Week 5: Cloud and API deployment

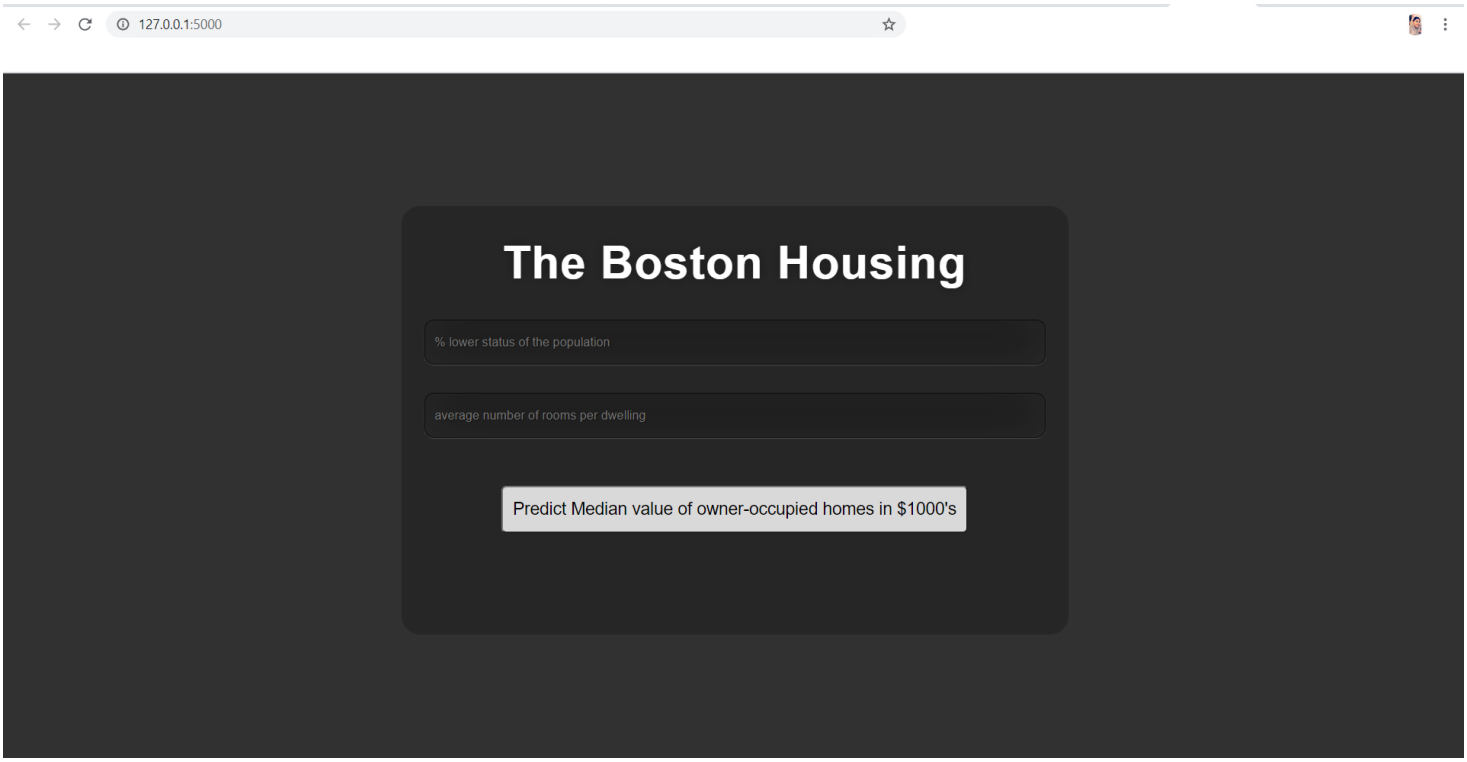
Name: Niyusha Baghayi
Batch Code: LICAN01
Submission Data: 3/27/2021
Submitted to: Data Glacier

