Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Analysing property sales data using Data Science

Author:
Wenxiang Luo

Supervisor:
Chiraag Lala

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing of Imperial College London

June 2022

Abstract

Your abstract.

Acknowledgments

Comment this out if not needed.

Contents

Chapter 1

Introduction

Nowadays, there is a substantial amount of data generated every second. The daily lives of humans are producing it, and some other fields, such as research, health care, economic activities, and environmental information from various sensors, also generate a vast amount of data. Obtaining the relationship between some features or the patterns underlying these massive amounts of data might benefit the entire world. For instance, new causes of diseases might be identified, and technological advancement could be accelerated.

However, this extensive data can be one of the main obstacles for analysis as it is approximately impossible for humans to obtain insights into the data manually. Under this circumstance, artificial intelligence (AI), a technique that empowers the computer to imitate human intelligence and manner, could be one of the methods to mitigate this issue. It can extract patterns from large datasets and use them to make predictions based on future data and even identify which data components are responsible for the results.

In this project, some AI techniques will be applied to property and demographic data to gain insights and understand the factors influencing a homeowner's likelihood to sell. The factors might include the proximity to schools, hospitals, or supermarkets, the accessibility to public transportation, and the property types (flats or houses). It could be highly advantageous to estate agents who would discover homeowners with more potential to become clients and provide them with business.

Chapter 2

Literal Review

Python is one of the most popular programming languages in the world since it is simple to develop, and there are extensive packages for various functionalities. In this project, Python and its packages would be used for loading data, preprocessing data, and constructing and evaluating machine learning models.

2.1 Machine Learning

Machine Learning (ML), a subset of AI, is a technique that the computer can learn and improve from data without explicit programming. The reason for utilizing ML is that its performance is sometimes better than the conventional approach. For example, ML techniques would simplify the solution to a problem that comprises a long list of rules (spam mail detection).

ML can be divided into three categories, one of which is supervised learning. In supervised learning, the dataset contains features (input to the model) and targets (ground truth of the output), and the model's parameters are randomly initialized. Then the features are passed to the model, and the differences between the current output and the ground truth are used to update the parameters until the differences are acceptable.

In this project, a supervised learning model will be implemented for data analysis, and the steps are listed below.

1. Data Preprocessing: Some data from the dataset may be missing, and these values must be handled appropriately before being passed to the model.

2. Standardization: In real life, different features usually have different ranges, and this will cause a problem in ML, which is that high magnitude features would have more weight than low magnitude features (Fandango, 2017). One of the solutions is standardization, which could scale all the features to the same magnitude.

3. Feature encoding: ML models require numerical values, whereas the categorical features in the dataset do not satisfy this requirement. Therefore, these features should be converted into numerical values.

4. Training & Testing: The parameters of the model are updated, and it is expected that the loss will converge during training. The performance of the model is validated when testing.

2.2 Handle Text Information

The format of texts in this project could be classified as HTML and plain text. Although Python standard libraries provide some string processing capabilities, they are insufficient for this situation. Therefore, ***Beautiful Soup*** and ***parse***, two third-party packages, will be used to extract information from multiple texts.

2.2.1 Beautiful Soup

Beautiful Soup is a Python library for extracting data from markup languages, such as HTML and XML. It can accomplish this with a user-specified parser, for example, ***html.parser***, to navigate, search and modify the parse tree, which would save considerable time (Richardson, 2007).

In this project, this library will be used primarily to obtain a property's room information, including names and dimensions. In addition, other information in HTML format will be processed to acquire other features, such as council tax band and price.

2.2.2 parse

The ***format*** function in the Python standard library formats a string, whereas the ***parse*** module provides functions with an oppositse effect, i.e., extract information. In this project, this module will be applied to acquire information such as numerical values from various strings.

2.3 NumPy

Numerical Python (***NumPy***) is a scientific computing package utilizing an optimized C/C++ API to reduce computation time compared to pure Python computations (McKinney, 2012). It provides some data structures, one of the most important structures is ***ndarray***. Unlike Python lists, NumPy arrays are homogenous, meaning all the elements in them are of the same type, and their sizes are constant (Fandango, 2017). In addition, NumPy provides various built-in functions for a variety of purposes, including statistics, linear algebra, transforms, and element-wise operation.

In this project, ***Numpy*** will be used for data preprocessing. For example, converting categorical data into integers.

