Data Mining For Urban Planning

Modeling Seminars and Projects

Asarbaev A., Feofanov V., Laguel Y.,
Lewandowski M., Räisänen S., Rhalimi M.

January 28, 2018

Chapter 1

Introduction

1.1 Purpose

The world population is increasing exponentially since 1950, when 30% of globe population was estimated to live in the cities. It is estimated that by 2050 this percentage of urban population will more than double, i.e. almost 70% of the global population is expected to be living in cities. This increasing urbanization level brings new challenges on how to perceive and organize the cities that are about to host that many residents. This question raises among others the problem of economical planning of the cities.

This project concerns what is widely called urban planning, for which various mathematical tools are able to give an unbiased estimation of factors which have an affect upon pricing. In order to build a reasonable model about house prices the largest possible datasets should be used in order to build models which are less likely to be affected by outliers as set out in the law of large numbers. One may consider utilizing real estate prices and information, which give a lot of valuable informations about the economic aspects of the city. Models mentioned above may be able to predict some of the cities dynamics.

However, the problem is that these data sets are not easily available as they are owned by real estate agencies or notaries and as a consequence access to these data is rather expensive or simply not possible. In spite of this, it is assumed that enough data may be obtained from Internet pages where people rent, buy, or sell houses and apartments.

1.2 Overview

The project consisted of four main parts. These were:

1. Data extraction
2. Text mining

3. House prices estimates
4. Data visualization

A brief introduction to these aspects of the project are presented. The data extraction part was dedicated to mining the relevant informations from the adequate websites. This involved focused efforts on scraping the data according to the scraping policies of the websites utilized. It was decided to investigate data gathered from the following websites:

- `Leboncoin`
- `Airbnb`

The region of Grenoble, France was focused upon in this application, but due to the features of the presented model, data can be gathered from any city which has data available from either website used. Additional websites could be utilized, but this would involve creating webscrapers custom fit to extracting the data. The goal was to extract as many features as possible, particular attention was focused on gathering the following:

- Geo-localization
- Rental price
- Surface area
- Housing type (i.e. studio, T1, T2, ...)

The project was started with the idea of gathering more data, among which energetic efficiency, building age, car parking availibity etc., but it turned out that people are not willing to share all the details via Internet or they simply do not bother with these aspects, possibly as it may not affect the final price in a significant manner. One of the initial challenges was that it is necessary to crawl the webpages really carefully - most web services track and ban crawlers when too many request per minute are being sent.

The second part of the project, namely text mining, was the one which was believed to have the possibility of adding more information for the project. However, it was not straightforward as on the web-pages users are likely to provide long description of the house or apartment, but usually they do not bother with parsing the data with respect to the given feature. Furthermore, the data given was very varied and focused on various features depending on the entry, which made it very difficult to extract features which had any effect on a lot of the data and thus was not specifically useful in the model for predicting prices. This is a part which could be approached in a further extension of the work, but would also involve finding a dataset where there is lots of additional information in the descriptions. In this

aspect it would be much more useful if it were possible to gain a dataset from Realtors as they generally will expound more upon the details of the housing units they are attempting to sell or rent.

The third part, land price estimates, was aimed to model prices as a function of available features. Given the dataset which was able to be extracted, the features used in the prediction of land prices were the longitude, latitude, home type, and person capacity. Given more complete datasets, this method could easily be extended into situations for further features.

Finally the fourth part of the project was about the visualization of the results on an interactive plot. This was a challenging part, obstacles arose partly to the desire to streamline the report in Python environment. Presented here is a starting point of the visualization part for the sake of comparison with the final product.

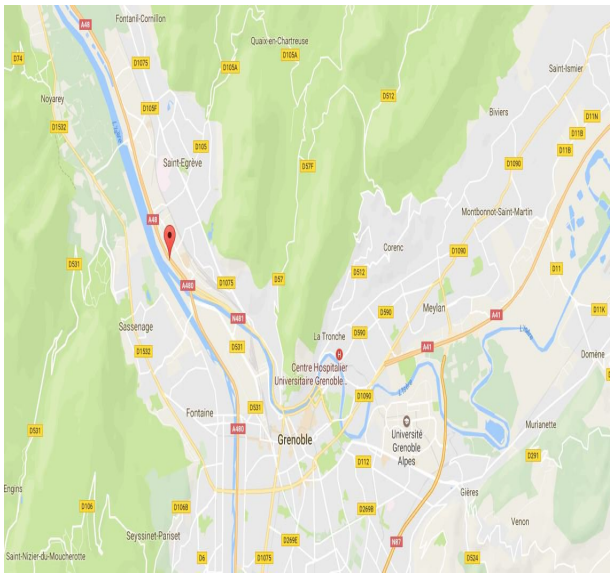

Figure 1.1: Map of Grenoble region before visualization part.

Chapter 2

Mining data from airbnb.com

Chapter 3

Mining data from Leboncoin.fr

3.1 Crawl system

As mentioned above, the main goal of the first part of the task was creation web crawler that allows us to obtain as much useful data from the advertising posts from sites https://www.airbnb.ru/ and https://www.leboncoin.fr/ as it possible. In this chapter it will be presented how It was implemented for **https://www.leboncoin.fr/**.

At the beginning of solving this task there was faced to problem related to web-site structure and information access. To realize data scraping system It was necessary analyze advertising post and understand where is the information located on the web-site and how to retrieve It.

After studying and analyzing advertising, it was found that all parameters are shown on figure (except of geolocation data) are located in the HTML-code of the web-page in the form of text and to extract this information it was necessary to access the HTML-code programmatically, parse it and write ones in the .csv file. Solution of the problem with geolocation data will be presented in the next chapter.

