Project Report
Real Estate Prediction

Supervisor, Dr David Rossiter
Marc Lamberti - marclamberti.ml@gmail.com

Table of contents

1. Introduction

1.1 Context

This project has been done as part of my course for the MSc Information Technology at Hong Kong University of Science and Technology. Supervised by Dr David Rossiter, I had three months to fulfill the requirements in order to succeed the module. Every three weeks, a meeting was organized to show and report my progress and fix the next objectives.

1.2 Motivations

Being extremely interested in everything having a relation with the Machine Learning, the independant project was a great occasion to give me the time to learn and confirm my interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use Machine Learning in Finance, Medicine, almost everywhere. That's why I decided to conduct my project around the Machine Learning.

1.3 Idea

As a first experience, I wanted to make my project as much didactic as possible by approaching every different steps of the machine learning process and trying to understand them deeply. Known as "toy problem" defining the problems that are not immediate scientific interest but useful to illustrate and practice, I chose to take Real Estate Prediction as approach. The goal was to predict the price of a given apartment according to the market prices taking into account different "features" that will be developed in the following sections.

1.4 Sources

Because I truly think that sharing sources and knowledges allow to help others but also ourselves, the sources of the project are available at the following link:
https://github.com/marclamberti/RealEstatePredictor
Feel free to give me your point of view or ideas for anything you want.

2. The Project

2.1 Data

The crucial element in machine learning task for which a particular attention should be clearly taken is the data. Indeed the results will be highly influenced by the data based on where did we find them, how are they formatted, are they consistent, is there any outlier and so on. At this step, many questions should be answered in order to guarantee that the learning algorithm will be efficient and accurate.

Many sub steps are taken to get, clean and transform the data. I am going to explain each one of them to show how they have been applied on my project why they are useful for the machine learning part.

2.1.1 Getting the data

The first problem was where can I get the data to build a large enough dataset since I want to be able to predict the price for a given apartment according to the real estate agency chosen.

Firstly, I tried to find a dataset already done on the web but unfortunately nothing was available or corresponded to what I was looking for. To address this problem, I decided to use the “web scraping“ which is a technique of extracting information from websites. The idea is to simulate the human behavior on different websites and parse the information to save them. To do this, I used a framework in python called Scrapy that is very intuitive and fast to realize this kind of tasks. It works as following :

1. Make one spider per website that is simply the simulation of a human behavior in order to navigate through predefined web pages and extract the information.

2. The information are extracted according to a user-defined pattern where each time the pattern is found in the current page, the information contained are extracted and saved as an item.

3. An item is next send to different pipelines made by the user where we can apply operations on it such as, formatting, cleaning, etc.

The figure 2.1.1-1 presents the implemented process.

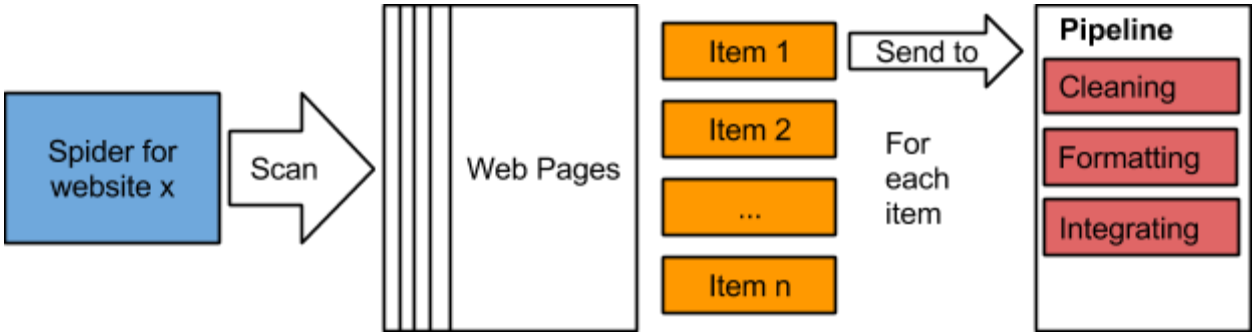


Figure 2.1.1-1: Illustration of the preprocessing pipeline.

The pipeline's modules are the following :

- **Cleaning**, is the first module called to clean the item and verify that all the information in it correspond to the pattern used to extract it. For instance, suppose that we have several estates on a given web page. Each estate is presented using images and many basic information such as, the gross area, the saleable area, the price, etc. When the spider extract them, it is not unlikely that some noise (like extra characters) was present with the value, or just the value does not exist. The cleaning module removes the noise, and check that all the values are not empty, otherwise the item is dropped. This is done for simplicity, indeed, it could be better to try to inference them later. After the cleaning part done, the item is sent to the formatting module.

- **Formatting**. the second module is used to format the item's values as we want. A basic example could be for the price, initially got being string type, is converted as float. This is done for every numeric values. The item formatted is then sent to the last module called Integrating.

- **Integrating**, this module, the last one, is basically the one in charges of saving the items in the format that we want. It also checks that there is no redundancies between the tuples. In my case, I decided to save them in an excel sheet for each website.

The process presented is used on three sources that I have chosen according to their similarities in terms of presentation of the estates and if they give the same information about them. The websites are, **hongkonghomes.hk**, **okay.com** and **squarefoot.com.hk**.

The six **features** extracted for each estate were the following :

1. area
2. gross area
3. saleable area
4. number of bathrooms
5. number of bedrooms
6. price

At the end, I had three excel sheets containing every estate with its corresponding features for each web site.

2.1.2 Attribute Types

After having collected the data to build our dataset regrouping the three websites, an important step is the data preprocessing. What is done during this phase will influence directly the result of the machine learning algorithm. I was firstly interested by what are my attribute types, how can I describe my data with basic statistical measures to get first insights, how my attributes are similar to each other and what kind of techniques can I use to visualize my data.

The figure 2.1.2-1 shows the attributes with their respective types.

Attribute	Area	Gross Area	Saleable Area	Nb Bathrooms	Nb Bedrooms	Price
Type	Nominal	Numeric	Numeric	Discrete	Discrete	Numeric

Figure 2.1.2-1: Table of attributes with their types.

The difference between continuous and discrete variable is that for the first one, the variable can take any value in a given range (like float, 1,2334...) and it is infinite while the other one can take only certain value which are countable (1,2,3,4) and can be also infinite. It is necessary to unified the type of our variable if we want to get correct results as we will see in the Machine Learning part.

This gave me also the information that the Price, that is the variable that I want to predict, is continuous meaning I have a **regression problem** and not a classification problem that would be the case if Price was discrete.

2.1.3 Basic Statistical Measures

With the type of my attributes in mind, I used basic statistical measures on my data such as the mean, the mode, the standard deviation, etc. For the rest of the rapport I will be focus only on the data coming from hongkonghomes to keep the reading easy to follow.

The results of the measures are shown in the figure 2.1.3-1 below.

	gross_area	saleable_area	nb_bedrooms	nb_bathrooms	price
count	1468.000000	1468.000000	1468.00000	1468.000000	1.468000e+03
mean	1487.931880	1159.188011	2.58515	1.910763	3.464037e+07
std	950.474067	751.984001	1.04054	0.807814	4.331363e+07
min	300.000000	223.000000	0.00000	1.000000	0.000000e+00
25%	808.000000	610.000000	2.00000	1.000000	1.300000e+07
50%	1231.000000	956.500000	3.00000	2.000000	2.100000e+07
75%	1906.750000	1509.000000	3.00000	2.000000	4.000000e+07
max	10061.000000	7534.000000	7.00000	8.000000	5.500000e+08

Figure 2.1.3-1: Basic statistical measures of the attributes.

