ZHIMIN ZHANG

ID Number: 4276764

Supervisor: KE ZHOU

Module Code: G53IDS

2018/04

Machine Learning for Nottingham Housing Price Prediction

Submitted April 24th 2018, in partial fulfilment of
the conditions for the award of the degree Computer Science.

ZHIMIN ZHANG

ID 4276764

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature ___ ZHIMIN ZHANG_____

Date _24_/__04_/_2018__

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive. *

Public access to this dissertation is restricted until: 24/04/2018 **

Abstract

Property transactions have long been an important theme in people's life. In addition, it is always involved with house price prediction. For example, estate transactions including housing mortgage, property owner's insurance and house sale all require the estimation of the house value. Thus, it is necessary to provide an accurate price estimation technique under these circumstances to maximize the profits of the parties involved.

To realize this goal, I have designed and implemented a set of tools to explore the whole process of using machine learning to do house price prediction based on house features such as number of bedrooms, property type and so on. The work ranges from data fetching to machine learning model training and also including a web interface aiming to demonstrate my research result. Those created tools and research conclusions, ideally, would give an insight into property prediction in the real world.

Acknowledgements

I would like to express my sincere thanks to my supervisor Zhou Ke. He has inspired me a lot during the process of the project design and implementation. Without his technique advice and continued encouragement, I would not have finished this project so successfully.

A special thank is also extended to my senior Zhen Fang for sharing his own dissertation experience. His kind reminder has helped me greatly in managing the working time. In addition, his tip on academic writing has contributed to relieve my writing anxiety.

1 Introduction

1.1 Background and Motivation

1.1.1 House price and its prediction

House is the symbol of wealth nowadays. People regard house price as an important standard to measure the economics of an area. It is common to see various trend analysis around house market. Even government sometimes releases policies to control house price. Under the circumstances, the technique to evaluating the value of a house is needed by many occasions. On one hand, more and more real estate companies and property transaction websites start to provide professional price estimation service. Similarly, real estate agents tag prices to second-hand houses for reselling. On the other hand, the technique is also needed by banks and governments. For example, banks need to appraise a house before giving a mortgage loan. Governments need it for taxation purpose [1]. Hence, it is meaningful to think about how people can do the job accurately and efficiently.

In the past, the job is usually done by professional house agents. They examine the house and predict a price depending on the information such as the location of the house, the quality of the house. But the disadvantage is also obvious. It is hard for a human to do house prediction fairly without personal preference. Moreover, it is a tedious and time-consuming work so that it is almost impossible to do it over a large scale within a short time. Hence, some new solutions have been proposed over the past few years. Doing house price prediction based on machine learning is one of the solution that has not been widely researched yet. Thus, the process of working out such an algorithm has profound and lasting implications.

1.1.2 Machine Learning

The rapid development of machine learning nowadays has subverted many industries. Complicated tasks including image or speech recognition, data mining and quantitative analysis have gradually been taken over it. Hence it is not hard to come up with the idea to solve the price prediction problem using machine learning technique.

As the SAS news said, machine learning is very much the “topic of the moment”. [2] Thanks to the constant update of computing techniques as well as the exponential growth of data volume, this field has experienced a blowout development. And it has achieved great success in computer vision, nature language processing and voice recognition fields.

Using machine learning to do this job is very feasible. Briefly speaking, machine learning model is a function of which the input is several influencing factors i.e. independent variables and it will output the result after a series of calculations depending on those factors. For example, using the simplest machine learning algorithm – basic linear regression to predict house price can be illustrated as:

$$price = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \cdots + w_n * x_n$$

Where $x_1, x_2, \dots x_n$ are different influencing factors such as house type, house size and house location and $w_1, w_2, \dots w_n$ are their weights. Factors of which the value are not number such as house type will be preprocessed into number before putting into the model. More details will be mentioned in later chapter. Considering the defects of human agents, using machine learning to do house prediction can greatly reduce interference of man-made ingredient. On the top of that, since it uses the same standard for anyone while keeping a high prediction speed, the implementation of this estimation technique can also help to fill an important information gap and improve the efficiency of the real estate market [3].

1.2 Project Aims

1.2.1 Related work

As it has been mentioned above, the traditional way to do house price prediction is by those house agents. My goal is to combine machine learning technique to finish the job. Even though using machine learning to do house price prediction is not a new topic, related researches are in fact very limited. I have read some papers related to the topic and here is a brief analysis about the merits and demerits of different methods.

Human House Agent

The following merits and demerits are summarized by comparing with non-human solution.

Merits:

1. Not like machine which only list various numerical data, the human agent will explain his prediction in a way that can be understood by non-professionals.

Demerits:

1. The prediction result may vary with different agents which is caused by personal preference.
2. Human agent cannot do house prediction in a large scale. In other words, they can only be professional within a given region. For example, Agent in Paris is not likely to predict the house price in London.

Non-human Solution

The first paper mainly discusses building a mobile application that can predict house price in London [4]. It uses the Gaussian Processes (GP) model with the data downloaded from the London Datastore[1].

Merits:

1. The research is basically complete including contents from the model training to the final mobile application developing.
2. The developing of the mobile application has increased the practical significance of the research.

[1] http://data.london.gov.uk/dataset/average-house-prices-borough

Demerits:

1. Features from the London Datastore are very limited. The lack of features has led to a poor model performance.
2. Only traditional machine learning algorithms like Bayesian Regression and Relevance Vector Machines have been discussed. However, it has ignored the newly emerging neural network/deep learning methods.
3. The mobile application's function is nowhere near practical because of the poor accuracy and reliability of the model.

The second paper concerns fitting regression model using programming language R. The data is provided by Kaggle[2]. This is a pure theoretical analysis dedicating to house price prediction.

Merits:

1. The paper has objectively analyzed and compared different machine learning models such as random forest trees, (generalized) linear regression model in detail.
2. The problem it discusses is still available on Kaggle (including the data), hence the process of the research can be easily verified and reimplemented.

Demerits:

1. Like the previous paper, it has not mentioned models like deep learning or neural network which is one of the focus in my research.
2. It merely contains the outcomes calculated based on data from Kaggle. This is not comprehensive because data from real life may not be able to reach that ideal performance.

1.2.2 Aims and Objectives

The aim of this dissertation is to design and implement a series of tools from sketch to finish all the work that will occur in house price prediction. It includes: house data downloader which is used to fetch house data; a feature engineering framework to do house feature analysis and preprocessing; a detailed comparison of several machine learning models' performance; a visualization tool for demonstrating and online predicting.

To reach the goal mentioned above, as well as taking the advantages and disadvantages of others work into consideration, the objectives of my project are as follows:

1. An http crawler that can automatically crawl house data from the UK's property website zoopla[3]. The data includes property type, number of rooms, house image, seller's description and the area information.

---

[2] Kaggle is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users.

[3] https://www.zoopla.co.uk/

2. Feature engineering process covering various feature preprocessing and feature analysis algorithms.
3. Using data to train and then compare several different machine learning models including Basic Linear Regression, Lasso Regression, Gradient Boosting Decision Tree and Deep Learning.
4. A web server for visualizing the house price according to different property locations by Google Map API as well as an online prediction tool providing online house price prediction for Zoopla.

2 Methodology and project overview

The project has followed the research-through-design (RtD) methodology. This approach is initially designed for interactive systems [4]. The core principle of it is latter design will highly depend on former implementation. This is exactly the case in this research-based project. For instance, it is impossible to foresee which feature will be significant for prediction house price. Hence the next focus of the research cannot be confirmed until the former result comes out. However, since numerous techniques have been involved in my dissertation and different parts differ a lot in methodology, the corresponding methodology of each part will be mentioned in its related chapter. To help reader find corresponding content more easily, and illustrate the overall paper structure, an overview has been given below:

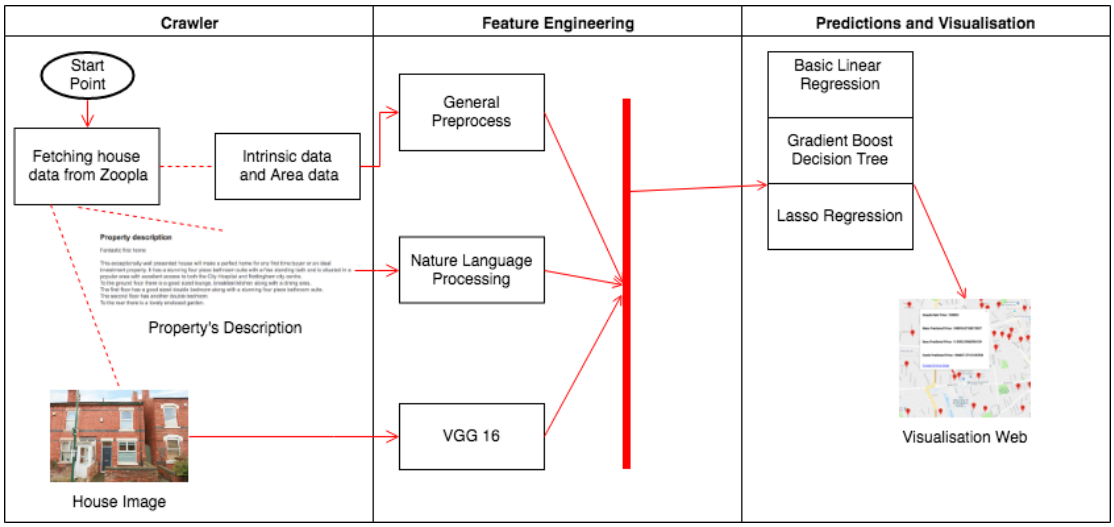


Figure 2.1 Design Overview

The project will be basically divided into three parts, different part will involve different methodology.

1. The first part is mainly based on web crawler including HTTP and text parsing. The principle will be explained in chapter 3.1.2. And the implementation details including the platform chosen will be mentioned in 4.1.
2. The second part involves feature engineering including how different form of features can be converted to number and the what algorithms will be used for feature selection. The knowledge behind will be explained in chapter 3.2. And the implementation will be discussed in chapter 4.2.

3. Final part is about the prediction model and visualization web application. The comparison and analysis of the result will be illustrated in chapter 5.