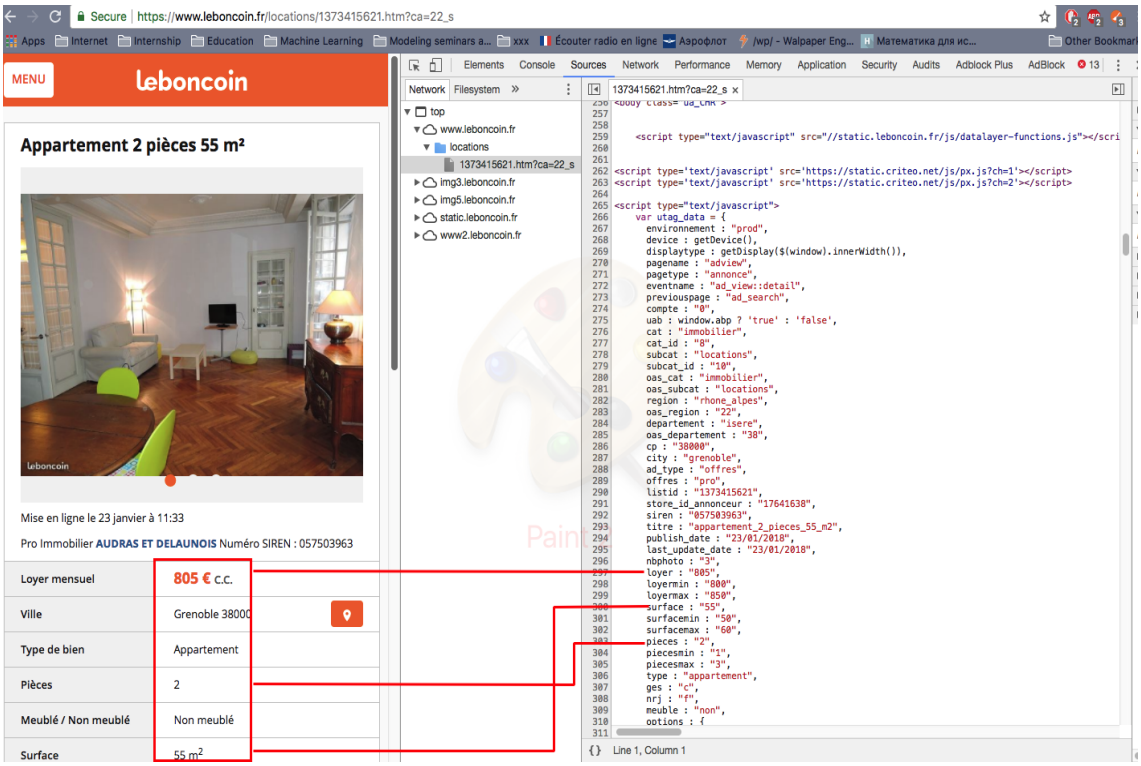

Figure 3.1: Example of HTML-code consisting advertising required parameters

It was received a recommendation from the teaching staff to study "Beautiful Soup" Python library that allowed us to solve above tasks.

Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping[6].

3.1.1 Solution

To solve above described problem was written program code (Appendix A) with using Beautiful Soup library, the main idea of which consists of the next steps:

1. Initializing of cities and postal codes lists. This step allows us to specify the cities for which we are interested to retrieve data.

2. For chosen city from cities list: using function **urllib.urlopen('website address').read()** from **urllib** library and **BeautifulSoup** function from **bs4** library It extracts HTML-code from web-site whose

address was used as the variable of urllib.urlopen().read() function. In our case it is used link of website for Grenoble rental advertising.

3. Using extracted HTML text, Crawler finds numbers and links of all pages that consist of rental advertising for chosen city (in our case chosen city is Grenoble). It saves these links and number of last page for further iterating.

4. For each link of page that was found in the 3 step, using the same functions (urllib.urlopen().read() and BeautifulSoup) from 2nd step, Crawler extracts links of all advertisements on this page and stores this information.

5. For each link of rental advertisements that was found in the 4th step, using the same functions from 2nd step, Crawler extracts necessary parameters and stores these parameters in a dictionary.

6. After the Crawler extracts parameters from the last page and write all parameters in the dictionary, It writes this dictionary in .csv file.

For a more intuitive understanding below is shown a block diagram of the Crawler.

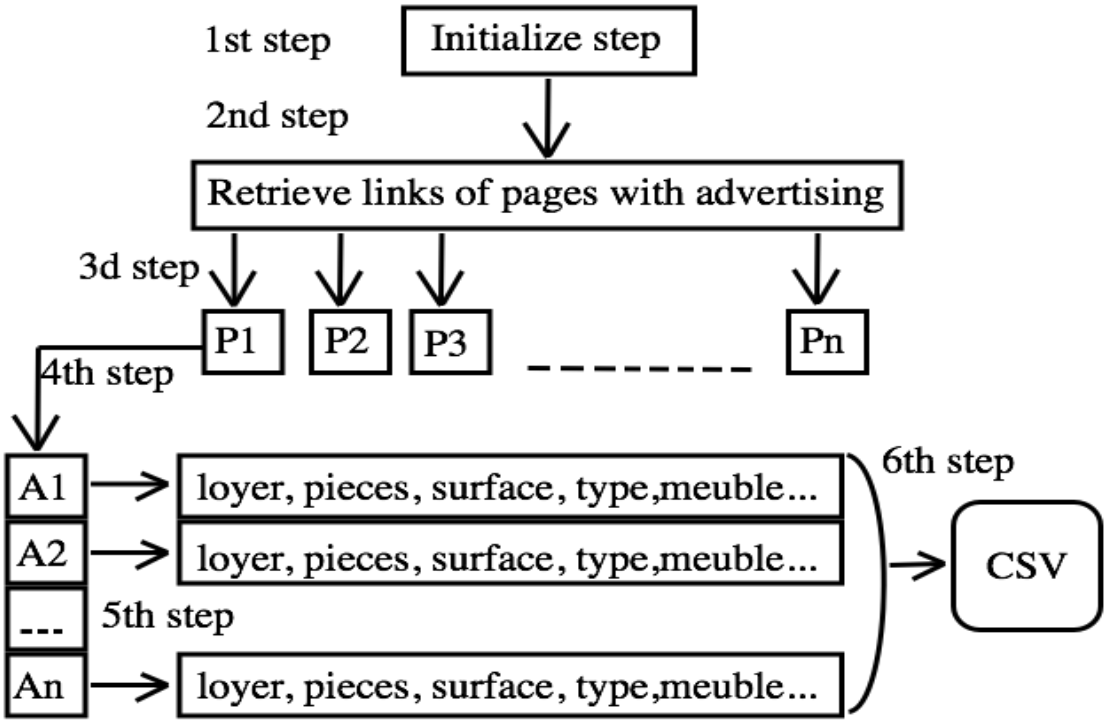

Figure 3.2: The Crawler block diagram

It is worth noting that the main problem in this part was searching of necessary links of pages (step 3), links of advertising (step 4), parameters (step 5) in HTML-code. In our case it was very helpful **soup.find_all()** function from **BeatifulSoup** library and also that the structure of HTML-code was very stable.