With the following result I was able to notify some interesting such as :

- The is no missing value.
- The mean for the number of bedrooms and number of bathrooms is not meaningful.
- Some estates don’t have any bedroom but all have at least one bathroom.
- The Q1, Q2, Q3 (25%, 50%, 100%) are good indicators of the shape of the different attribute. The Q2 being the median that is not influenced by outliers unlike the mean.
- According to the max values, there are obviously outliers like for gross area where 75% of the data have a median about 1906, 10061 appears clearly to be an outlier as well as the max value for number of bedrooms.
- The standard deviation gives also an indication about what should be considered as outliers, but it is not a robust technique since the standard deviation use the mean to be computed.
- Finally, the minimum value of the price is 0 meaning there is at least a noisy tuple in the dataset.

Notice that the variable area is not present because it is a nominal attribute so the measures above would be meaningless. Nonetheless, the figure below shows the information.

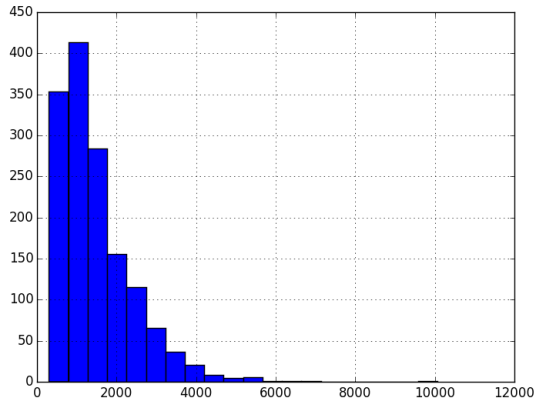
```
count                  1468
unique                   29
top       Mid-levels West
freq                    322
Name: area, dtype: object
```

Figure 2.1.3-2: Basic statistical measures for nominal attribute.

Here, I learnt that as well as the other attributes, there is no missing value, there are 29 different area, the area that appears the most often is Mid-levels West with a frequency of 322 (it appears 322 times).

2.1.4 Visualizing The Data

Next, I tried to figure out what kind of chart I could use and it turns out that each one has its own advantage depending on what we want to visualize. Let's start with histogram.
A histogram is graphical representation of the **distribution of a continuous data**. It estimates the probability distribution. It is composed of bins, representing a range of values, on the x-axis there are the values, on the y-axis there are the frequencies of the bins. The histograms gave me a better intuition of how my variables were distributed.



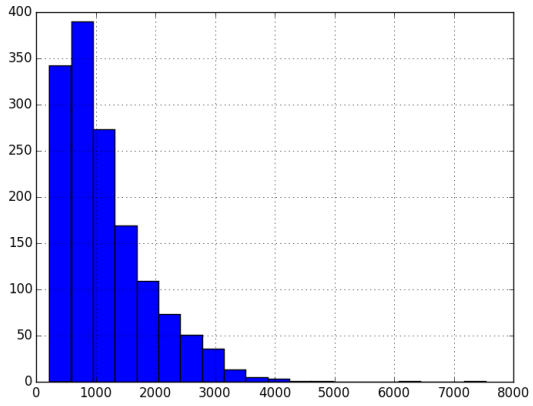


Figure 2.1.4-6 / Figure 2.1.4-7: On the left, the distribution of the variable gross area; On the right the distribution of the variable saleable area.

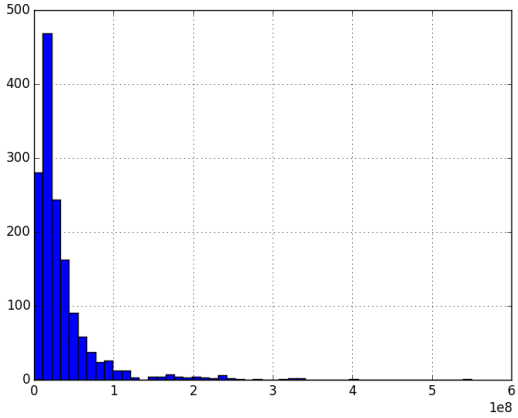


Figure 2.1.4-8: Distribution of the price variable.

Among these variable distributions (respectively, gross area, saleable area and price) I clearly confirmed my first intuition that they are not normally distribution and in my case, they are all skewed on the right (positively skewed).

About the discrete variables, areas, number of bathrooms and number of bedrooms, I used another chart called bar charts.

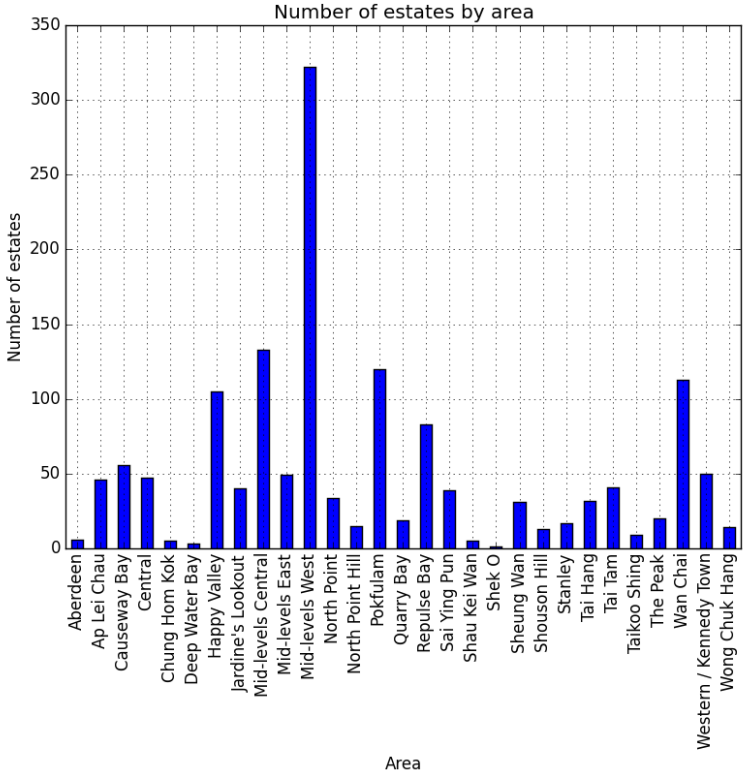
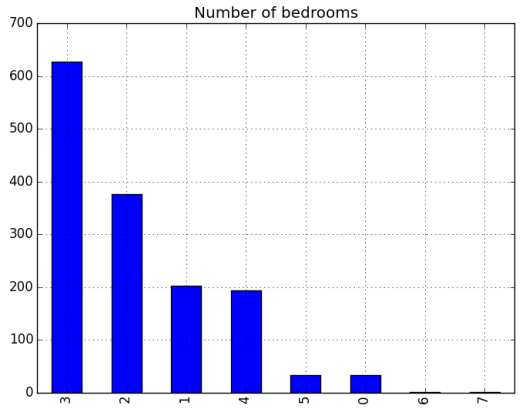


Figure 2.1.4-9: Number of estate over the areas.



