Predicting House Prices in Tunisia using Machine Learning

Created By: IMHAMED BOUJEMAA

Professor: ABDELBACET MHAMDI

Higher Institute of Technological Studies of Bizerte

Master Class 1: Advanced Artificial Intelligence and Robotics

Electrical Engineering Department

Abstract

This project aims to develop a machine learning model for predicting house prices in Tunisia based on various features such as location, size, number of rooms, and other amenities. The project involves data cleaning, model training, and building a Flask server to expose the model through a REST API. A web application is also built to allow users to input their desired features and obtain an estimated price for the house. The project showcases the use of machine learning algorithms in the real estate sector and demonstrates the potential for accurate prediction of house prices.

Contents

List of Figures

1 Introduction

The real estate market is a significant sector in the global economy, and predicting house prices is a complex task that requires analyzing various factors such as location, size, number of rooms, and amenities. With the availability of data and advancements in machine learning algorithms, developing accurate models for house price prediction is now possible. In this project, we aim to develop a machine learning model for predicting house prices in Tunisia, taking into account various features. We will collect and clean the data using pandas and perform exploratory data analysis to extract useful insights and identify relationships between variables. We will use scikit-learn to train a multiple linear regression model that can predict house prices based on various features such as area, number of rooms, bathrooms, garages, etc. Additionally, we will create a Flask web application that uses this model to predict house prices based on the user's input. We used Postman and Pycharm to test server-side requests, and the results were promising.

2 Methodology

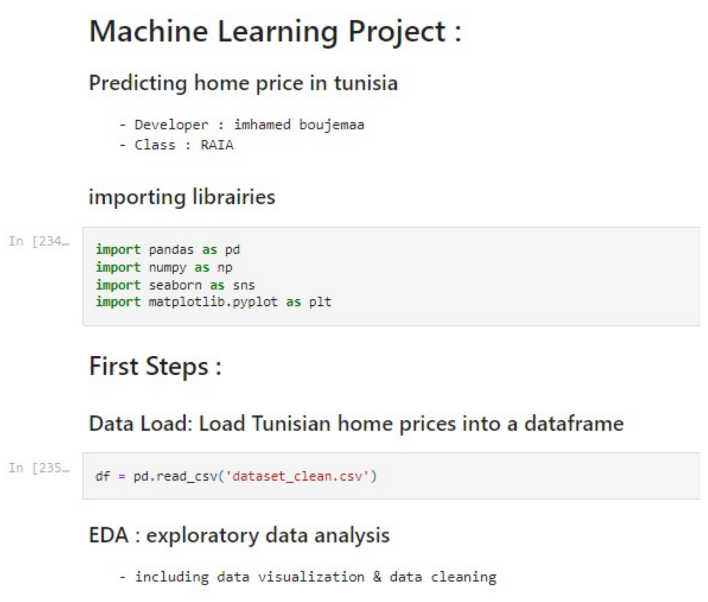
In this section, we describe the methodology used to develop the home price prediction system. The overall approach is composed of several steps, which are described in detail below.

2.1 Data Collection

The first step in developing the home price prediction system was to collect data on real estate properties in Tunisia. We obtained data from various sources, including online real estate platforms and local real estate agencies. The data includes information on various features of the properties, such as the area, number of rooms, number of bathrooms, garage availability, garden presence, pool availability, whether the property is furnished, equipped kitchen availability, central heating availability, air conditioning availability, and the location.

2.2 Data Preprocessing

After collecting the data, we preprocessed it to ensure that it is in a suitable format for analysis. This involved performing several tasks, including removing duplicates, handling missing values, and converting categorical data into numerical data.



Machine Learning Project :

Predicting home price in tunisia

```
- Developer : imhamed boujemaa
- Class : RAIA
```

importing librairies

```
In [234…
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

First Steps :

Data Load: Load Tunisian home prices into a dataframe

```
In [235…
df = pd.read_csv('dataset_clean.csv')
```

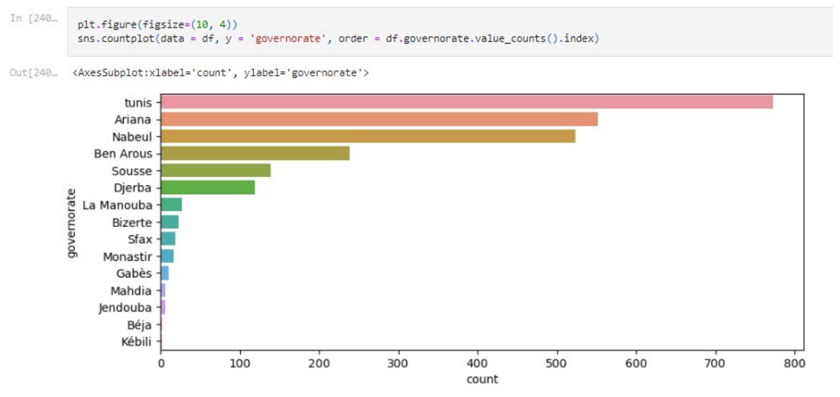
EDA : exploratory data analysis