3.2 Retrieval of location coordinates

3.2.1 Essence of the problem

Mining of the coordinates on the `leboncoin.fr` is separated as another issue. In fact, HTML-code of an offer page does not store geolocation of the place explicitly. Therefore, it is not enough to use tools for crawling that were described in the section 3.1. There are two approaches to resolve the problem, namely, text analysis of the offer description and automation of actions in a web-browser. The former one tries to find an address into the description. Then, Google Geocoding API can be used for finding coordinates. The latter one gives a possibility to click on the buttons that opens a map with the location. Further, queries for the online map service are analyzed.

3.2.2 Mercator Projection

In 1569 Gerardus Mercator proposed a cylindrical map projection of the globe. The main advantage of this projection is that it preserves local angles around points. At the same time, preservation of area does not take place. It means that the closer an area is to the poles, the more distorted this area is. For example, in the Mercator projection Greenland is larger than Africa, although it is 14 times smaller in land area [1].

Today, for online map applications an adaptation of the Mercator projection is used. Web Mercator represents the globe as a grid of images that are called tiles. Each tiles is created by the Mercator projection [2]. Depending on a scale level, the world is divided by a certain number of tiles. The number is defined as 2^z rows by 2^z columns, where $z = 0, 1, 2 \dots$ is a zoom level.

Retrieving the row and column numbers (y and x) of a point in the Web Mercator given the latitude φ and the longitude λ is achieved by computing the following formulas:

$$x = \frac{\pi}{360^\circ}(2^z - 1)\left(1 + \frac{\lambda}{\pi}\right),$$

$$y = \frac{\pi}{360^\circ}(2^z - 1)\left(1 - \frac{1}{\pi}\ln\tan\left[\frac{\pi}{4} + \frac{\varphi}{2}\right]\right). \tag{3.1}$$

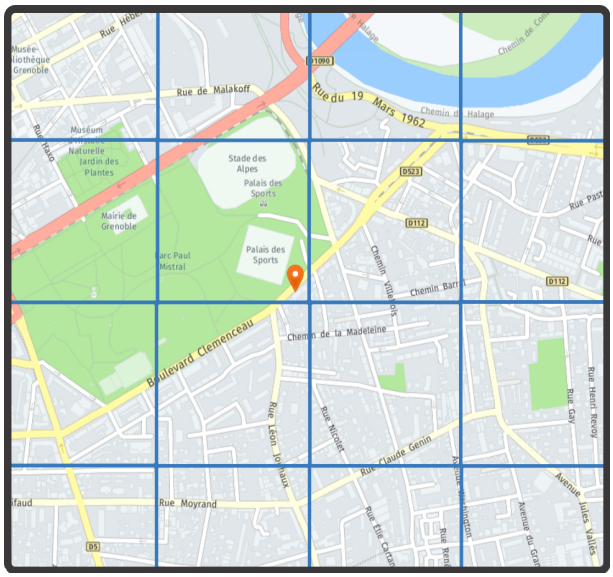

Figure 3.3: In an online-application a map is assembled from tiles: PNG images (usually, 256x256) that have their own coordinates in the Mercator projection.

To have unity in the measurements of the data features, it is desirable to be able to convert Mercator projection coordinates to the standard geographical coordinates, which are latitude and longitude. From the equation 3.1 it is not hard to deduce the necessary formulas for this conversion:

$$\lambda = 360^\circ \frac{x}{2^z - 1} - \pi,$$

$$\varphi = \frac{360^\circ}{\pi} \arctan\left\{\exp\left(\pi - \frac{2\pi y}{2^z - 1}\right)\right\} - \frac{\pi}{2}. \tag{3.2}$$

Chapter 4

Learning

The results of this section are derived from two references : an article that introduced Support Vector Regression [4] and the components of the course Advanced Learning Models relating to kernels which was presented by Julien Mairal in Grenoble INP 2017-2018.

4.1 Objective

The main goal of this section was to estimate from the data retrieved by the two webcrawlers the price of any house in the region studied. It was decided to perform a Support Vector Regression as a result of the great performances of this method as well as the quantity of kernels which are available to be tested.

For educational purpose it was decided to implement an independently produced version of the support vector regression based on the article. A version based on `scikit-learn` library was also utilized in order to compare with the results obtained from the version produced for this project.

4.2 Kernels

Kernel-based methods constitute, with neural network, state of the art methods for many machine learning problems. Support Vector Machines are classes of kernel-based methods popularized in the 1990's. Still very popular today thanks to the theoretical motivations behind them as well as the ability to fully understand them; they continue to offer great performance comparable or even better than deep neural networks for some problems.

The main idea behind kernel-based methods is to map the data which is desired to be classified (or regressed) into a featured space where the task is made easier. This can be very efficiently done thanks to what is called a kernel.

4.2.1 Definition and Examples

First it is necessary to define what a kernel is and give some examples:

Definition 4.2.1. A positive definite kernel on a given set $\mathcal{X}$ is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is symmetric:

$$\forall (x, x') \in \mathcal{X}^2, K(x, x') = K(x', x)$$

and which satisfies for all $N \in \mathbb{N}$, $(x_1, x_2, ..., x_N) \in \mathcal{X}^N$, $(a_1, ..., a_N) \in \mathbb{R}^N$:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(x_i, x_j) \geq 0$$

This second property is called the positive-definite property.
Intuitively, a kernel can be seen as the generalization of a dot product to an arbitrary set : it helps for instance to measure a kind of distance between the elements of this set with the identity : $||x - y||^2_{\mathcal{X}} = [K(x, x) + K(y, y) - 2K(x, y)]$. In what follows $\mathcal{X}$ will play the role of an input space. Hence, kernels can be seen as tools for similarity measure between the inputs.