3 Project Design

3.1 Data and Crawler

3.1.1 Deciding data source

For a house prediction dissertation based on machine learning, the first important part is the data source. The quality and amount of data will determine the level of machine learning models [5]. I decided to crawl data from Zoopla using crawler technique. An important reason is the house data from UK's Land Registry is limited. Some details not available from Land Registry are available on Zoopla. Another reason is Zoopla is one of the UK's biggest property websites. Crawling data on property transaction website can guarantee the authenticity of data. The following screenshot (Figure 3.1) has shown how Zoopla displays a typical on sale house:

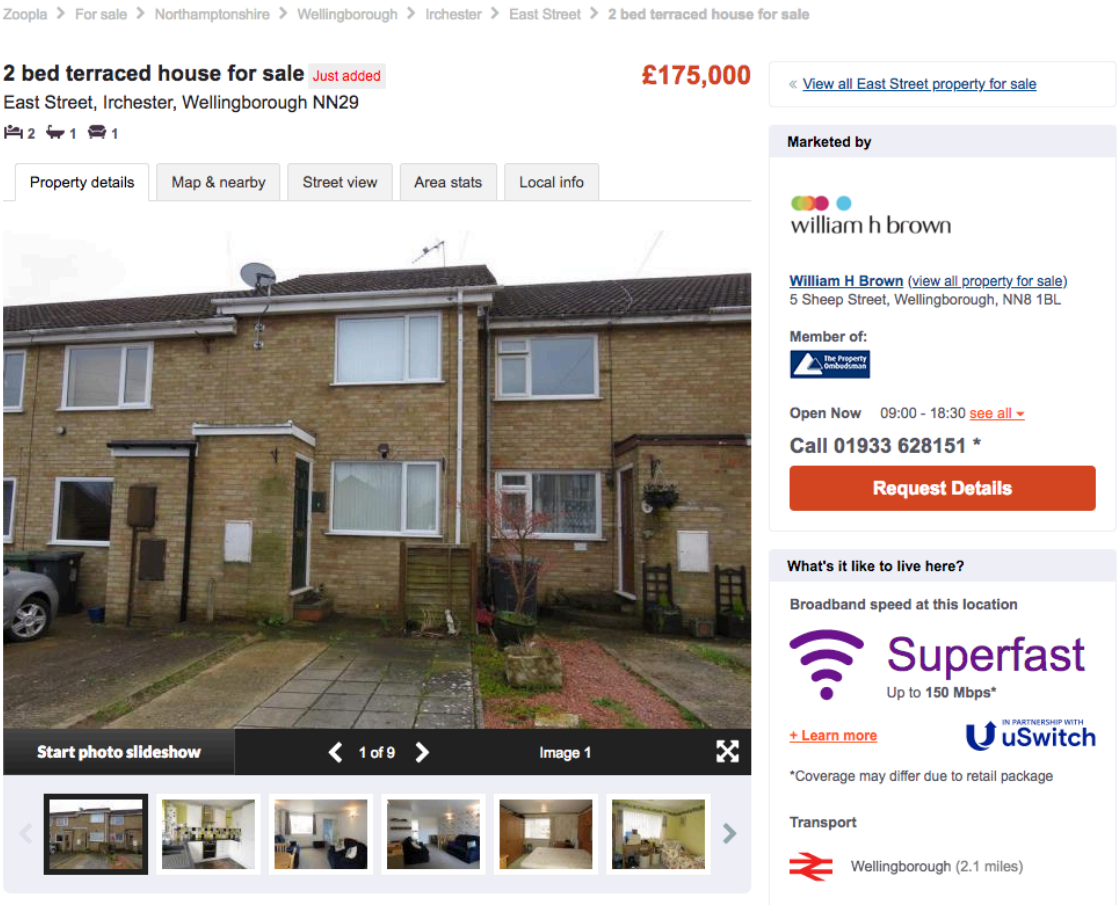


Figure 3.1 An on-sale house on Zoopla

For simplicity, I have used the on-sale price as the label in this problem. Although there will be more or less error between the real value and the seller given price, these two are still very close. And it will not influence the feasibility of this house prediction

project because in general, the seller given prices conform to the law of normal distribution around the real price.

The chart has listed the features that are available on Zoopla.

Feature Name	**Usage**
House ID	House identifier, unique for each house.
Postcode	Data integration will be based on it if we have more features from other source.
On-sale price	It will be used as our label, i.e. the “y”.
Latitude and longitude	It can be used for the final visualization.
Property type	House Feature
Number of bedrooms	House Feature
Number of bathrooms	House Feature
Number of receptions	House Feature
Description	Seller’s description for the house, can be used to do the nature language processing.
Rating	Users’ rating about this area in different aspects.
Demographic	Age distribution of this area.
Education	Education quality distribution of this area.
Crime	Crime distribution of this area.
Council tax	Council tax distribution of this area.
Housing	Housing owning distribution of this area.
Employment	Career distribution of this area.
Family	The family structure distribution of this area.
Interest	Locals’ interest distribution of this area. (cinema, diy, eating out, sports, football supporter, gardening, music, foreign travel)
Newspaper	A list of newspaper preference of this area.
Image	First photo of the house, will be used for deep learning

Chart 3.1 Features available from Zoopla

3.1.2 The web crawler

Zoopla does not provide a public API for the complete dataset it owns. Hence the first tool needed to build is a web crawler. The crawler's idea is inspired from considering how the information can be fetched manually:

1. Start from the index page that list all the houses on sale. i.e. https://www.zoopla.co.uk/for-sale/property/nottingham/
2. Iterate houses on the list one by one and collect their information.
3. Go to the next page and repeat above processes

However, this is an extremely slow and tedious process that contains many repetitive actions. This is the reason why a web crawler is needed. Here is the definition of web crawler from Techopedia [6]:

"Web crawlers collect information such the URL of the website, the meta tag information, the Web page content, the links in the webpage and the destinations leading from those links, the web page title and any other relevant information."

In short, web crawler is used to substitute human to handle HTTP Protocol. It sends HTTP requests to web servers and resolves the content returned by the servers. In fact, when user visits a web page, the things happening in behind is exactly HTTP Protocol i.e. client side (user) requests the web page and the website server returns the page. As the following figure shows, the only difference here is that the client side is no longer human but crawler that can gather millions of house data in several hours, which is impossible for a human.

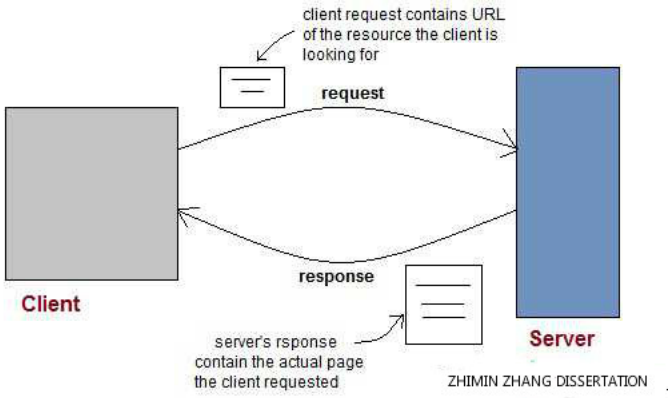


Figure 3.2 HTTP Process

3.2 Feature Engineering

3.2.1 Data Preprocessing

Right between downloading all the data and analysis them using various machine learning algorithms, another important step is to do data preprocessing. The preprocessing exists for the following reasons. On the one hand, Irregular data need to be converted suitable for different algorithms. On the other hand, the performance of

supervised machine learning algorithm can often be lifted by data preprocessing [7]. It includes several sub-steps:

1. Data Cleaning: all the missing values are filled in this step and some inconsistent data and noisy data are deleted.
2. Data Integration: various data from different sources are merged together without conflicts.
3. Data Transformation: some specific features are normalized or generalized for a better training speed and model performance.
4. Data Quantitation: the listed data, categorical data are converted into numerical data to fit different machine learning algorithms.

In addition to these preprocessing measurements, another preprocessing task is using the seller's house descriptions to do nature language processing. To further simplify the question, I plan to use a simple model called 'bag of words' to extract useful features. This model takes no account of grammar even word order, but word frequency is calculated. For example, consider the following two documents:

(1) Bob likes to play basketball, Jim likes too.

(2) Bob also likes to play football games.

Based on these two documents, construct the following dictionary:

Dictionary = {1: 'Bob', 2. 'like', 3. 'to', 4. 'play', 5. 'basketball',
6. 'also', 7. 'football', 8. 'games', 9. 'Jim', 10. 'too'}.

Depending on the dictionary, each document can be converted into vector form:

(1) : [1, 2, 1, 1, 1, 0, 0, 0, 1, 1]

(2) : [1, 1, 1, 1 ,0, 1, 1, 1, 0, 0]

Rule

1. The index of the list matches the index of keyword in the dictionary i.e. position one of the list matches 'Bob' and position two matches 'like'.
2. The value of each position in the list matches the frequency of that keyword i.e. the word 'to' occurs once in the first document, and its key in the dictionary is '3'. Hence, the value of the third position for the first vector should be one.

In summary, each document is regarded as a bag that contains several words, and then all the bags are identified by the frequency of the words it contains, that's why the model is called 'bag of words model'. The most significant benefit of this model is that the idea of this model is clear and the implementation of it is painless.

3.2.2 Feature selection

According to the definition from Wikipedia, feature selection describes the process of selecting a subset of relevant features that will be used in model training [8]. Specific

to the topic, feature selection can also help to find the features that are more likely to affect the value of a house.

In fact, feature selection can also be regarded as a dimensionality reduction process since those unrelated or redundant features will be filtered out. There are several reasons why it should be taken. Firstly, it saves the storage space required and lifts the data loading and training speed. Secondly, it improves the performance of the machine learning model because of the removal of multi-collinearity. Finally, it is much easier for people to visualize and understand the data with unnecessary features excluded.

From another perspective, this step also verifies the correctness of the previous work i.e. whether the features the crawler collected can influence the house price. Specifically, it shows which features are significant and will be primarily considered by house buyers. All of those can contribute to increasing the interpretability of my research.

The algorithm that will be used is "Chi-squared Test". Statistically speaking, it is widely applied to test the degree of correlation of two events [9]. This is the formula of Chi-squared Test:

$$x^2 = \frac{\sum_{i=1}^{n}(A^i - T^i)^2}{T_i}$$

Where A denotes the actual value and T is the expected value or hypothesis [10]. The following example has given a detailed explanation of Chi-squared Test.

Imagining there are two different medical treatments for a disease. And the goal is examining whether treatment A and B have any difference.

Efficacy of two treatments

Treatments	Effective	Not Effective	Sum	Cure rate
Treatment A	19	24	43	44.2%
Treatment B	34	10	44	77.3%
Sum	53	34	87	60.9%

1. First, set up a hypothesis that treatment A and B have the same effect which means all those differences above are caused by a sampling error.
2. Then combine the results of these two treatments to get the overall cure rate i.e. 53/87=60.9%.
3. Based on overall cure rate continue to calculate the expected value of those entries in the table:

   Treatment A is Effective: 43 * (53/87) = 26.2

   Treatment A is not Effective: 43 * (34/87) = 16.8

   Treatment B is Effective: 44 * (53/87) = 26.8

   Treatment B is not Effective: 44 * (34/87) = 17.2

4. Combine these two results:

Treatments	Effective	Not effective	Sum
Treatment A	19(26.2)	24(16.8)	43
Treatment B	34(26.8)	10(17.2)	44
Sum	53	34	87

Where the number out of the brackets is the actual value, and the number inside the bracket is the expected value.

