June - August 2021

Institut des Humanités Numériques

Data Science

Text extraction from images and classification using neural network with Python

an internship authored by

Emirkan Karaguzel

supervised by

Hanane Ariouat

Acknowledgements

First of all, I would like to thank the whole Bachelor group, for all the help that they provide me in this hard period of my life. Their support and their caring attention have given me the strength and the desire to continue to move forward. I would particularly like to thank Hanane Ariouat, my tutor and internship supervisor who listened, helped and supported me by making this internship possible despite my weak physical condition. And finally, I would like to thank my parents for their support, without whom I would never have tried to hope for recovery and without whom I would never have and live such an experience...

Abstract

Nowadays, there are hundreds of projects based around the house pricing field. As well as huge quantity of different text recognition algorithm. We can find any kind of text recognition script or even a data-set of any type of houses. However, when it comes to floor-plans very few studies exist, so far there is no OCR built around the floor plans and each data table that deals with houses is never filled with the dimension of each room. The objective of this internship is to build a generously sized floor plan data-set from photos and then successfully apply machine learning and deep learning models to it.

We produce floor plans data everyday and our goal is to extend the great playground it represents from photos into a data-set that will help us create a satisfactory prediction model.

Contents

Chapter 1

Introduction

When an individual purchases a home, they simultaneously purchase its structural features, such as the rooms and the sizes. These features, are most of the time not available when dealing with data-sets. While others, such as the total area or location, are quite often available. Despite the well-known impact of room characteristics on house prices, little attention has been paid to the systematic quantification of these characteristics.

Two issues have led to this neglect. Not only do few text recognition methods exist that can read data from floor-plans effectively, but that the collection of such data is quite challenging.

Fairly easy to find and available for most cities, we show that floor plans can provide us with this invaluable information and improve home price estimation.

We propose to use an OCR text recognition model on floor plans to extract and build a dataset by room size.

We then use a machine learning model and a deep neural network model to estimate house prices in each arrondissement of Paris. We use traditional housing characteristics such as total area, location, and price, as well as less traditional characteristics such as the area of each room to estimate the housing price model. We find encouraging results where learning to characterize each room size improves house price prediction, even when generalizing to each arrondissements in Paris.

1.1 Institut des Humanités numériques

During this internship I have been immersed in a world that every data scientist would dream of inside the computer science section of the IDHN[4].
According to their website they are defined as a federal structure in Digital Humanities which was created in 2017 in connection with the Area of Major Interest Sciences of the text and new knowledge led by Paris Sorbonne University and of which CY Cergy Paris Université is a founding member and partner. The IDHN has a Joint Research Unit working in the field of computer science, which has sixty permanent staff, including eight engineers and forty doctoral students. The four laboratories of the structure cover the majority of disciplines in SHS: geography, history and civilisations, literature, language sciences as well as the fields of artificial intelligence: data science, knowledge representation, and data management.

But for me, through my experience as an intern in this lab, the IDHN is a place where ideas come to life and where sharing, teaching and learning are considered very important. My supervisor Mrs. Ariouat took me under her wing and made me discover a whole new aspect of data-science. During this internship, I have been working in the data science domain of the IDHN to further the field of computer science in text recognition, data visualization and neural networks for predictions over house pricing and floor-plans. In this project, I will be mostly focusing the deep learning and recognition algorithm while my colleague and friend Leo will work on the machine learning part. You can check both of the reports, to get a perfect knowledge about how these models works. Putting our results together we found some strong and interesting observations.

1.2 State of the art

If you have an analytical mindset, and ever went on a virtual apartment-hunting trip, you have probably run into the issue that you can almost never search by property size, let alone by individual room sizes. Nevertheless, this information is quite often available in the form of a floor-plan image.

In general, text data present in images and video contain useful information for automatic annotation, indexing etc... However, the challenge of automatic text extraction is particularly difficult due to variances in text size, style, orientation, and alignment, as well as low image contrast and a complicated background. By going through these issues we will use something called a TIE, Text Information Extraction.

A TIE system use images or a sequence of images as an input. They can be in various type such as grayscaled/colored, compressed/uncompressed, static/motion and so on. Applied on floor-plans these problems can be described into several subproblems:

- **Text detection**, refers to the determination of the presence of text in a given plan.
- **Text localization**, process determining the location of text in the image and generating bounding boxes around it.
- **Text extraction and enhancement**, the stage where the text components are separated from the background we require to enhance the extracted text components because most of the time the text region has low-resolution and is prone to noise.
- **OCR - Recognition**, final part when the extract is transformed into plain text using OCR technology. In our case Python-tesseract Google's Tesseract-OCR Engine.

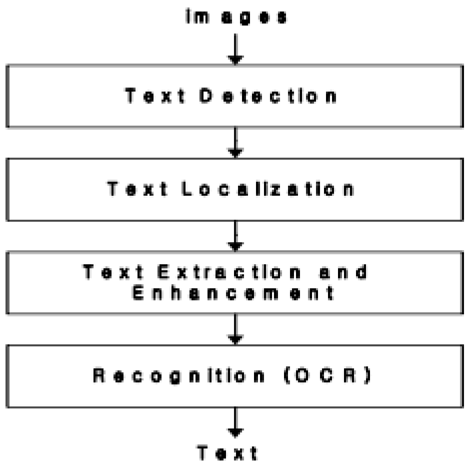


Figure 1.1: Figure describing the structure of our TIE model.