2.4 Pandas

The ***pandas*** is a fast and compelling package capable of handling data of various types (numerical values, strings, and time) and from multiple sources (CSV, Excel, and MqSQL database). One of the ***pandas*** data structures is ***DataFrame*** which is appropriate for handling tabular data with columns of different types. In addition, it could manage various operations, such as manipulating missing values, creating pivot tables (table summaries), and grouping data from different columns (Fandango, 2017).

Pandas would be used for data loading and preprocessing in this project. Data is initially loaded and represented as a ***DataFrame***. Then the attributes of the data, such as distribution, should be inspected. Finally, the data is preprocessed.

2.5 Scikit-learn

Scikit-learn is a Python ML package that provides various techniques for data mining, modeling, and analyzing. It could be commonly used in multiple ML tasks, such as classification, regression, and clustering. This project will use the package to analyze the performance of the models.

2.6 PyTorch

PyTorch was developed by Facebook, and it is one of the popular libraries for constructing ML models. It is also the basis of numerous packages that the developers could take advantage of (Godoy, 2021). In addition, this library is capable of performing auto differentiation by using graphic processing unit (GPU) acceleration, resulting in a reduction in training time.

In this project, some of the ***PyTorch*** components will be utilized.

- ***nn.Sequential*** is the container that contains all the layers of the ML model
- An appropriate loss function would be selected to compute the loss and perform backpropagation.
- One of the optimizers would be used to update the model's parameters.

2.7 Matplotlib

Matplotlib is one of the Python plotting packages for plotting different types of figures, including histograms, pie plots, and line plots. This library will be utilized primarily for data visualization in this project. For instance, the distribution of the raw data should be visualized to determine the procedures of preprocessing and plotting the training loss and validation loss during model evaluation.

Chapter 3

Project Plan

3.1 Data Preprocessing

3.1.1 Handling Missing Values

In this project, two methods will be attempted to preprocess the missing values.

1. Drop missing values: This method deletes the messing values for data analysis.

2. Fill the missing values: In this approach, the missing values are filled with mean, median, or default values, for example, zero or some constants.

3.1.2 Standardization

In this project, two different approaches will be attempted.

1. Standard scaling: By applying this method, the mean of the feature is removed and then divided by the standard deviation.

2. Min-max scaling: For this approach, the minimum and maximum of the raw data are used to transform it into a specific range.

3.1.3 Feature Encoding

In this project, label encoding will be applied for this goal. It would convert the categorical values into a sequence of integer values.

3.1.4 Separating Dataset

The original dataset should be shuffled and divided into three parts which will be used for training, validation, and testing.

3.2 Build Model with PyTorch

3.2.1 Model architecture

- **Basic Model**: The number of the input neural of the model should be the same as the number of features, and then there are hidden layers. Finally, the output layer is a sigmoid function as this model should output the possibility of the selling.
- **Advanced Model**: The advanced model contains more components, such as dropout or batch normalization, and these additional layers might reduce overfitting and enhance accuracy.

3.2.2 Training

1. Use ***DataLoader*** to fetch batches of data for training, and the batch size could be set to 128 if there is sufficient memory.
2. The loss function used for model training is ***MSELoss*** since this is a regression problem.
3. The optimizer for updating parameters could be ***SGD*** or ***Adam***.
4. Train the model iteratively, and the number of iterations (epochs) will be set to 500 and updated if necessary.

3.2.3 Hyperparameters Tuning

- Learning rate: The initial learning rate will be set to 0.001, and it will be increased/decreased depending on the performance. Moreover, learning rate schedulers, for example, ***LambdaLR*** and ***StepLR***, could be applied during training.
- The number of epochs: If the initial value causes overfitting or underfitting, then the epochs should be decreased or increased.
- Activation functions: The initial activation function will be ReLU, but other activation functions, such as TanH and Parametric ReLU, will also be tested to enhance the performance.

3.3 Evaluation

After constructing and training models, the next step is to evaluate their performance. The mean squared error (MSE) could be used to evaluate performance numerically. The lower the MSE, the higher the performance. In addition, a line plot containing the training loss and validation loss against epochs should be produced.

This figure could determine if the model is overfitting/underfitting. The optimal best model is then selected and tested using the test dataset to generate the final result.

Chapter 4

Implementation

4.1 Data Preprocessing

The data that was retrieved for this project originated from two different sources: a raw dataset and online APIs.

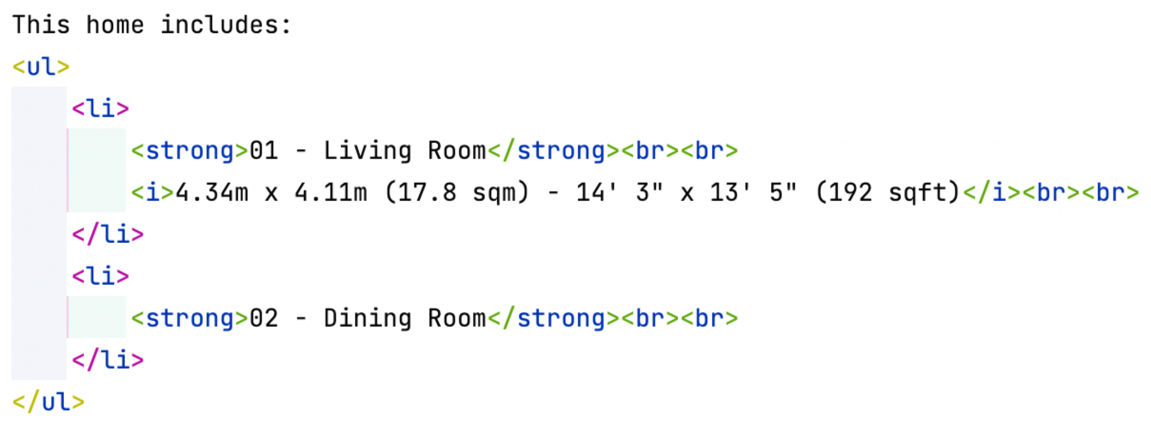
4.1.1 Extract values

There are two types of data in the raw dataset: HTML texts containing the name and area of a property's rooms and categorical keywords describing a particular feature.

Handle Room Descriptions (HTML)

Acquire room name and dimension

The structure of the HTML texts containing the room information in a property is shown in figure 4.1. The rooms are separated by tag <*li*>, and the room name and its dimension is denoted by <*strong*> and <*i*> tags, respectively. Therefore, a function (***EweMove_Description_S3_Rooms***) was implemented to split the HTML text by utilizing ***Beautiful soup***, and its flowchart is shown in figure 4.2.



```
This home includes:
<ul>
    <li>
        <strong>01 - Living Room</strong><br><br>
        <i>4.34m x 4.11m (17.8 sqm) - 14' 3" x 13' 5" (192 sqft)</i><br><br>
    </li>
    <li>
        <strong>02 - Dining Room</strong><br><br>
    </li>
</ul>
```

Figure 4.1: The layout of HTML texts

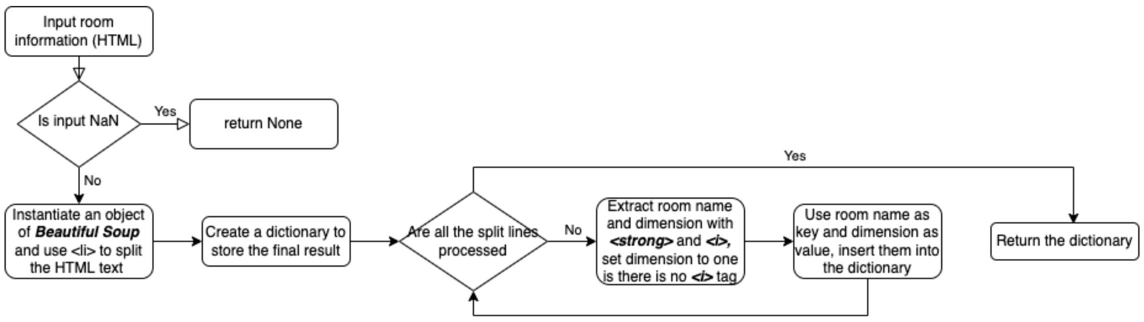

Input room information (HTML)
Is input NaN
Yes
return None
No
Instantiate an object of ***Beautiful Soup*** and use <li> to split the HTML text
Create a dictionary to store the final result
Are all the split lines processed
No
Extract room name and dimension with ***<strong>*** and ***<i>***, set dimension to one is there is no ***<i>*** tag
Use room name as key and dimension as value, insert them into the dictionary
Yes
Return the dictionary

Figure 4.2: Flowchart for extracting room information from HTML

The names of rooms are analyzed after the room descriptions in the dataset have been processed. As a consequence, there are over 200 unique room names for approximately 3600 records, some of which are exceptionally uncommon across the entire dataset. For instance, only two properties have cinema rooms and one has a lift, which is less than 0.1% of all entries.