5. Now consider the Chi-squared value:

$$x^2 = \frac{\sum_{i=1}^{n}(A^i - T^i)^2}{T_i} = \frac{(19-26.2)^2}{26.2} + \frac{(24-16.8)^2}{16.8} + \frac{(34-26.8)^2}{26.8} + \frac{(10-17.2)^2}{17.2} = 10.01$$

Hence it can be understood by saying that Chi-squared Test calculates the difference between the actual value and the expecting value. The smaller the difference is, it is more likely that the hypothesis is correct.

3.3 Machine learning

After all the data has been well prepared, the core step of the dissertation is to do house price prediction using different machine learning models. The basic idea is to try several different models and compare their performance. There are four different models will occur in my dissertation for the following reasons.

Basic liner regression: this model will be used as a baseline to evaluate the performance of other three advanced algorithms.

Lasso regression: this can be regarded as an advanced linear model which contains feature selection functionality. The conclusion can be verified if the performance of basic linear regression can get close to Lasso regression when feature selection has already been taken.

Gradient Boost Decision Tree: this is a mature and widely-used nonlinear model. This can be used to verify if nonlinear model can have better performance for house price prediction.

Deep Learning: the hottest model at the time I wrote the paper (2018), I have not seen any paper introducing this model into house price prediction. Hence, my research will be an explorative attempt.

3.3.1 Basic linear regression

Basic linear regression is one of the simplest models in machine learning. This is the formula of this model:

$$f(x, w) = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \cdots + w_n * x_n \quad \{3.1\}$$

As I have mentioned in introduction chapter. $x_1, x_2, \ldots x_n$ are different influencing factors such as house type, house size and house location and $w_1, w_2, \ldots w_n$ are their weights. The process of training the model training is trying to find the value of $w_1, w_2, \ldots w_n$.

To find the best $w_1, w_2, \dots w_n$, **error function** is also needed:

$$S = \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \quad \{3.2\}$$

Where y is the label and $f(x_i, w)$ is the prediction value calculated by the model function i.e. formula 3.1. The value of S is the error between the model prediction value and the actual value. The smaller S is, the closer the predicted value is to the actual value. That's why formula 3.2 is named 'least square error function'.

Then the best parameter $w_1, w_2, \dots w_n$ must have the smallest error i.e. we want to find the parameter $w_1, w_2, \dots w_n$ to minimize the error function. A common way to solve this question is using **gradient descent**.

The basic idea of gradient descent is:

1. Take the derivative for the error formula with respect to w.
2. Find the corresponding $w_1, w_2, \dots w_n$ to make the derivative formula be 0. Because in that case, the error function is exactly at the extreme point (minimum point) which is expected.
3. More specifically, adjust $w_1, w_2, \dots w_n$ to the negative gradient of the error function current point:

$$w^{n+1} = w^n - \gamma \nabla S(w) \quad \{3.3\}$$

Where γ is the length of each step which controls the speed of gradient.

3.3.2 Lasso regression

Lasso regression is the enhanced linear regression model that performs both variable selection and regularization. Robert Tibshirani firstly introduced it in 1996 on Leo Breiman's nonnegative garrote [11]. It keeps using the model function (formula 3.1) that basic linear regression used. However, using least square error method cannot avoid the issue brought by multicollinearity. Multicollinearity is defined as [12]:

> *'...a situation in which two or more explanatory variables in a multiple regression model are highly linearly related'.*

In general, it will greatly influence the performance of the linear model. Lasso regression is one of the improved algorithms that can solve this problem. This is the error function of lasso regression:

$$S = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i, w))^2 + \lambda |w| \quad \{3.4\}$$

The $\frac{1}{2}$ in the front of the sum exists for calculating convenience. The only difference between formula 3.4 and formula 3.2 is that an extra term $\lambda |w|$ is plugged. This leads to a smaller weight w for every feature after gradient descent. On one hand, this introduction of this extra term contributes to avoid overfitting. On the other hand, the weight w can be adjusted to 0 in situations which contain multicollinearity. This means features which may lead to multicollinearity will be weighted 0 since the weights in front of them are 0.

Comparing with those advanced models, Lasso regression intelligible and explainable. Comparing with basic linear model, it is much powerful with feature selection self-contained. In summary, lasso regression is a typical linear regression method that is enough to handle most of the simple situations. That's why I have involved it in my research.

3.3.3 Gradient Boosting Decision Tree

After having two linear models, I have decided to use a nonlinear algorithm to check whether a better performance can be reached if the house data has nonlinear relationship with house price. The model I plan to use is Gradient Boosting Decision Tree. It contains two concepts: boosting and decision tree.

Boosting belongs to a more general concept – Ensemble learning. Ensemble learning is the idea that aims to improve the final performance by combining several machine learning models together. In ensemble learning those bottom models are called base models. According to the 2017 statistical report from Kaggle, ensemble learning tends to have better results comparing with those single model machine learning algorithms [13]. Ensemble learning contains two sub concepts, bagging and boosting. Here only boosting will be used. Conceptually, boosting learning keeps all the base models in a line, and uses the data to train them one by one. However, not like the traditional training method, every time a base model finishes training, the strategy of training or the weight of each data instance will be updated accordingly. For example, those house instances of which the previous models have poor performance on will get a higher weight in later trainings. In this way, the later model will pay more attention or lay more bias on this kind of data. When we combine all the models together, the performance will improve because the data bias of a single model has been adjusted by other models. This is the general model function of boosting learning:

$$y = \sum_{j=1}^{m} \partial_j f_j(x, w) \qquad \{3.5\}$$

Where m denotes the number of base models, and ∂ denotes the weight of each base model function.

Decision Tree is a famous nonlinear machine learning algorithm which can be used for both classification problems and regression problems. However, in real situations, decision tree is rarely used alone. It is often used as one of the base models in ensemble learning. For example, gradient boost decision tree which I plan to use, in simple terms, is a typical boosting learning algorithm that its base models are all decision trees. There are several reasons why I choose to use Gradient Boost Decision Trees.

1. It performs excellently for nonlinear problems as I already have a linear algorithm (Lasso Regression).
2. It is a robust algorithm, i.e. the initial value of the parameters will not influence the final model performance. Hence, I do not need to pay too much attention on how to set the initial value properly.
3. The algorithm has been widely used in both industry and various machine learning competitions, which means a plethora of related tutorials and documents are available.

3.3.4 Deep Learning

Deep learning is one of the hottest topics nowadays which has been applied in fields including computer vision and speech recognition and audio recognition [14]. My goal here is using the convolutional network based on those house images to predict house price. However, I decided not to build my own neural network from scratch because the task is demanding. Additionally, training and optimization process is also time consuming. Hence, I decide to use the pre-trained model – VGG16.

VGG 16 is a 16-layer network used by VGG team in the ILSVRC-2014 competition. It is also called the 'very deep convolutional networks for large-scale image recognition'. To better understand VGG16, I want to give a definition of the term 'pre-trained' and then explain why it is useful. A pre-trained model is a model that has been trained on a large amount of data of which the weights and biases are well tuned. Since different image recognition models contain similar structures to distinguish common geometrical shapes like edges and lines, it is not hard to imagine that it can be used to accomplish different image recognition tasks. According to the relevant document of VGG16 from the author, the model indeed has reasonable performance when used as an image feature extraction tool [15]. The following figure demonstrates the structure of VGG16:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3.3 VGG Architectures

Having the basic knowledge about Deep Learning and VGG16, I would like to illustrate how I have designed my prediction tool based on it. Firstly, I will use the model as a feature extraction tool to extract meaningful features from VGG16, i.e. using the model to do prediction depending on the pixels vectors (converted from the house image). And then the model's prediction result will be the input of a previous regression model (lasso

regression or gradient boost decision tree). The basic principle can also be described by the following flow chart:

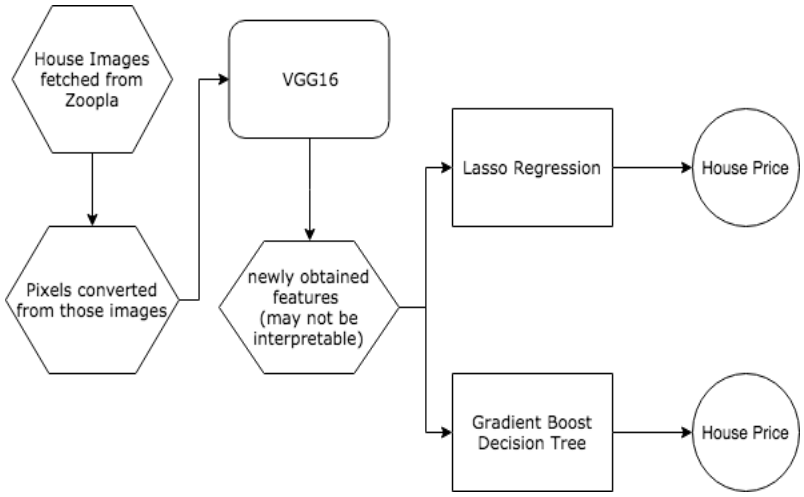


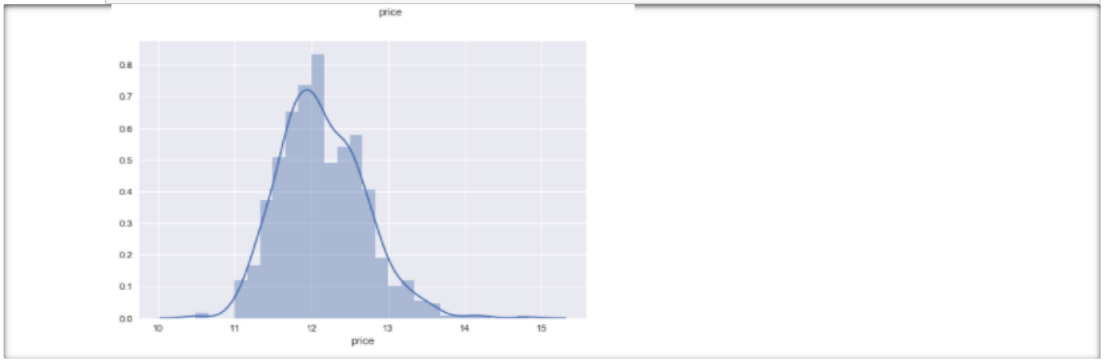
Figure 3.4 House Image Analysis Workflow

4 Project Implementation

4.1 Programming Language and Tools

The combination of Jupyter Notebook and Python programming language have been used as a tool to verify my hypothesis during the whole research process. To say it informally, Jupyter Notebook is an enhanced interactive interpreter that can show the result of python scripts on the web. It supports several script languages such as Python, R, Julia and Scala. Considering that Python has larger numbers of scientific computing and machine learning packages, I have chosen it among the above candidates. The following screenshot explains why this interactive way is extremely helpful for a machine learning project. As can be seen in the Figure 4.1, I load the data in line 3 and check what features it contains in line 4. Then some basic features analysis (plot) and preprocessing (drop unnecessary features) have been undertaken.

```
In [3]: # load data and then check the shape of the data
        train = pd.read_json("data/house.jl", lines=True)
        train.shape

Out[3]: (753, 26)

In [4]: train.columns

Out[4]: Index(['counciltax', 'crime', 'cs_rating', 'demographic', 'description',
               'education', 'employment', 'en_rating', 'family', 'house_id', 'housing',
               'interests', 'latitude', 'longitude', 'newspapers', 'num_of_bathrooms',
               'num_of_bedrooms', 'num_of_receptions', 'overall_rating', 'postcode',
               'pr_rating', 'price', 'property_type', 'rs_rating', 'sp_rating',
               'tt_rating'],
              dtype='object')

In [4]: sns.distplot(train.price)
        plt.show()

        sns.distplot(np.log1p(train.price))
        plt.show()
```



Meta data only

```
In [5]: meta = train.drop(["postcode", "price", "house_id", "description"], axis=1)
        meta = process_list(meta)
        meta = meta.fillna(0)
        meta = pd.get_dummies(meta)
        meta.shape

Out[5]: (753, 86)
```

Figure 4.1 Jupyter Notebook Screenshot

In this way, the workflow can be:

1. Open Jupyter Notebook
2. Load data
3. Plot or process data
4. Try different models

Rather than:

1. Open text editor and write some code
2. Run the code and check the result
3. Modify the code and rerun it

Since this is a machine learning project, and the size of data is usually huge which means loading or processing data will be time consuming, the first workflow can save large amount of time by loading data only once. In addition, it is much easier to monitor validation error by using this interactive way.