Here's a link of a very good explaining study about TIE available on IJERT[1].
Well, during this internship we figured out that this problematic would be an excellent opportunity to perform image recognition and create a script that can turn images into a nice and clean data table.
The motivation of the current project is that in short, once we have a large enough data-set, we open up the possibility of applying the vast techniques of data science to create a never-seen-before database of houses by room size, price and location.

1.3 Motivation

So far, if you search for similar projects around floor-plan analysis on the internet you won't find explicit data set about each room and moreover you won't find a text recognition system that will work efficiently for this same purpose. Our main objective is to generate a script that will be able to read, extract and create a database based on the image of the plans we provide. With this new data-set, we will apply the algorithms from the field of data science, in our case perform some predictions around house pricing using our own generated data-set with data visualization and deep/machine learning model using **Python**.

Chapter 2

Methods

First inspired by the work of MatePocs on GitHub, creating a program that will be able to collect data from random floor-plans is quite a difficult task. As he said in his GitHub post he created *"A script that automatically collects room sizes from apartment floor-plans... or at least it tries"*.[7]

The complexity of this task comes from the fact that there is a wide variety of floor-plans on the Internet, the language, the room names, the unit of measurement and, most of the time, the result per room is not indicated in one single integer but in a form of multiplication.

Our first goal will be to extract data from the floor-plans data set using text recognition and collect enough data to let us to perform predictions models and visualisations around the housing market and room sizes.

For this project we will use Python which is one of the best programming languages in data science. as it is very complete in terms of external libraries such as pandas[10], numpy[9], tesseract[3], scikit-learn[6], keras[5], matplotlib[8] and plotly[11]. Python is providing us with very powerful and useful tools that will be perfect for this project.

2.1 Text extraction from images

Knowing the main disadvantages of working with different floor-plans, we will create a **semi-automatic script**, that will in the end, **convert the floor-plans into a data-table.** We are going to use pytesseract, a Python wrapper of Google's Tesseract OCR, which will be our main text recognition library as well as pandas, numpy and pillow to build our script.

Tesseract OCR

pandas

Our script will be based on the raw extract from Tesseract, once our OCR worked on the process part with already enhanced images, the script will be mainly focusing the cleaning part of the extract.

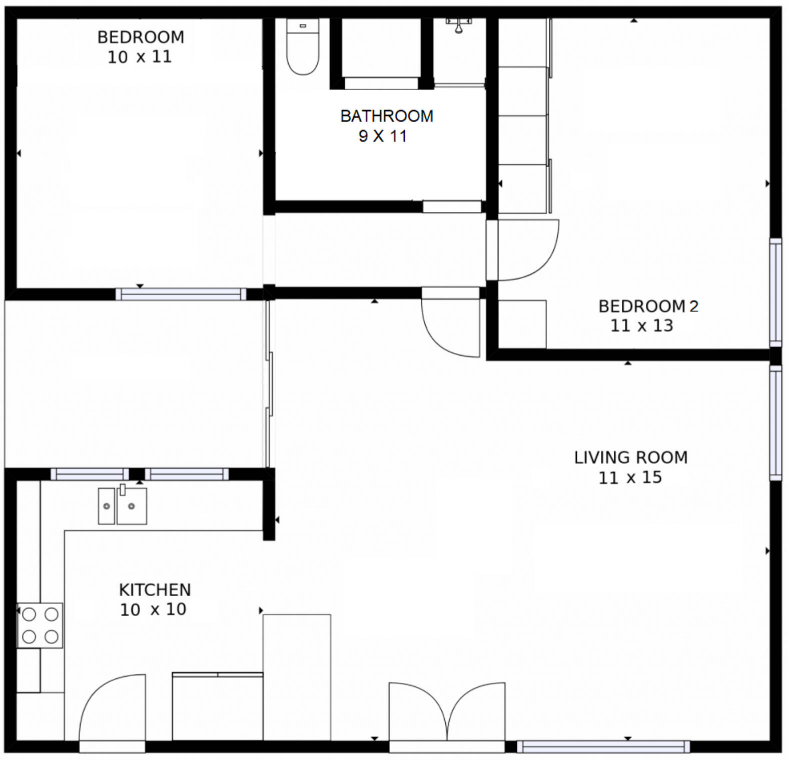
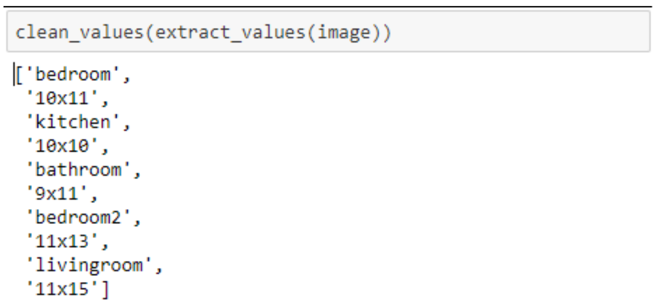


Figure 2.1: Picture showing the floor-plan n°47 in our data-set

Basically, when the quality is good enough, we run the text extraction script we built directly, otherwise we enhance the image before using our OCR. We'll use floor plan 47 from our data-set as an example to partially explain how our script works. In order and by function, here is how a single floor-plan image is completely transformed into a data table.

