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.

1.1 Aim & Objective

1.2 Layout of the Report

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 plaintext. Although Python standard libraries provide some string processing capabilities, they are insufficient for this situation.

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).

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 from formatted strings.

2.2.3 Regular Expression

A regular expression is a sequence of ordinary and special characters representing textual patterns. The ordinary characters are identical throughout the expressions and texts, while the special characters specify the pattern, including number, location, and type of characters (Stubblebine, 2007). One of the primary disadvantages of the regular expression is its obscure syntax, which results in difficulty specifying a pattern.

2.2.4 Library Usage

In this project, HTML texts are used extensively in the raw dataset to describe property summaries, property layouts, and council tax. This is the optimal scenario for *Beautiful Soup* which is employed to extract plaintext by specifying tags. Then, *parse* is applied to obtain the information, such as room names, from the plaintexts since they are in the same format. In addition, due to its limitations, the regular expression is only used to acquire numerical values in this project.

2.3 Data Manipulation

2.3.1 NumPy

Numerical Python (***NumPy***) is a scientific computing package designed to support large multidimensional matrices. It uses an optimized *C/C++* API to reduce computation time compared to pure Python computations (McKinney, 2012). A substantial number of complex tasks of data analytics can be simplified by numerous *numpy* features. For example, it provides robust matrix operations, facilitates the construction of multidimensional objects, and serves as the foundation of other packages, including *matplotlib* and *seaborn*.

2.3.2 Pandas

The ***pandas*** is an open-source and compelling package that was developed primarily for data analysis and data manipulation and is built on *numpy*. It is capable of handling data of various types (numerical values, strings, and time) and from a variety of sources (CSV, Excel, and MqSQL database). ***DataFrame*** is one of the *pandas* data structures that is appropriate for handling tabular data with columns of different types. Additionally, it could manage various operations, such as manipulating missing values, creating pivot tables, and grouping data from different columns (Fandango, 2017).

2.3.3 Library Usage

In this project, the dataset provided is in CSV format hence it could be loaded by ***Pandas*** since it is suitable for tabular data. Then the package is utilized for preprocessing, such as handling missing values and grouping columns of data.

Numpy is appropriate for manipulating numerical data and acts as an intermediary between various packages. Therefore, it could be employed to evaluate the performance of ML models and transmit data to plotting packages.

2.4 ML Frameworks

2.4.1 Scikit-learn

Scikit-learn is a popular open-source ML framework that employs *Numpy*. It contains traditional ML algorithms, including clustering, classification, and regression, as well as a variety of utilities that can be applied to preprocess data and evaluate the performance (Géron, 2019). The drawback of this library is that it does not natively support GPU acceleration and is not a neural network framework.

2.4.2 PyTorch

PyTorch is one of the popular ML frameworks developed by Facebook, which is designed to implement neural networks with flexibility and speed (Godoy, 2021). It provides various components for model construction and training. For instance, there are numerous types of modules that comprise a model, such as linear layers, dropout, and activation functions, as well as a variety of loss functions and optimizers that can be employed in model training.

Furthermore, it can be beneficial to construct and train a model with *Pytorch*. It has a Pythonic nature which means that its syntax is similar to Python, making it more straightforward for Python programmers to develop neural networks than other ML frameworks. Moreover, it is a rapidly expanding framework for developing neural networks with a vast ecosystem, meaning that a substantial number of utilities have been developed on top of it (Godoy, 2021). Additionally, *PyTorch* supports automatic differentiation and GPU acceleration which can be advantageous for model training.

2.4.3 TensorFlow

TensorFlow is another ML framework produced by Google that specializes in deep learning and neural networks. It provides approximately the same components as *PyTorch* and also supports automatic differentiation and GPU acceleration. One of the appealing characteristics of *TensorFlow* is called ***TensorBoard***, which is an interactive visualization system that can display the flowchart of the data manipulation and plot the tendency of the performance (Shukla and Fricklas, 2018).

2.4.4 Library Usage

This project aims to construct a neural network which means *scikit-learn* is not applicable at this stage. Although *TensorFlow* provides the same capabilities as *PyTorch* and is superior in visualization, the model construction and training will use *PyTorch* due to its Pythonic syntax and compatibility with *TensorBoard*.

However, *scikit-learn* can be used to preprocess datasets and evaluate performance. It provides various utilities that can be helpful before training, for example, encoding categorical features and splitting the dataset into training and validation. In addition, it offers features for model evaluation, such as confusion matrix, accuracy, and recall.