Due to the large number of room names, it is impossible to use it as the input of the model. Therefore, the rooms are divided into seven categories: bedrooms, bathrooms, kitchens, living/reception rooms, dining rooms, work areas, and other rooms so that the data can be generalized.

Generalize room information

A class, ***ExtractRooms***, was developed to acquire and integrate the room information, especially the area in square meters, and its UML diagram is shown in figure 4.3. The member variable *rooms* is a list containing the result of invoking *EweMove_Description_S3_Rooms*, *room_set* comprises all the room names, *current_rooms* consists of the room names that have been processed, and *extract_area* is a formatted string for acquiring room area.

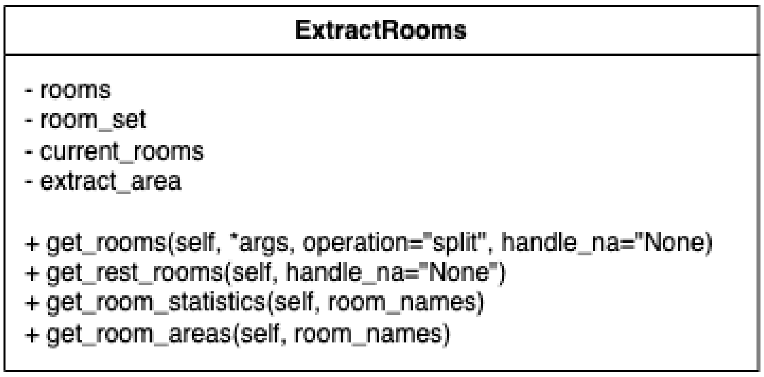

ExtractRooms
- rooms
- room_set
- current_rooms
- extract_area
+ get_rooms(self, *args, operation="split", handle_na="None)
+ get_rest_rooms(self, handle_na="None")
+ get_room_statistics(self, room_names)
+ get_room_areas(self, room_names)

Figure 4.3: The UML diagram of the class (*ExtractRooms*)

Member functions

- get_rooms

This is the core of the class, and its flow diagram is shown in figure 4.4. It should be noted that ***args*** is a variable-length argument, which means that it can accept as many arguments as possible, and it is used to select room names from *room_set*. For instance, all the names containing ”living” or ”reception” will be selected if **args = [”living”, ”reception”]*.

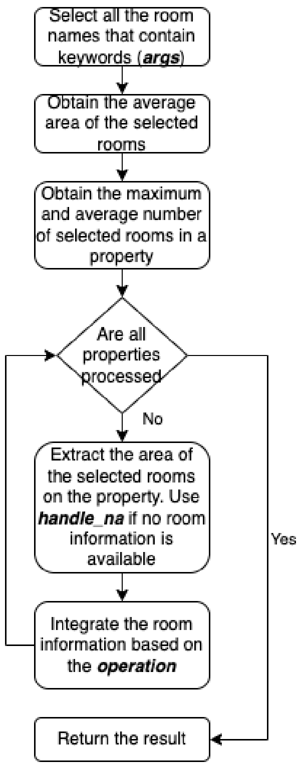

Select all the room names that contain keywords (args)
Obtain the average area of the selected rooms
Obtain the maximum and average number of selected rooms in a property
Are all properties processed
No
Extract the area of the selected rooms on the property. Use handle_na if no room information is available
Yes
Integrate the room information based on the operation
Return the result

Figure 4.4: Flowchart of *get_rooms*

Chapter 5

Testing the Implementation

Chapter 6

Performance

Chapter 7

Experimental Results

Chapter 8

Conclusion

Bibliography

Fandango, A. (2017). *Python Data Analysis*. Packt Publishing Ltd. pages 2, 3, 4

Godoy, D. V. (2021). *Deep Learning with PyTorch Step-by-Step: A Beginner's Guide*. pages 4

McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.". pages 3

Richardson, L. (2007). Beautiful soup documentation. *Dosegljivo: https://www. crummy. com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018]*. pages 3