- `extract_values(), clean_values()`

  This functions will use the Tesseract library to read and recognize text inside the floor-plans and then clean the extract since it's most of the time filled with blanks and misinterpretation from the algorithm, in a single list with all the elements that we will need for our data-set.



```
clean_values(extract_values(image))
```

```
['bedroom',
 '10x11',
 'kitchen',
 '10x10',
 'bathroom',
 '9x11',
 'bedroom2',
 '11x13',
 'livingroom',
 '11x15']
```

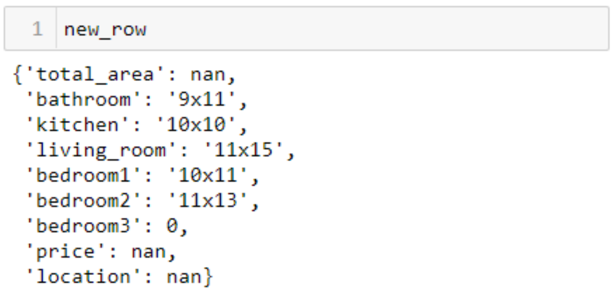
- `deep_c()`

  This function is the 'manual' part of our script it will be used when `clean_values()` didn't manage to clean the list enough to write the data this way :

  `['room1','size1','room2','size2',etc...]`

- `row()`

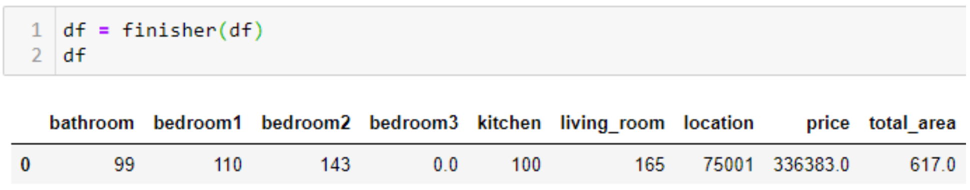
  Finally, when the extract is cleaned, this function will go through the list, recognize the name of the rooms and sort them with their own sizes. `row()` will then return a single dictionary that is ready to fill our data-set.



```
new_row
```

```
{'total_area': nan,
 'bathroom': '9x11',
 'kitchen': '10x10',
 'living_room': '11x15',
 'bedroom1': '10x11',
 'bedroom2': '11x13',
 'bedroom3': 0,
 'price': nan,
 'location': nan}
```

We still have to deal with the diversity of floor plans, so we created our `row()` so that it can work with any language and built functions like `computer() and finisher()` to calculate the values when the room size is written in the form yyy x yyy and finalize the data set.



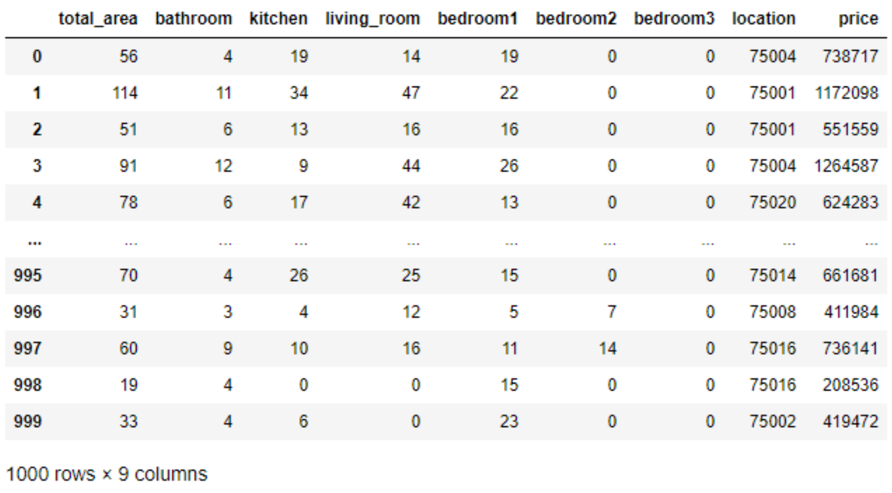
```
df = finisher(df)
df
```

	bathroom	bedroom1	bedroom2	bedroom3	kitchen	living_room	location	price	total_area
0	99	110	143	0.0	100	165	75001	336383.0	617.0

Now, using this script, we repeated the process with new floor plans for as long as it took to get a consistent amount of data and built a csv file of a thousand rows.
From this newly generated data-set, we can build any data science model around it and during this project we will use a linear regression and a neural network model, and try to understand which model will be more interesting in this case. But first let's see what our data looks like and what it says...

2.2 Data-set and Visualization

Before moving on the predictions and computations, we need to have a closer look to our data-set. Our data-table is composed of one thousand rows and nine columns as seen below:



	total_area	bathroom	kitchen	living_room	bedroom1	bedroom2	bedroom3	location	price
0	56	4	19	14	19	0	0	75004	738717
1	114	11	34	47	22	0	0	75001	1172098
2	51	6	13	16	16	0	0	75001	551559
3	91	12	9	44	26	0	0	75004	1264587
4	78	6	17	42	13	0	0	75020	624283
...	...	...	...	...	...	...	...	...	...
995	70	4	26	25	15	0	0	75014	661681
996	31	3	4	12	5	7	0	75008	411984
997	60	9	10	16	11	14	0	75016	736141
998	19	4	0	0	15	0	0	75016	208536
999	33	4	6	0	23	0	0	75002	419472

1000 rows × 9 columns

Figure 2.2: Figure showing the data-set.

The first seven columns represent the total areas and the sizes of the rooms from the houses in meters, while the two last columns give us the location by districts in Paris and the prices of the houses.

We will then take a look on how our data is distributed and if it's in a good shape that we can produce predictions model out of it.

