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

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 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 that was 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 fundation 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 both 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.

Chapter 3

Project Plan

3.1 Data Preprocessing

3.1.1 Extract information from HTML texts

HTML texts contain the property layouts and prices, and the information inside should be acquired at the beginning.

In order to obtain the property layouts, *Beautiful soup* will be used to extract plaintext by specifying tage. For instance, the room names and their dimensions are enclosed by <*li*> and <*i*>, respectively. Then the function in *parse* package should be used to extract the area from dimension strings. The prices are contained by simple HTML texts, hence the regular expressions are used for extraction.

3.1.2 Handling Missing Values

By inspecting the dataset, more than 30% of the data is missing. Therefore, these missing values should be removed. There is another method which is to fill in mean values. This is not applicable in this case since it will significantly change the distribution of original data.

3.1.3 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.

3.1.4 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.

3.1.5 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. The output of this model is a possibility of the selling and its price, which indicates that the activation functions are sigmoid and ReLU.

- **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 dataset used in this project is small, so cross validation should be applied.

3. The loss function used for model training is ***MSELoss*** since this is a regression problem.

4. The optimizer for updating parameters could be ***SGD*** or ***Adam***.

5. 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.

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