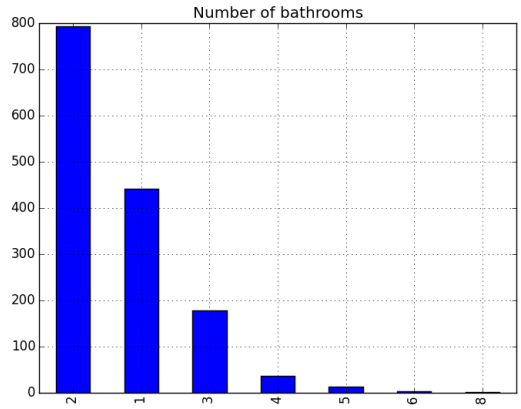
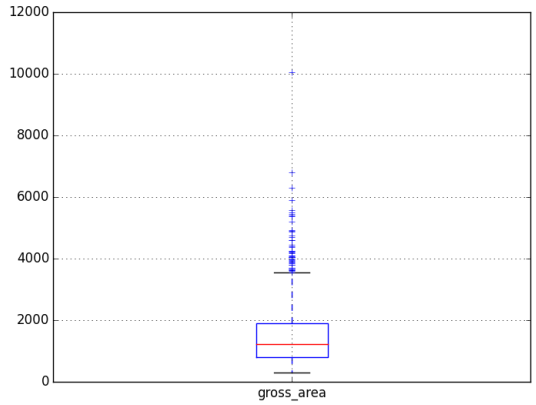


Figure 2.1.4-10 / Figure 2.1.4-11: On the left, distribution of the number of bedrooms; On the right distribution of the number of bathrooms.

The second chart that is extensively used is the boxplot.



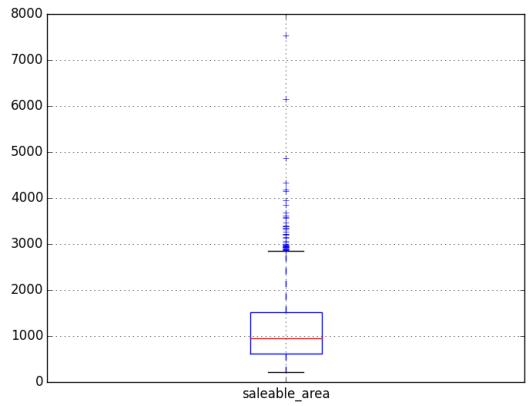


Figure 2.1.4-1 / Figure 2.1.4-2: On the left, box plot of gross area. On the right, box plot of saleable area.

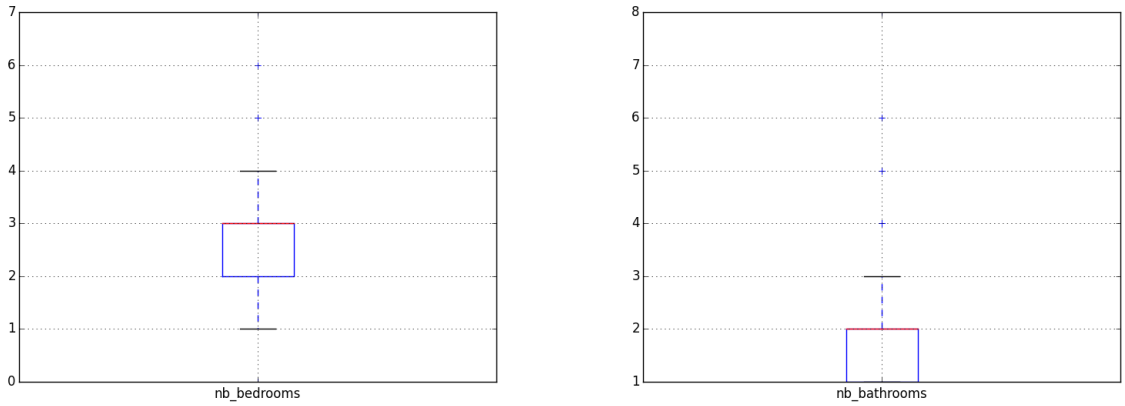


Figure 2.1.4-3 / Figure 2.1.4-4: On the left, box plot of number of bedrooms. On the right, box plot of number of bathrooms.

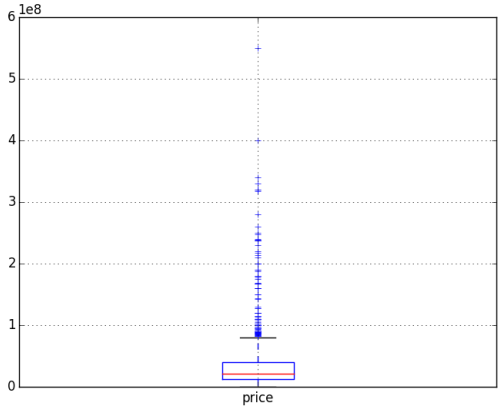


Figure 2.1.4-5: Box plot of the price attribute.

The boxplot is is a convenient way of graphically depicting groups of numerical data through their quartiles. The quartiles are in the box where Q1 is the 25% of the data, Q2 50% of the data, also known as the median (the red line), and Q3 is the 75% of the data. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles. The end of the whiskers represent respectively the highest data within 1.5 IQR of the upper quartile and the lowest data within 1.5 IQR of the lower quartile. The points plotted outside the whiskers are marked as outliers. Furthermore, the position of the box also indicates how the data are distributed given a variable. If the box is on left, the distribution is skewed on the right, if the box is on the right, the distribution is skewed on the left. So just using the boxplot, I had a better vision of how are structured my data according the variable specified, there are outliers and only number of bedrooms seems to be normally distributed.

Another approach that I used to see more precisely the distribution of my continuous variables was by using density plot. Unlike histogram that use bins, density plot use a curve to represent the distribution more accurately.

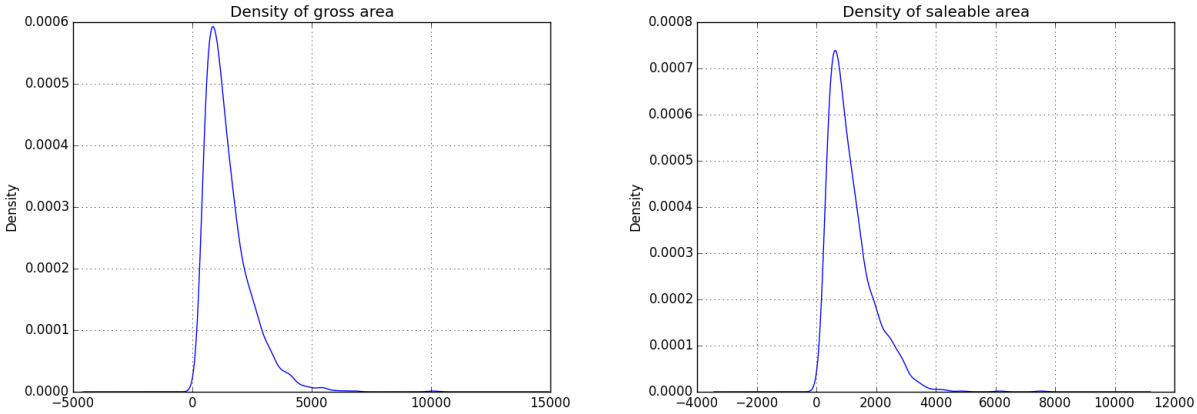


Figure 2.1.4-12 / Figure 2.1.4-13: On the left, density distribution of the gross area variable; On the right density distribution of the saleable area variable.