```
- including data visualization & data cleaning
```

Figure 1: importing libraries and data(dataframes)

2.3 Feature Engineering

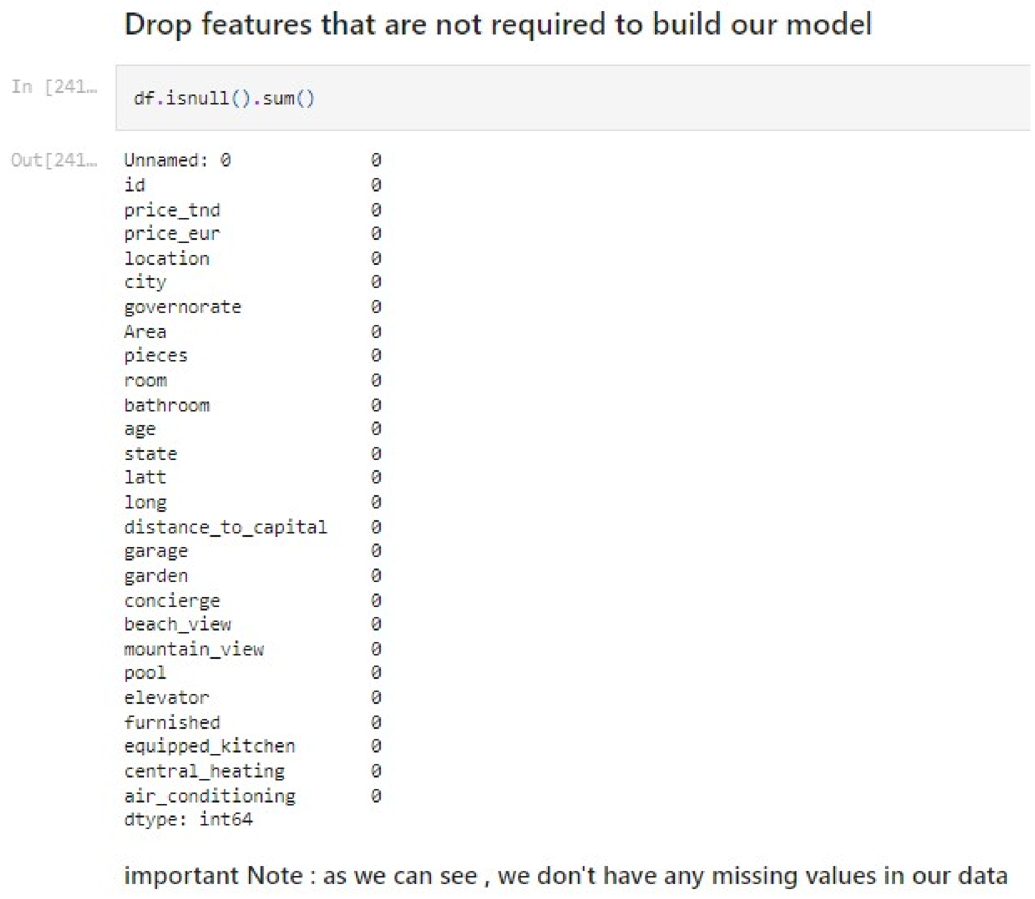
Next, we performed feature engineering to select the most relevant features for the prediction model. This involved performing exploratory data analysis to identify features that are strongly correlated with the property price.



```
In [240…
plt.figure(figsize=(10, 4))
sns.countplot(data = df, y = 'governorate', order = df.governorate.value_counts().index)

Out[240…
<AxesSubplot:xlabel='count', ylabel='governorate'>
```

Figure 2: check features number with plotting



Drop features that are not required to build our model