Some classical examples of kernels are as follows :

Examples.
The following applications can be shown to be kernels :

- Dot product : For $\mathcal{X} = \mathbb{R}^n$, the first classical example of kernel is the dot product :

$$\forall x, y \in \mathbb{R}^n, K(x, y) = x^T y$$

- Polynomial Kernel $\mathcal{X} = \mathbb{R}^n$,

$$\forall x, y \in \mathbb{R}^n, K(x, y) = (x^T y)^n, n \in \mathbb{N}^*$$

- Gaussian Kernel $\mathcal{X} = \mathbb{R}^n$,

$$\forall x, y \in \mathbb{R}^n, K(x, y) = e^{-\frac{||x-y||^2}{2\sigma}}, \sigma > 0$$

Moreover, the following basic properties are also applicable on kernels :

Proposition 4.2.1.
If K_1 and K_2 are positive definite kernels, then :

- *$K_1 + K_2$ is a positive definite kernel.*
- *$K_1 K_2$ is a positive definite kernel.*

- *cK_1 is a positive definite kernel.*

If $(K_i)_{i\geq 1}$ is a sequence of positive definite kernels that converges pointwisely to a function K :

$$\forall x, y \in \mathcal{X}^2, K(x,y) = \lim_{n\to\infty} K_i(x,y)$$

then K is also a positive definite kernel.

4.2.2 Mercer's theorem

In this subsection a little bit more is explained regarding the parallels between a kernel and a dot product.

Definition 4.2.2 (Reproducing Kernel Hilbert space)**.**
Let $\mathcal{X}$ be a set and $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ be a class of function forming a real Hilbert space with inner product $\langle .,. \rangle_{\mathcal{H}}$. The function $K : \mathcal{X}^2 \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$ if :

1. $\mathcal{H}$ contains all functions of the form

$$\forall x \in \mathcal{X}, K_x : t \mapsto K(x,t)$$

2. For every $x \in \mathcal{X}$ and $F \in \mathcal{H}$, the reproducing property holds :

$$f(x) = \langle f, K_x \rangle_{\mathcal{H}}$$

If a reproducing kernel exists then $\mathcal{H}$ is called a reproducing kernel Hilbert space.

The article decided to introduce kernels directly through the following property that formalized this parallel :

Theorem 4.2.2 (Aronszajn's Theorem)**.**
K is a positive definite kernel on a set $\mathcal{X}$ if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that,

$$\forall x, y \in \mathcal{X}, K(x,y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

Aronzajn's theorem is actually a generalization of Mercer's theorem, which was established only for $\mathcal{X}$ being a compact set and K continuous. In its proof, the Hilbert space $\mathcal{H}$ that appears is nothing else than the reproducing Kernel Hilbert space of K and the mapping ϕ is simply defined by $\phi : x \mapsto K_x$.

4.3 Support Vector Machine Regression

Now the approach proposed by Support Vector Machine (or SVM) for regression is presented. In what follows, let define :

- $\mathcal{X}$ as an input space.
- $\mathcal{Y}$ as the output space. In the project's case, $\mathcal{Y} = \mathbb{R}$.

In both case, the objective of the algorithm is to find a function $f : X \to Y$ belonging in a designed space $\mathcal{F}$, that estimates as precisely as possible the right output for a given output. Equivalently, given a dataset $(x_i \in \mathcal{X}, y_i \in \mathcal{Y})_{1 \leq i \leq n}$, it is needed to find a function $f \in \mathcal{F}$ that is a solution of the problem :

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, f(x_i)) + \lambda ||f||_{\mathcal{H}}$$

where,

- l is a loss function such as $l(y, t) = (y - t)^2$
- λ is a regularization term here to prevent overfitting.

Most of the time, the set $\mathcal{F}$ is a parametrized set. For instance, if it is desired to perform a linear regression, F will be $\{x \mapsto \langle w, x \rangle + b, w \in \mathcal{X}, b \in \mathbb{R}\}$

4.3.1 The Representer theorem

The Representer theorem is the essential results that motivates kernel-based approach for classification or regression :

Theorem 4.3.1 (Representer Theorem)**.**

- *Let $\mathcal{X}$ be a set endowed with a positive definite kernel K, $\mathcal{H}$ the corresponding reproducing kernel Hilbert space and $S = \{x_1, ..., x_n\} \in \mathcal{X}$ a finite set of points in $\mathcal{X}$.*
- *Let $\psi : \mathbb{R}^{n+1} \to \mathbb{R}$ be a function of $n + 1$ variables, strictly increasing with respect to the last variable.*

Then, any solution of the optimization problem :

$$\min_{f \in \mathcal{H}} \psi(f(x_1), ..., f(x_n), ||f||_{\mathcal{H}})$$

belongs to the finite-dimensional subspace $Span(K_{x_1}, ..., K_{x_n})$

For particular interest in regards to the problem, the proof of this well known result is provided :

Proof.

Let $\zeta : H \to R$ defined by $\zeta(f) := \psi(f(x_1, ..., f(x_n), ||f||_{\mathcal{H}})$. We want to show that the minimum of ζ holds in $H_S := Span(K_{x_1}, ..., K_{x_n}) = \{\sum_{i=1}^{n} \alpha_i K(., x_i), (\alpha_1, ..., \alpha_n) \in \mathbb{R}^n\}$.
Since, H_S is a finite dimensional space, it generates with its orthogonal the whole space $\mathcal{H}$: any function $f \in \mathcal{H}$ can be uniquely decomposed as :

$$f = f_S + f_\perp$$

with $f_S \in \mathcal{H}_S$ and $f_\perp \in H_S^\perp$. Since H is a reproducing kernel Hilbert space, thanks to the reproducing property mentioned above, we have :

$$\forall i \in \{1, ..., n\} f_\perp(x_i) = \langle f_\perp, K_{x_i} \rangle = 0 \text{ since } K_{x_i} \in H_S$$

Hence,

$$\forall i \in \{1, ..., n\} f(x_i) = f_S(x_i)$$

Moreover Pythagoras theorem gives that :

$$||f||_{\mathcal{H}} = ||f_S||_{\mathcal{H}} + ||f_\perp||_{\mathcal{H}} \geq ||f_S||_{\mathcal{H}}$$

Since ψ is increasing along its last variable, we get that $\zeta(f) \geq \zeta(f_S)$ with equality if and only if $||f_\perp||_{\mathcal{H}} = 0$ Hence, solutions of the problem are all necessarily in $\mathcal{H}_S$

□

This representer theorem is essential insofar as complexity of the Hilbert space H is no longer an obstacle to the optimization problem since we will only look for solutions in its finite dimensional subspace, generated thanks to the dataset $(x_1, x_2, ..., x_n)$.