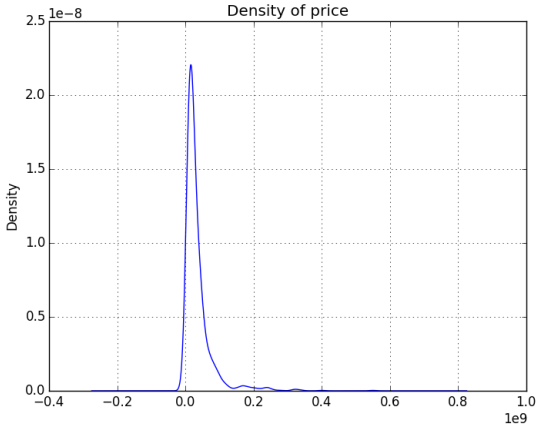


Figure 2.1.4-14: Density distribution of the price variable.

2.1.5 Correlation Analysis And Redundancies

After seeing the distribution of my variables, I computed the correlation between the different attributes and how each one is correlated to the price. Indeed, two attributes highly correlated (not using price) could be useless because they will not have a great impact on the regression result and should be reduced to one (cf: Principal Component Analysis). On the other hand, an attribute low correlated with the dependent variable (Price) could not be really influent on the result.

The figure 2.1.5-1 shows the correlations between the attributes.

	gross_area	saleable_area	nb_bedrooms	nb_bathrooms	price
gross_area	1.000000	0.983416	0.729341	0.826256	0.843920
saleable_area	0.983416	1.000000	0.730483	0.816521	0.812467
nb_bedrooms	0.729341	0.730483	1.000000	0.738505	0.537868
nb_bathrooms	0.826256	0.816521	0.738505	1.000000	0.669899
price	0.843920	0.812467	0.537868	0.669899	1.000000

Figure 2.1.5-1: Correlations between the different attributes.

A better way to see that is graphically using a heat map as shown below.

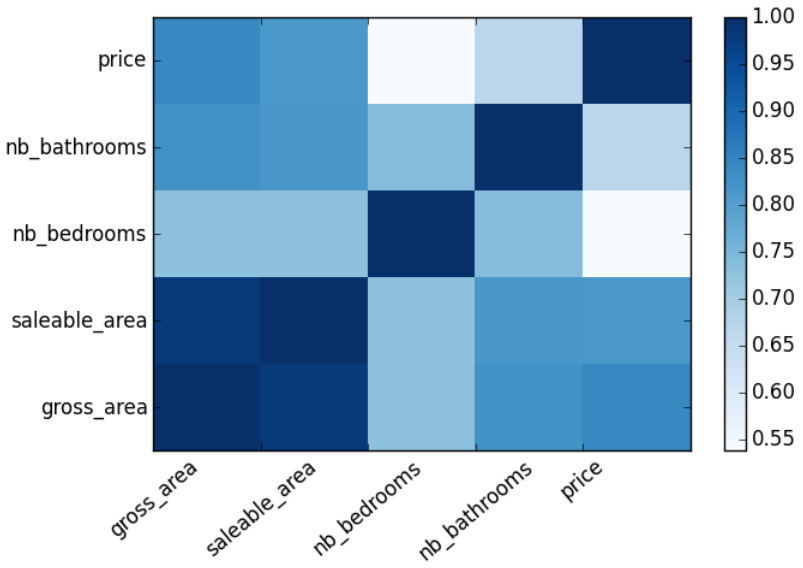


Figure 2.1.5-2: Correlations between the different attributes using heat map.

The diagonal is obviously equal to 1, since it represents the correlation between the same attributes. Gross area and saleable area seem to be highly correlated, meaning by adding saleable area to my learning algorithm I would not have a major changes in terms of results. As I thought, while the gross area increases the price increases too. Number of bedrooms has the lowest correlation to the price among all the variables.

2.1.6 Outliers Detection

According to different measures and charts of my data, there was no doubt about the presence of outliers. The outliers, in my case, are tuple taking abnormal value such as very large or very small, even 0, in one or many of variables. These outliers can affect greatly the results of my learning algorithm. They are several types of outliers :

1. **Univariate**, outliers having an extreme value on one variable.

2. **Multivariate**, outliers having a combination of unusual scores on at least two variables.

The univariate outliers can be detected using the **z-score technique** corresponding to the number of standard deviation the data is from the mean. If negative, below the mean, if positive, above the mean. Z-score technique can be used if the data are **normally distributed**, if the score is $\pm$ 3.0 or beyond, the tuple is considered as outliers. Another technique is by using the rule of IQR (Interquartile Range) where everything below $Q1 - 1.5 * IQR$ and above $Q3 + 1.5 * IQR$ is considered as outliers. The advantage is that it does not depend on the variable's distribution.

The multivariate outliers can be detected using Mahalanobis D2 distance that is a multidimensional version of z-score. It measures the distance of a case from the centroid (multidimensional mean) of a distribution, given the covariance (multidimensional variance) of the distribution. A tuple is considered as outlier if its probability associated is 0.001 or less.

In my case, I used z-score for numerical variable and Tukey's 1.5 IQR rule for discrete to detect the outliers among my data. Since the variables are skewed, I had to apply transformations on them in order to get them normally distributed. Depending on the variables is positively skewed or negatively skewed the following transformation can be applied :

- **Negatively Skewed**, apply power greater than 1 for every value of a given attribute $(x^2,\ x^3,\ etc..)$

- **Positively Skewed**, apply power less than 1 for every value of a given attribute $(x^{2/3},\ x^{1/2},\ x^{1/3}, etc...)$, or try $ln(x)$.

Applying this I got the distributions depicted below.

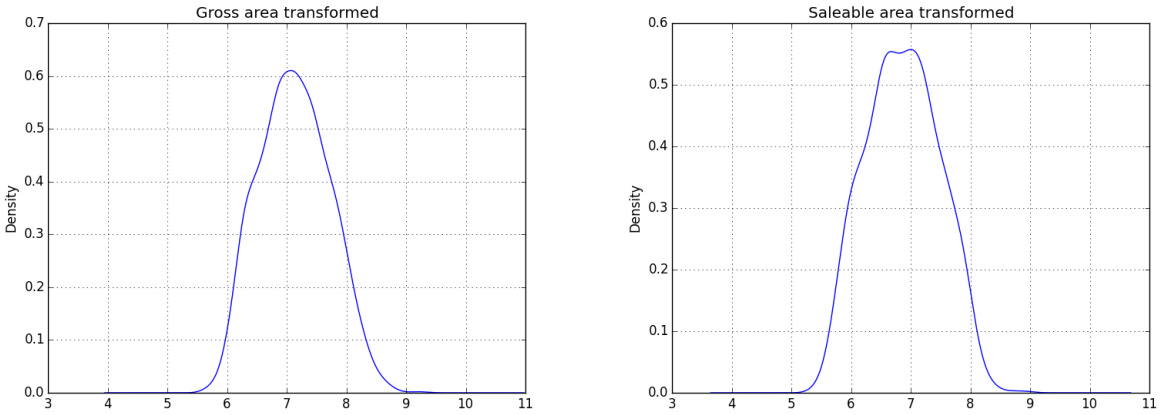


Figure 2.1.6-1 / 2.1.6-2: On the left, the gross area distribution transformed using log; On the right the saleable area distribution transformed using log.