Table of Contents

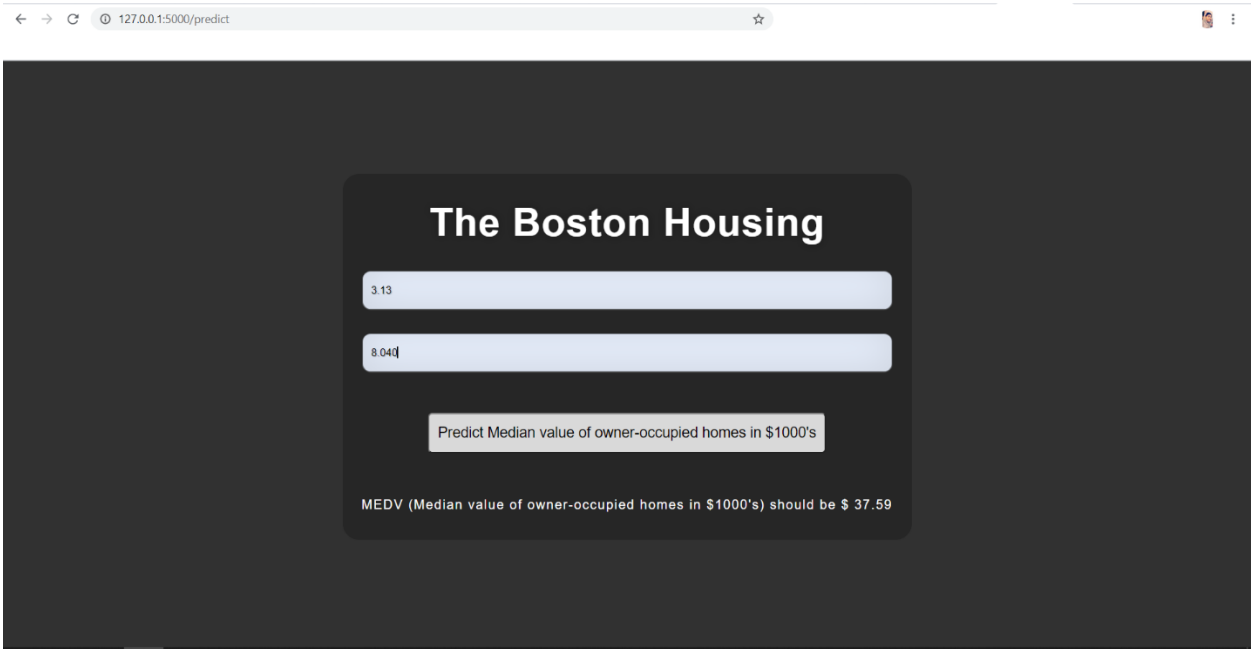
Introduction

The flask web app which is implemented do in the way that get the value of two variables from clients and after that predict the result and show it. The dataset that I have used for this project is “Boston Housing” dataset, I trained a model to predict the “MEDV” by two features “LSTAT” and “RM”. In the following sections I will explain each part of the project separately in detail.

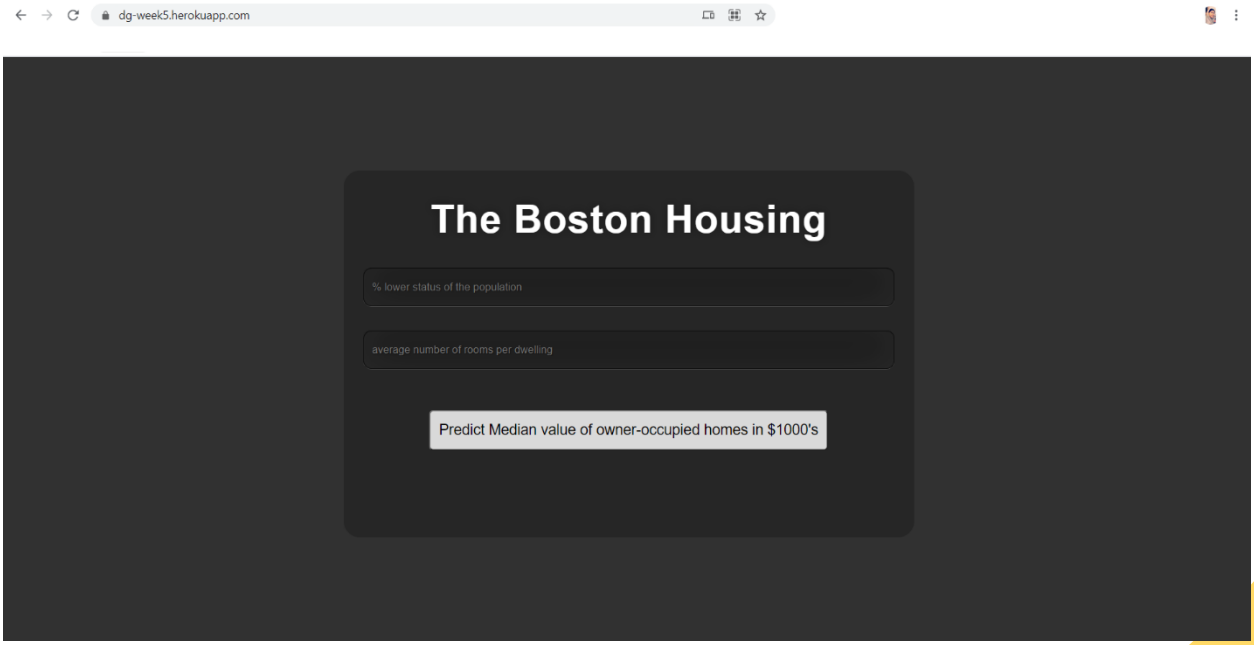
Here is the picture of the web app interface:



After a user put values and press the predict button, the answer will come in a short time:



At last, I have deployed this project on Heroku so we can see the web application on https://dg-week5.herokuapp.com/ :



And also we can use API to send the parameters to the application and get response of it :

```
PS C:\Users\Niu\Flask- Boston\Flask> python .\request.py
37.59
```

Dataset

The dataset that I used it is “Boston Housing” Dataset. The below picture illustrates all features of this dataset.

The Boston Housing Dataset

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

[1]

[1] https://www.kaggle.com/prasadperera/the-boston-housing-dataset

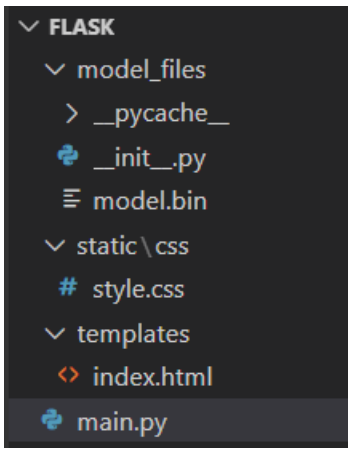
Model

I have deployed a model to predict “MEDV” based on “RM” and “LSTAT” as they have almost linear correlation with the target (MEDV). In this regard I applied Linear Regression to predict the target and tarin the model with whole dataset as train data. You can find the code in the linked picture.

The codes are written in Jupyter Notebook, at last I stored the model to the .bin file that I used it in the Flask.

Flask Deployment

For deploying Flask first of all we should install it, after that we can use it in python codes. The below picture show the structure of the flask deployment in this project:



As you can see here, we have some important files like .py, .html and .css files, we will explore each of these files in detail:

model.bin

Is the model that I stored as .bin file.

style.css

This file helps to have more beautiful view of this app. Here is the complete .css file:

static > css > # style.css > .login

Explorer (Ctrl+Shift+E)

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);

html { width: 100%; height:100%; overflow:hidden; }

body {
    width: 100%;
    height:100%;
    font-family: 'Helvetica';
    background: #000;
    color: #fff;
    font-size: 24px;
    text-align:center;
    letter-spacing:1.4px;

}
.login {
    position: absolute;
    top: 40%;
    left: 50%;
    margin: -150px 0 0 -350px;
    width:700px;
    height:450px;
    text-align: center;
    background:  rgb(38, 38, 38);
    border-radius: 20px;
}

.login h1, .login h2 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px; text-align:center; }
```

static > css > # style.css > body

```
input {
    width: 90%;
    height: 25px;
    margin-bottom: 30px;
    background: rgba(0,0,0,0.3);
    border: none;
    outline: none;
    padding: 10px;
    font-size: 13px;
    color: #fff;
    text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
    border: 1px solid rgba(0,0,0,0.5);
    border-radius: 10px;
    box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
    -webkit-transition: box-shadow .5s ease;
    -moz-transition: box-shadow .5s ease;
    -o-transition: box-shadow .5s ease;
    -ms-transition: box-shadow .5s ease;
    transition: box-shadow .5s ease;
    margin-left: 0;
    margin-right: 0;
}

.btn_submit {
    height: 50px;
    border-radius: 5px;
    font-size: 18px;
    padding: 10px;
    margin-top: 20px;
    background: #d9d9d9;
}

.prediction_text{
    font-size: 16px;
}
```

index.html

This is the main view page that contains all static elements which we can see in the UI as a client:

templates > index.html > html > head > title

```
<h1>The Boston Housing</h1>

<!-- Main Input For Receiving Query to our ML -->
<form action="{{ url_for('predict')}}" method="post">
<input type="text" name="LSTAT" placeholder="% lower status of the population" required="required" />
<input type="text" name="RM" placeholder="average number of rooms per dwelling" required="required" />
<button type="submit" class="btn btn-primary btn-block btn-large btn_submit">Predict Median value of owner-occupied homes in $1000's</button>
</form>

<br>
<br>
<div class="prediction_text">{{ prediction_text }}</div>
</div>
</body>
</html>
```

main.py

This is the last and the most important file of this project, first of all I imported all necessary libraries, after that the model is read from the file, also we should declare something to do for each rout and also, we should declare some routs that we can see webpages only by their routes.

The first rout shows the index.html.

The second rout which has “/predict” and that is the destination URL of the form that mentioned in “index.html”, when the predict button is pressed the form leads us to this URL and the following steps will happen:

Read data from the form (the values for “LSTAT” and “RM” entered by user), use the model and predict the “MEDV”, at last load the “index.html” with the result value of “MEDV” and show it.

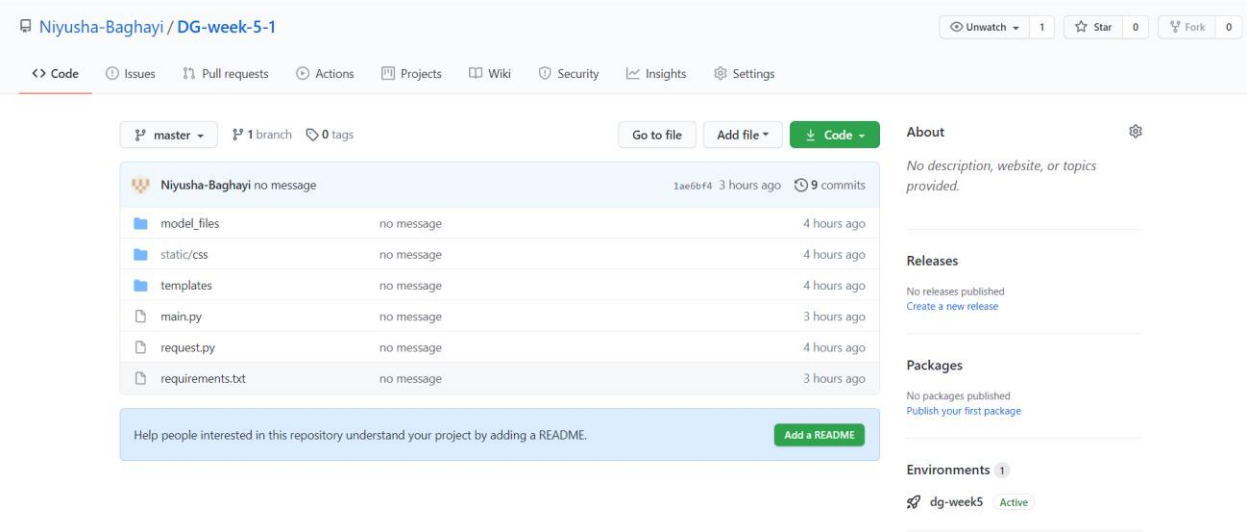
The last route is for pass the inputs by calling API, it will collect the parameters which pass trough POST method and run the same model on it and return the result of prediction and we can see the result in terminal after its running finished.

main.py > ...

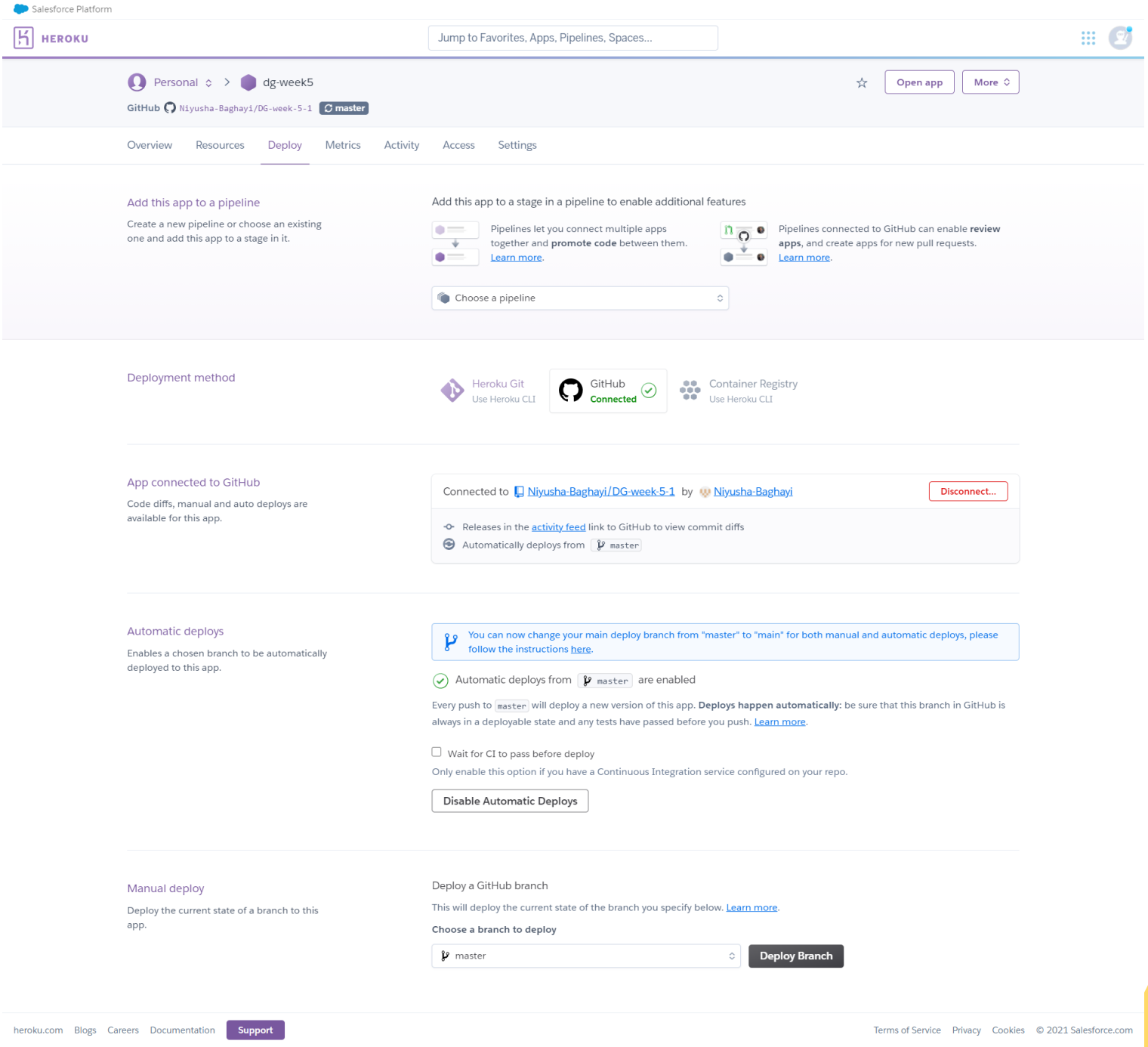
```python
from flask import Flask, render_template, request, jsonify
import pickle
import pandas as pd
import numpy as np

##creating a flask app and naming it "app"
app = Flask(__name__)

##loading the model from the saved file
with open('model_files/model.bin', 'rb') as f_in:
    model = pickle.load(f_in)

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

@app.route('/predict',methods=['POST'])
def predict():

    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text="MEDV (Median value of owner-occupied homes in $1000's) should be $ {}".format(output))

@app.route('/results',methods=['POST'])
def results():

    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = round(prediction[0], 2)
    return jsonify(output)

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

Deploying the Web App on Heroku

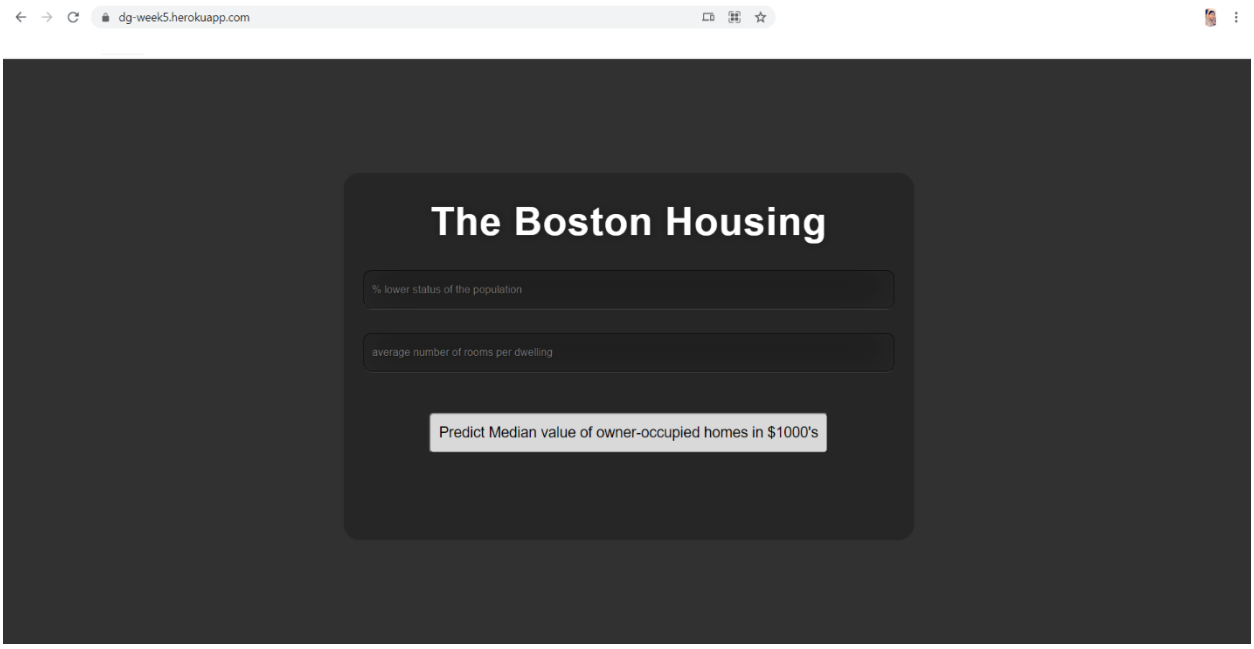
For deploying the application on Heroku, we can push all data to GitHub and by connecting the Heroku account to GitHub and fetching the data from there, deploy the application:



After that go to Heroku and connect the GitHub to it:


Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
dg-week5
Open app
More
GitHub Niyusha-Baghayi/DG-week-5-1 master
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.
Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and promote code between them. Learn more.
Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. Learn more.
Choose a pipeline
Deployment method
Heroku Git
Use Heroku CLI
GitHub
Connected
Container Registry
Use Heroku CLI
App connected to GitHub
Code diffs, manual and auto deploys are available for this app.
Connected to Niyusha-Baghayi/DG-week-5-1 by Niyusha-Baghayi
Disconnect...
Releases in the activity feed link to GitHub to view commit diffs
Automatically deploys from master
Automatic deploys
Enables a chosen branch to be automatically deployed to this app.
You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.
Automatic deploys from master are enabled
Every push to master will deploy a new version of this app. Deploys happen automatically: be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. Learn more.
Wait for CI to pass before deploy
Only enable this option if you have a Continuous Integration service configured on your repo.
Disable Automatic Deploys
Manual deploy
Deploy the current state of a branch to this app.
Deploy a GitHub branch
This will deploy the current state of the branch you specify below. Learn more.
Choose a branch to deploy
master
Deploy Branch
heroku.com
Blogs
Careers
Documentation
Support
Terms of Service
Privacy
Cookies
© 2021 Salesforce.com

At the end, we can see the application on Heroku https://dg-week5.herokuapp.com/



API

For calling the application by API, I have created a python code to send the values in JSON format through POST method :

request.py > url

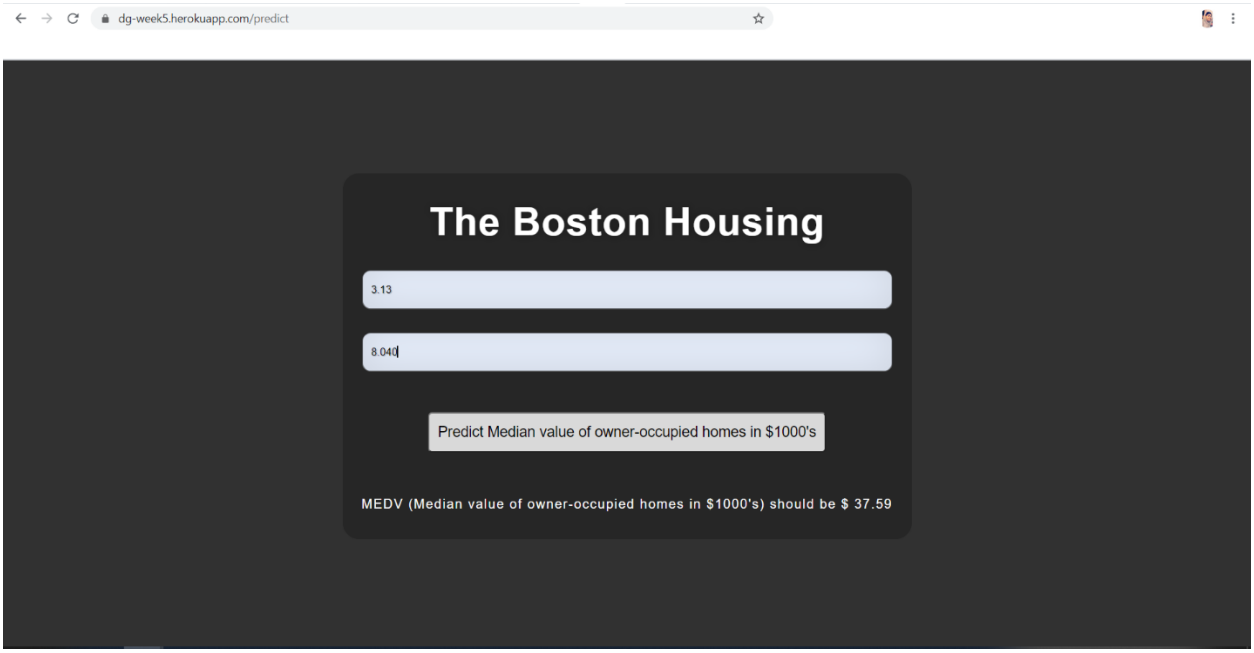
```
1  import requests
2
3  url = 'https://dg-week5.herokuapp.com/results'
4  r = requests.post(url,json={'LSTAT':3.13, 'RM':8.040})
5
6  print(r.json())
```

As you can see the URL is the URL of the application on Heroku and the path(/results) which we have defined a function for that to run the model for this data in “main.py”.

So we can run this python file by terminal:

```
PS C:\Users\Niu\Flask- Boston\Flask> python .\request.py
37.59
```

And also we can get the same response as we put the same value in web application by web UI :


dg-week5.herokuapp.com/predict
The Boston Housing
3.13
8.040
Predict Median value of owner-occupied homes in $1000's
MEDV (Median value of owner-occupied homes in $1000's) should be $ 37.59