It was decided to present the proof because it points out the notion of perpendicularity in $\mathcal{H}$. Quite apart from proving this theorem, perpendicularity can be used to reduce again the set of possible functions by applying a principal component analysis on the family $(K_{x_1}, ..., K_{x_n})$. This is indeed very useful when the dataset provides to many examples to run the algorithm.

As a result of the interest and applications of Kernels for the problem the details of the algorithm of the SVM for regression is presented in the following section.

4.3.2 SVM for Regression

Assume there is a training dataset of l variables in $\mathcal{X} := \mathbb{R}^n$ $(n > 0)$, and their associated values in $\mathcal{Y} := \mathbb{R}$. It is desired to find a function f, that will

minimizes the empirical risk associated to this training set.

For an educational purpose, article [**?**] starts with f being linear :

$$f \in F_{lin} := \{(x, w) \mapsto \langle x, w\rangle + b, w \in \mathbb{R}^n, b \in \mathbb{R}\}$$

SVM regression proposes an empirical risk error of the form

$$\mathcal{E}(f) = \frac{1}{2}||w||^2 + \frac{C}{l}\sum_{i=1}^{l} l_\epsilon(y_i, f(x_i)) \tag{4.1}$$

where $C > 0$, ϵ is a small positive number and $l_\epsilon(y, f(x))$ is a particular loss function :

$$l_\epsilon(y, f(x)) := \begin{cases} 0 & \text{if } |y - f(x, w)| \leq \epsilon \\ |y - f(x, w)| - \epsilon & \text{otherwise} \end{cases}$$

It can be demonstrated that this problem can be reformulated into a simple quadratic problem with linear constraints.

In what follows, consider b as a fixed for the moment and the value of b will be discussed later. If K is considered as being the dot product on $\mathbb{R}^n$, it is known that K is a Kernel on X, and the associated reproducing kernel Hilbert space is simply the dual of $\mathbb{R}^n$. Hence, the representer theorem can be applied with :

$$\psi : (a_1, ..., a_n, a_{n+1}) \mapsto \frac{1}{2}a_{n+1}^2 + \frac{C}{l}\sum_{i=1}^{l} l_\epsilon(y_i, a_i + b)$$

and it is known that (when b is fixed) the solution is of the form $f : x \mapsto \sum_{i=1}^{l} \alpha_i K(x, x_i) + b = \sum_{i=1}^{l} \alpha_i \langle x, x_i\rangle + b$.

Hence, the minimization problem can be reformulated as :

$$(P) \min_{\alpha \in \mathbb{R}^n} \frac{1}{2}\alpha^T K\alpha + \frac{C}{l}\sum_{i=1}^{l} l_\epsilon(y_i, [K\alpha]_i + b) \tag{4.2}$$

where K is the kernel gram matrix associated to $(x_1, ..., x_1)$: $K = (K(x_i, x_j))_{1 \leq i,j \leq l}$ and $[K\alpha]_i$ is the i^th coefficient of the vector $K\alpha$: $[K\alpha]_i = \sum_{j=1}^{l} \alpha_j K(x_i, x_j) = f(x_i)$.

The linear constraints hidden by the function l_ϵ can thus be dualized and relaxed by introducing some variables $\xi_i, \xi_i^\star$:

$$(P') \min_{\alpha \in \mathbb{R}^n} \quad \frac{1}{2}\alpha^T K\alpha + \frac{C}{l}\sum_{i=1}^{l} \xi_i + \xi_i^\star$$

$$\text{subject to} \quad \begin{cases} y_i - [K\alpha]_i - b & \leq & \epsilon + \xi_i \\ [K\alpha]_i + b - y_i & \leq & \epsilon + \xi_i^\star \\ \xi_i, \xi_i^\star & \geq & 0 \end{cases}$$

The Lagrangian associated to this problem can now be written for any $(\alpha, \xi, \xi^\star, \eta, \eta^\star, a, a^\star) \in (\mathbb{R}^l)^3 \times (\mathbb{R}^l_+)^4$:

$$\begin{aligned} L(\alpha, \xi, \xi^\star, \eta, \eta^\star, a, a^\star) :=& \frac{1}{2}\alpha^T K\alpha + \frac{C}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^\star) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^\star\xi_i^\star) \\ & - \sum_{i=1}^{l} a_i(\epsilon + \xi_i - y_i + [K\alpha]_i + b) \\ & - \sum_{i=1}^{l} a_i^\star(\epsilon + \xi_i^\star + y_i - [K\alpha]_i - b) \end{aligned} \tag{4.3}$$

and the primal problem is :

$$\min_{\alpha,\xi,\xi^\star\in\mathbb{R}^l} \max_{\eta,\eta^\star,a,a^\star\in\mathbb{R}^l_+} L(\alpha, \xi, \xi^\star, \eta, \eta^\star, a, a^\star) \tag{4.4}$$

Solutions of this dual problem are known to be saddle points of L. Hence, the maximizer of the dual is obtained as :

$$\begin{aligned} \partial_b L &= \sum_{i=1}^{l}(a_i - a_i^*) = 0 \\ \partial_\alpha L &= K(\alpha - (a - a^\star)) = 0 \\ \forall i \in \{1, ..., n\} \partial_{\xi_i} L &= \frac{C}{l} - a_i - \eta_i = 0 \\ \forall i \in \{1, ..., n\} \partial_{\xi_i^\star} L &= \frac{C}{l} - a_i^\star - \eta_i^\star = 0 \end{aligned}$$

If it is assumed that the kernel is strictly definite positive, then all of the eigenvalues of the gram matrix K are positive and second condition becomes

$$\alpha = (a - a^\star)$$

Therefore the final formulation is obtained :

$$\begin{aligned} &\text{maximize} \quad -\frac{1}{2}(a - a^\star)K(a - a^\star) - \epsilon\mathbb{1}^T(a + a) + y^T(a - a^\star) \\ &\text{subject to} \quad \begin{cases} \mathbb{1}^T(a - a^\star) = 0 \\ 0 \le a, a^\star \le \frac{C}{l} \text{ (componentwise inequality)} \end{cases} \end{aligned} \tag{4.5}$$

It is noticed here that the maximization does not come from any consideration of the dual problem but rather from the change of sign of the objective function.