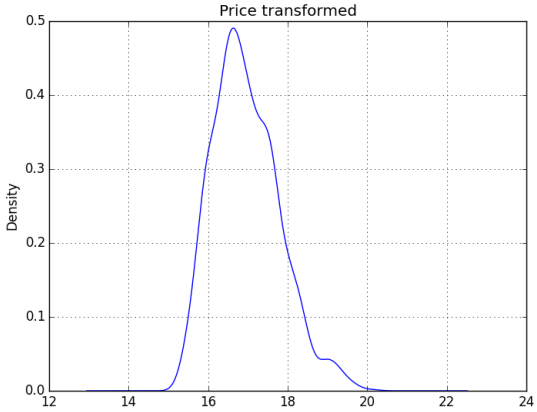


Figure 2.1.6-3: Price distribution transformed using log.

2.1.7 Standardization

The standardization is about feature scaling. It is useful to speed up the learning algorithm and it is usually a requirement. Indeed, when we have several variables with different scale like gross area and number of bedrooms, gross area has a bigger range and it is likely that it will have a bigger impact on the result because of distance measures used in many learning algorithm. The standardization rescales the features so that they will have the properties of the standard normal distribution with $\mu = 0$ and $\sigma = 1$.

So I rescaled each of my variables by applying the following formula on every value :

$$z = x - \mu / \sigma$$

Formula 2.1.7-1: z-score

2.1.8 Nominal Variables

Since my project was about predicting a continuous value (price) that is consequently a regression problem, every variable should be treated as continuous. To deal with that, I used what it is called as “Dummy variable or Indicator variable”. Each value of area is became a binary variable taking 0 or 1 if the given appartement has been sold in the area. So starting from five features (without counting area) I finally had, 29 + 5 = 34 features / variables.

With the data processing done, the next part is the Machine Learning.

2.2 Machine Learning

The Machine Learning part is about trying to find the best learning algorithm for a given problem even if it is highly conditioned by how well the data has been processed and tune some parameters to improve it. Depending on the problem, if it is supervised (meaning we build a model from labeled training set, the value of the dependent variable is known) or if it unsupervised (the model is built on unstructured and unlabeled data), if it is a regression or classification problem, many learning algorithm exist each with their benefits and drawbacks.

2.2.1 Sampling

Given my dataset, I applied a sampling technique in order to divide it into different subset having each its own utility. It is commonly assumed that more we have data to build a model more it will have tend to give good results. Usually the dataset is divided as follow with their respective utility :

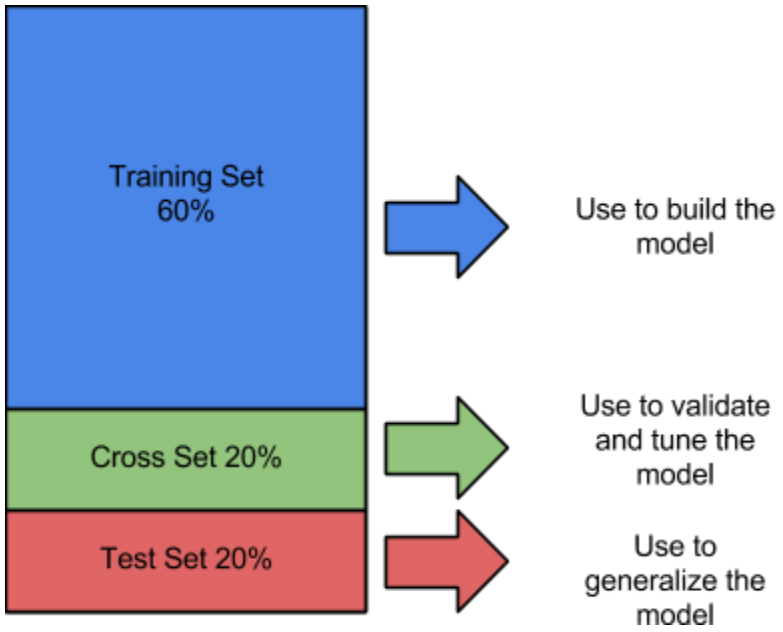


Figure 2.2.1-1: Sampling over the dataset.

The training set as its name indicates it is used to train the learning algorithm. The cross validation set is used to validate the model and make optimizations. Indeed, because the model is built from the training set, we can't check it on the same set because the result would be overly optimistic. It can be also useful for optimization as I will show in the following sections. The test set is used **only** to see how well the learning algorithm is generalized, meaning how it performs with unknown data.

The figure 2.2.1-2 below illustrates the use of the different sets.

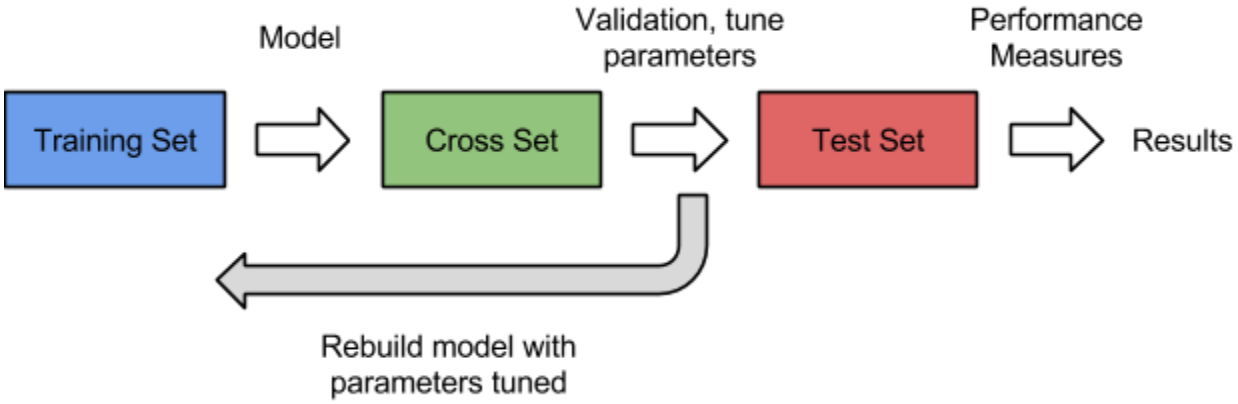


Figure 2.2.1-2:
Illustration of how the different sets are used in a Machine Learning process.

2.2.2 Learning Algorithm

Among the large choice of learning algorithm I chose to use Linear Regression because my dependent variable is continuous. The goal was to model a relationship between y , the dependent variable and $x_1, \dots , x_p$, the independent variable where p is the number of them. The general form of this is given by the following equation :

$$y = f(x) + \varepsilon$$

Formula 2.2.2-1: General form of Linear Regression.

Since I had several independent variable (features) to predict the dependent variable (Price),I was in the case of multiple linear regression (the ^ just means estimation), given by :

$$\hat{y} = \hat{f}(x) + \varepsilon$$
$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$$

Formula 2.2.2-2: Linear Regression with multiple variables.