4.2 Web crawler

4.2.1 The crawler framework -- Scrapy

Scrapy, as described in the official website, is a fast and powerful framework for scraping and web crawling. This is an overview of the framework:

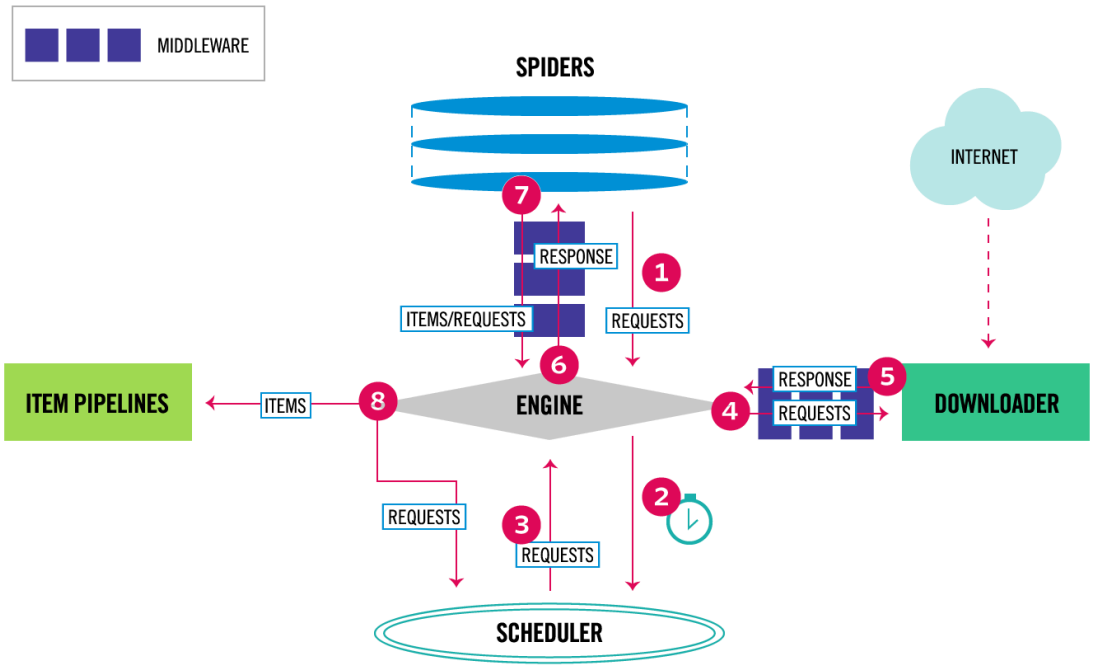


Figure 4.2 Framework Overview

Detailed workflow:

1. The framework starts from receiving requests.
2. And then all the requests will be managed by a scheduler. The scheduler runs in an asynchronous and non-blocking manner. This manner guarantees the overall speed.
3. Requests sent out by scheduler will be converted to the real HTTP requests by another component named downloader. This component is responsible for sending HTTP requests and receiving responses from remote servers.
4. Once the response has been received, it will be passed as parameters into the callback function that users registered earlier in their spiders. After that, all the text parsing and information extraction work will take place inside the callback function.

Usually it is hard for a framework to be powerful while keeping its usability. Comparing with another famous crawler package – Requests[4], which only provides features to handle various web requests and responses, Scrapy has predefined the complete behaviors that can schedule huge amount requests and responses in short time. In other words, users do not need to define any handle logic to guarantee the crawling speed. Additionally, Scrapy already contains HTML parsing features in box, no more extra HTML parsing library such as Beautiful Soup[5] is needed. All the above reasons have given Scrapy a strong competitiveness.

4.2.2 Analysis of Zoopla's web structure

Two different kinds of pages will be involved throughout the analysis.

1. **List page:** Several houses (emphasized in red circle) are listed in this page. It can have more than one page, so page turning must be considered.

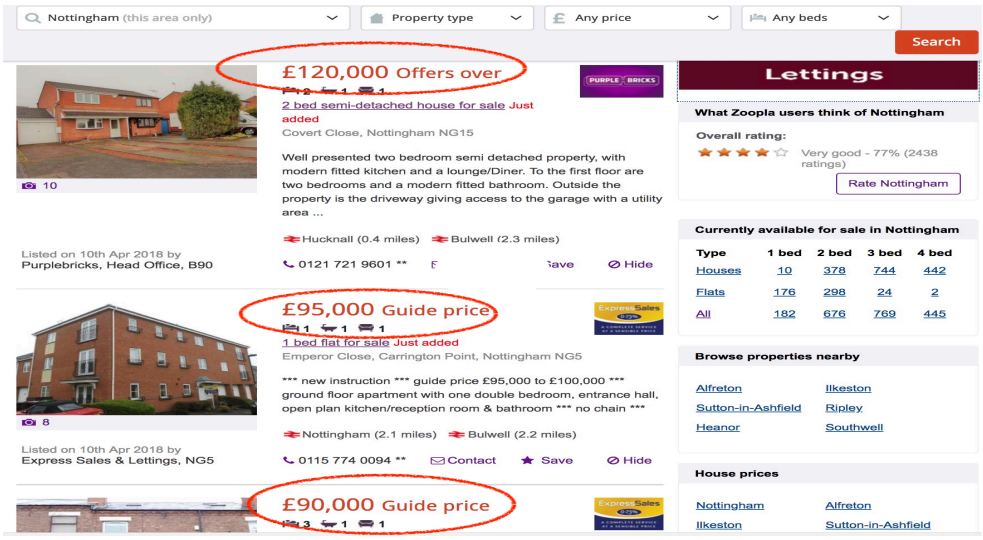


Figure 4.3 Screenshot of List Page

---

[4] Requests is a famous HTTP library for Python. http://docs.python-requests.org/en/master/

[5] Beautiful Soup is a Python library for pulling data out of HTML and XML files. https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.html

2. **House page:** House page can be entered through list page. The page gives detailed information about the house, and the information will be saved by crawlers.

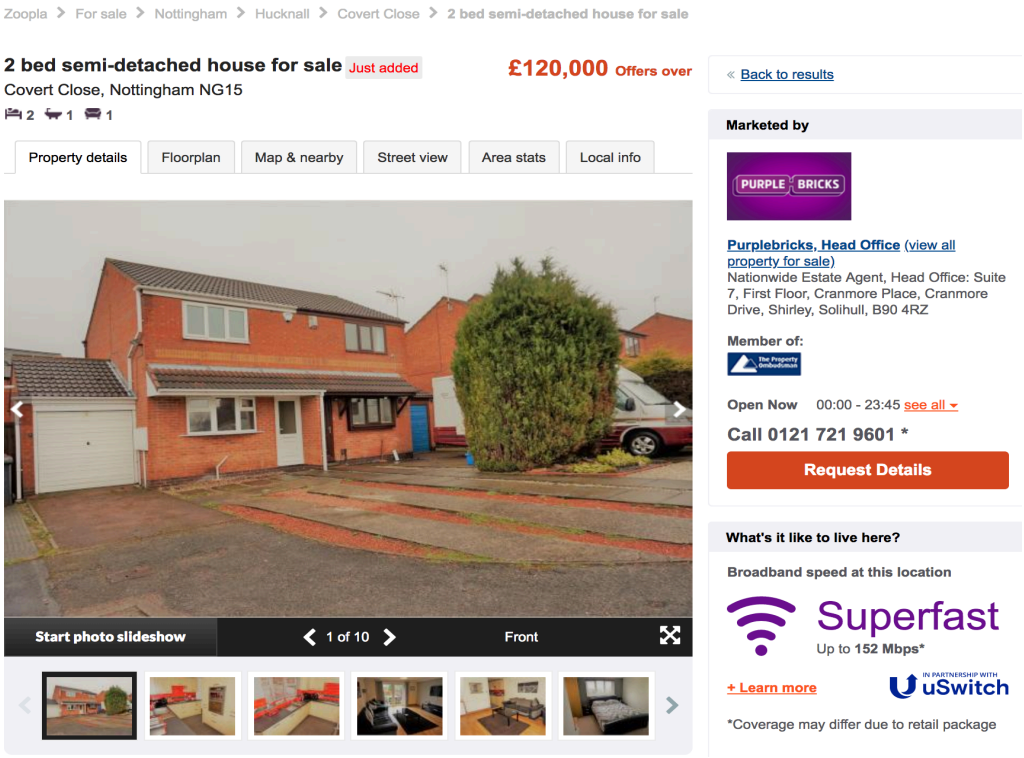


Figure 4.4 Screenshot of House Page

Hence, combined with the crawler Scrapy, the whole crawling process should be:

1. Start from the first list page.
2. Enter each house page and save the house information respectively.
3. Go to the next page and repeat the above steps.