Hence, a quadratic problem which can be solved by various methods, such as interior point methods, is obtained.

Solution of this problem is hence :

$$f : x \mapsto \sum_{i=1}^{l} (\alpha_i - \alpha_i^\star) K(x, x_i) + b$$

4.3.3 Implementation

This part of the project was implemented in `python` in order to maintain a similar pipeline with the other project components. This implementation was object oriented in order to have a maintainable code. The structure of the code is presented through the following diagram :

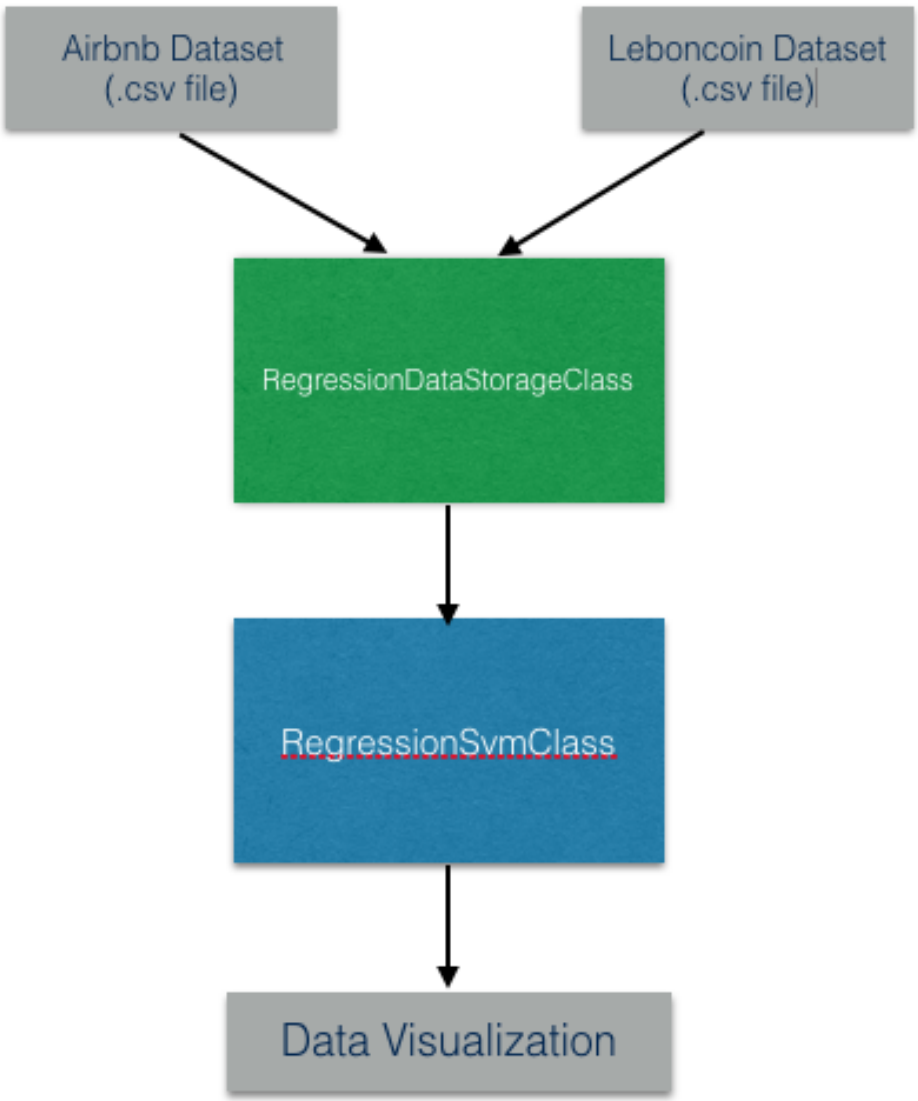

Figure 4.1: Orthogonal Wavelet Transform applied on signal 'Bumps'

`RegressionDataStorageClass` basically retrieve data from the csv files containing the data. Four features for regression were utilized : longitude, latitude, type of home(single room or entire home) and person capacity.

`RegressionSvmClass` performed the approach described above. The quadratic problem is solved thanks to a `cvxopt` which is a python solver.
A second `RegressionSvmClass` was implemented with `scikit-learn` and which was pretty straightforward since the library proposes itself a regression class called `SVR` (in submodule `sklearn.svm`). Performances were equivalent

for the result, and computation time was not a differentiating element since the data size remained relatively small.

4.3.4 Results for Regression

After shuffling the dataset, it was split into two subsets with a ratio of 80%/20% for the training set and the testing set.

For the Airbnb dataset,for each sample of the testing set the absolute value of the difference between the real value and the estimate given by the regression was computed. This result obtained a mean error of 8 Euros for a nightly pricing.

Chapter 5

Data Visualization

Visualization of results in a manner which makes the results easy to interpret is an important, yet an often overlooked, facet of any project working with data. It involves an overlap in the area of art and science with the science behind many aspects not specifically known or poorly understood. This means it is difficult to have an objectively 'best' model for visualization, but there are instances where some graphs are 'better' than others.

In the case of modeling for urban planning visualization becomes important as it allows viewers to gather basic information as to how prices are spread throughout the city and draw additional inferences which are not easily represented, or simply not present in a distinguishable manner, in the data.

New inferences could be things noticed such as certain areas tend to have higher prices than others and can lead to other observations and new questions when the problems are further examined. Are people willing to pay more for an apartment which is located next to a nice park? How about next to a river? What about in a quiet neighborhood? Similar questions and observations can be stirred by examining features and price spread on maps which visualize the data in a manner such that the spread of prices can be inferred not only as a combination of specific features, but as a more complex combination of location features. For instance in the United States of America, a public school which is rated very highly also correlates very highly with higher prices in the immediate vicinity where the school district is located. In various locales these instances are bound to be based upon different results based upon the culture of the specific locales.