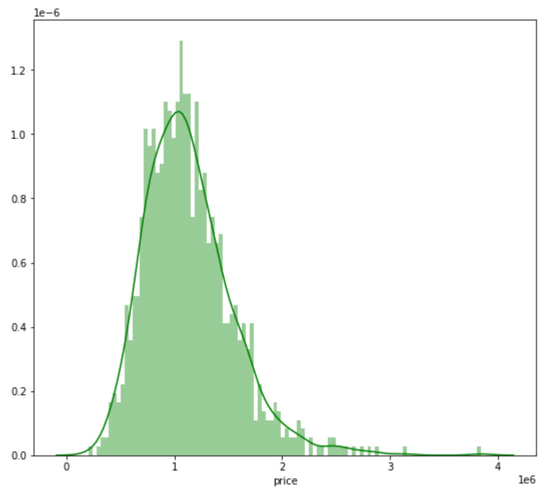
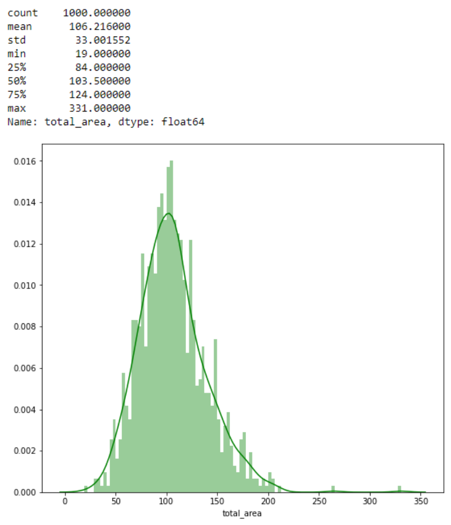


Figure 2.3: Bar-plot showing the distribution of the price.

With this graph we can see that the prices are mostly around one million and some outliers lies above three million. We will eventually want to get rid of them to get a normal distribution of the independent variable ('price') for machine learning or neural network purposes.



```
count    1000.000000
mean      106.216000
std        33.001552
min        19.000000
25%        84.000000
50%       103.500000
75%       124.000000
max       331.000000
Name: total_area, dtype: float64
```

Figure 2.4: Bar-plot showing the distribution of the total area.

Same remark as before, the total area mostly lies around a hundred squared meter. And we can see some area outliers that we will get rid of at the same time as the price outliers since they are correlated. The total area mean is 106 with a maximum of 331.

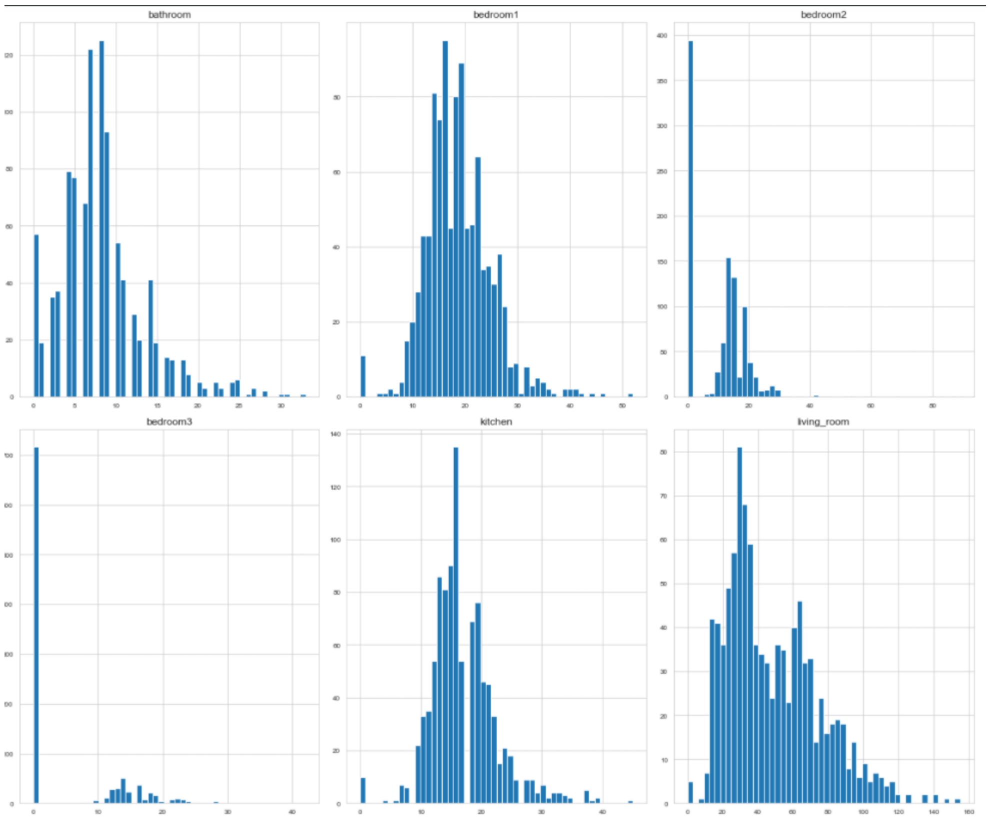


Figure 2.5: Bar-plot showing the distribution of rooms values.

We observe that for most of the rooms, the value is defined on zero when the information isn't available. This is something that we will need to take in count when building our models. We must perform some data cleaning before jumping into the mathematical computations.