2.5 Data Visualization

2.5.1 Matplotlib

Matplotlib is a Python package for 2D plotting that produces high-quality figures. It supports interactive and non-interactive plotting and can save images in multiple

formats, including PNG and JPEG. It can also generate numerous types of graphs, such as line plots, scatter plots, and pie plots.

2.5.2 Seaborn

Seaborn is a Python library for creating statistical graphs that integrates with *pandas* to offer a high-level interface to *matploblib*. If a dataset is provided, *seaborn* can automatically generate the figure with appropriate plotting attributes, such as color and legend. Additionally, it is capable of generating comprehensive graphics with a single function call and a minimum number of arguments (Waskom, 2021).

2.5.3 Library Usage

In this project, data visualization would be beneficial during preprocessing data and performance evaluation. For preprocessing, the distribution of the raw data should be inspected, hence *seaborn* could be an optimal choice since the input is a *DataFrame* and its syntax is concise. During evaluation, the model output will be converted to *Numpy* arrays. Therefore, *matplotlib* can be used in this case, as it is interactive and the figure can be further adjusted to illustrate the performance.

2.6 Methodology

Chapter 3

Implementation

3.1 Data Preprocessing

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

3.1.1 Extract Values

There are two types of data in the raw dataset: HTML texts containing the property information and categorical keywords describing features, such as parking spaces and heating.