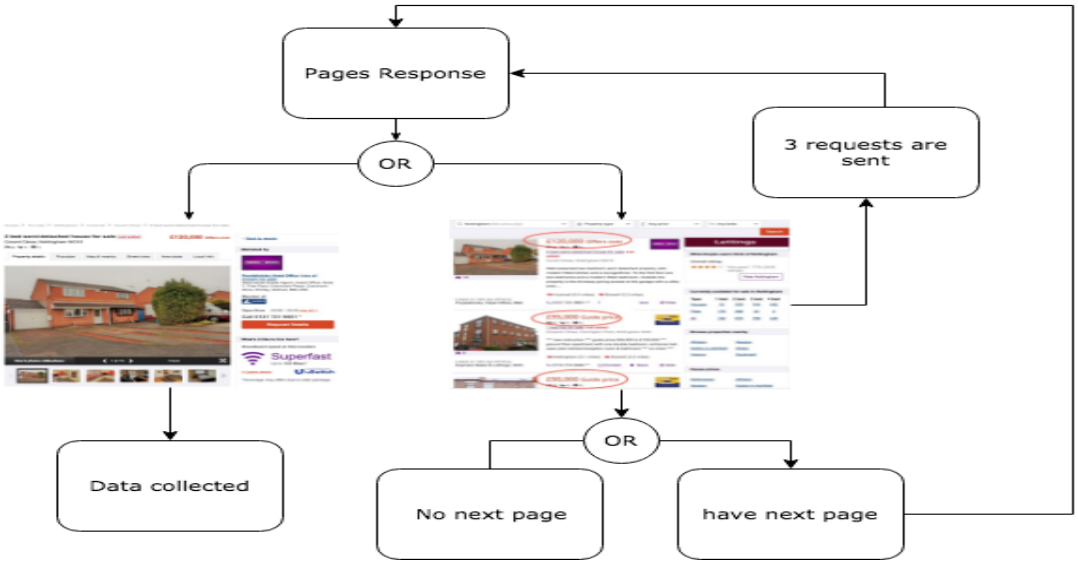


Figure 4.4 Crawler Design

4.2.3 The Code Part

Following the above idea, the spider has been implemented. It is mainly responsible for defining link rules and parsing information.

```
14  class CrazySpider(CrawlSpider):
15      name = "crazy_spider"
16      allowed_domains = ['zoopla.co.uk', 'lid.zoocdn.com']
17      start_urls = ["https://www.zoopla.co.uk/for-sale/property/nottingham/?identifier=nottingham&q=Nottingham&search_source=refi
18      rules = (
19          Rule(LinkExtractor(allow=('/for-sale/property/nottingham',),
20                             deny=('tel',))),
21          Rule(LinkExtractor(allow=('/for-sale/details/[0-9]+',),
22                             deny=('image', 'play_movie')),
23               callback="parse_info")
24      )
25      custom_settings = {'LOG_LEVEL':'INFO',}
26                         #'LOG_FILE': 'log.txt',}
27
28
29      def parse_info(self, response):
30          '''Parse the house information'''
31
32          image_addr = response.css("#images-main img::attr(src)").extract_first()
33          if(image_addr.find('noimage') != -1):
34              return
35
36          item = CrazyItem()
37
38          item['price'] = str2num(response.css(
39              ".text-price > strong::text").extract_first().strip().replace(",",
40              "").replace("£",""))
41
```

Figure 4.5 Code Snippet of Crawler

1. The crawler starts by wrapping the attribute *'start_url' (line 17)* to the initial HTTP request.
2. After response has been received, the engine will handle it depending on the predefined rules *(line 18)*. House page and list page will be handled in different logic.
3. If the responding page is a house page, the function 'parse_info' *(line 29)* will be called to parse it and then save the information. The function mainly uses CSS selector for information extraction. For example, line 32 shows how to use CSS selector to extract house image link from HTML.
4. On the other hand, if the page is a list page, several house page URLs can be found. Then those URLs will be converted into new requests one by one.

4.3 Feature engineering

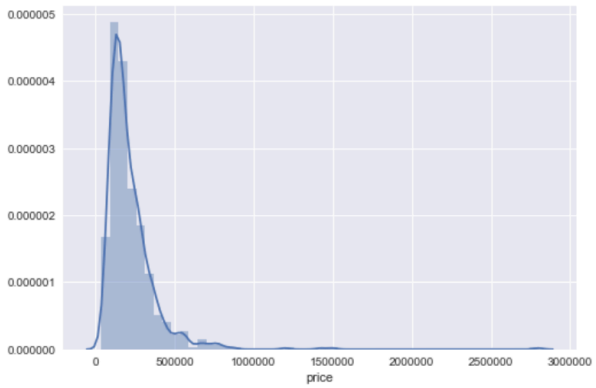
4.3.1 Log transformation

After loading all the data, the first problem I meet is that the data distribution of the house price does not conform to normal distribution. There are several reasons why data should conform to normal distribution. In short, well distributed data can greatly save training time and lift model performance (especially for linear model) [16]. Additionally, well distributed data also contributes to finding the relationship between

features more easily. This is because the sample of normal distribution are uniformed and normalized so that some statistic inference can be applied.

Back to the house price, the original price distribution plot has a high kurtosis and it is positively skewed. In other words, the distribution of price does not conform to normal distribution well:

```
sns.distplot(train.price)
plt.show()
print("Skewness:", train.price.skew())
print("Kurtosis:", train.price.kurt())
```

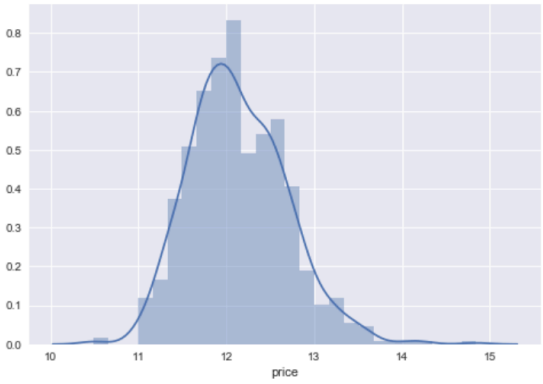



```
Skewness: 6.12187303657
Kurtosis: 70.5775911966
```

Figure 4.6 Price Distribution Plot

To adjust the distribution of the house price, log transformation has been taken. This is the distribution plot of the price after transformation, and now the data is well distributed. This can also prove that the effect of log transformation:

```
sns.distplot(np.log1p(train.price))
plt.show()
print("Skewness:", train.price.skew())
print("Kurtosis:", train.price.kurt())
```




```
Skewness: 6.12187303657
Kurtosis: 70.5775911966
```

Figure 4.7 Price Distribution Plot After Log Transformation

4.3.2 Data flattening

As mentioned in previous chapter, all the features will be converted to number in this phase. Figure 4.8 is a typical example describes why a feature need to be converted.

The feature describes the employment around that area where the house located. Not like the features occurred before, such as *'number of bedroom'* or *'house price'*, this feature contains more than one data point (i.e. Higher manager, Professional, Office Worker…). Hence, it will be downloaded as a list. In the following case, the value of this feature will be downloaded as *'[64, 84, 95, 87, 113, 128, 98]'*. However, the expected form should be '64, 84, 95, 87, 113, 128, 98' in seven features in total. Except 'Employment', other features with similar form will be flattened as well.

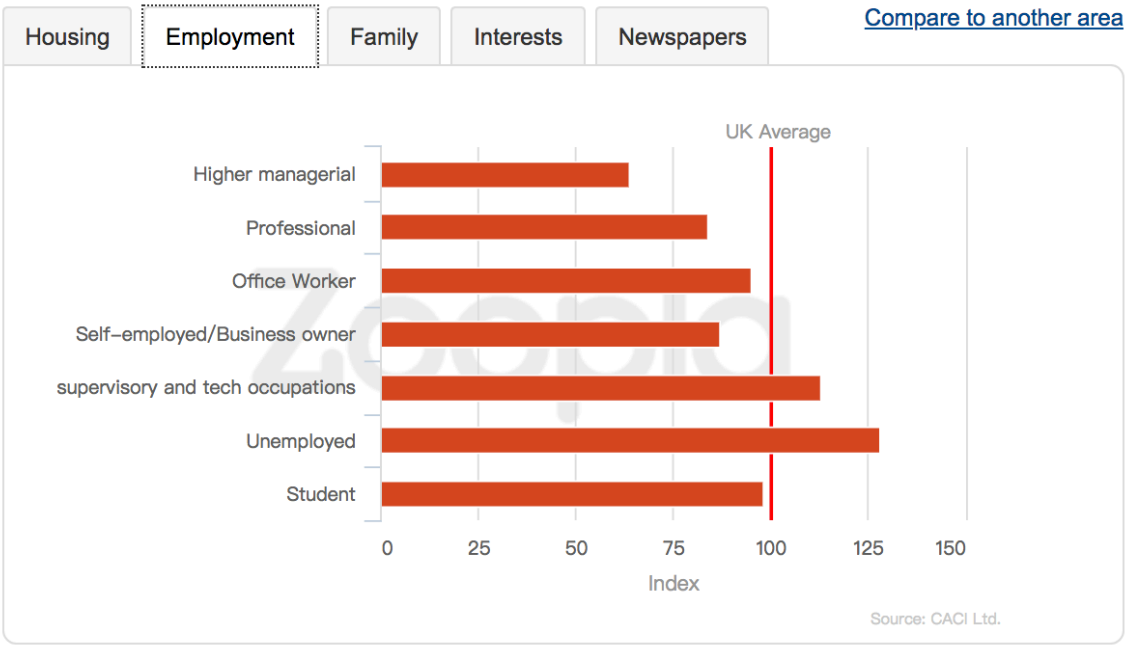


Figure 4.8 Employment in the area of the house location

The function *'process_list'* is defined to handle this kind of features of which value is a list:

```
1   def process_list(df):
2       to_flat = [
3                'counciltax', 'crime',  'demographic', 'education',
4                'employment', 'family', 'housing', 'interests', 'newspapers',
5               ]
6
7       for each in to_flat:
8           new_added_df = pd.DataFrame(df[each].values.tolist()).add_prefix(each)
9           df = pd.concat([df.drop(each, axis=1), new_added_df], axis=1, join='inner')
10
11      return df
12
```

Figure 4.9 Function to flat data

The list *'to_flat'* contains all the features that need to be flatten. And for each feature, its value will be taken out and be converted to an independent matrix (line 8). After that, the matrix is merged back to the original matrix. The process can also be described by the following example:

1. This is a matrix which contains three house instances and each house instance has four features.

Original matrix

index	Num_of_bed	Property_type	Employment	Price
0	2	Flat	[70, 80, 90]	100
1	1	Detached	[75, 85, 95]	200
2	2	Semi-det	[60, 80, 100]	200

2. This is the new matrix after 'Employment' column being taken out and converted.

New matrix converted from 'Employment' (line 8)

Index	Employment_1	Employment_2	Employment_3
0	70	80	90
1	75	85	95
2	60	80	100

3. The new matrix is merged back to the original one, and the previous 'Employment' column has been deleted.

Matrix after conversion (line 9)

index	Num_of_bed	Property_type	Employment_1	Employment_2	Employment_3	Price
0	2	Flat	70	80	90	100
1	1	Detached	75	85	95	200
2	2	Semi-det	60	80	100	200

4.3.3 Dummy coding and empty value filling

All the categorical data will be converted to numerical data at this stage. Unlike list feature of which value is a list, the value of categorical data can only be one of several possible candidates. In this project, '*Property_type*' is a categorical data of which possible values are listed by the *'candidates_type'* below:

```
56          candidates_type = ['detached', 'semi_detached', 'flat', 'terraced',
57              'detached_bungalow', 'end_terrace', 'town_house', 'bungalow',
58              'cottage', 'semi_detached_bungalow', 'maisonette']
59          item['property_type'] = response.css(
60              "html").re('property_type", "(.*)"')
61          if (len(item['property_type']) == 0 or
62              item['property_type'][0] not in candidates_type):
63              return
64          else:
65              item['property_type'] = item['property_type'][0]
```

Figure 4.10 Code Snippet of Filtering House Property

The common way to achieve numerical conversion is dummy coding. Basically, detailed conversion follows those two steps:

1. Create a matrix of which the columns are made up of the possible value of this feature.
2. Since a data instance can only match a specific value, only that corresponded column will be 1, and other columns should all be 0.

For example, there is a feature called subject, and its possible value can be 'Math', 'History' or 'Physics':

Index	Subject
0	Math
1	History
2	Physics

After dummy coding, it should be:

Index	Subject_Math	Subject_History	Subject_Physics
0	1	0	0
1	0	1	0
2	0	0	1

The last preprocessing phrase should be filling those empty value. However, as the crawler has already given up most of the data containing empty value while crawling, so no extra action is needed here. This greatly simplifies our work.

4.3.4 Feature Selection

According to my original design, the 'chi-square' algorithm will be used for feature selection. This is the result:

Feature Selection

In [43]:
```
X_new = SelectKBest(chi2, k=2).fit(meta, train.price)
features = []
for i, column in enumerate(meta.columns):
    features.append((X_new.pvalues_[i], column))
features.sort()
print("Top 10 unrelated features: \n\n", features[:10])
print("\nTop 10 related features: \n\n", features[-10:])
```

```
Top 10 unrelated features:

 [(0.0, 'employment0'), (0.0, 'employment1'), (0.0, 'employment2'), (0.0, 'employment3'), (0.0, 'emp
loyment4'), (0.0, 'employment5'), (0.0, 'employment6'), (0.0, 'family0'), (0.0, 'family1'), (0.0, 'f
amily2')]

Top 10 related features:

 [(0.9999999999999822, 'education1'), (1.0, 'demographic0'), (1.0, 'demographic2'), (1.0, 'demograp
hic3'), (1.0, 'demographic4'), (1.0, 'demographic5'), (1.0, 'education0'), (1.0, 'education3'), (1.
0, 'education4'), (1.0, 'property_type_semi_detached_bungalow')]
```

Figure 4.11 Features Selection Result

where the higher value means higher degree of correlation. Since the same selection process will undertake right before model training, this section is for demonstration only. It is easy to get the knowledge that some features such as 'employment' and 'family' do not have any relation with house price. The detailed conclusions will be explained at next chapter.

4.4 House price prediction

There are several machine learnings techniques such as Basic Linear Regression, Lasso Regression, Gradient Boost Decision Tree and Deep Learning model VGG16 involved during the whole process. However, they have not been used parallelly but are combined in a complex way. The following relation chart is aimed to clarify the case happened here much clearer.

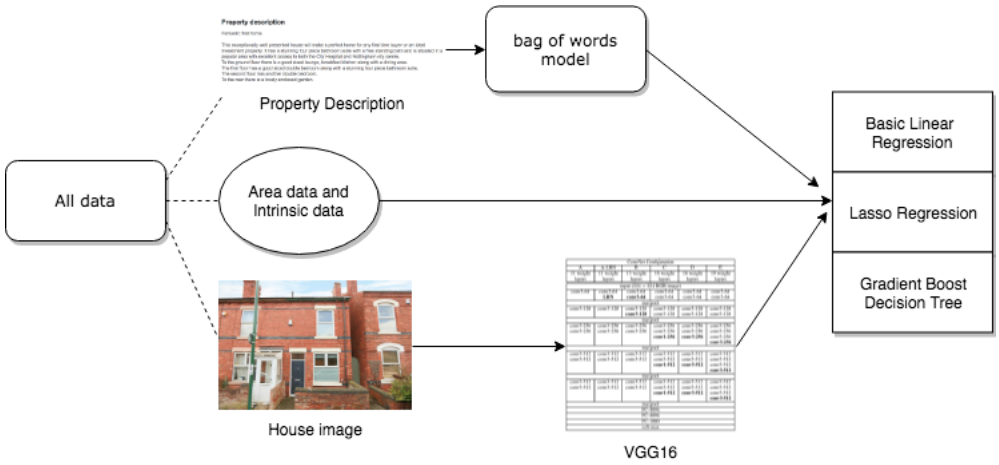


Figure 4.12 Prediction Relation Chart

Looking at the big picture, all data will go through Lasso Regression and Gradient Boost Decision Tree in the end. The only difference is that different kinds of house features may experience different data preprocessing techniques i.e. VGG16 for images, bag of words model for description text. It is better to understand those techniques are only playing the roles of features extractors and their output will be used as the input of Lasso Regression or GBDT to do the final prediction work.

To switch my training models flexibly and adjust the model parameters conveniently, I have abstracted those steps into a function to avoid duplicate code:

```
1  def regressor(X, y, model, n):
2
3      # use all features without feature selection
4      if n == -1:
5          preproc_pipe = [('scaler', StandardScaler()), # column-wise
6                          ('normalizer', Normalizer())] # row-wise
7      else:
8          preproc_pipe = [('scaler', StandardScaler()),
9                          ('lsa', TruncatedSVD(n)),
10                         ('normalizer', Normalizer())]
11
12     preproc_pipe = Pipeline(preproc_pipe)
13     X = pd.DataFrame(preproc_pipe.fit_transform(X))
14
15     pipeline = []
16     if model == 'GB':
17         pipeline.append(('estimator', GradientBoostingRegressor(n_estimators = 300)))
18     elif model == 'LASSO':
19
20         pipeline.append(('estimator', Lasso(alpha=2e-5, max_iter=2000)))
21     else:
22         print('Error Model')
23         return None
24     pipeline = Pipeline(pipeline)
25
26     MAE_train = []
27     MAE_test = []
28
29     kf = KFold(n_splits=5, shuffle = True)
30     for train_index, test_index in kf.split(X):
31         X_train, X_test = X.iloc[train_index], X.iloc[test_index]
32         y_train, y_test = y.iloc[train_index], y.iloc[test_index]
33
34         pipeline.fit(X_train, np.log1p(y_train))
35         y_train_predict = np.expm1(pipeline.predict(X_train))
36         y_test_predict = np.expm1(pipeline.predict(X_test))
37
38         MAE_train.append(mean_absolute_error(y_train, y_train_predict))
39         MAE_test.append(mean_absolute_error(y_test, y_test_predict))
40
41     return np.mean(MAE_test),
```

Figure 4.13 Code Snippet of Regressor

The function gives user the freedom to choose models i.e. Lasso or GBDT (line 15) and allows user to pass feature selection parameters e.g. the parameter 'n' used by *TruncatedSVD* (line 3). Furthermore, the features normalization and log transformation have also been included in the pipeline. Another point worth mentioned is that 5-fold cross validation is applied to make sure that the model's performance has been fairly assessed. On the top of it, the standard performance measurement for this dissertation is 'Mean absolute error' (line 38).

However, I did not learn any advanced algorithms for parameter choosing. This has caused a long and boring parameter choosing process because the most primitive method – brute force has been used. The following screenshot has demonstrated how I use brute force to find the parameter alpha of Lasso regression. I tried several different parameters and then draw the plot and then find out the optimal parameter is *0.00002*:

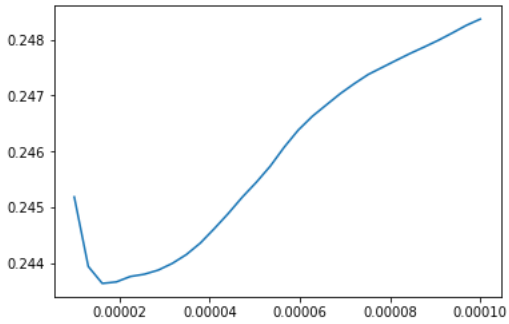


Figure 4.14 Choosing parameter alpha

4.5 Data Visualization

A web server has been set up to demonstrate the project result. I did not implement everything by myself but have used the Google Map API instead. On one hand, it has provided enough functionality that can meet my visualization requirement without worrying the internal implementation. On the other hand, it is too hard to implement the similar function from scratch in limited time. I used Node.js as my backend language combined with frontend JavaScript to build this web application. The web application is simple but powerful. Figure 20 introduces how the web has interacted with Google and the local server:

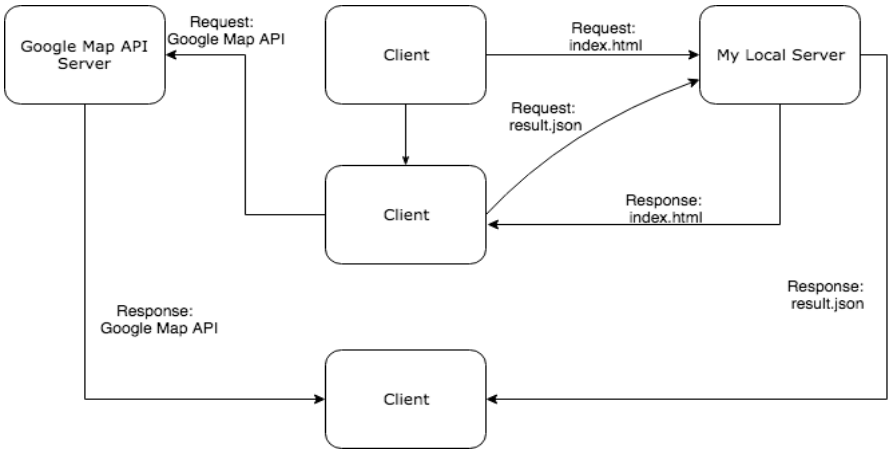


Figure 4.15 Web Framework

As can be seen, there are two servers in the graph, local server and Google Map API server/provider. Local server is responsible to provide the web pages and the necessary data for visualization based on the AJAX technique. Before the page is fully loaded, it will also request some extra scripts from Google server to make sure those API functions is obtained. After all of those, the page displays an interactive map with all the houses inside. Users are free to click to check the detailed information. This is a screenshot of my visualization page:

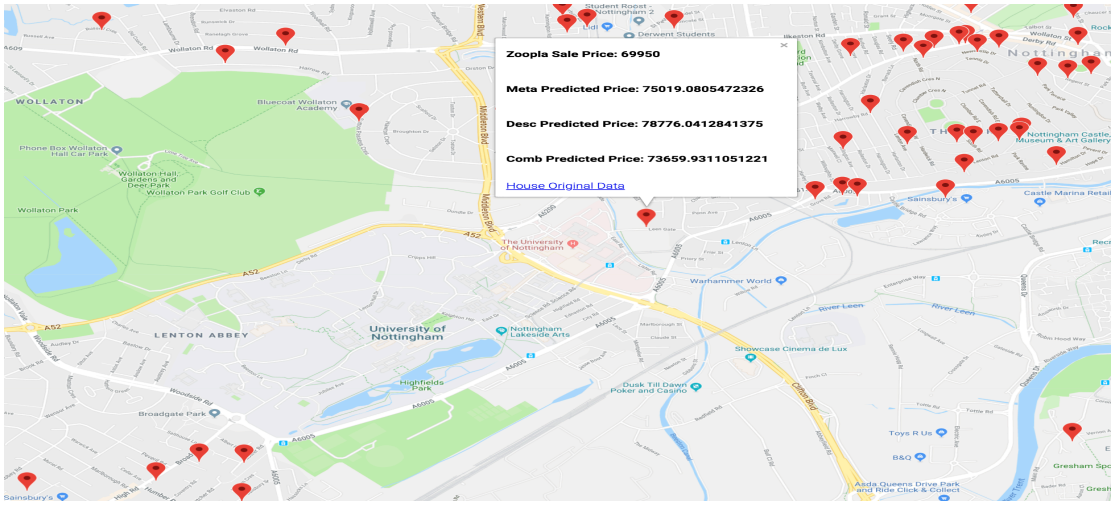


Figure 4.16 Visualization Screenshot

Another part of my visualizations is the online predictor. The final effect is represented in the following screenshot.