5.0.1 Background

The reason proper visualization of data is important is that by adjusting the way that high dimensional data is presented, connections between data aspects which are not completely obvious can be shown in an easily understood manner. This is due to the very quick visual processing facilities

which are inherent in people who have properly functioning visual facilities, which includes the majority of people. In scientific terms this quick visual processing is known as pre-attentive processing. It refers to the subconscious accumulation of information from an environment. Pre-attentive processing works in a similar manner to the way a neural network works via processing data from small-scale to global-scale attributes and making connections between various data points. The most important thing to take advantage of from pre-attentive processing is that this process takes place over the course of 250-300ms. The attributes picked up in this initial time span include details such as color, length, width, orientation, shape, size, position, and intensity. However, various studies have shown that not all attributes are picked up with the same speed. For instance, variations in color are much more quickly determined than variations in shape. [5]

Interaction with things also allow for an individual to study the information and notice interesting aspects that stick out to them which other people may not have noticed or seen as important. By being able to interact with the data consciously while also presenting the data in such a manner as to maximize the connections made from pre-attentive processing, the maximum number of connections and inferences can be made from the visualized data.

5.0.2 Methods

Tools Utilized

Given that the rest of the code was all written in `python`, it was decided that `python` would also be utilized in the visualization component in order to keep a steady pipeline. However, the typical interactive visualization libraries for interactivity are usually based on the powerful `javascript` library `D3`. Given that interactivity was a desired component, there were a few possibilities for making the desired outputs. These included libraries such as `Bokeh`, `Seaborn`, and `Plotly`. Given that the desirability was for the data pipeline to be independent of city input, it was decided to use a map component of a library which allowed for all map features to be solely dependent upon the data entered and did not require downloading multiple files for each city represented.

It was decided that `Plotly` would be utilized for a few specific reasons. First of all, it provides the most powerful connections between the `python` and `javascript` languages without requiring significant portions of code to be written in`javascript`. The main attraction behind the usage of `Plotly` was that it contains a nice class allowing data to be plotted upon a map. In fact, `Bokeh` was also attempted for the project, but there was a bottleneck in presenting the data which was simply unable to be breached. This was mainly due to the impossibility of adding multiple representations while also

adding correct annotations which gave the indicators as to how different things are spread. The downside of utilizing `Plotly` is that it requires an API access. There are several different tiers of access, but the most basic one, which was the one selected for this project, makes the plots created available for anybody to access at any time. If this problem is followed more, a higher level API access should be selected to ensure there is no private usage of the code and visualizations unless that is a desired feature.

Features of the Visualizations

Each visualization was composed of the following features:

- Map data
- House data
- Color based upon specific values
- Interactivity

The first layer of the visualization was the map data, this was important as it gives the framework upon which the prices are plotted. The second layer consists of the gathered data plotted onto the map data using the collected longitude and latitude. Given the plotted gathered data it was desired to present the variability of pricing in an intuitive manner. This was generated by producing a color-scale which changed values based upon the requested value of the data. Finally interactivity of the figure was gained by allowing zooming, panning, and the display of information when hovering over a selected data point.

5.0.3 Visualizations Produced

For the visualizations of this project, the data worked with was the mined data from the websites `Airbnb` and `Leboncoin` regarding the city of Grenoble. Since one dataset was working with daily data and the other with monthly data, the datasets were not able to be assigned simultaneously to a single graph which does limit slightly the power of the visualization due to the limited size of the datasets which were gathered. One artificial dataset was also produced from the kernel regression method used to study the prices and plotted.

First of all, histograms of the daily and monthly rents were produced as shown in figures 5.1 and 5.2 in order to give the viewer some idea of the distribution of rental prices and examine whether or not there are significant differences between the distributions of rental prices for the two different periods. It does seem that the daily rental prices does have a peak which is a bit further towards the minimum in comparison with the monthly rental

prices which could also be due the fact that daily rental prices could be based upon a slightly more competitive market. However, overall the spread of the rental prices both seem to be quite similar.

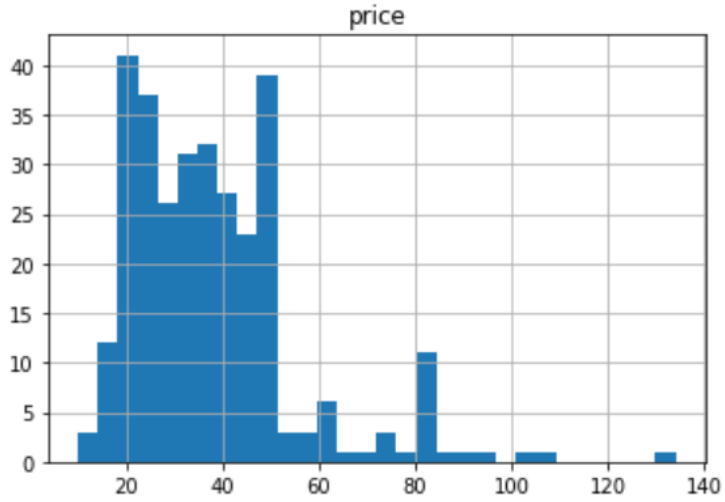

Figure 5.1: Histogram of daily rental prices

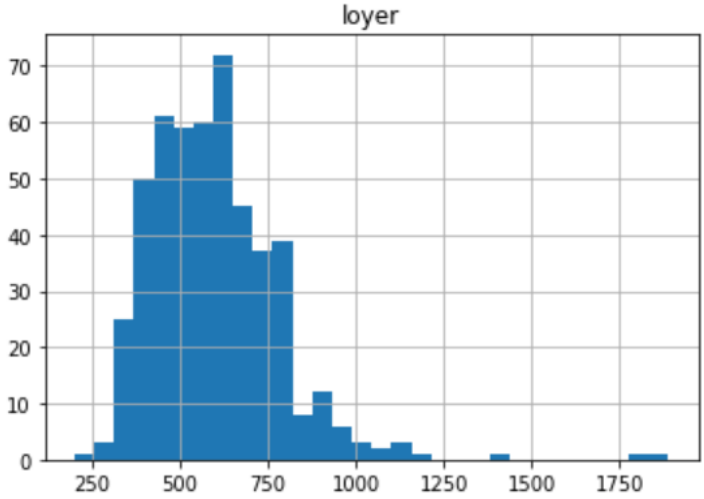

Figure 5.2: Histogram of monthly rental prices

The final visualization was made with tabbed views allowing the viewer to select different representations of the data. All of the data was gathered into one chart in order to enable easy usage and comparison of all features. All different charts can be activated by selected the specific desired tab from the left side of the map. Furthermore, when pointing to a specific data point, all relevant information is shown.

