

Name: İrem Tanrıverdi

Batch code: LISP01

Submission date: 20/03/2021

Submitted to: Data Glacier

Name of the dataset is “kc_house_data.csv” taken from Kaggle. This dataset includes information about characteristics of the houses like how many bathrooms, bedroom, etc. and what is the price of these house.

Index	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
0	221900	3	1	1180	5650	1	0	0	3	7
1	538000	3	2.25	2570	7242	2	0	0	3	7
2	180000	2	1	770	10000	1	0	0	3	6
3	604000	4	3	1960	5000	1	0	0	5	7
4	510000	3	2	1680	8080	1	0	0	3	8
5	1.225e+06	4	4.5	5420	101930	1	0	0	3	11
6	257500	3	2.25	1715	6819	2	0	0	3	7
7	291850	3	1.5	1060	9711	1	0	0	3	7
8	229500	3	1	1780	7470	1	0	0	3	7
9	323000	3	2.5	1890	6560	2	0	0	3	7
10	662500	3	2.5	3560	9796	1	0	0	3	8
11	468000	2	1	1160	6000	1	0	0	4	7
12	310000	3	1	1430	19901	1.5	0	0	4	7
13	400000	3	1.75	1370	9680	1	0	0	4	7
14	530000	5	2	1810	4850	1.5	0	0	3	7

This is the snapshot of the dataset.

Waterfront: There is waterfront=1 ; There is not waterfront=0

```
Users/İrem/Desktop/Deployment-flask-master-2/model2.py
x app.py x request.py x index.html x model2.py* x model.py
1 # Importing the libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import pickle
6
7
8 dataset = pd.read_csv('kc_house_data.csv')
9
10 X = dataset.iloc[:, [3,4,7,8,9,10,11]] # selecting covariates from dataset
11 y = dataset.iloc[:, 2] # selecting response from dataset
12
13 from sklearn.linear_model import LinearRegression
14 regressor = LinearRegression()
15
16 #Fitting model with trainig data
17 regressor.fit(X, y)
18
19 # Saving model to disk
20 pickle.dump(regressor, open('model.pkl','wb'))
21
22 # Loading model to compare the results
23 model = pickle.load(open('model.pkl','rb'))
24 print(model.predict([[2, 2, 2,1,2,3,6]]))
25
```

Name	Type	Size
dataset	DataFrame	(21613, 21)
model	linear_model._base.LinearRegression	1
regressor	linear_model._base.LinearRegression	1
X	DataFrame	(21613, 7)
y	Series	(21613,)

- Linear regression model had conducted to predict house price, taking some characteristics of the houses as predictors.

```

/Users/irem/Desktop/Deployment-flask-master-2/templates/index.html
x app.py x request.py x index.html x model2.py x model.py
1 <!DOCTYPE html>
2 <html>
3 <!--From https://codepen.io/frytyler/pen/EGdtg-->
4 <head>
5   <meta charset="UTF-8">
6   <title>ML API</title>
7   <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
8   <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
9   <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
10  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
11  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
12 </head>
13
14 <body>
15   <div class="login">
16     <h1>Prediction of House Price</h1>
17
18     <!-- Main Input For Receiving Query to our ML -->
19     <form action="{{ url_for('predict')}}" method="post">
20       <input type="text" name="bedrooms" placeholder="Bedrooms" required="required" />
21       <input type="text" name="bathrooms" placeholder="Bathrooms" required="required" />
22       <input type="text" name="floors" placeholder="Floors" required="required" />
23       <input type="text" name="waterfront" placeholder="Waterfront" required="required" />
24       <input type="text" name="view" placeholder="View" required="required" />
25       <input type="text" name="condition" placeholder="Condition" required="required" />
26       <input type="text" name="grade" placeholder="Grade" required="required" />
27
28       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
29     </form>
30
31     <br>
32     <br>
33     {{ prediction_text }}
34
35   </div>
36 </body>
37 </html>

```

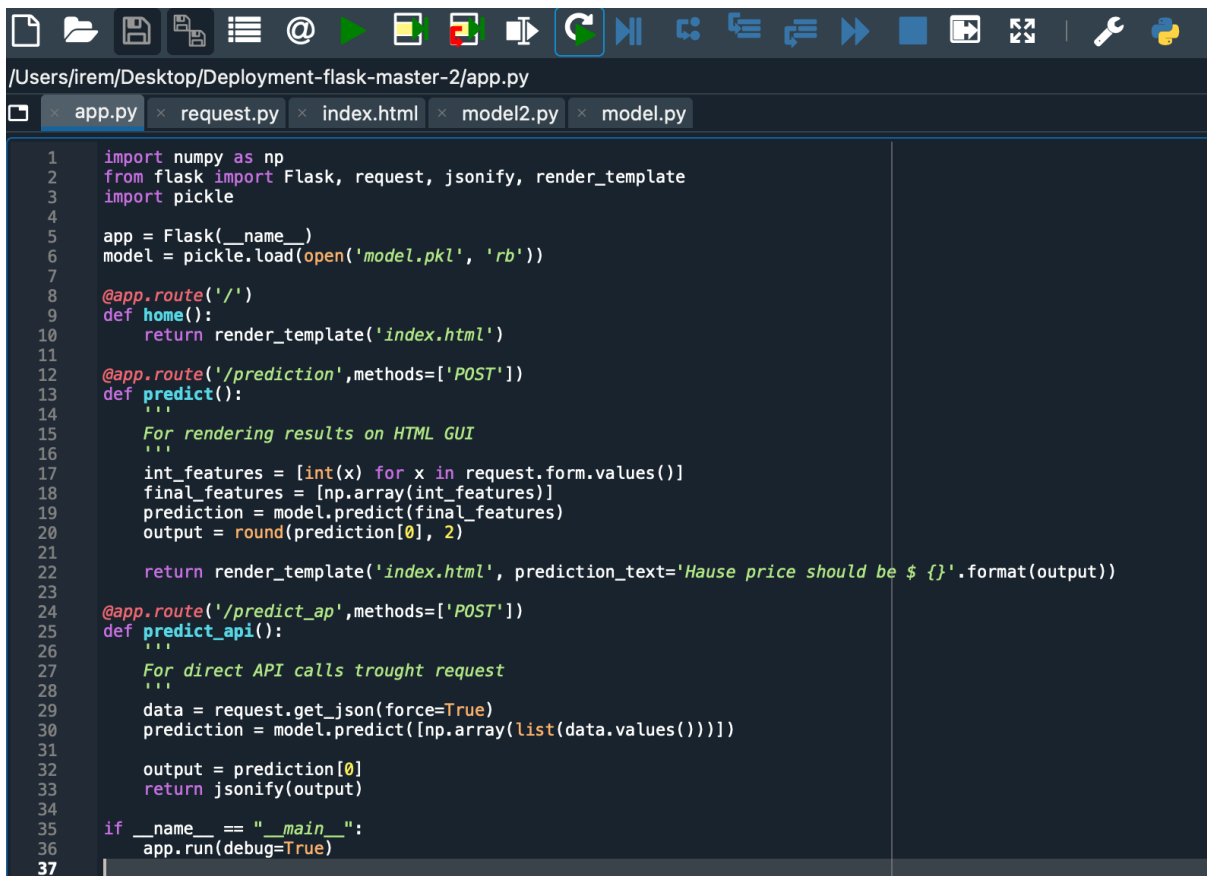
- This is the root node where API URL should go.
- index.html file is like our home page.

```

/Users/irem/Desktop/Deployment-flask-master-2/request.py
x app.py x request.py x index.html x model2.py x model.py
1 import requests
2
3 url = 'http://localhost:9000/predict_ap'
4 r = requests.post(url,json={'bedrooms':2, 'bathrooms':2, 'floors':2,'waterfront':1,
5                             'view':2, 'condition':3,'grade':6})
6
7

```

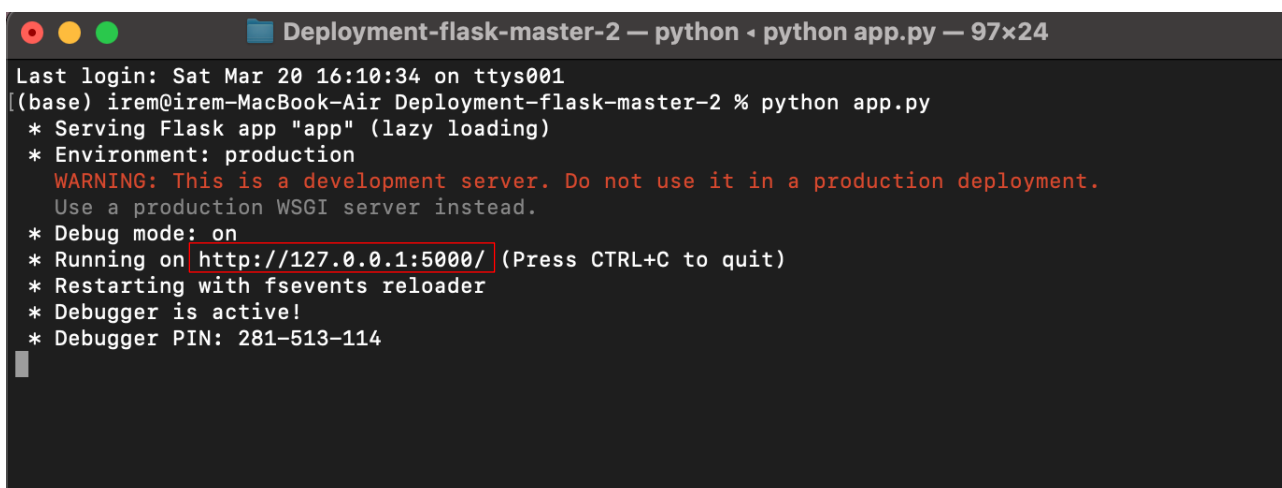
- We create request.py file to give URL. We are just saying that request or post URL and we give json values.



```
1 import numpy as np
2 from flask import Flask, request, jsonify, 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('/prediction', methods=['POST'])
13 def predict():
14     """
15     For rendering results on HTML GUI
16     """
17     int_features = [int(x) for x in request.form.values()]
18     final_features = [np.array(int_features)]
19     prediction = model.predict(final_features)
20     output = round(prediction[0], 2)
21
22     return render_template('index.html', prediction_text='Hause price should be $ {}'.format(output))
23
24 @app.route('/predict_api', methods=['POST'])
25 def predict_api():
26     """
27     For direct API calls through request
28     """
29     data = request.get_json(force=True)
30     prediction = model.predict([np.array(list(data.values()))])
31
32     output = prediction[0]
33     return jsonify(output)
34
35 if __name__ == "__main__":
36     app.run(debug=True)
37
```

- In this app.py file, we create flask environment where we will be creating our API, and where we will read this file and then we will give the input to the file.

Then we open the terminal, and the read app.py file on the terminal.

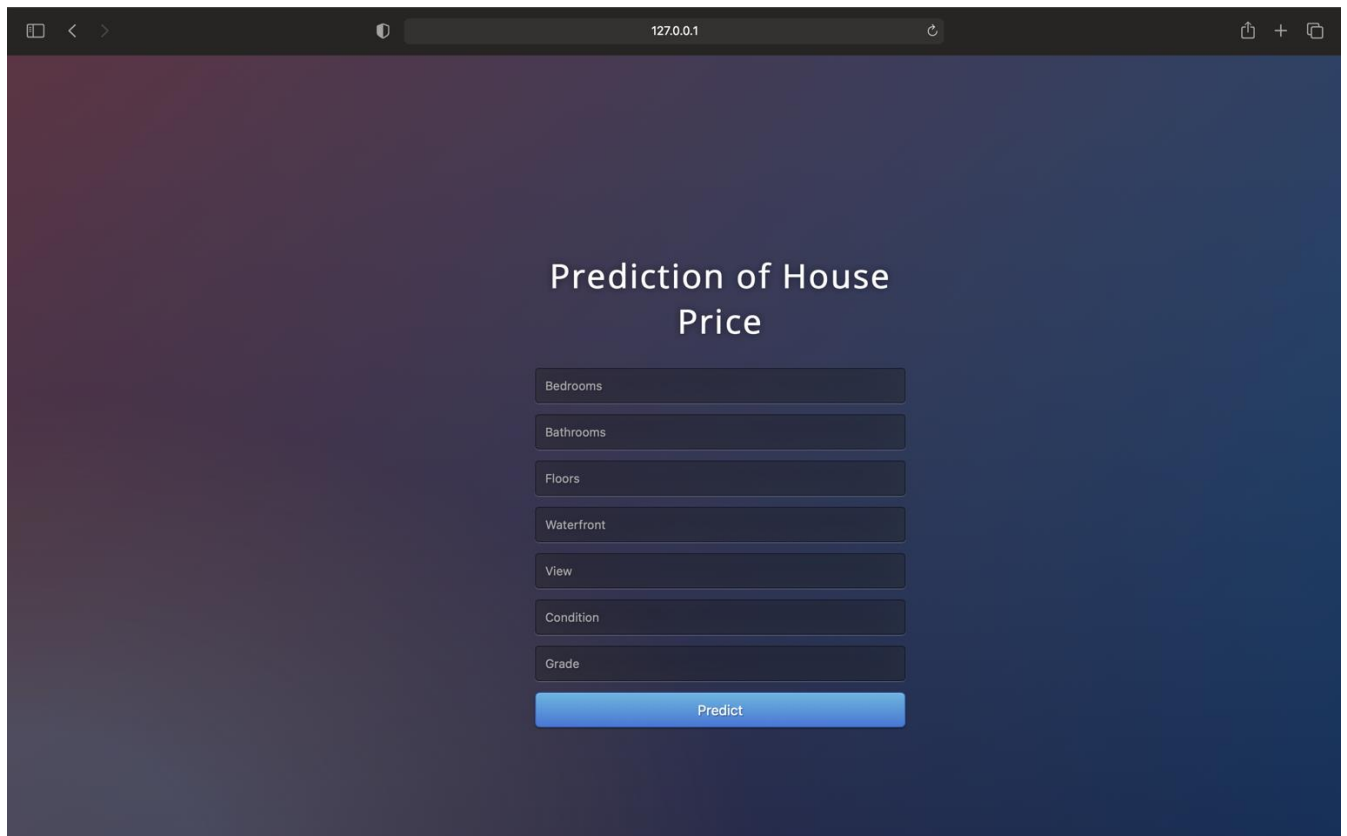


```
Deployment-flask-master-2 — python < python app.py — 97x24
Last login: Sat Mar 20 16:10:34 on ttys001
(base) irem@irem-MacBook-Air Deployment-flask-master-2 % python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with fsevents reloader
* Debugger is active!
* Debugger PIN: 281-513-114
```

We see that address of the demo is <http://127.0.0.1:5000/>

Then we go to this address.

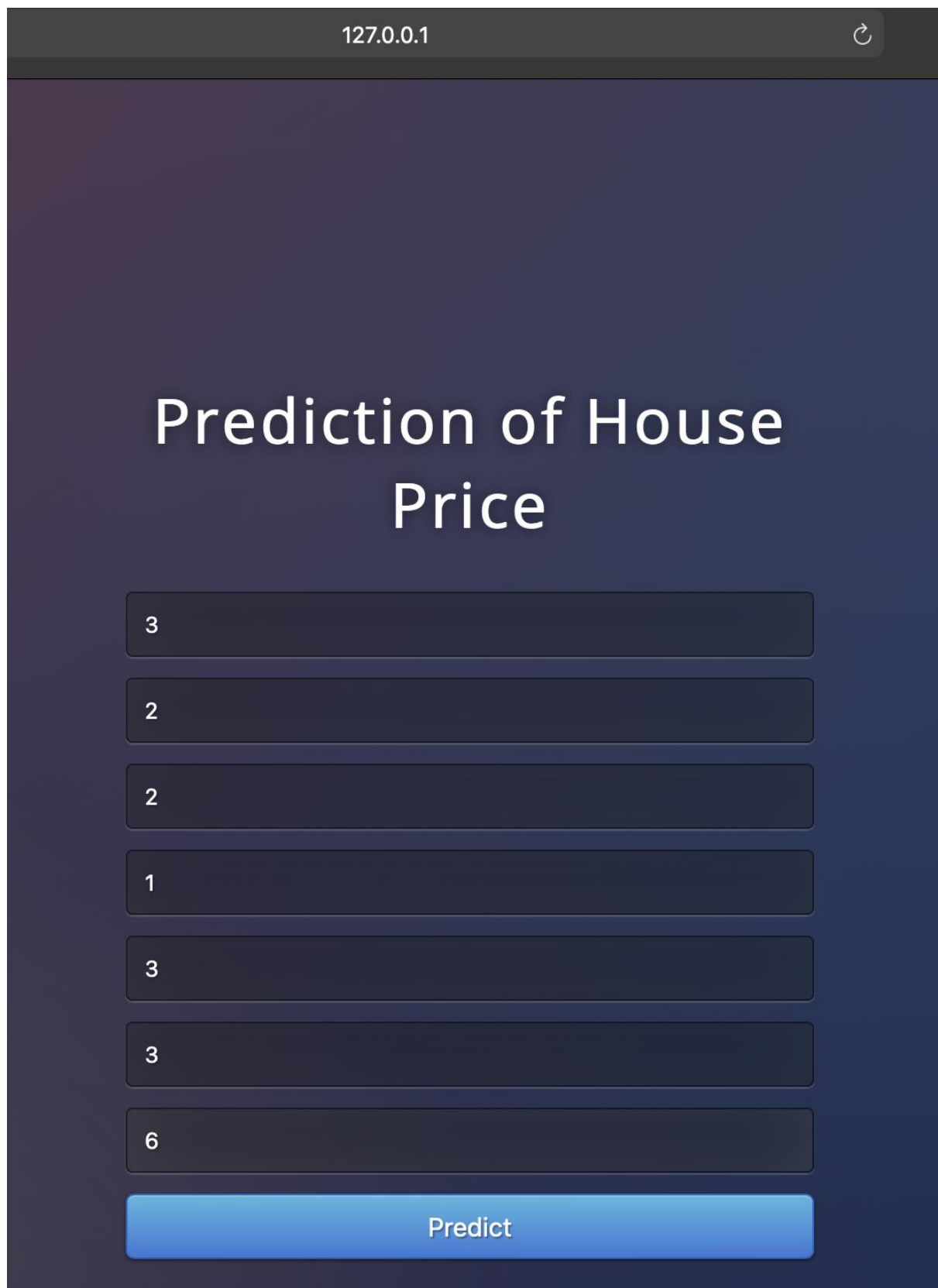
This is our home page.



The screenshot shows a web browser window with a dark theme. The address bar displays '127.0.0.1'. The main content area has a dark blue gradient background. Centered on the page is the title 'Prediction of House Price' in white text. Below the title is a vertical stack of eight dark blue input fields, each with a light blue border and placeholder text: 'Bedrooms', 'Bathrooms', 'Floors', 'Waterfront', 'View', 'Condition', and 'Grade'. At the bottom of this stack is a blue button with the text 'Predict' in white.

Let's try some values to see the prediction. Let's see what the difference between the price of the houses with waterfront is and without waterfront keeping the other covariates same.

Price of the house with waterfront:



127.0.0.1

Prediction of House Price

3

2

2

1

3

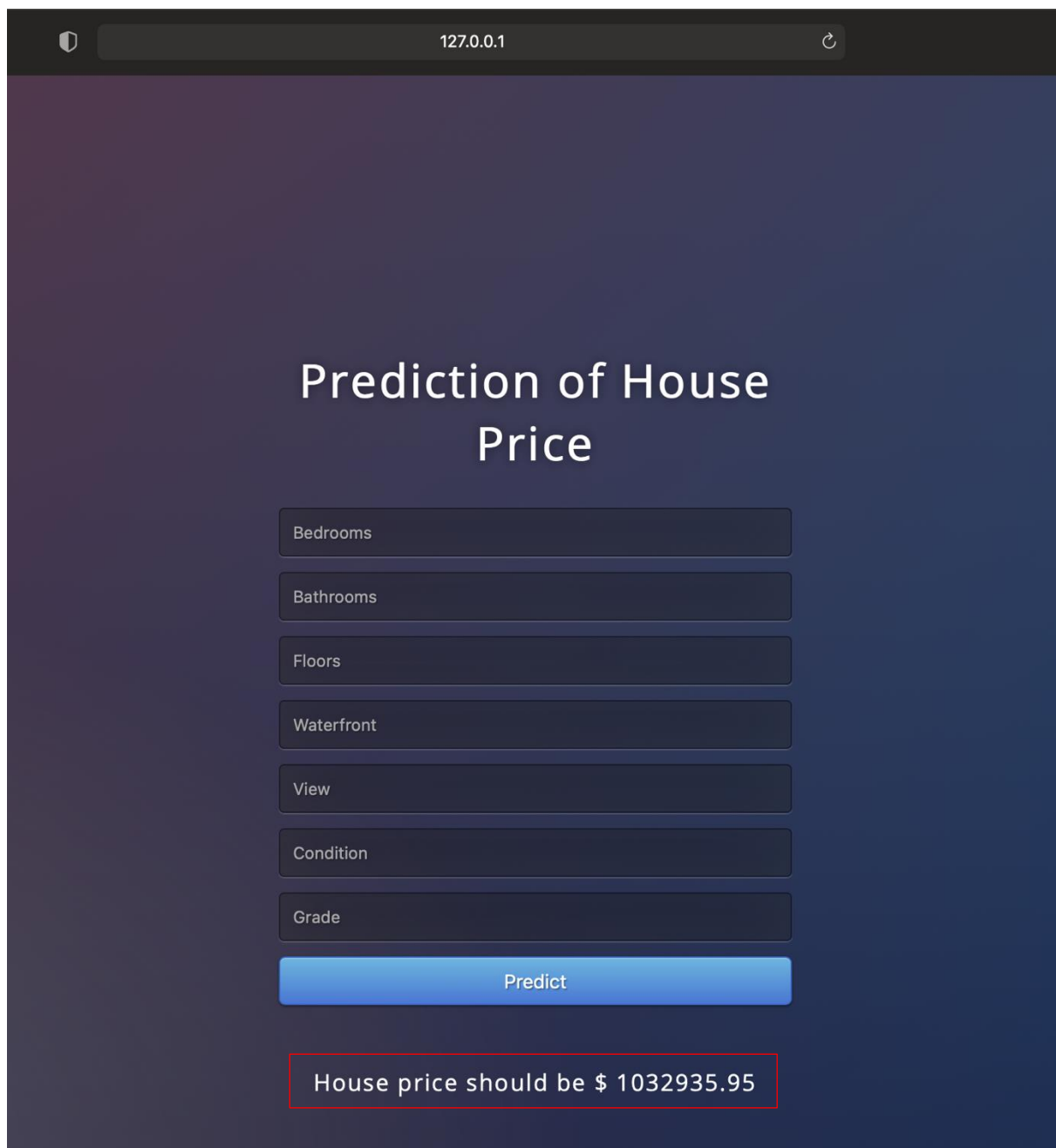
3

6

Predict

First, we enter the characteristics of the house.

Then we predict:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1'. The page has a dark blue gradient background. The title 'Prediction of House Price' is centered in white. Below the title, there are seven input fields stacked vertically, each with a light blue border and a dark blue background. The fields are labeled 'Bedrooms', 'Bathrooms', 'Floors', 'Waterfront', 'View', 'Condition', and 'Grade'. Below these fields is a prominent blue button with the text 'Predict'. At the bottom, a white rectangular box with a red border contains the text 'House price should be \$ 1032935.95'.

Prediction of House Price

Bedrooms

Bathrooms

Floors

Waterfront

View

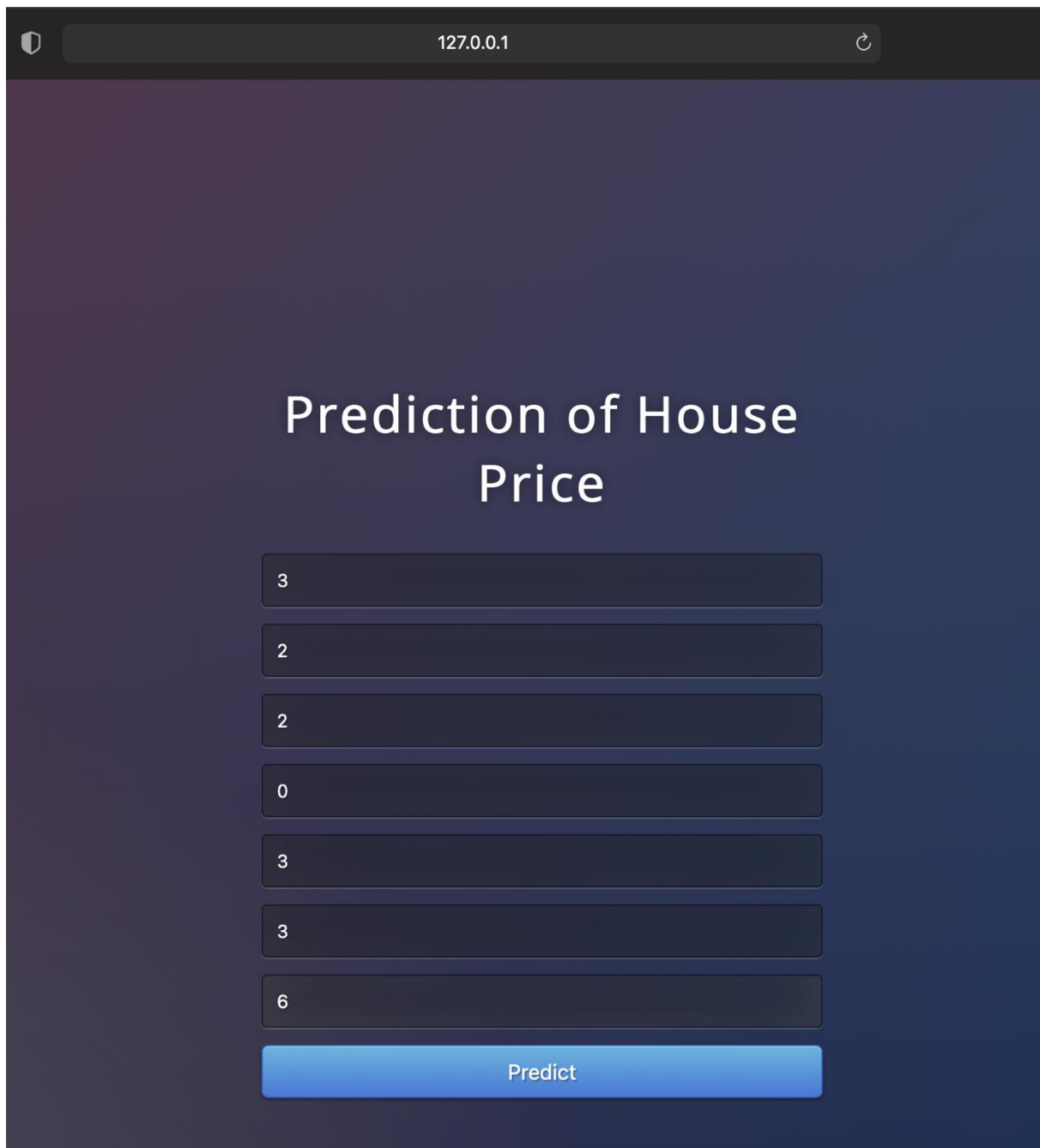
Condition

Grade

Predict

House price should be \$ 1032935.95

Price of the house without waterfront:

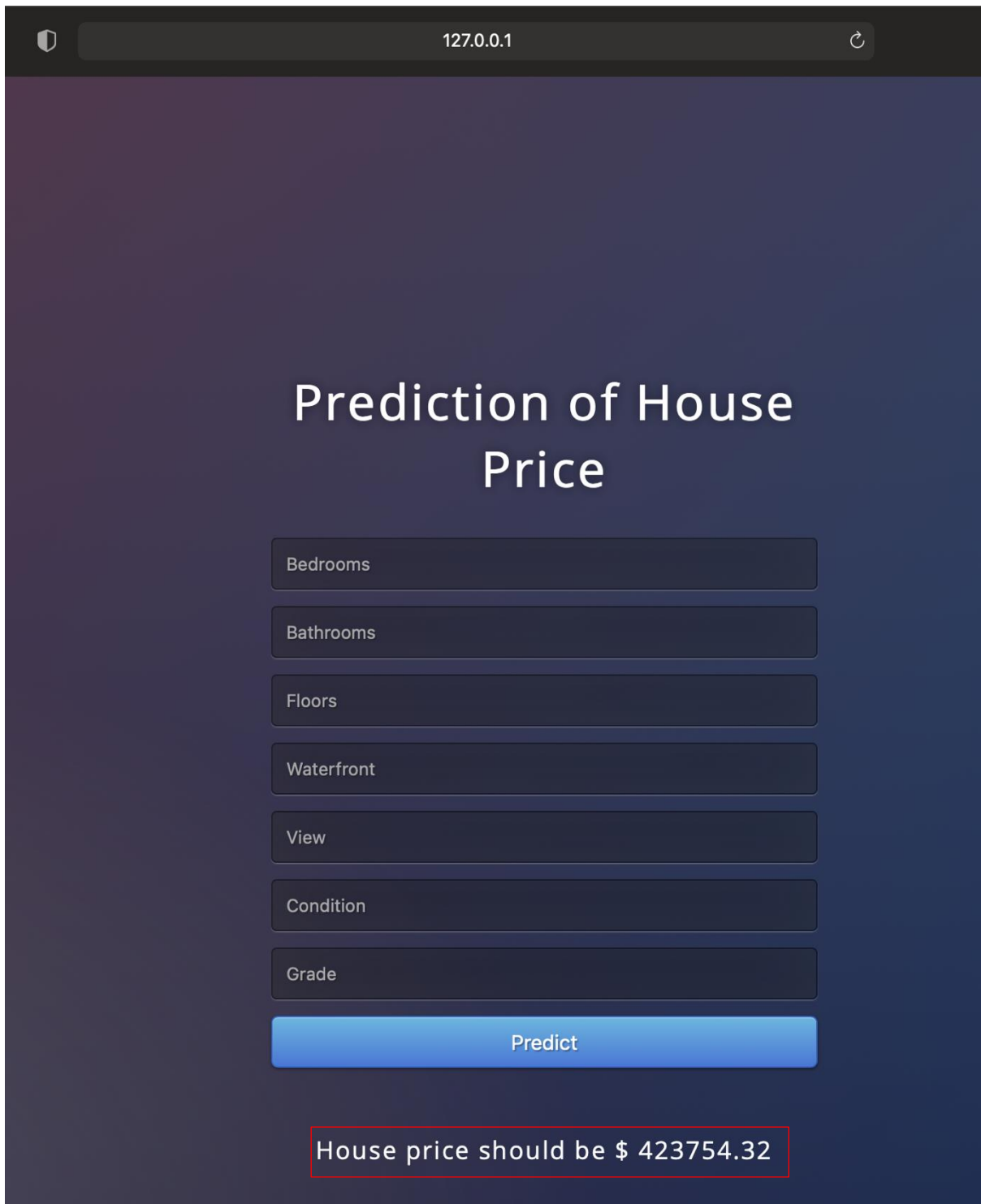


The screenshot shows a web browser window with the address bar displaying '127.0.0.1'. The page has a dark blue gradient background. The title 'Prediction of House Price' is centered in white. Below the title, there are seven input fields, each containing a number. At the bottom, there is a blue 'Predict' button.

Feature	Value
Number of bedrooms	3
Number of bathrooms	2
Number of floors	2
Number of parking spaces	0
Number of living rooms	3
Number of dining rooms	3
Number of kitchens	6

Predict

Then we predict:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1'. The page has a dark blue gradient background. The title 'Prediction of House Price' is centered in white. Below the title are seven input fields, each with a light blue border and a dark blue background, containing the following text from top to bottom: 'Bedrooms', 'Bathrooms', 'Floors', 'Waterfront', 'View', 'Condition', and 'Grade'. Below these fields is a large blue button with the text 'Predict'. At the bottom, a red-bordered box contains the text 'House price should be \$ 423754.32'.

Feature	Value
Bedrooms	
Bathrooms	
Floors	
Waterfront	
View	
Condition	
Grade	

Predict

House price should be \$ 423754.32

Then we see that while the house price with waterfront is \$1.032.935, the house price without waterfront is \$423.754.