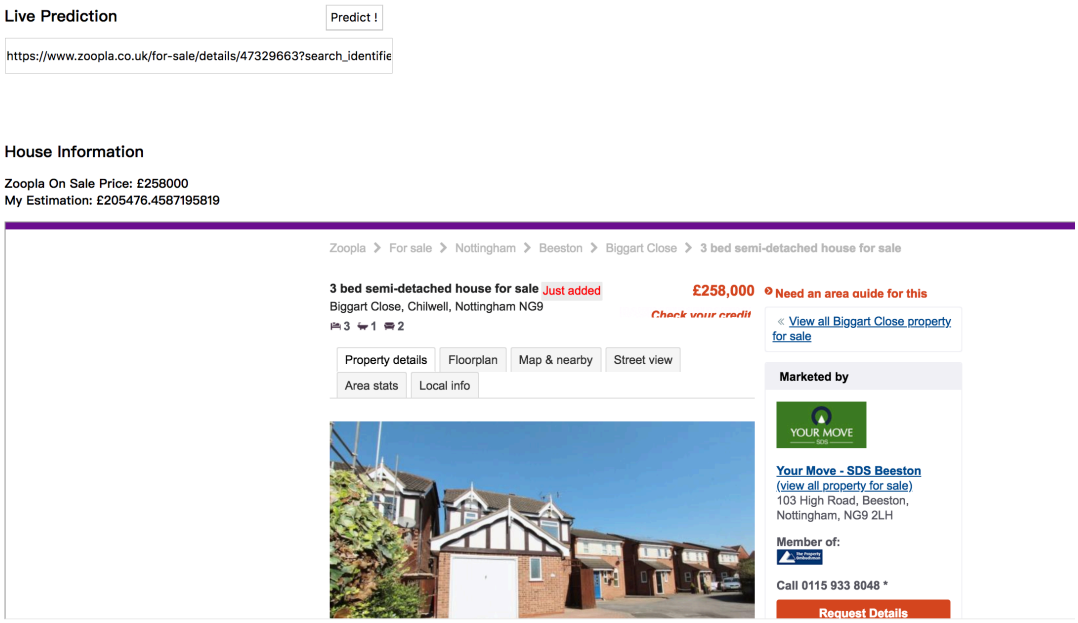


Figure 4.17 Screenshot of online predictor

After all the models has been trained, the implementation of the predictor is obvious. First, the crawler will download the corresponding house information depending on the URL provided by web user. Second, use the house prediction model to predict the price and send the prediction result back. Finally, the result will be displayed on the website.

5 Result Analysis and evaluation

In this chapter I will give a detailed analysis about the dissertation result including those meaningful features found by Chi-Squared algorithm, some significant keywords from sellers' description and the performance comparison of different model.

5.1 Meaningful Features

The following table is the complete scores of different features calculated by Chi-Squared Test. Where score closing to 0 means low correlation and closing to 1 means high correlation:

Unrelated Features: Features with 0 scores, which means those features have no correlation with house price.

No Correlation Features

employment	family	housing	interests	newspapers	council tax

Scores of Related Features: Features with at least one value point have non-zero scores, which means those features have correlation with house price.

Correlation Score of Crime

Anti-social behavior	Burglary	Criminal damage & arson	Drugs	Public order	Robbery	Violence and sexual	Vehicle crime
0	0.999	0.004	0	0	0.999	0	0.999

Correlation Score of Demographic (Aged)

0-14	15-24	25-34	35-44	45-54	55-64	65+
1.000	0	0.004	0	0	0.999	0

Correlation Score of Users Rating

Community & safety	Entertainment & nightlife	Parks & recreation	Schools & public services	Restaurants & shopping	Transport & travel	Overall
0	0	0.372	0.780	0.806	0.999	0.999

Correlation Score of Education

Level 4+ English and Maths	level 5+ English and Maths	Achieving A-C	Achieving A-G	2+ A-Level passes
1.000	0.999	0.564	1.000	1.000

Correlation Score of Property Type

Detached	Flat	Terraced	Maisonette	Detached bungalow	End terrace
0	0.001	0.071	0.228	0.301	0.596
Semi-detached	Bungalow	Town house	Cottage	Semi-detached bungalow	
0.809	0.883	0.995	0.999	1.000	

Correlation Score of Rooms

Number of bedrooms	Number of receptions	Number of bathrooms
0.252	0.677	0.729

According to the feature selection result, house data like employment, family structure, housing situation (i.e. owned, rented, with or without mortgage) and council tax of house area cannot be seen having any relation with house price. Additionally, the interests and newspaper tendency of people are also unrelated.

However, the crime rate, demographic structure, people's rating and education level of that area do have an impact on house price. On the other hand, the intrinsic features of the house, such as property type, number of bedrooms/bathrooms/receptions, unsurprisingly, are also significant in deciding house value. A summary of the result has been given below:

Summary of the significance of intrinsic and area data

Feature Name	**Significant**
Property type	Yes
Number of bedrooms	Yes
Number of bathrooms	Yes
Number of receptions	Yes
Rating	Yes
Demographic	Yes

Education	Yes
Crime	Yes
Council tax	No
Housing	No
Employment	No
Family	No
Interest	No
Newspaper	No

5.2 Meaningful words

There are quite a lot of meaningful words that do have an influence on house price. I used word of bags model to convert the document into a vector of keywords. The following result shows the some of the words that can make sense in house prediction selected by Chi-Squared Test:

Part of the meaning words

oven	washing	machine	heaters	perimeter	Insulation
Drainer	brilliant	near	secures	supermarkets	Friendly
Later	identification	matters	descriptions	Care	apply
planning	Ready	reliable	refurbishment	obtaining	domestic
externally	Hood	guidance	professional	mile	housing
roads	Mentioned	endeavour	fact	especially	

Those keywords are in fact very interesting. It is easy to dig out something that is consistent with people's perceptions of evaluating house value. For example, from adjectives such as `'brilliant'`, `'friendly'`, the model is in fact evaluating the overall quality of the house. And nouns such as `'oven'`, `'washing'`, `'machine'`, `'heaters'` are listing the home appliances or equipment of the house. Additionally, `'insulation'` can be associated with sound insulation. `'near'` is considering the geographical location. Even `'supermarkets'` can give the information that the model is learning how to take convenience of the house into consideration when doing house prediction.

5.3 Performance of different model

In this chapter, models trained by 4 different data combinations will be compared.

1. Area data and intrinsic house data without description and image
2. Description only (preprocessed by bag of words first)

3. Area data, intrinsic house data and description without image
4. Image only (preprocessed by VGG 16 first)

The performance of models is evaluated using Root Mean Square Error (RMSE). It is usually used for compared the differences between actual observing value and our model prediction value. This is the calculation formula of RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)}{m}}$$

Where m is the number of instances, here it simply means the number of the house recordings. $\hat{y}_i$ means the value predicted by the model (predicted price) and y_i means the actual value (actual price).

The performance of different models

#	1	2	3	4
Basic Liner Regression	50801	61597	54278	92000
Lasso Regression	49234	61570	48611	90603
GBDT	47166	55000	44309	90648

Here are the conclusions derived from above results:

1. The performance of Basic Linear Regression is close to Lasso Regression. However, in Lasso Regression all the features have been used i.e. no feature selection has been undertaken, and in Basic Linear Regression I have used feature selection. Finally those two models have roughly the same effect, this again verified what I have mentioned in chapter 3.3.2, the only difference between Basic Linear Regression and Lasso Regression is that Lasso Regression naturally comes with feature selection functionality.
2. Using combination 2 (description only) to finish the prediction job can also have an acceptable result of which the RMSE is close to combination 1. This proves that the hidden information inside the description is equivalent to the intuitive information from combination 1. It also demonstrates the value of those descriptions, even not that objective, but are indeed informative.
3. Case 3 combined the data in combination 1 and combination 2. Hence it is not surprised that the model trained by case 3 will have the best performace.
4. Case 4 (images only) gives a relatively poor performance. This is reasonable because only one image has been used and the content of the image has not been guaranteed. Some images show the front yard of the house and others show the inside room. Obviously, it cannot give enough information about the house it is currently predicted. On the other hand, pretrained model VGG16 has not been well tuned to perfectly suitable for those house images. In short, more images and more model optimization work need to be done to get a better performance.

Single Feature Prediction

The result of feature selection has shown some significance features. The idea here is using single feature to train the model, and then if the above conclusion is correct, then related features should have relatively small RMSE. And here is the result:

Feature Name	**RMSE**
Property type	76532
Number of bedrooms	79155
Number of bathrooms	80430
Number of receptions	81449
Rating	83738
Demographic	85430
Education	86578
Crime	87444
Council tax	90212
Housing	2.384e15
Employment	4.245e18
Family	6.571e9
Interest	3.756e21
Newspaper	3.443e10

All the significant features found by the feature selection process have a reasonable RMSE and those unrelated features have led to a RMSE which is extremely large.

5.4 Comparison with Zoopla

To best evaluate the model's performance, I have step further to compare my model with Zoopla. Zoopla also have its estimation service[6]. Additionally, I have calculated the Zoopla's estimation RMSE and my lowest one i.e. 44309.