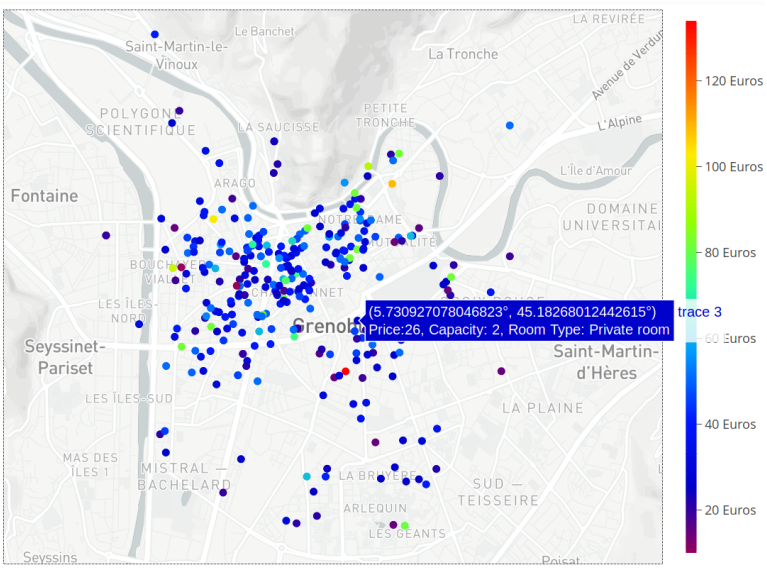

Figure 5.3: Map with daily rental prices and color-scale related to price

First the data representing the daily rental prices are shown in 5.3. It was noticed that there are however a few outliers which make the bulk of the variation hard to distinguish, due to the staggered tail of the rental prices and an attempt to cut off this tail by eliminating the highest rental prices was attempted by removing all data points which were more than three standard deviations from the mean price. The result is shown in 5.4. It is easily noticeable that there is more information able to be obtained once the outliers are removed.

Given the success in representing the data without present outliers, the data for monthly rental prices is presented in 5.5 with all outliers removed.

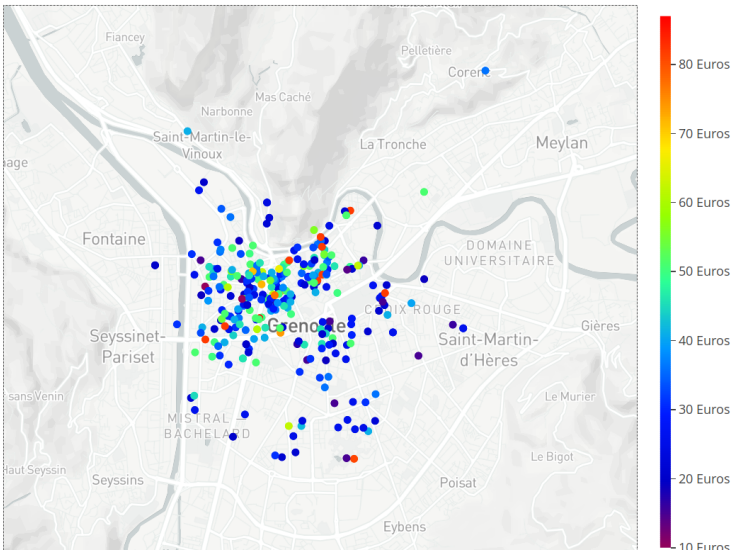

Figure 5.4: Map with daily rental prices minus outliers and color-scale related to price

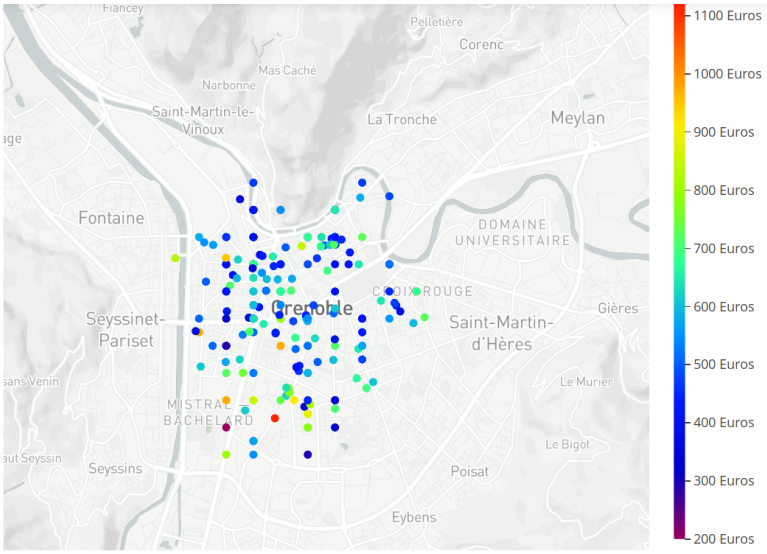

Figure 5.5: Map with monthly rental prices minus outliers and color-scale related to price

Chapter 6

Conclusions

Firstly, let us briefly discuss possible extensions of the project. Surely, the text mining part would add a lot of to accuracy of created models - the more features we have the more adequately we are able to estimate the price. In our opinion it would be possible to create interactive map based on Google Maps such that clicking on the desired region would result in estimated price for that region altogether with offers already existing. Unfortunately, we were not able to reach to that step.

Finally, let us summarize what we have done. Firstly, we managed to build and quite successfully crawl the Leboncoin and Airbnb websites in order to get the data. Secondly, we used crowled data to build the model with which we could estimate prices based on observed features. Thirdly, we consider ourselves quite successful in presenting the results on the interactive map.

Bibliography

[1] https://en.wikipedia.org/wiki/Mercator_projection

[2] https://manialabs.wordpress.com/2013/01/26/converting-latitude-and-longitude-to-map-tile-in-mercator-projection

[3] https://developer.here.com/documentation/traffic/common/map_tile/topics/mercator-projection.html

[4] Smola, Schölkopf, "A tutorial on Support Vector Regression", 2004,Statistics and Computing,volume 14, "3", pages 199-222

[5] Meng, Xianglin and Wang, Zhengzhi. "A Pre-attentive Model of Biological Vision", 2009, IEEE International Conference on Intelligent Computing and Intelligent Systems, pages 154-158

[6] https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser)