough')
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()

making pipeline for running onehotencoding and training
[ ] pipeline = make_pipeline(ct, classifier)

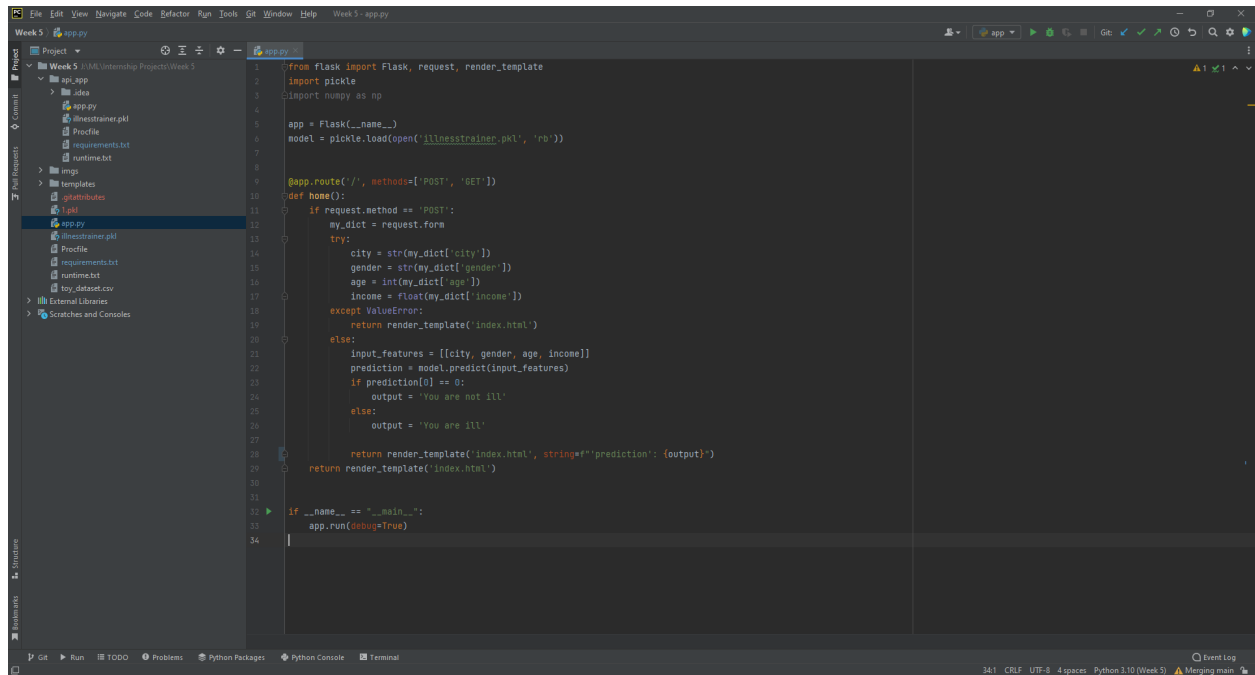
predicting data to see accuracy
[ ] pipeline.fit(X_train,y_train)
y_pred = pipeline.predict(X_test)

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

accuracy is 91.99 %

saving model
[ ] pickle_out = open('illnesstrainer.pkl','wb')
pickle.dump(pipeline, pickle_out, protocol=pickle.HIGHEST_PROTOCOL)
pickle_out.close()
```

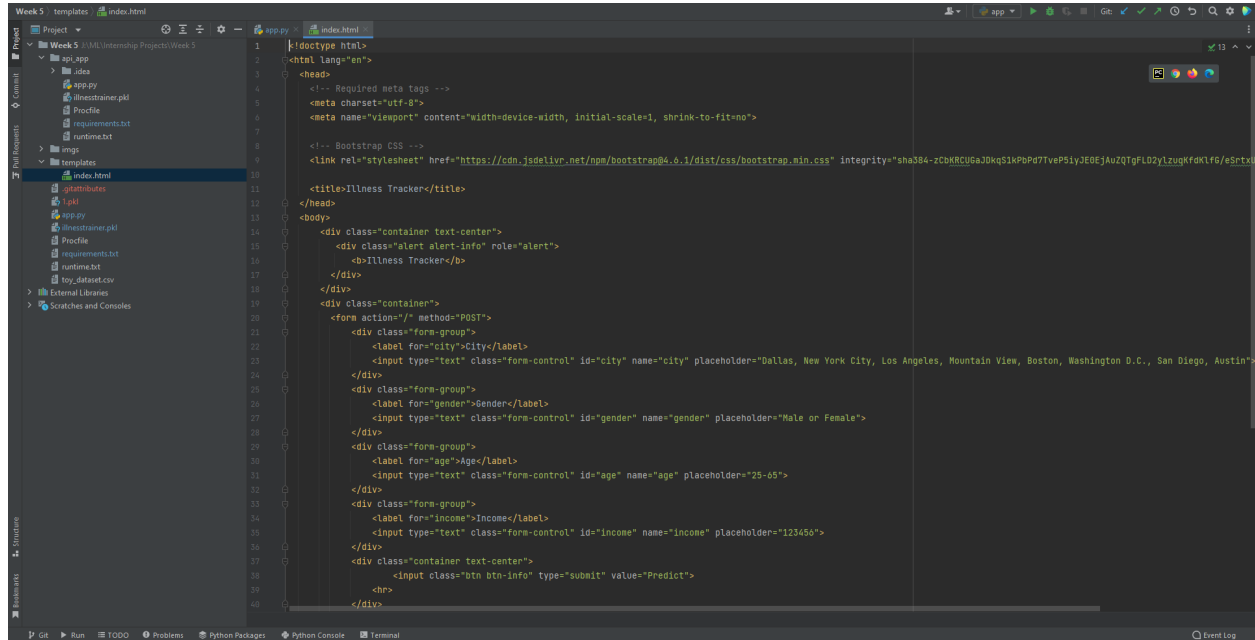
### Step 3: Building Flask App



```
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 @app.route('/', methods=['POST', 'GET'])
9 def home():
10     if request.method == 'POST':
11         my_dict = request.form
12         try:
13             city = str(my_dict['city'])
14             gender = str(my_dict['gender'])
15             age = int(my_dict['age'])
16             income = float(my_dict['income'])
17             except ValueError:
18                 return render_template('index.html')
19             else:
20                 input_features = [[city, gender, age, income]]
21                 prediction = model.predict(input_features)
22                 if prediction[0] == 0:
23                     output = 'You are not ill'
24                 else:
25                     output = 'You are ill'
26                 return render_template('index.html', stringf'prediction': {output})
27             return render_template('index.html')
28
29 if __name__ == '__main__':
30     app.run(debug=True)
```

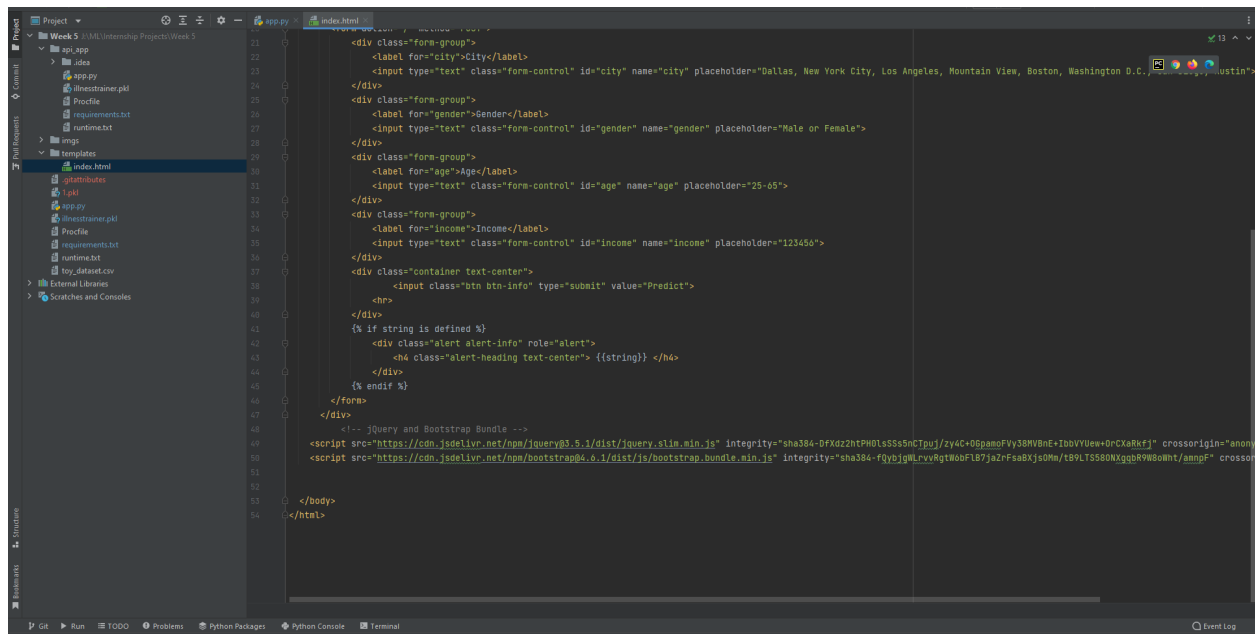
## Step 4: Building HTML file

(1)



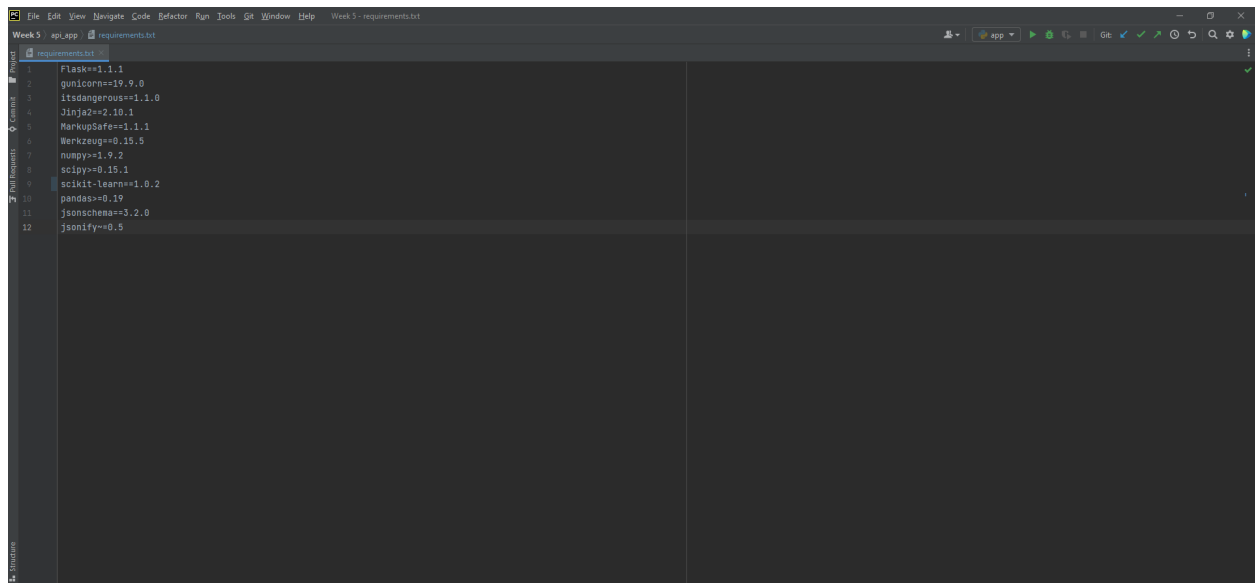
```
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <!-- Required meta tags -->
5 <meta charset="utf-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8 <!-- Bootstrap CSS -->
9 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css" integrity="sha384-zCBKJwUaZcKjp5GRu9Mc7ypPq77vFpSiYJ0EJAUQZTgFLDZyLuzgkf0K1f6/egrt" crossorigin="anonymous">
10
11 <title>Illness Tracker</title>
12 </head>
13 <body>
14 <div class="container text-center">
15 <div class="alert alert-info" role="alert">
16 <b>Illness Tracker</b>
17 </div>
18 </div>
19
20 <div class="container">
21 <form action="/" method="POST">
22 <div class="form-group">
23 <label for="city">City</label>
24 <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">
25 </div>
26 <div class="form-group">
27 <label for="gender">Gender</label>
28 <input type="text" class="form-control" id="gender" name="gender" placeholder="Male or Female">
29 </div>
30 <div class="form-group">
31 <label for="age">Age</label>
32 <input type="text" class="form-control" id="age" name="age" placeholder="25-65">
33 </div>
34 <div class="form-group">
35 <label for="income">Income</label>
36 <input type="text" class="form-control" id="income" name="income" placeholder="123456">
37 </div>
38 <div class="container text-center">
39 <input class="btn btn-info" type="submit" value="Predict">
40 </div>
41 </div>
42
43 </body>
44 </html>
```

(2)



```
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>
46 </div>
47
48 <!-- jQuery and Bootstrap Bundle -->
49 <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js" integrity="sha384-Df4zH16l6D5o1206184XCU2j/47iH24c7EDRhZGqOJ29721Z1f4tW3j0Z143344" crossorigin="anonymous"></script>
50 <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-fQVgZq3MEKjE6z6FdjiYWRcOpUk2lZyFehhcbfCun03J01yAGlf7deQSNc09Ulu/O" crossorigin="anonymous"></script>
51
52 </body>
53
54 </html>
```

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



The screenshot shows a code editor window titled "Week 3 - requirements.txt". The editor contains a file named "requirements.txt" with the following content:

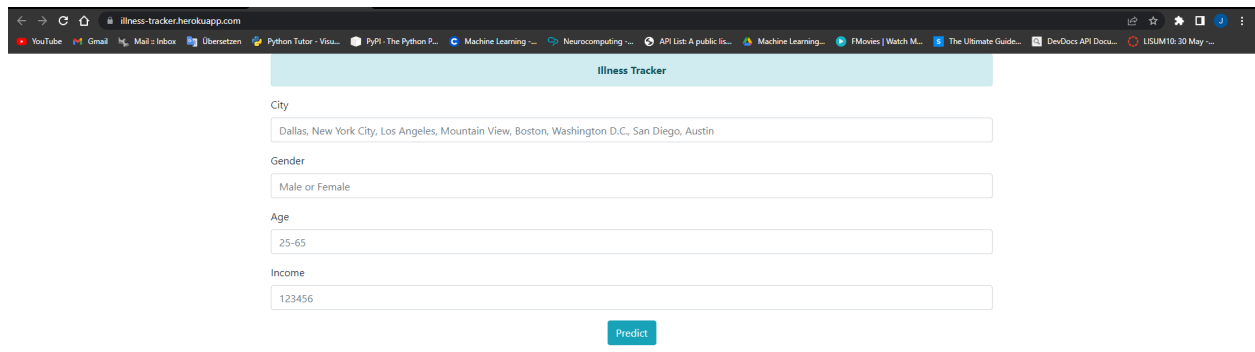
```
1 Flask==1.1.1
2 gunicorn==19.9.0
3 itsdangerous==1.1.0
4 Jinja2==2.10.1
5 MarkupSafe==1.1.1
6 Werkzeug==0.15.5
7 numpy==1.9.2
8 scipy==0.15.1
9 scikit-learn==1.0.2
10 pandas==0.19
11 jsonschema==3.2.0
12 jsonify==0.5
```

The editor interface includes a sidebar on the left with a "Project" view showing the file structure, and a top toolbar with various icons for running, debugging, and other development tasks.

## Step 6: Deploying Web app on Heroku

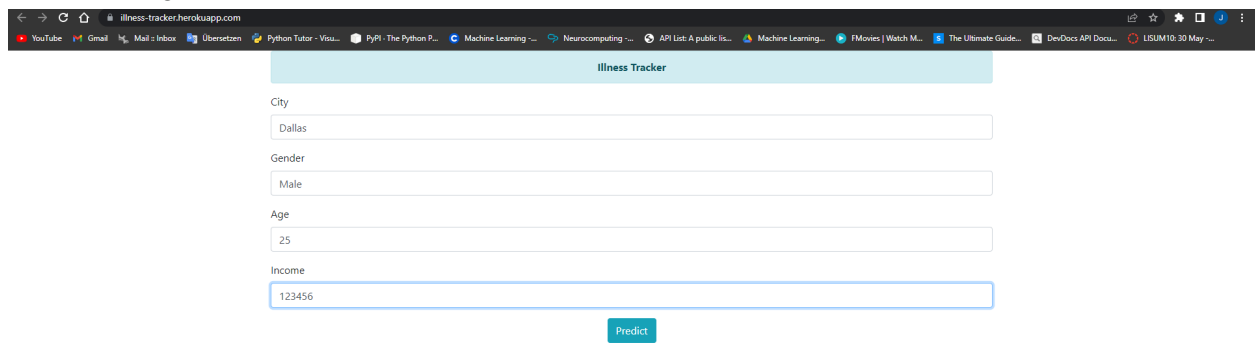
App link : <https://illness-tracker.herokuapp.com/>

(1) Web app Home Page:



The screenshot shows the web application's home page in a browser. The page has a light blue header with the title "Illness Tracker". Below the header, there are four input fields for user information: "City", "Gender", "Age", and "Income". The "City" field contains the text "Dallas, New York City, Los Angeles, Mountain View, Boston, Washington D.C., San Diego, Austin". The "Gender" field contains "Male or Female". The "Age" field contains "25-65". The "Income" field contains "123456". Below these fields is a teal "Predict" button. The browser's address bar shows the URL "illness-tracker.herokuapp.com".

(2) Filling the fields:



This screenshot shows the same web application as the previous one, but with the input fields filled with specific values. The "City" field now contains "Dallas". The "Gender" field contains "Male". The "Age" field contains "25". The "Income" field contains "123456". The "Predict" button remains visible below the fields. The browser's address bar still shows the same URL.

Show desktop

### (3) Predicting the result

The screenshot shows a web browser window with the address bar displaying `illness-tracker.herokuapp.com`. The browser's tab bar includes several open tabs: YouTube, Gmail, Mail: Inbox, Übersetzen, Python Tutor - Visu..., PyPi: The Python P..., Machine Learning ..., Neurocomputing ..., API List: A public li..., Machine Learning..., PMovies | Watch M..., The Ultimate Guide..., DevDocs API Docu..., and LISUM10: 30 May ...

The main content area of the browser displays a web application titled "Illness Tracker". The application features a form with the following fields:

- City**: A text input field containing the value "Dallas, New York City, Los Angeles, Mountain View, Boston, Washington D.C., San Diego, Austin".
- Gender**: A text input field containing the value "Male or Female".
- Age**: A text input field containing the value "25-65".
- Income**: A text input field containing the value "123456".

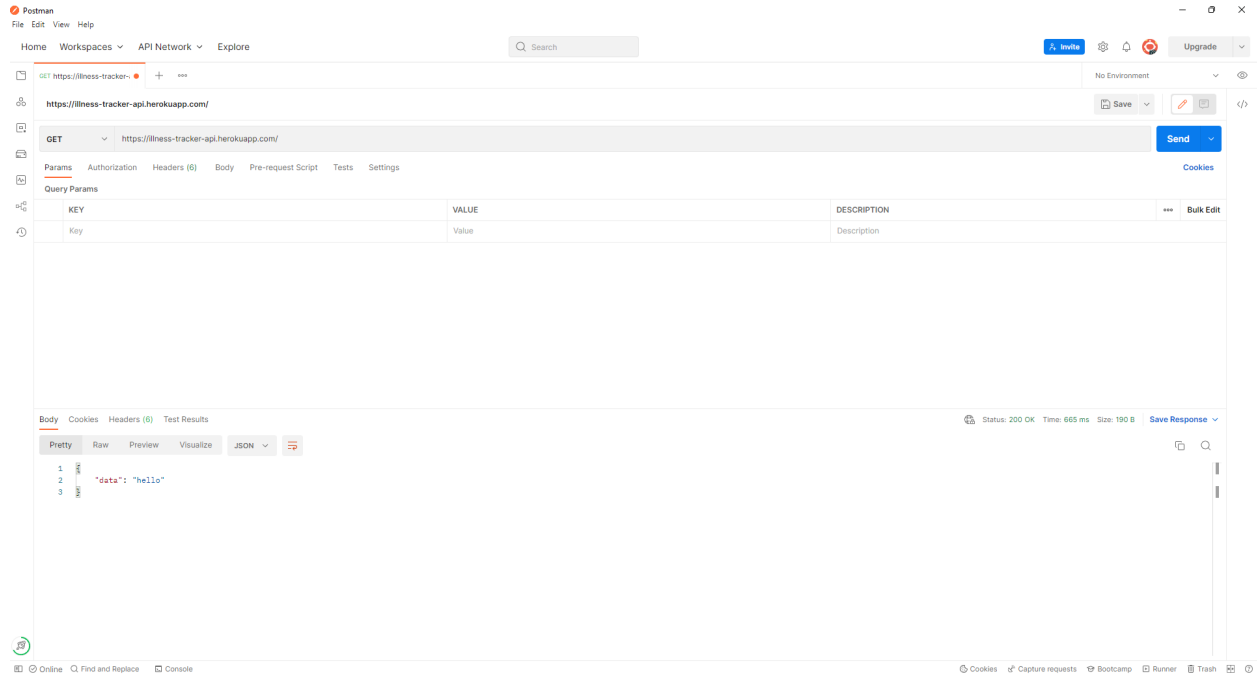
Below the form is a blue button labeled "Predict".

Below the "Predict" button, a light blue box displays the prediction result: `'prediction': You are not ill`.

## Step 7: Deploying Api Based app on Heroku and check it on Postman app

App link : <https://illness-tracker-api.herokuapp.com/>

### (1) Home Page (returning default json)



### (2) After Entering Key and Data (returns json format prediction)