Comparison Result

	Mine (GBDT)	Zoopla
RMSE	44309	35425
Percentage Error	22.43%	15.37%

[6] https://www.zoopla.co.uk/house-prices/browse/nottingham/

6 Summary and Reflections

6.1 Contributions and reflection

6.1.1 Personal achievements and contributions

In this project I have implemented several tools including:

1. A crawler based on Scrapy. It can crawl up to 5000 house data from Zoopla in one hour without breaking the robot rule released by Zoopla. It does not have a high requirement for the performance of computer. In fact, it can even run in Raspberry Pi[7] in a considerable speed.
2. A complete house prediction framework including the work from data preprocessing to result exporting using Python. The framework has been combined with Jupyter Notebook, which means all the processes I mentioned above will be taken in an interactive way.
3. House prediction models with an 22% error percentage (Zoopla 15%).
4. A web server for visualization developed by Node.js. It is designed to be powerful as well as keeping a very high extensibility for further development in the future. Google Map API is also integrated in to avoid repetitive coding. And also the online house price predictor is implemented which can be regarded as a combination of those three tools.

Most current work in this field, as I mentioned at the beginning of my paper, is still using data provided by others. On one hand, the authenticity of the data is doubtful. They can be nice made up only for machine learning use. Getting wonderful performance based on these data does not have any practical significance. On the other hand, data cleaning may already happen before it is downloaded. This seems to reduce the difficulty of data preprocessing but in fact leads to an incomplete experimental process. In my dissertation, all the data are gathered by the crawler from Zoopla which is a website for housing transaction. All the house data downloaded are 'on sale' at that moment. Hence, one of the notable point my dissertation is the absolutely authenticity of data. Even I did not have an extraordinary result. But the result is still valued for true.

Another notable point is that I have used four kinds of different data including intrinsic data, area data, image data and nature language data (house description). On one aspect, I explored the most common way to solve this prediction problem by Lasso Regression or GBDT. In another aspect, only quite a few papers have been found to finish the same task by deep learning. The similar situation also happens to nature language processing. My research work has filled the hole of using deep learning and nature language processing to do machine learning.

---

[7] The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

6.1.2 Some changes

During this whole year, I have struggled to finish the tasks as much as I can. Although the final goal of this project has been almost reached. There are still some aspects that I could not manage to achieve because of lacking time or unsolved the difficulty I met. For example, the original plan is to crawl house data over the whole UK region. However, after keeping running the crawler for one month, over 400 million house data have been downloaded but the crawler was still half way to finish. After carefully thinking about the trade off, I decided to make a change. Insisting on the old plan will cause bigger trouble not only in the process of crawling but also the in model training. On one aspect, the crawling time required may increase exponentially but I do not have sufficient time to wait. And worse than that, a higher level of computing resource and training time could delay my progress terribly. Hence, I gave up the original plan and change the project's region to Nottingham.

Another change is about the final visualization. I originally intended to build a mobile application as a platform for demonstrating my dissertation result. However, when I realized that both two smart phone systems (Android and iOS) are widely used, and there are also many PC users. I decided to develop a web application instead to make it more applicable. Therefore, people from various platforms can have a look of the result conveniently. This wise change not only eased my programming burden (do not need to develop different tools for different platforms) but also achieved the same goal in different way.

6.1.3 Project Appraisal

As the first long-term project I have involved, I am very satisfied with what I have done. On one aspect, the great success in the dissertation has given me confidence that I have the ability to design, manage and implement a scientific project. Additionally, it also gives me the inspiration on applying my knowledge in machine learning to the real life. During the process of dissertation, I tasted the joys and sorrows of life. I am sure this experience will stay forever in my memory and be one of the greatest moment in my life.

Specific to my study, I improved my programming skills in Python by using several scientific packages including Numpy, Pandas, Matplotlib and Scikit-learn to set up the framework. They are useful coding experience which has deepened my understanding of machine learning. At the same time, using crawler framework Scrapy to get the house data by myself is also a good practice.

6.1.4 Future consideration

Even quite a lot of work has been done, there is still plenty of scope to push that further. If more time is given, I believe I can do better in the following aspects.

Data: Currently I hold roughly two thousand house records within Nottingham. The volume of the dataset is relatively small for a project containing heavy model like deep learning. I believe the project will be more complete if the whole UK's house data has been used. On the other hand, only one single feature selection algorithm has been used. In fact, I have not done too much features mining work or features engineering. I can

dig deeper and get more comprehensive knowledge from those data if more time is given. Except the depth of data, I also want to extend the breadth of data. 'right move'[8] is another website in UK for house sell and rent, I plan to download more information from this website and then merge all the data together to get a comprehensive dataset. In this way, I am sure the final performance of my model will be better.

Model: Although I have tried and compared those above four models, it still cannot be regarded as an all-sided project. In other words, for both model optimization and models choosing, there are extra space of improvement. Detailed to the project process, I have just used brute force (loop) to compare different parameters. This is neither wise nor scientific. The best way should be use parameter choosing algorithms to find the most suitable parameter. Additionally, more machine learning techniques like Random Forest, Bayesian, Support Vector Machine can be tried during the process. However, only four models are involved in my project. In short, I can take more machine learning techniques into my consideration and have a more scientific parameter choosing process.

Web: The final web application is used for visualization only. This does not have too much practical significance. If enough time is given, I plan to write a browser extension/plugin for Zoopla. It can predict the price of the house which the users are currently browsing using the model I trained. Then users can get a rational reference of the house price. From another point of view, to convince users to trust the result of my prediction, more evidences should be displayed in the website to support my result.

6.2 Project management

The version control system -- git has been used throughout the project. It can be very helpful when sometimes the work need to roll back. In addition, to achieve the final goal of this dissertation before the deadline, a detailed schedule has been made early. The whole work has been spited into nice parts and is planned to be finished within 27 weeks. Here are the goals and the completion result of each part.

Week 1

Goal: a. Write the project proposal.

b. Review Python, the language grammar.

c. Prepare the web knowledge for writing web crawler.

Result: All the above goals have been finished at the first three days. I wrote several scripts in Python and got to know the famous crawler framework Scrapy. And then I tried to find some related machine learning packages in Python.

Week 2 – Week 3

Goal: a. Implement a crawler for the house data on Zoopla using the framework Scrapy.

---

[8] http://www.rightmove.co.uk/

b. Get familiar with data processing and machine learning related machine learning packages.

Result: I underestimated the difficulty of implement the crawler, I spent large amount of time on reading the official document of Scrapy and cannot finish it in two weeks.

Week 4 – Week 6

Goal: a. Continue to learn these machine learning and data processing packages.

b. Finish the web crawler and start to crawl all those data.

Result: The crawler has been finished but the crawling process, even much quicker than human, was still time consuming. In the other hand, I have learnt to use several packages including Numpy, Pandas, Matplotlib and Scikit-learn.

Week 7 – Week 9

Goal: a. Start to use Python to preprocess the house data collected from Zoopla. The ideal result is to get a clean and integrated result that is ready to do data analysis.

b. Finish interim report before deadline.

Result: The actual process was not as easy as planned, sometimes the program crashed for unknow reason. I was always confused by the obscure track information because of lack of experience. However, I have struggled to finish my interim report.

Week 10 – Week 15

Goal: a. Clear all those unfinished or overdue tasks during the Christmas.

Result: Even spent too much time on preparation for the master interview, I have managed to finish the data preprocessing and some basic analysis after that. I was ready to go further to the next step.

Week 16 – Week 18

Goal: a. Abstract above process into several functions to improve the readability of my code and build up my process framework on the top of it.

b. Allow time to catch up other modules coursework.

Result: Above goals have been achieved in time. The only job is considering how to design function parameter and its functionality. There is not too much technique research need to do during those two weeks.

Week 19 – Week 21

Goal: a. Review mathematical knowledge including linear algebra, calculus and Probability which is required for the purpose of understanding those machine learning algorithms.

b. Have a comprehensive understanding of different machine learning models based on the math.

Result: It went as planned. Too much mathematical content was involved. I was struggled to understand questions like 'how Gradient Descent works', 'how deep learning can update its parameter by backward propagation'.

Week 22 – Week 24

Goal: a. Use different models including Lasso Regression, Gradient Boost Decision Tree and VGG16 to predict house price and compare their performance.

b. Set up visualization frame to better demonstrate my dissertation result.

Result: Since all the work has been well prepared, everything at this stage was smooth. I tried different models and exported their result into a JSON file. And then this file was used to do visualization in my web framework. Node.js is easy to get started and Google Map API document is really user friendly which made it possible for me to finish my work as planned.

Week 25 – Week 27

Goal: a. Finish the final dissertation and get everything done.

Result: this part is exactly what I am currently doing.

As can be seen, the schedule provides a standard reference during the whole process. It helps me to adjust my working pace and gives me the freedom to update my plan accordingly. Although sometimes I could not finish the goals as planned, I would manage to catch up in the next few weeks. In this way, I kept my pace as expected in the end and finished the project successfully.

7 Bibliography

[1] Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H.-T. (2012).

[2] Caroline Hermon, “Machine learning: how new, and how hot?” [Online]. Available: https://blogs.sas.com/content/hiddeninsights/2018/01/04/machine-learning-how-new-and-how-hot/#comments

[3] Limsombunchai, V. (2004, June). House price prediction: hedonic price model vs. artificial neural network. In New Zealand Agricultural and Resource Economics Society Conference (pp. 25-26)

[4] Godin, D., & Zahedi, M. (2014). Aspects of research through design: a literature review. Proceedings of DRS, 281.

[5] Josh Somma (2017) “Why data is so important when it coms to all” [Online]. Available: https://www.squiz.net/learn/blog/why-data-is-so-important-when-it-comes-to-ai

[6] Techopedia “What is web crawler” [Online]. Available: https://www.techopedia.com/definition/10008/web-crawler

[7] Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data preprocessing for supervised leaning. International Journal of Computer Science, 1(2), 111-117.

[8] Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S., ... & Wang, W. (2006). Data mining curriculum: A proposal (Version 1.0). Intensive Working Group of ACM SIGKDD Curriculum Committee, 140.

[9] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

[10] Greenwood, P.E.; Nikulin, M.S. (1996). A guide to chi-squared testing. New York: Wiley. ISBN 0-471-55779-X.

[11] Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the lasso". Journal of the Royal Statistical Society. Series B (methodological). Wiley. 58 (1): 267–88. JSTOR 2346178.

[12] Belsley, David A.; Kuh, Edwin; Welsch, Roy E. (1980). Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. New York: Wiley. ISBN 0-471-05856-4.

[13] Vadim Smolyakov (2017). “Ensemble Learning to Improve Machine Learning Results” [Online]. Available: https://blog.statsbot.co/ensemble-learning-d1dcd548e936

[14] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85-117.

[15] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[16] Will G Hopkins (2000). “Log Transformation for Better Fits” [Online]. Available: http://www.sportsci.org/resource/stats/logtrans.html

[17] Ng, A., & Deisenroth, M. (2015). Machine learning for a london housing price prediction mobile application. Technical Report, June 2015, Imperial College, London, UK.

[18] Nde, S. M. (2017). Fitting a Linear Regression Model and Forecasting in R in the Presence of Heteroskedascity with Particular Reference to Advanced Regression Technique Dataset on kaggle. com.

[19] Scott Chacon and Ben Straub "Git –local-branching-on-the-cheap" [Online]. Available: https://git-scm.com/