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 3.1. The rooms are separated by tag *<li>*, and the room name and its dimension is denoted by ** 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 3.2.

```
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 3.1: The layout of HTML texts

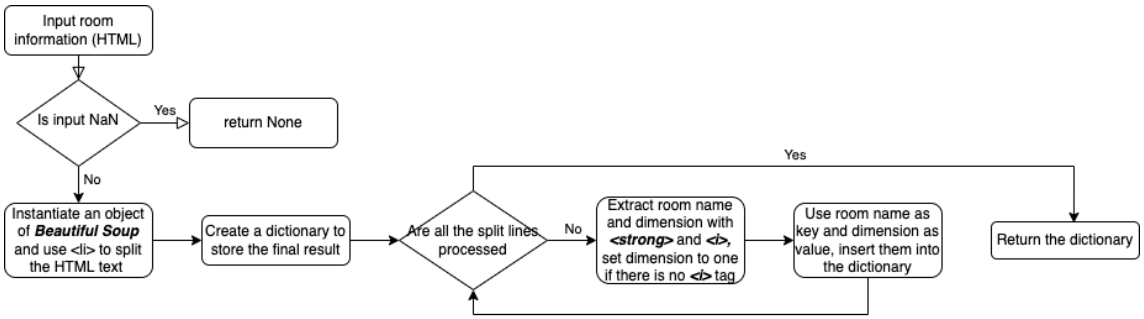


Figure 3.2: Flowchart for extracting room information from HTML

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 3.3. 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.

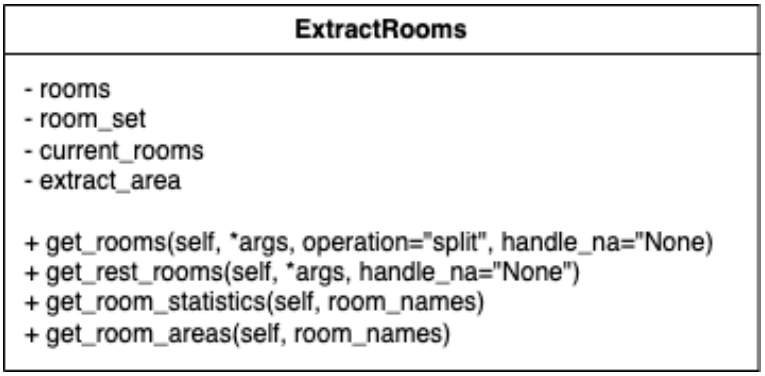


Figure 3.3: The UML diagram of the class (*ExtractRooms*)

Key member functions

- get_rooms

The flow diagram of this method is shown in figure 3.4 and the arguments are listed below.

1. args: It should be noted that this 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, **args = ["living", "reception"]* will select all names containing "living" or "reception".
2. operation: The argumen determines the types of the final result and the valid inputs include *"sum"*, *"mean"*, *"split"*, and *"number"*. For example, if args is "bedroom", then the function can return the sum of bedroom areas, the average bedroom area, the area of each bedroom and the number of bedrooms.
3. handle_na: This parameter specifies how to manage missing values, either by ignoring them or by filling the mean value if the input is "None" or "mean", respectively.

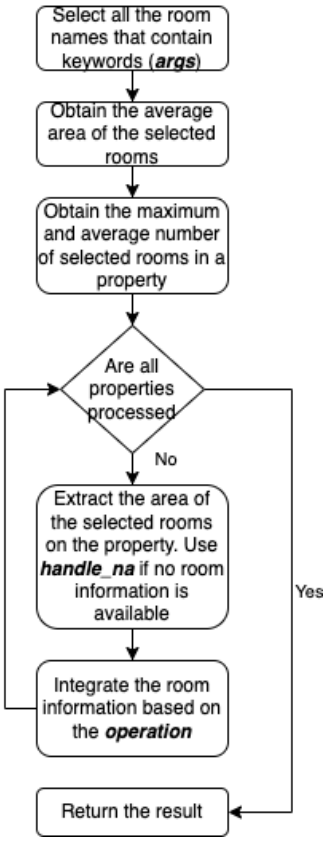


Figure 3.4: Flowchart of *get_rooms*

- get_rest_room:
  This method is identical to ***get_room*** with two exceptions. The parameter ****args*** is used to disgard the room names containing the keywords, and only the number and the total areas of other rooms are returned.

3.1.2 Manipulate Categorical Keywords

In the dataset, four features are characterized by categorical keywords, including parking, heating, accessibility, and outdoor spaces. Figure 3.5 is a snippet of the parking dataset that is used as an example to demonstrate the manipulation of the keywords. The first three rows indicate that there is one parking space for the first property, which is allocated off-street parking for residents, the second row has none, and the third property has one on-street parking space.

RTD3307_parking1 - Parking Description	RTD3307_parking2 - Parking Description	RTD3307_parking3 - Parking Description
Allocated	Off Street	Residents
On Street		
Driveway	Garage	Off Street
Driveway	Garage	Off Street
Driveway		
Driveway	Garage	
Garage	Driveway	

Figure 3.5: The parking spaces in the first five properties

Figure 3.6 illustrates the UML diagram of the class ***GeneralizeDataset***, which was developed to determine how these features of each property are described and the number of keywords within the description. The core of this class is member function ***get_feature_types***, and its flowchart is displayed in figure 3.7. In addition, function ***get_feature_num*** can be used to determine the number of keywords for each property.

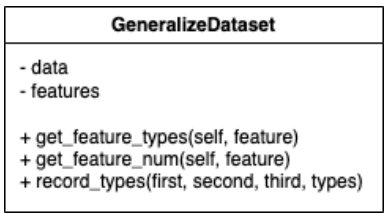


Figure 3.6: The UML diagram of *GeneralizeDataset*

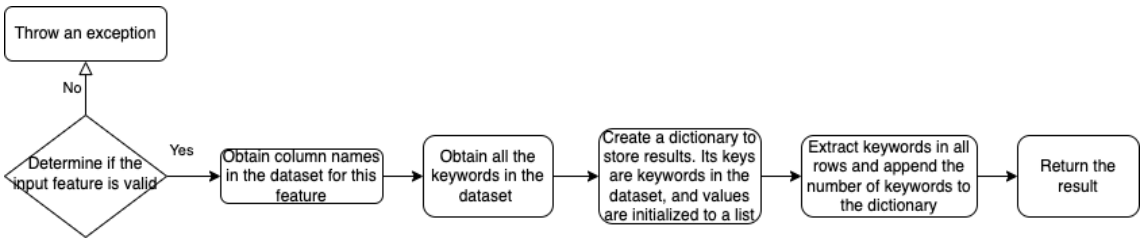
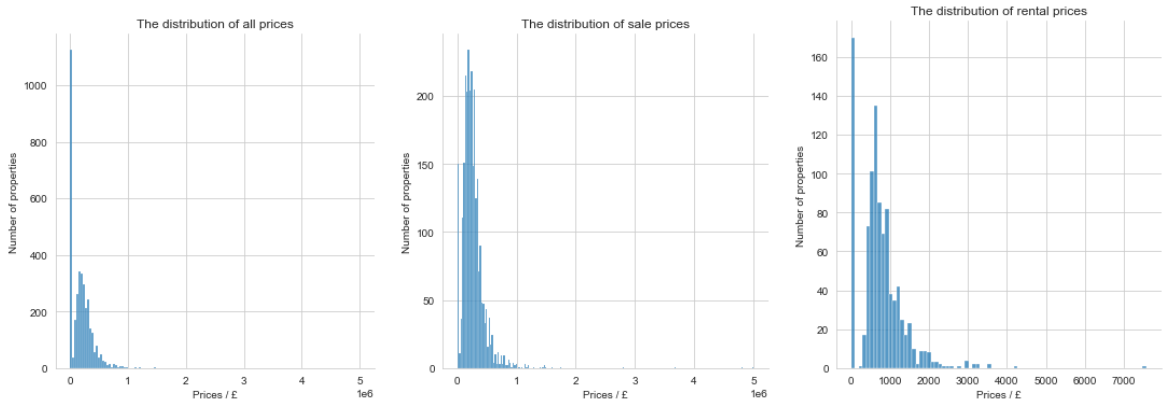


Figure 3.7: Flowchart of *get_feature_types*

3.1.3 Manage Outliers

The price is crucial to the project, and the distribution of the raw data was inspected, as shown in figure 3.8. The spike on the left side of the figure indicates that numerous properties have zero prices, which is illogical. Therefore, these abnormal values should be addressed as they could significantly impact the performance of the model.



(a) All Prices (b) Sale Prices (c) Rental Prices

Figure 3.8: The distribution of prices from raw data

Chapter 4

Testing the Implementation

Comprehensive testing was conducted throughout the implementation but it is documented in this separate chapter for the sake of illuatration.

4.1 Data Preprocessing

4.1.1 Extract Values

Obtaining Room Information

There were two tests for this objective, the first test examing if the room name and its dimension can be acquiare from HTML texts and the second test focuses on whether the room areas can be obtained and integrated correctly.

Acquire room name and dimension
This test utilized two HTML texts, which are illustrated in figure 4.1. By invoking the function ***EweMove_Description_S3_Rooms*** with two snippets, it can be verified that all the room names and dimensions can be extracted without error, and if the dimension is unavailable, it is set to one.

Generalize room information
The information obtained from HTML texts (figure 4.1) was used to access the behavior of *ExtractRooms*, especially its member functions, *get_rooms*. During testing, the selected type is the bedroom, and all the operations of *get_room* were inspected. Additionally, the other rooms were used to test *get_rest_rooms*.

It is guaranteed that if the operation is ”split”, then all the bedroom areas can be listed; the sum and average of areas as well as the number of bedrooms will be returned if the function is invoked with corresponding parameters. In addition, the number of other rooms and their total area will be computed correctly.

```
This home includes:
<ul>
    <li>
        <strong>01 - Entrance Hall</strong><br><br>
    </li>
    <li>
        <strong>02 - Living/Dining Room</strong><br><br>
        <i>6.58m x 3.78m (24.8 sqm) - 21' 7" x 12' 4" (267 sqft)</i><br><br>
    </li>
    <li>
        <strong>03 - Kitchen</strong><br><br>
        <i>2.68m x 2.14m (5.7 sqm) - 8' 9" x 7' (61 sqft)</i><br><br>
    </li>
    <li>
        <strong>04 - Bedroom 1</strong><br><br>
        <i>3.37m x 2.45m (8.2 sqm) - 11' x 8' (88 sqft)</i><br><br>
    </li>
    <li>
        <strong>05 - Bedroom 2</strong><br><br>
        <i>2.54m x 2.45m (6.2 sqm) - 8' 4" x 8' (67 sqft)</i><br><br>
        The second double bedroom is bright and well-sized, with room for all required furniture.<br><br>
    </li>
    <li>
        <strong>06 - Bathroom</strong><br><br>
        <i>2.14m x 2.04m (4.3 sqm) - 7' x 6' 8" (46 sqft)</i><br><br>
    </li>
    <li>
        <strong>07 - Garden</strong><br><br>
        Communal Gardens.<br><br>
    </li>
    <li>
        <strong>08 - Parking</strong><br><br>
        2 allocated parking spaces.<br><br>
    </li>
</ul>
```

(a) Snippte 1

```
This home includes:
<ul>
    <li>
        <strong>01 - Entrance Porch</strong><br><br>
    </li>
    <li>
        <strong>02 - Lounge Diner</strong><br><br>
        <i>6.76m x 4.04m (27.3 sqm) - 22' 2" x 13' 3" (293 sqft)</i><br><br>
    </li>
    <li>
        <strong>03 - Kitchen</strong><br><br>
        <i>2.97m x 2.36m (7 sqm) - 9' 8" x 7' 8" (75 sqft)</i><br><br>
    </li>
    <li>
        <strong>05 - Bathroom</strong><br><br>
    </li>
    <li>
        <strong>07 - Bedroom (Double)</strong><br><br>
        <i>4.05m x 3.25m (13.1 sqm) - 13' 3" x 10' 7" (142 sqft)</i><br><br>
    </li>
    <li>
        <strong>08 - Bedroom (Double)</strong><br><br>
        <i>3.28m x 2.36m (7.7 sqm) - 10' 9" x 7' 8" (83 sqft)</i><br><br>
    </li>
    <li>
        <strong>09 - Bedroom (Double)</strong><br><br>
        <i>4.3m x 2.44m (10.4 sqm) - 14' 1" x 8' (112 sqft)</i><br><br>
    </li>
    <li>
        <strong>10 - Bathroom</strong><br><br>
    </li>
</ul>
```

(b) Snippet 2

Figure 4.1: The HTML snippets for testing

4.1.2 Manipulate Categorical Keywords

The data utilized for this test is displayed in figure 3.5. Initially, an invalid feature, ”Distance to School” was input into function *get_feature_types*, and it can be assured that an exception would be thrown. The feature was then set to ”parking”, and the result should be a table with ten rows and six columns, as there are ten properties in the snippet and six keywords in total. In addition, it could be verified that *get_feature_num* will return the number of keywords for each property when invoked.

4.1.3 Manage Outliers

During preprocessing, both filling the mean value and removing outliers were attempted. It can be assured that the spike on the left side will decrease significantly in both methods. For filling the average prices, there will be a peak at the mean values, and the original price distribution will be altered. If they are removed, then the original distribution will be retained.

Chapter 5

Experimental Results

5.1 Data Preprocessing

5.1.1 Extract Values

Obtaining Room Information

Acquire room name and dimension

After calling the function and retrieving the results (shown in tables 5.1 and 5.2), it is evident that the room names in two HTML snippets could be obtained, and the dimensions were acquired if available otherwise, the value was set to one, hence this function can pass the test.

Table 5.1: Information from HTML snippet in figure 4.1(a)

Entrance Hall	1
Living/Dining Room	6.58m x 3.78m (24.8 sqm) - 21’ 7” x 12’ 4” (267 sqft)
Kitchen	2.68m x 2.14m (5.7 sqm) - 8’ 9” x 7’ (61 sqft)
Bedroom 1	3.37m x 2.45m (8.2 sqm) - 11’ x 8’ (88 sqft)
Bedroom 2	2.54m x 2.45m (6.2 sqm) - 8’ 4” x 8’ (67 sqft)
Bathroom	2.14m x 2.04m (4.3 sqm) - 7’ x 6’ 8” (46 sqft)
Garden	1
Parking	1

Generalize room information

The results of calling *get_rooms* are displayed in tables 5.3 and 5.4. It is obvious that all the bedrooms and their areas in tables 5.1 and 5.2 were successfully extracted. In addition, the numerical values could also be obtained without error if the operations were configured appropriately. These behaviors demonstrated that the functionality and the design are identical.

Furthermore, the result of invoking *get_rest_rooms* is shown in table 5.5. It is clear that there is no statistical inconsistency using the information from tables 5.3, 5.1, and 5.2, hence this function can pass the test.

Table 5.2: Information from HTML snippet in figure 4.1(b)

Entrance Porch	1
Lounge Diner	6.76m x 4.04m (27.3 sqm) - 22’ 2” x 13’ 3” (293 sqft)
Kitchen	2.97m x 2.36m (7 sqm) - 9’ 8” x 7’ 8” (75 sqft)
Bathroom	1
Bedroom (Double)	4.05m x 3.25m (13.1 sqm) - 13’ 3” x 10’ 7” (142 sqft)
Badroom (Double)	3.28m x 2.36m (7.7 sqm) - 10’ 9” x 7’ 8” (83 sqft)
Bedroom (Double)	4.3m x 2.44m (10.4 sqm) - 14’ 1” x 8’ (112 sqft)
Bathroom	1

In conclusion, the performance of the two crucial member functions in class *ExtractRooms* meets expectations.

Table 5.3: Room information (”split”)

	Bedroom 1	Bedroom 2	Bedroom 3
0	8.2	6.2	0.0
1	13.1	7.7	10.4

Table 5.4: Bedroom information integrated by different operations

(a) Mean

	Average area
0	7.2
1	10.4

(b) Sum

	Total area
0	14.4
1	31.2

(c) Number

	Number of rooms
0	2
1	3

5.1.2 Manipulate Categorical Keywords

For the initial test, an exception was thrown if an invalid feature, in this case ”Distance to School”, was entered. Next, the result of calling function *get_feature_types* is shown in table 5.6. Apparently, the shape of the table is correct, and its elements are consistent with the dataset snippet. For the first property with an **allocated off-street** parking space for **residents**, the three keywords in the first row are set to one while the others are zero, and this conclusion holds true for the remaining properties. In addition, the number of keywords associated with each property can be obtained accurately by calling *get_feature_num*.

In summary, the member functions in this class are capable to determine the types and number of keywords, hence the implementation can pass the tests.

Table 5.5: The number and total area of other rooms

	Number	Area
0	6	34.8
1	5	34.3

Table 5.6: The keywords for each property

	Allocated	Driveway	Garage	Off Street	On Street	Residents
0	1	0	0	1	0	1
1	0	0	0	0	0	0
2	0	0	0	0	1	0
3	0	0	0	0	0	0
4	0	1	1	1	0	0
5	0	0	0	0	0	0
6	0	1	1	1	0	0
7	0	1	0	0	0	0
8	0	1	1	0	0	0
9	0	1	1	0	0	0

5.1.3 Manage Outliers

The distribution of the prices after applying two methods are shown in figure 5.1 and 5.2. For filling the average value, it is apparent that the spike is reduced, meaning that there is no price of zero. A new spike in the middle indicates that the original distribution has been modified. If zero prices were removed, the peak on the left would also be reduced, and the distribution would remain unchanged.

Therefore, removing zero prices was applied to create input datasets for models as it retains the original distribution, whereas filling the mean value causes the data to shift. Consequently, the model will learn more from average, which results in poor performance.

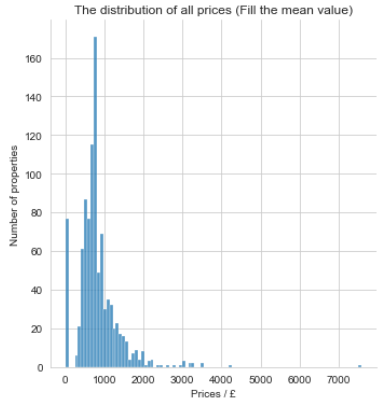
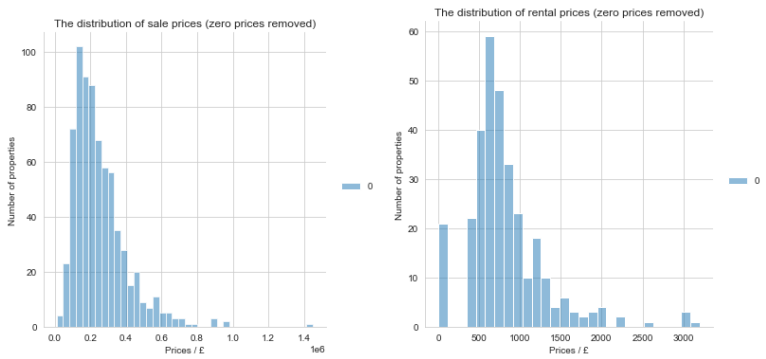


Figure 5.1: The distribution of prices with mean value filled



(a) Sale Prices (b) Rental Prices

Figure 5.2: The distribution of prices with zero prices removed

Chapter 6

Conclusion

Bibliography

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

Godoy, D. V. (2021). *Deep Learning with PyTorch Step-by-Step: A Beginner's Guide*. pages 5

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.". pages 4

McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.". pages 4

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

Shukla, N. and Fricklas, K. (2018). *Machine learning with TensorFlow*. Manning Greenwich. pages 5

Stubblebine, T. (2007). *Regular Expression Pocket Reference: Regular Expressions for Perl, Ruby, PHP, Python, C, Java and. NET*. " O'Reilly Media, Inc.". pages 3

Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021. pages 6