Where ε is the error term, $\hat{f}(x)$ is the estimation of the function of x , $\hat{y}$ the predicted result, $\beta_{0 \dots p}$ are the unknown parameters that we want to find, $x_{1 \dots p}$ the independent variable.

This learning method is parametric because I assumed that the shape of my model (function) was linear that have simplified greatly the problem because I only needed to estimate a set of parameters. Nonetheless, the disadvantage of using a parametric method is that the model chosen will usually not match the true unknown form of f (function connecting the input variable x to the output variable y). This problem can be addressed by turning the model into something more flexible that can fit many different functional forms for f but can lean to other problems such as overfitting because we need to estimate more parameters.

A precision about the error term. Actually it exists two kind of error, the reducible error also known as residual term (distance between the estimated regression line and the data points), and the irreducible error also known as the error term (distance between the true regression line and the data points). **The reducible error is what I tried to minimize as much as possible and it is actually the goal of the optimization algorithm used to find the unknown parameters**.

2.2.3 The Cost Function

The cost function is what we want to minimize as much as possible. In regression problem the commonly cost function is the mean squared error that quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation, given the following formula :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2$$

Formula 2.2.3-1:
Mean Squared Error used to measure the accuracy of the learning algorithm.

where $\widehat{f}(x_i)$ is the prediction that $\widehat{f}$ gives for the i th observation. This allow to assess the accuracy of the model. A small MSE indicates that the predicted response is closed to the true response variable while a large MSE indicates that the predicted response differ substantially to the true response variable.

2.2.4 Gradient Descent

After having defined the model as linear and the cost function to evaluate its accuracy, I had to find a way to automatically determine the best parameters that minimize the cost function and consequently improve the performance of my model. A well known optimization algorithm is the gradient descent. Basically it tries to find the local optima (or global optima if the function is convex) of the function by taking “steps” until it hopefully converges. The “step” is actually done by taking the derivative (the line tangent to a function) of the cost function. The slope of the tangent is the derivative at that point and gives the direction to move towards.

The figure 2.2.4-1 depicted below shows the gradient descent process.

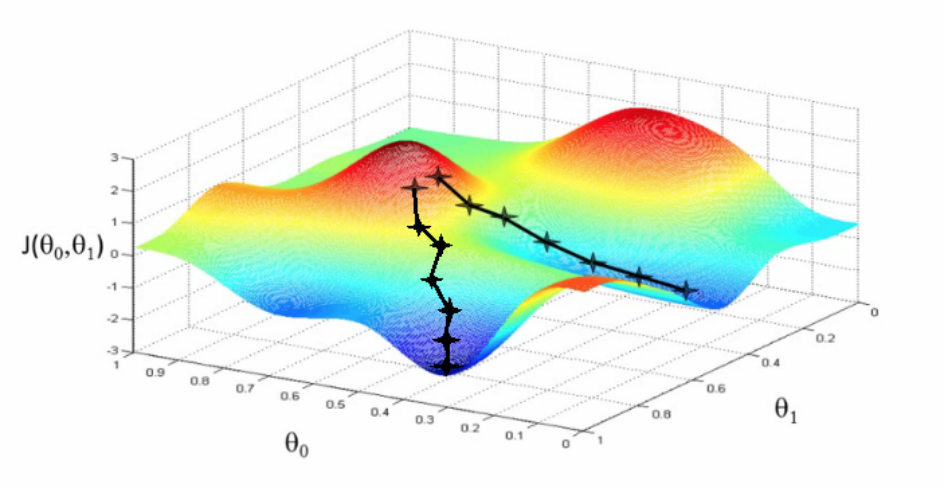


Figure 2.2.4-1: Illustration of gradient descent using two parameters.

where θ_0 and θ_1 are the parameters respectively on the x axis and the y axis and $J(\theta_0, \theta_1)$ the cost function on the z axis. For simplicity, the value of the parameters are given. The crosses represent each step taken by the gradient descent. Here the gradient descent has been run twice with different starting values for parameters θ_0, θ_1.

The gradient descent is composed by two parameters that can be tuned, the learning rate α allowing to converge (or not) quickly by taking large values, and the number of iterations, meaning the number of times the gradient descent will iterate on the observations to approach the local optima. The learning rate has to be chosen carefully because if it is too large, the gradient descent will fail to converge. Usually it takes values such as, 0.001, 0.003, 0.01, and so on.

In the case of linear regression, the concrete algorithm of gradient as following :

$$
\begin{aligned}
&\textit{Repeat until convergence } \{ \\
&\quad \textit{for } j = 0, \ldots, p \\
&\quad\quad \beta_j := \beta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{f}(x_i) x_i^j) \\
&\}
\end{aligned}
$$

where p is the number of parameters, x_i^j is the value of the independent variable j for the ith observation with a special consideration for β_0 since it corresponds to the y-intercept in that case and has no independent variable x, its x_i^j is a constant equals to 1. Due of that, I had to add a constant column equal to 1 in first position for all of my sets. It is important to understand that the parameters have to be updated simultaneously.

In my case I set the alpha rate at 0.0003 and the number of iteration at 100000. Since the linear function is convex, meaning it has no more than one minimum and so it is global. Thanks to that, the gradient descent will find the optimal solution.

A way to check that the gradient descent works well with the given is by plotting the cost function over the number iterations. Indeed, since the gradient descent take a step toward the global optima and the function is convex, the cost function should decrease after each iteration until convergence where the decreasing will be smoother. The results are shown in the following figure 2.2.4-2.

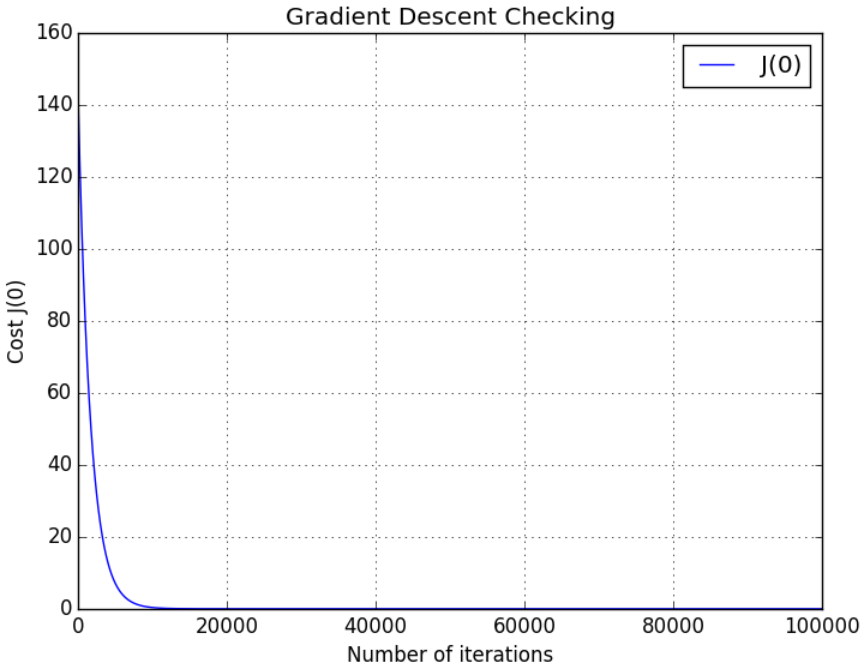


Figure 2.2.4-2:
Cost function over the number of iterations showing the gradient descent converging.

It is clear that the gradient descent has converged to the global optima.

2.2.5 Regularization

The regularization is a technique to address overfitting / high variance problem. Overfitting arise when the model fits the data point too precisely and catch also noisy or outlier points. The model does not generalize well with new unknown data. This problem can be due to a complex model with many features. The regularization will penalize the parameter β_p by reducing them.

Given the initial cost function :

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2$$

Formula 2.2.5-1: Cost Function without regularization.

Regularized it becomes :

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2 + \lambda\sum_{j=1}^{p}\beta_p^2$$

Formula 2.2.5-2: Cost Function with regularization.

where λ is the regularization term determining how much the cost of the parameters will be inflated. Larger λ is, more the function will be smoothed, but a value too large will cause underfitting. Underfitting is when the model fails to catch the trend of the data (to fit the data), and so produce a large MSE. Note that we don't penalize β_0.

The regularization is also applied to the gradient descent and becomes as follow :

$$\begin{aligned}
&Repeat\ until\ convergence\ \{\\
&\quad for\ j = 0\\
&\qquad \beta_0 := \beta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i)x_i^0)\\
&\quad for\ j = 1,...,p\\
&\qquad \beta_j := \beta_j - \alpha\left[\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i)x_i^j\right) + \frac{\lambda}{n}\beta_j\right]\\
&\}
\end{aligned}$$

Like for the cost function, β_0 is not penalized in the gradient descent. Regularization is a very good practice and should be always integrated.

One question could be how to choose the regularization term λ. To answer this, a way is to plot the cost function over a set of λ using the training set and the cross validation set.

Concretely, we learn a model using a given regularization term and the training set and then we plot the cost function for the cross validation set and the training set using the model learnt without regularization. This should give this kind of plot :

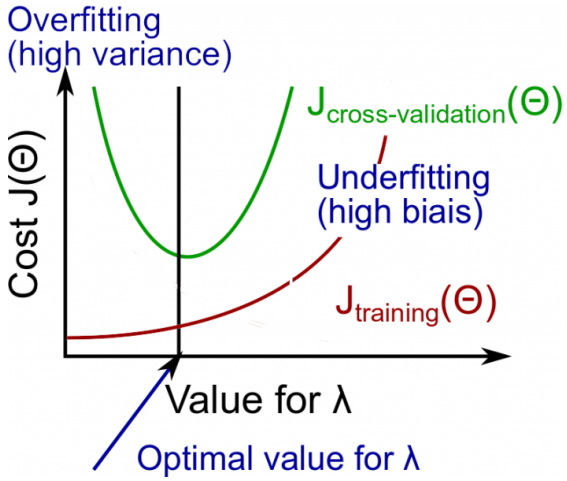
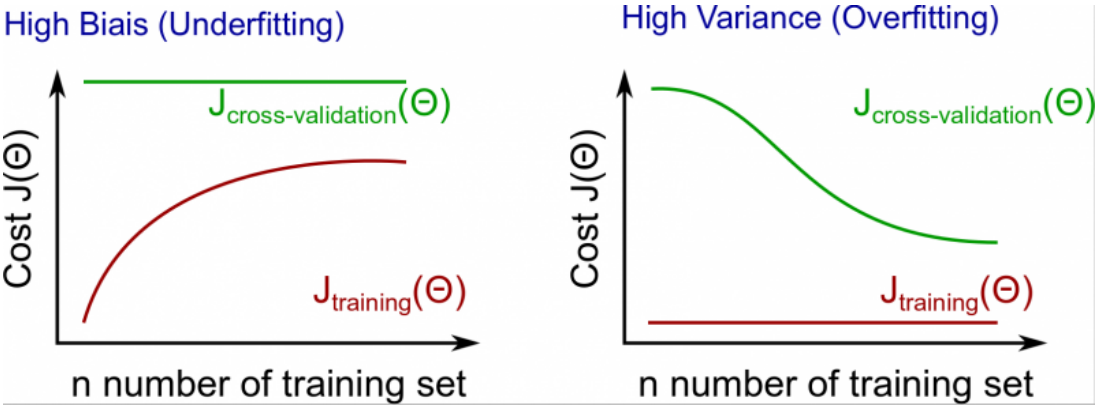


Figure 2.2.5-1:
Illustration showing the curves of the training set and cross validation against the cost function over the regularization term λ.

2.2.6 Learning Curves

Finally, to check how my learning algorithm behaves and if it does not suffer of overfitting or underfitting I used what it is called “Learning Curve”.

Behind this term, it simply refers to the plotting of the cross validation error and the training error (given by the cost function without regularization) over the number of training examples. For $i = 1, ..., n$ we build a model using the ith observation in the training set and we compute its training error. We compute also its cross validation error but using the entire cross validation set and not just the ith observation in the cross validation set. After plotting the curves, there are two possible cases that give intuition about what the learning algorithm suffers :



Figures 2.2.6-1 / 2.2.6-2:
Illustrations showing the training and the cross-validation curves. On the left, when the learning algorithm suffers of underfitting. On the right when it suffers of overfitting.

If the the cross validation curve decrease slowly and the training curve increase slowly along the number of training set, this means that the learning algorithm works well.

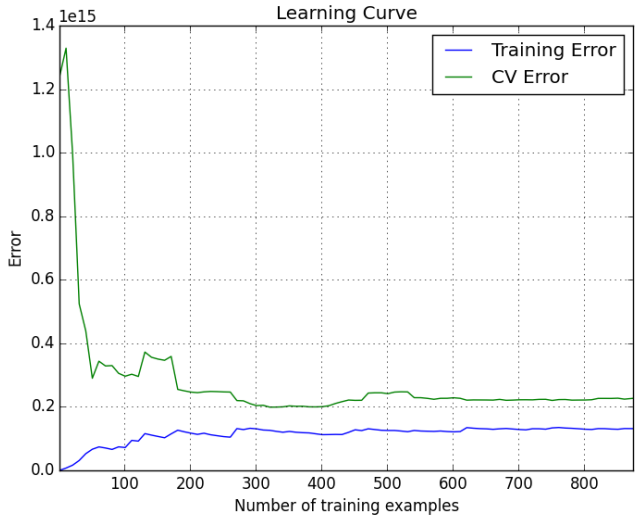


Figure 2.2.6-3:
Chart showing the training and the cross validation error over the number of training examples used to build the model.

Usually, here are the way of addressing overfitting/underfitting problem :

- Overfitting
  - Reduce number of features.
  - Increase λ.
  - Add training examples.
- Underfitting
  - Increase number of features.
  - Polynomial feature.
  - Decrease λ.

3. Results

To check my results, since I coded everything, I wanted to have a reference to know how well is my learning algorithm compared to if I used a library to do the whole machine learning process. As reference, I used scikit-learn and a linear regression with the same features. The figure 3.1-1 below shows what I obtained using polynomial feature.

```
Coefs:
[[ 1.51258936 -0.66501464 -0.14479887 -0.00513912 -0.00620638]]
Intercept:
[ 16.94255589]
Residue
[ 69.2351298]
Residual sum of squares: 69.24
Variance score: 0.87
(Natural Log transform) Original price 16.91 with the predicted one 16.67
(Real) Original price 22000000.00 with the predicted one 17354924.59
==================================
gross_area
[0 1 2]
saleable_area
[0, 3]
nb_bedrooms
[0, 4]
nb_bathrooms
[0, 5]
(Natural Log transform) Original price 16.91 with the predicted one 16.78
(Real) Original price 22000000.00 with the predicted one 19317070.46
Coefs:
[[ 16.94255589]
 [  0.56970874]
 [ -0.27104379]
 [  0.34907431]
 [  0.04527904]
 [  0.03586186]]
Residual sum of squares: 80.26
Variance R^2: 0.85
```

Figure 3.1-1: Results of the scikit library and my learning algorithm.

Where the first results are given from the library and the second results are given using my model.
Consequently the **model** modelizing the relationships between the attribute is :

$\beta_0 = 16.9422589$
$\beta_1 = 0.56970874$
$\beta_2 = -0.27104379$
$\beta_3 = 0.34907431$
$\beta_4 = 0.04527904$
$\beta_5 = 0.03586186$

$$\hat{y} = \beta_0 + grossArea * \beta_1 + grossArea^2 * \beta_2 + saleableArea * \beta_3 + nbBedrooms * \beta_4 + nbBathrooms * \beta_5$$

Formula 3.1-1: Regression model

The variance score gives how well the model fits the data from 0 to 1 and it is computed as follow :

$$R^2 = 1 - \frac{Residual\ Sum\ Of\ Squares}{Total\ Sum\ Of\ Squares}$$

Formula 3.1-2: R^2 formula to check how the regression line fits the data.

Finally, I obtained the following regression curve depicted in the figure 3.1-2.

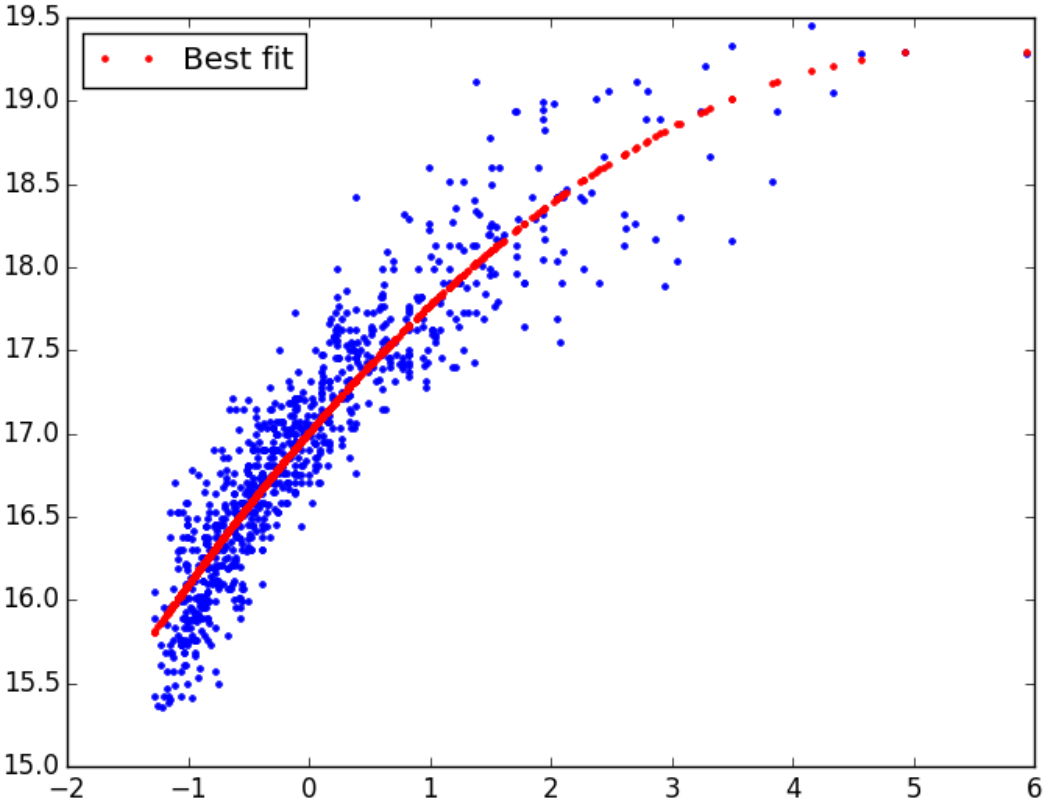


Figure 3.1-2: Chart showing the regression curve.

4. References

To conduct this project the following tools have been used :

- Python 2.7
- Pandas (Library) : http://pandas.pydata.org/
- Numpy (Library) : http://www.numpy.org/
- Scikit-learn (Library) : http://scikit-learn.org/

The techniques used to visualize and preprocess the data has been inspired from the book "**Data Mining Concepts and Technique**".

The Machine Learning part has been greatly inspired by the **Machine Learning course teached by Andrew Ng of Coursera** (https://www.coursera.org/course/ml) and the book "**An introduction to Statistical Learning**".

The following figures have been taken from the Machine Learning course of Coursera:

- *Figure 2.2.4-1: Illustration of gradient descent using two parameters.*
- *Figure 2.2.5-1: Illustration showing the curves of the training set and cross validation against the cost function over the regularization term* λ.
- *Figures 2.2.6-1 / 2.2.6-2: Illustrations showing the training and the cross-validation curves. On the left, when the learning algorithm suffers of underfitting. On the right when it suffers of overfitting.*

5. Appendix

5.1 Minutes of the first meeting

Date : 02/03/2015

Time : 13:40

Place :
Room 3512

Attending :
Marc Lamberti
Professor David Rossiter

Recorder :
Marc Lamberti

5.1.1. Approval of minutes

This is first formal group meeting, so there were no minutes to approve.

5.1.2. Discussion Items

- Selecting the different sources of data
- Developing a crawler for each sites in order to get back the data
- Determining which feature do we keep
- Cleaning the data and dropped items with missing data
- Saving the data in an excel sheet

5.1.3. Meeting adjournment and next meeting

- Scaling the data
- Set up website front end at least to visualize the data

5.2 Minutes of the second meeting

Date : 27/03/2015

Time : 13:40

Place :
Room 3512

Attending :
Marc Lamberti
Professor David Rossiter

Recorder :
Marc Lamberti

5.2.1. Approval of minutes

Minutes approved.

5.2.2. Discussion Items

- Scaling the data
- Website to visualize the data
- First insights about machine learning

5.2.3. Meeting adjournment and next meeting

- Machine Learning to make first predictions

5.3 Minutes of the third meeting

Date : 10/04/2015

Time : 11:30 AM

Place :
Room 3512

Attending :
Marc Lamberti
Professor David Rossiter

Recorder :
Marc Lamberti

5.3.1. Approval of minutes

Minutes approved.

5.3.2. Discussion Items

- Machine Learning
- Linear Regression and possible improvements
- Results obtained

5.3.3. Meeting adjournment and next meeting

- Integrate the system with the website front end
- Write the report

5.4 Minutes of the fourth meeting

Date : 10/04/2015

Time : 14:00 PM

Place :
Room 3512

Attending :
Marc Lamberti
Professor David Rossiter

Recorder :
Marc Lamberti

5.3.1. Approval of minutes

Minutes approved.

5.3.2. Discussion Items

- Report
- Minor modifications