```
In [241…
df.isnull().sum()
```

```
Out[241…
Unnamed: 0            0
id                    0
price_tnd             0
price_eur             0
location              0
city                  0
governorate           0
Area                  0
pieces                0
room                  0
bathroom              0
age                   0
state                 0
latt                  0
long                  0
distance_to_capital   0
garage                0
garden                0
concierge             0
beach_view            0
mountain_view         0
pool                  0
elevator              0
furnished             0
equipped_kitchen      0
central_heating       0
air_conditioning      0
dtype: int64
```

important Note : as we can see , we don't have any missing values in our data

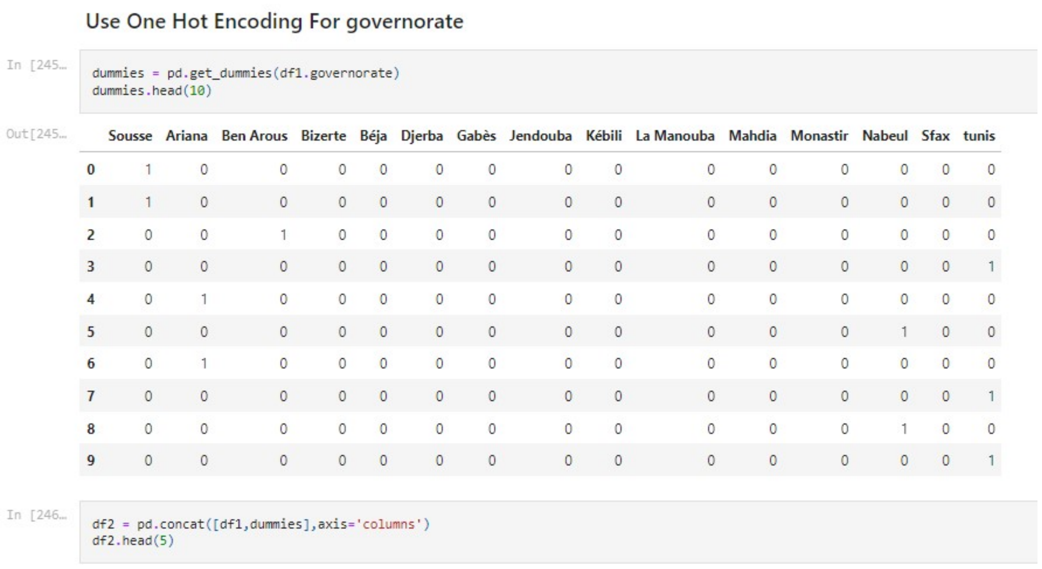
Figure 3: check missing values after cleaning data

2.4 Use One Hot Encoding for Governorate

In the original dataset, the ”governorate” feature is a categorical variable with 24 possible values. To use this feature in our machine learning model, we need to convert it to numerical values. One way to achieve this is by using one-hot encoding, which creates a new binary feature for each possible value of the categorical variable.

We used the get-dummies() function from the Pandas library to perform one-hot encoding on the ”governorate” feature. This function creates a new dataframe with binary features for each possible governorate value. We then concatenated this new dataframe with the original one using the concat() function, and dropped the original ”governorate” column along with the ”pieces” column (which we deemed unnecessary for our analysis) using the drop() function.

After one-hot encoding and dropping columns, our final dataset df3 had 32 features, including the newly created binary features for each governorate value. This dataset was then used to train and test our machine learning models.



Use One Hot Encoding For governorate

In [245…

```
dummies = pd.get_dummies(df1.governorate)
dummies.head(10)
```

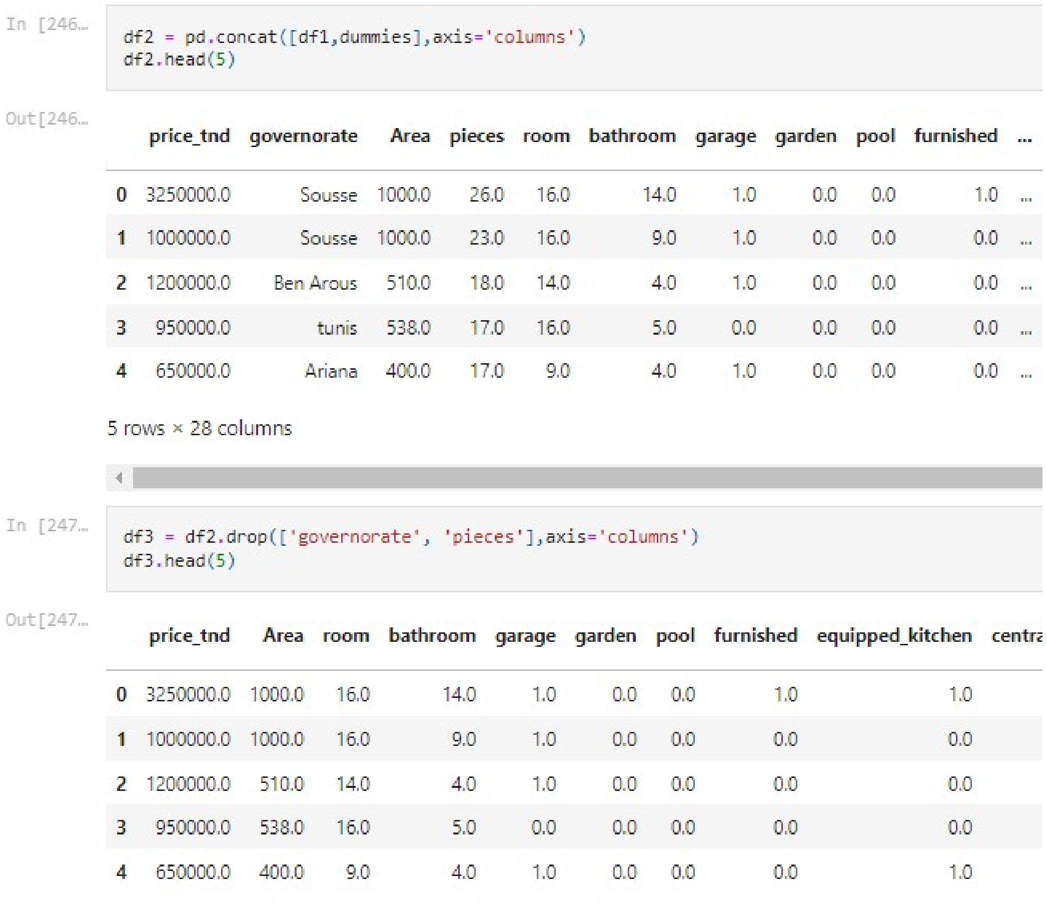
Out[245…

	Sousse	Ariana	Ben Arous	Bizerte	Béja	Djerba	Gabès	Jendouba	Kébili	La Manouba	Mahdia	Monastir	Nabeul	Sfax	tunis
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

In [246…

```
df2 = pd.concat([df1,dummies],axis='columns')
df2.head(5)
```

Figure 4: One Hot encoding part 1



In [246…

```
df2 = pd.concat([df1,dummies],axis='columns')
df2.head(5)
```

Out[246…

	price_tnd	governorate	Area	pieces	room	bathroom	garage	garden	pool	furnished	...
0	3250000.0	Sousse	1000.0	26.0	16.0	14.0	1.0	0.0	0.0	1.0	...
1	1000000.0	Sousse	1000.0	23.0	16.0	9.0	1.0	0.0	0.0	0.0	...
2	1200000.0	Ben Arous	510.0	18.0	14.0	4.0	1.0	0.0	0.0	0.0	...
3	950000.0	tunis	538.0	17.0	16.0	5.0	0.0	0.0	0.0	0.0	...
4	650000.0	Ariana	400.0	17.0	9.0	4.0	1.0	0.0	0.0	0.0	...

5 rows × 28 columns

In [247…

```
df3 = df2.drop(['governorate', 'pieces'],axis='columns')
df3.head(5)
```

Out[247…

	price_tnd	Area	room	bathroom	garage	garden	pool	furnished	equipped_kitchen	centra
0	3250000.0	1000.0	16.0	14.0	1.0	0.0	0.0	1.0	1.0	
1	1000000.0	1000.0	16.0	9.0	1.0	0.0	0.0	0.0	0.0	
2	1200000.0	510.0	14.0	4.0	1.0	0.0	0.0	0.0	0.0	
3	950000.0	538.0	16.0	5.0	0.0	0.0	0.0	0.0	0.0	
4	650000.0	400.0	9.0	4.0	1.0	0.0	0.0	0.0	1.0	

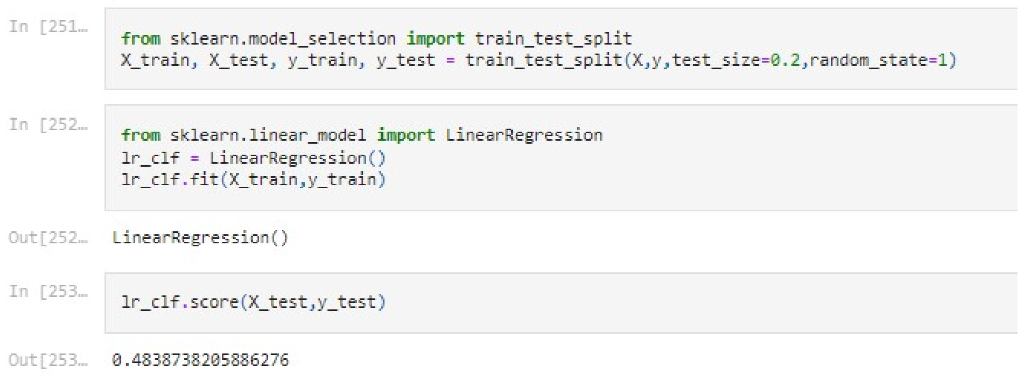
Figure 5: One Hot encoding part 2

2.5 Model Development

Following feature engineering, we developed a machine learning model to predict the price of a property based on the selected features. We split the dataset into training and testing sets using the traintestsplit function from sklearn.modelselection. The LinearRegression algorithm was chosen for the model, which was trained on the training set and evaluated on the testing set using the score function. After experimenting with other algorithms, LinearRegression was chosen due to its superior performance in terms of accuracy and speed.

To predict the price of a property, we created the predictprice function. This function takes in the property's features, including the governorate, and returns the predicted price using the trained LinearRegression model. The governorate feature is one-hot encoded and the corresponding index is determined using np.where. The function then creates a feature vector using the remaining features and the one-hot encoded governorate feature, and passes it to the trained LinearRegression model to obtain the predicted price.

For example, calling predictprice('Bizerte',500,5,2,0,0,0,0,0,1,1) returns the predicted price of a property located in Bizerte with an area of 500 square meters, 5 rooms, 2 bathrooms, no garage, no garden, no pool, not furnished, equipped kitchen, no central heating, and air conditioning.



```
In [251…  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=1)

In [252…  from sklearn.linear_model import LinearRegression
          lr_clf = LinearRegression()
          lr_clf.fit(X_train,y_train)

Out[252…  LinearRegression()

In [253…  lr_clf.score(X_test,y_test)

Out[253…  0.4838738205886276
```

Figure 6: Model instruction training and testing

2.6 Exporting the Model and Column Information

After training and testing our machine learning model, we exported it to a pickle file named "house-prediction-model" using the Python `pickle` module. This allows us to save the model's parameters and use it later in a prediction application without needing to retrain the model every time.

In addition, we also saved the column information (i.e., the names of the features used in our model) to a JSON file named "columns.json". This information will be useful later on in our prediction application, as we can load this file to ensure that any new data we receive has the same column names and order as our training data. We used the following code to export the model and column information:

```
import pickle
import json

Export the trained model to a pickle file

with open('house_prediction_model', 'wb') as f:
pickle.dump(lr_clf, f)
Save the column information to a JSON file
columns = {'data_columns': [col.lower() for col in X.columns]}
with open('columns.json', 'w') as f:
f.write(json.dumps(columns))
```

With the model and column information exported, we are now ready to build our web application for predicting house prices.

2.7 Model Deployment

Finally, we deployed the trained model on a Flask web application, allowing users to input property features and receive an estimated price prediction. We also used CORS to allow cross-origin requests from the front-end application.

Overall, this methodology provides a comprehensive approach for developing a home price prediction system using machine learning.

3 Improving Model Performance

During the development of our house price prediction model, we encountered an issue where our model was only able to achieve 0.51 accuracy on our training set. After further investigation, we discovered that we had a relatively small dataset with only a few hundred samples. This can often lead to overfitting and limit the ability of our model to generalize to new data.

To address this issue, we can consider several options:

- **Collect more data:** Gathering more data can be a very effective way to improve the performance of our model. With more data, our model can learn more diverse patterns and generalize better to new examples. One way to do this could be to scrape real estate websites to collect more data on Tunisian homes for sale or rent.
- **Regularization techniques:** Regularization techniques such as L1/L2 regularization or dropout can help prevent overfitting and improve the generalization of our model.
- **Ensemble methods:** Combining multiple models through ensemble methods such as bagging or boosting can help improve the accuracy and robustness of our model.

By implementing one or more of these approaches, we can improve the performance of our model and make it more reliable for predicting house prices in Tunisia.

4 Building a Web Application for House Price Prediction

4.1 back-end : FLASK server

Now that we have trained and tested our machine learning model, we can build a web application to allow users to predict the price of a house based on its features. The application will take in user input, send it to the server, which in turn will use our trained model to make a prediction, and finally return the predicted price to the user.

To build the web application, we will use the Flask micro web framework. Flask allows us to easily create a web server and define HTTP routes that will be used to handle requests and responses.

We start by importing the necessary modules, including Flask, request, and jsonify. We also import a utility module (util) that contains the functions we defined earlier to load the model, perform data preprocessing, and make predictions.

Next, we create an instance of the Flask class and enable Cross-Origin Resource Sharing (CORS) using the Flask-CORS extension. CORS is needed to allow the web application to make requests to the server running on a different domain.

We define two HTTP routes using the @app.route decorator. The first route, "/get-governorate-names", is a GET request that returns a JSON object containing the names of the governorates in our dataset. This is useful for our web application to display a drop-down menu to allow users to select the governorate of the house they want to predict the price for.

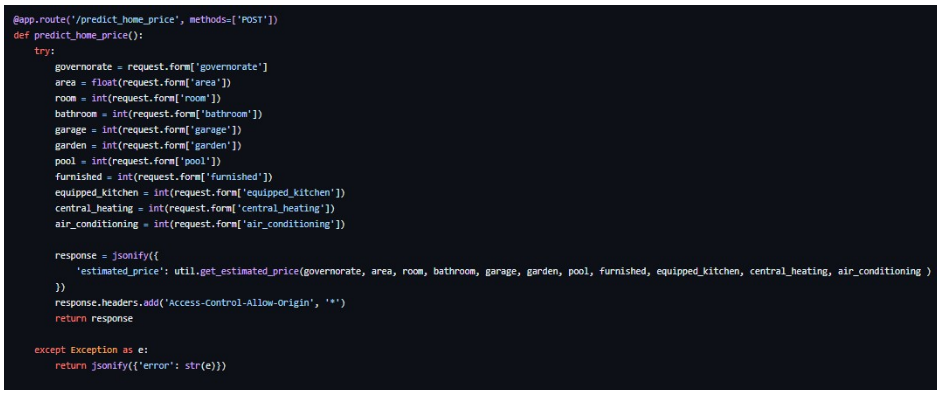
The second route, "/predict-home-price", is a POST request that takes in user input in the form of a JSON object and returns a JSON object containing the predicted house price. The input data includes the governorate, area, number of rooms, number of bathrooms, garage capacity, garden size, pool availability, whether the house is furnished, equipped kitchen availability, central heating availability, and air conditioning availability.

Inside the "predict-home-price" route function, we extract the user input from the request using the request.form method. We then pass the input to the util.get-estimated-price function, which preprocesses the data, loads the trained model from a pickle file, and makes a prediction using the input data.

Finally, we return the predicted price as a JSON object in the response.

The last few lines of code check if the module is being run directly (as opposed to being imported by another module) and starts the Flask server on the default port (5000) if it is.

Overall, the code demonstrates the use of Flask to build a web application that integrates machine learning to provide a prediction service based on user input.



```python
@app.route('/predict_home_price', methods=['POST'])
def predict_home_price():
    try:
        governorate = request.form['governorate']
        area = float(request.form['area'])
        room = int(request.form['room'])
        bathroom = int(request.form['bathroom'])
        garage = int(request.form['garage'])
        garden = int(request.form['garden'])
        pool = int(request.form['pool'])
        furnished = int(request.form['furnished'])
        equipped_kitchen = int(request.form['equipped_kitchen'])
        central_heating = int(request.form['central_heating'])
        air_conditioning = int(request.form['air_conditioning'])

        response = jsonify({
            'estimated_price': util.get_estimated_price(governorate, area, room, bathroom, garage, garden, pool, furnished, equipped_kitchen, central_heating, air_conditioning )
        })
        response.headers.add('Access-Control-Allow-Origin', '*')
        return response

    except Exception as e:
        return jsonify({'error': str(e)})
```

Figure 7: predict-home-price with POST method

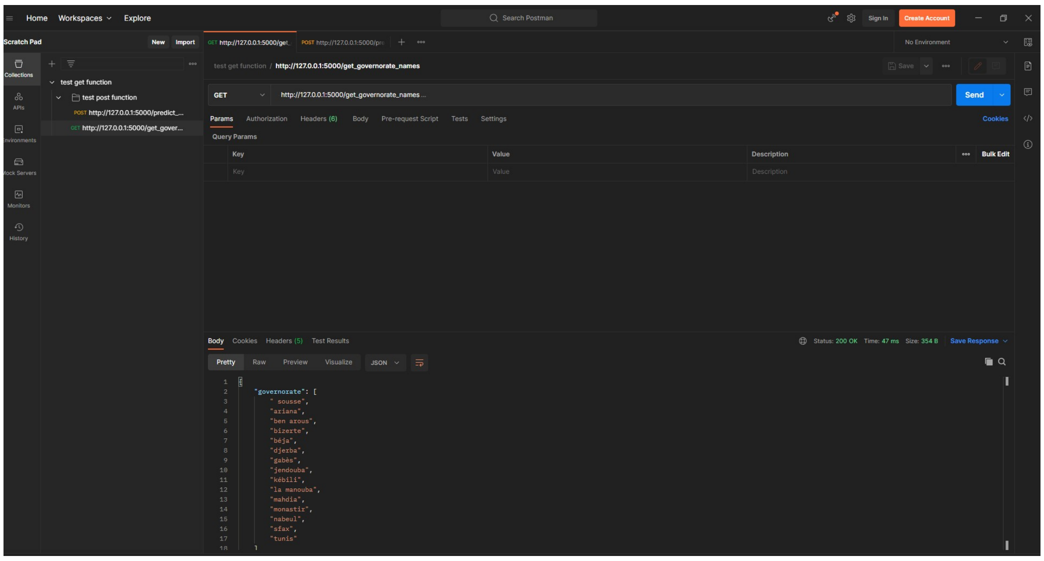
4.2 Testing the Server using Postman

To ensure that the Flask server is correctly handling requests and returning the expected responses, we can use a tool called Postman. Postman is a popular API development tool that makes it easy to send HTTP requests and test responses.

Using Postman, we can test various endpoints of the server and verify that the server is correctly processing our requests and returning the expected data. For instance, we can send a POST request to the /predict-home-price endpoint with the required parameters, and check if the server is returning the estimated price as expected.

Postman also provides features like request history, automated testing, and response visualization that make it a valuable tool for debugging and testing APIs. By using Postman, we can ensure that our server is working correctly and providing accurate responses to client requests.

In summary, Postman is an important tool for testing APIs and verifying that the server is handling requests and returning data as expected. It allows us to test various endpoints, parameters, and responses, and provides valuable features for debugging and testing our server.



test get function
test post function
POST http://127.0.0.1:5000/predict_...
GET http://127.0.0.1:5000/get_gover...

test get function / http://127.0.0.1:5000/get_governorate_names

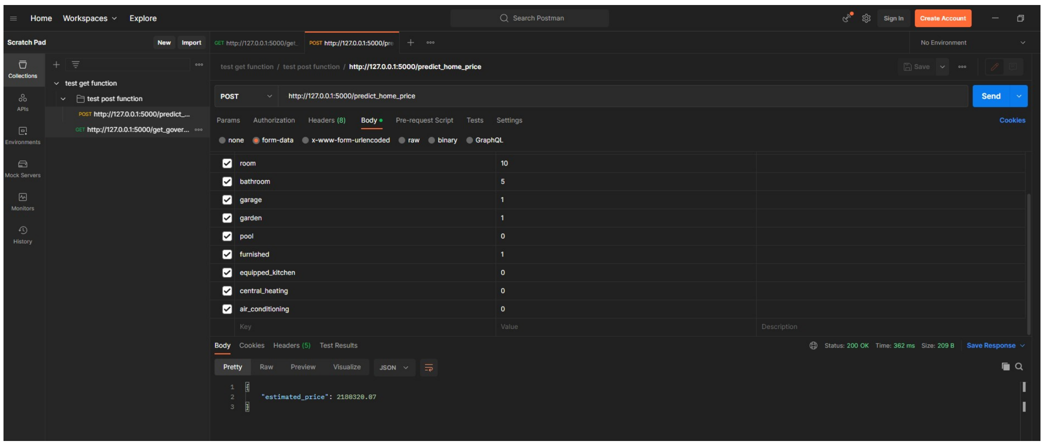
GET http://127.0.0.1:5000/get_governorate_names ...

Key	Value	Description
Key	Value	Description

Status: 200 OK Time: 47 ms Size: 354 B

```
{
    "governorate": [
        " sousse",
        "ariana",
        "ben arous",
        "bizerte",
        "béja",
        "djerba",
        "gabès",
        "jendouba",
        "kébili",
        "la manouba",
        "mahdia",
        "monastir",
        "nabeul",
        "sfax",
        "tunis"
    ]
```

Figure 8: Receive POST request from server



test get function / test post function / http://127.0.0.1:5000/predict_home_price

POST http://127.0.0.1:5000/predict_home_price

none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

Key	Value	Description
✓ room	10	
✓ bathroom	5	
✓ garage	1	
✓ garden	1	
✓ pool	0	
✓ furnished	1	
✓ equipped_kitchen	0	
✓ central_heating	0	
✓ air_conditioning	0	

Body Cookies Headers (5) Test Results — Status: 200 OK Time: 362 ms Size: 209 B

```
{
    "estimated_price": 2180320.07
}
```

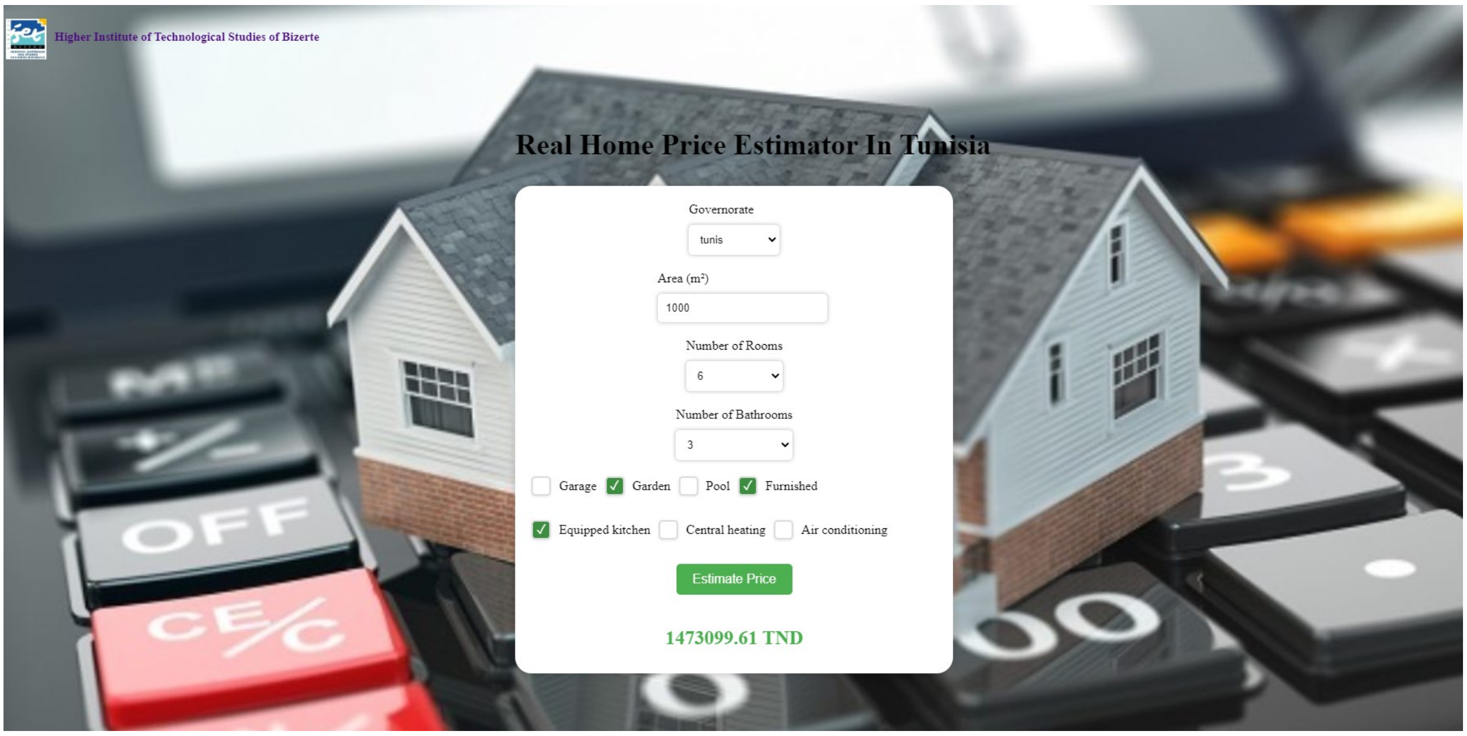
Figure 9: Receive GET request from server

4.3 front-end : user interface

The user interface is built using HTML, CSS, and JavaScript. The HTML and CSS files define the structure and style of the webpage, while the JavaScript code handles the user input and makes requests to the Flask server.

The important lines of code in the JavaScript file include:

- `var governorate = document.getElementById("uigovernorate").value;` This line gets the selected value of the governorate dropdown menu in the HTML file.
- `$.post(url, {...}, function(data, status) { ... });` This line sends a POST request to the Flask server using the jQuery library. The request includes the form data entered by the user, which is passed as a JSON object in the second argument. The third argument is a callback function that handles the server's response.
- `estPrice.innerHTML = "<h2>" + data.estimated_price.toString() + " TND</h2>"` This line sets the innerHTML of an HTML element with the ID ”uiEstimatedPrice” to display the estimated price returned by the Flask server. The estimated price is retrieved from the server's response, which is passed as the first argument of the callback function.



Real Home Price Estimator In Tunisia

Governorate: tunis

Area (m²): 1000

Number of Rooms: 6

Number of Bathrooms: 3

☐ Garage ✓ Garden ☐ Pool ✓ Furnished

✓ Equipped kitchen ☐ Central heating ☐ Air conditioning

1473099.61 TND

Figure 10: Web application for user interface

These lines of code are crucial to the functionality of the user interface, as they allow the user to input data and receive an estimated price based on that input.

5 Conclusion

In conclusion, we have developed a machine learning model to predict house prices in Tunisia based on various features such as area, number of rooms, bathrooms, etc. We have also created a Flask web application to deploy the model, and a user interface to make it easy for users to interact with the application. The web application allows users to enter the necessary input values for the model and receive an estimated price prediction based on those values. This project demonstrates the practical applications of machine learning in the field of real estate, and shows how it can be used to provide valuable insights to homeowners, real estate agents, and potential buyers.

6 Additional Resources

The source code and associated files for this project can be found on GitHub at the following address: `https://github.com/IMHAMEDBOUJEMAA/Master_Course_1_2/tree/main/10_projects/1_House_prediction_price_in_tunisia`. You can also find more information about me on my LinkedIn profile: `https://www.linkedin.com/in/imhamed-boujemaa-599ba7223/`.