Now that we have explored our data-set, we will use the libraries matplotlib and Plotly to visualise our data in a better form and we will try to find some explanations and information that are not obvious when looking at a table.

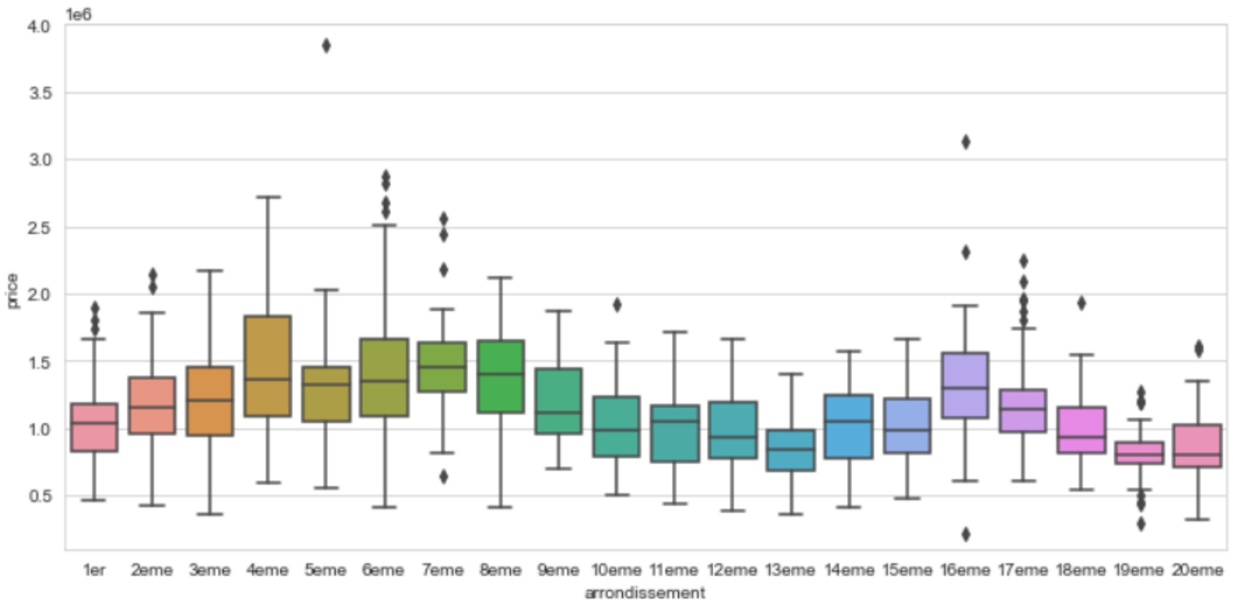


Figure 2.6: Box-plot comparing the price per location.

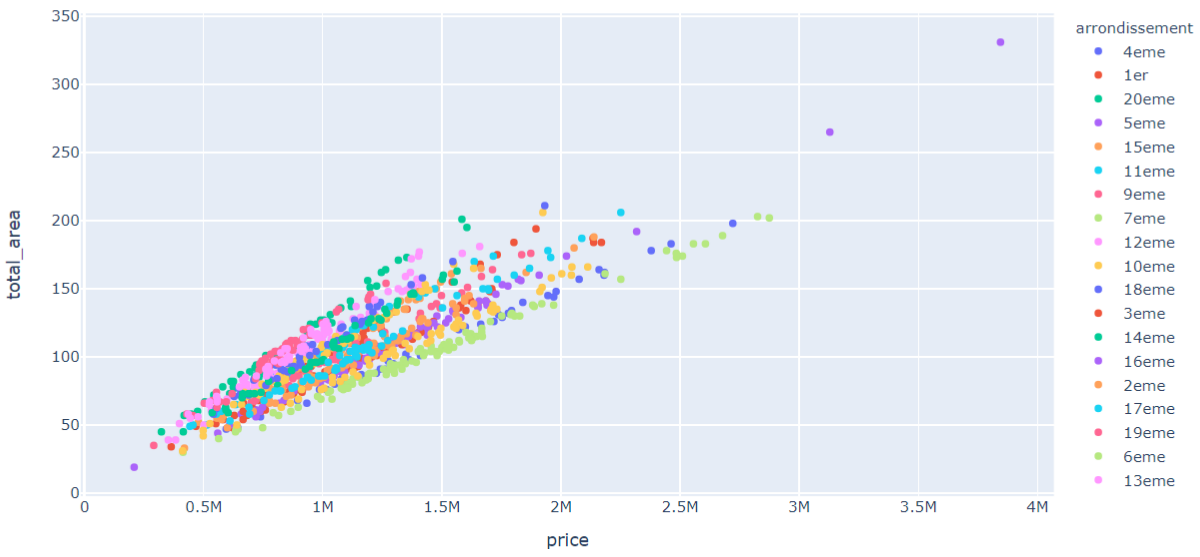


Figure 2.7: Scatter-plot comparing the price and total area per location.

The first thing that can be seen in these graphs is that the price is not evenly distributed by location. We can already see some trends in these two figures, as in the box-plot.
For example, the median price in the 20th arrondissement is lower than in the 7th arrondissement, which we can also see in the scatter-plot, the price is higher in the 7th arrondissement than in the 20th arrondissement for the same total area. The data is obviously distributed by location, we can easily spot the main line of each arrondissement and predict with the naked eye what the value of a house would be according to its location. This data-set gives us a good representation of the reality of the housing prices in Paris and where demand is the highest.
Now it's time to move on to the fun part of this project, building a neural network and a machine learning model to start predicting house prices.

2.3 Machine Learning and Deep Learning

Machine learning is a branch of artificial intelligence that involves algorithms that parse data, learn from it, and then use what they've learned to make better decisions. Deep learning is a subfield of machine learning that layers algorithms to construct a "artificial neural network" that can learn and make intelligent decisions on its own. So we can ask ourselves what is the difference between deep learning and machine learning ?
In fact, deep learning and machine learning works in a similar way. However, their powers are quite different.
While simple machine learning models get better over time at whatever task they are given, they still require some guidance. If an AI algorithm makes an incorrect prediction, an engineer must step in and make changes. Using its own neural network, a deep learning model can determine whether a prediction is accurate or not. In machine learning we can speak of supervised learning while in deep learning we will speak about unsupervised learning.
During this project we made the decision to use both of linear regression and neural network models on our data-set. Both methods have their own advantages and disadvantages. In this section we will see how we build and use them over our house price data.

2.3.1 Linear Regression

Regression analysis is a sub-field of supervised machine learning. It aims to model the relationship between a number of features and a continuous target variable. We use linear regression to obtain a quantitative answer, in our case, predicting the price of a house based on a single feature. We will use various Python libraries to implement linear regression, but we will be mostly based on scikit-learn. For this part I decided to use the easy package scikit-learn provide, using the gradient descent method to compute the regression, doing the regression this way is easier and faster than going through mathematical functions, this is why I chose to use the gradient descent model from scikit-learn. But if you want to know more details and more deeply how a linear regression model work through the matrices you can check Leo's[2] report, which is mostly based on this model.

$$X = \begin{bmatrix} x^{(1)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \qquad \theta = \begin{bmatrix} a \\ b \end{bmatrix}$$

It basically use as input the variables that we want to predict and compute the coefficient of our regression line step by step, in our case X is the total area and Y the price. We can perform multiple linear regression by using the whole data-set as input as well.

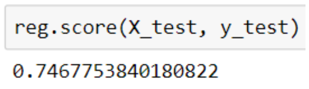
$$J(\theta) = \frac{1}{2m} \sum (X.\theta - Y)^2$$

Using cost functions to compute gradient,

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T.(X.\theta - Y)$$

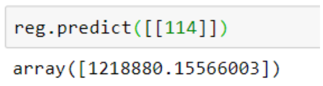
$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

We are building our linear regression model around two important features, the total area and the price, we have seen in the previous section that we have some outliers in our data-set and if we apply a linear regression model on it now the result will be biased and not at it's best accuracy. Here is our accuracy result on this data-set.



```
reg.score(X_test, y_test)
0.7467753840180822
```

Our prediction accuracy is about 74% which isn't very effective... Let's try to predict the house price for the first rows of our data-set to understand this fact. For a total area of 114 squared meters, the price in this data-table is set at 1 172 098€. We expect from our model something around this price but when making the computations, with the model we built the predicted price is 1 218 880€. It's a good prediction but we can do better.



```
reg.predict([[114]])
array([1218880.15566003])
```

By plotting our linear regression line we can see our inaccuracy problem. Most of the data is set between 50-150 squared meters with a fairly consistent price dispersion depending on the location.

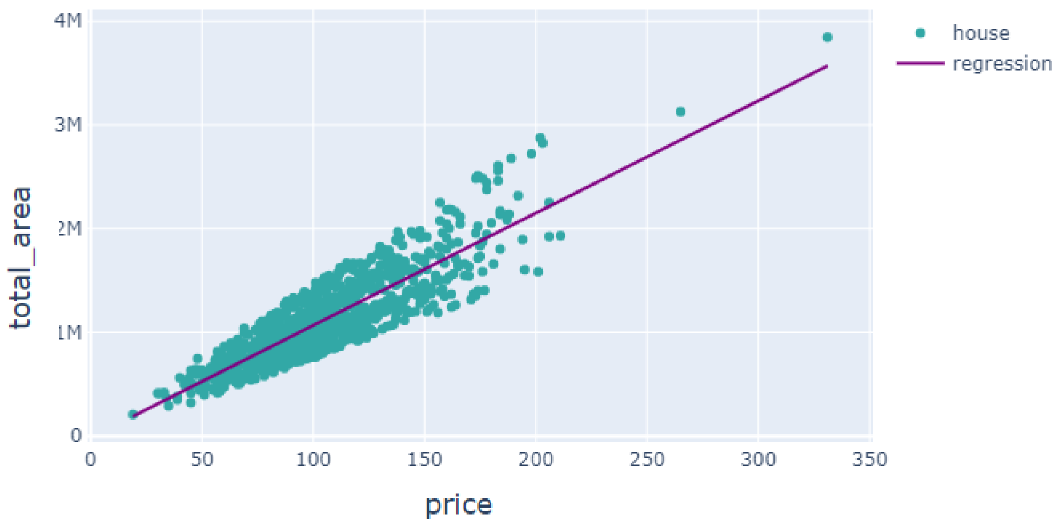


Figure 2.8: Scatter-plot showing linear regression applied over our data-set.

As we said above, some prediction models are not very effective based on the data we have and it is part of the data scientist's job to know which model will best fit a given data set. In this case looking at the distribution of our data, using this kind of predictions model will not be able to reach 85% of accuracy. To face this problem we will use a neural network model that should give us better results.

2.3.2 Neural Network

Neural networks are essentially self-optimizing functions that match inputs to correct outputs. It's a part of machine learning, we say that neural networks are deep learning / unsupervised learning model, unlike the linear regression method we have studied before which was a supervised learning model.

During this section we will work with Keras, which is a powerful Python library for developing and evaluating deep learning models. It's powered by Tensorflow an open source machine learning tool developed by Google.

Using Keras we built a model that take in count the size of each rooms of the house, the total area and the location. In our data set, the input is of eight values and output is of one values. So the input and output layer are of eight and one dimensions respectively.

The first thing we have to do is to set up the architecture, we want to build a neural network looking like this:

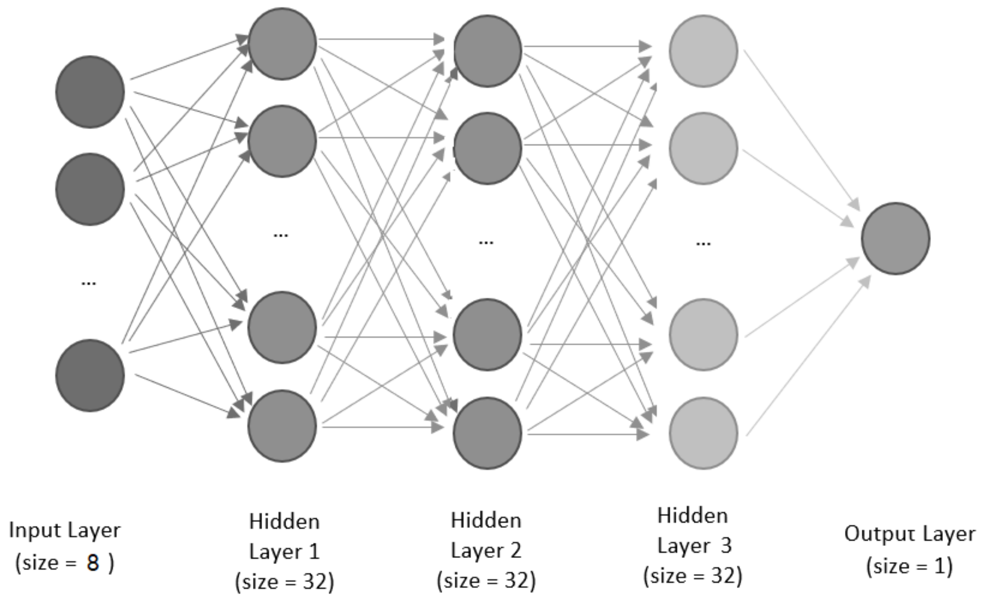

Input Layer (size = 8)
Hidden Layer 1 (size = 32)
Hidden Layer 2 (size = 32)
Hidden Layer 3 (size = 32)
Output Layer (size = 1)
...

Figure 2.9: Our neural network architecture.

We first use scikit-learn to create test and training sets. Once we have our variables ready we implement it in the network we just built. Since we have a data-set of a thousand row, running our model to learn won't take long, this is why we can easily set the epoch to five hundred. We can see the learning evolution of our model by plotting the loss of the test and train sets on the same graph.

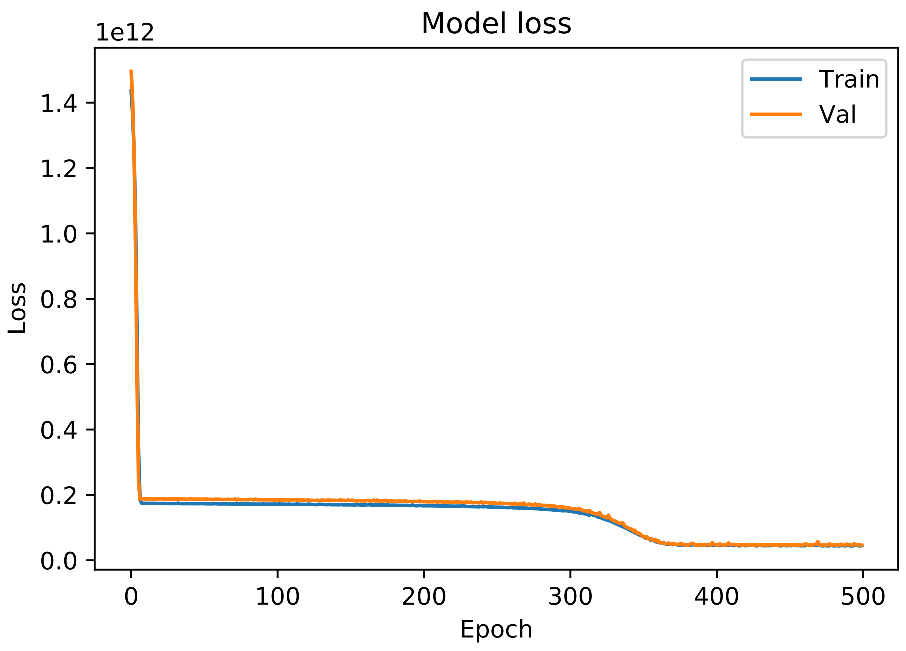
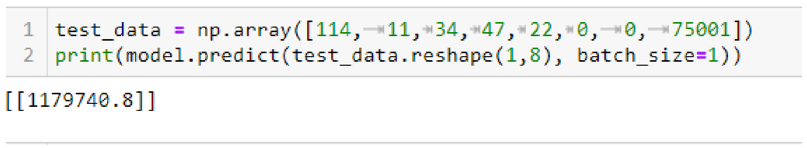


Figure 2.10: Scatter-plot showing the loss evolution by epoch.

We can see that our network has started to learn from the first 20 epochs and that from 400 epoch on-wards, the network does not improve any more. To save computing power, we can set the epoch a little lower, around 400.

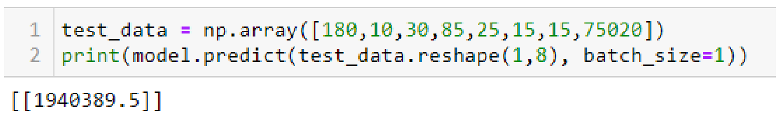
Now that we trained our model it's time for us to try it. Since we have created a network with 8 inputs, we have a model that can predict the price of a house for any custom room size. Let's try our network on an already known house from the data-set. As we did previously during the regression model, we will take the house of 114 squared meters. The expected price from the model should be 1 172 098€.



```
test_data = np.array([114,	11,	34,	47,	22,	0,	0,	75001])
print(model.predict(test_data.reshape(1,8), batch_size=1))
```

```
[[1179740.8]]
```

Our model predicted a price of 1 179 740€, which is really close to the reality, most of the time neural network models are more accurate than machine learning model, more particularly when we are dealing with big data-sets of million rows.
We can now predict the price of any house in Paris, for example let's assume we want a house in the 20th arrondissement with a total area of 175 squared meters, composed of one bedroom of 25 m^2, two of 15 m^2, one living room of 85 m^2, a kitchen of 30 m^2 and one bathroom of 10 m^2. Giving this information as input in our model we get this prediction.



```
test_data = np.array([180,10,30,85,25,15,15,75020])
print(model.predict(test_data.reshape(1,8), batch_size=1))
```

```
[[1940389.5]]
```

Our model predicts that a house that big in Paris 20th district will cost around 1 940 000€, which is very consistent with the current market prices in the 20th district of the french capital.
This model can be very useful for real estate agent or people that are interested into selling/buying a house. Moreover since this model is a very powerful and good tool to estimate house prices, another use may be the analysis of the prices by districts in Paris.

Chapter 3

Results and Discussion

Predictions by Model			
House area	Machine Learning	Deep Learning	Reality
25 m^2	255 485€	327 971€	330 000€
50 m^2	526 101€	552 440€	530 000€
75 m^2	796 718€	800 189€	820 000€
100 m^2	1 067 334€	1 071 009€	1 145 000€
125 m^2	1 337 951€	1 306 836€	1 290 000€
150 m^2	1 608 568€	1 590 665€	1 590 000€
200 m^2	2 184 852€	2 185 000€	2 150 000€

Figure 3.1: Table showing the predictions of our models.

Consequently, neural networks obtain more accurate results than linear regression, but the difference in the predictions between the two models is very small. However, the time needed to build them is more important, a linear regression model is straightforward to set up while a neural network requires a lot of computing power and some adjustments to get a decent accuracy. The choice of the model is not an exact science in this case, depending on the size of your data and how accurate you want to be, if you prefer quicker results and can afford to sacrifice some accuracy, you will opt for machine learning and linear regression. But if you have the time and want the most accurate model possible you will choose to go with a deep learning neural network.

3.1 Conclusion

To conclude, the objective of this project was to perform the multiple mechanisms that data science can represent over floor-plans data, throughout this journey we built a fairly good script of data mining from a database of floor-plans images using text recognition. Thanks to this program we generated a 1000 x 9 data-set with important features such as room sizes, house area, pricing and location in Paris. With the application of data visualization over it, we decided to investigate the price by location and we realised that they were correlated, each district of Paris in this data-set represent a given line that is giving us an idea for the price of the house by area and observed the fact that house prices are unevenly distributed across Parisian districts. Moreover, with the use of machine learning and deep learning we come out with two powerful and accurate model that can predict house pricing taking in count 8 features. Comparing the two model we built, we have an idea of whether to use machine learning or the deep learning model. Each of these models has its own advantages and disadvantages, machine learning is easier to apply but loses accuracy while deep learning is more complex to build but much more accurate. So the idea is that if we continue to grow our data set, we will continue working with our deep learning model knowing that in addition our neural network can predict the price of any house, by location and room sizes.

3.1.1 Limitations

As every ambitious project has its own limitations, we had to face many complications during this journey. First thing first, the plans given on the internet were most of the time too blurry for tesseract or some rooms were not labeled. As the project went on, we faced the issue of missing values NaN by replacing them with 0's. When using our script we quickly realised that there exists many types of rooms such as sun deck, office, garage or even houses with four bedrooms thus not allowing us to put them in our data-set with the correct total area. Moreover collecting data was a very long part of this project, since we had to clean manually the output of tesseract when it was wrong. However in our data-set the main problem was that some houses are way bigger than their total area in the original floor-plans since our data-set doesn't have enough columns to save data of each kind of room. During the visualisation we realised that the data was distributed like a cloud between 100-200 square metres, which was the main problem that prevented our linear regression model from being accurate enough. Finally since a deep learning neural network model requires a lot of data to be really accurate and we only had the time to collect data of only a thousand houses, our training and test set were very poor to build a good prediction model.

3.1.2 Possible extension

As mentioned above, we can improve the text recognition script so that it can recognize more room types and add more columns to the data set. With this modification we will kill two birds with one stone and solve the problem of wrong total area since we will have all rooms of each house in the data table, the total area will be correct. Last but not least we can collect even more data so our neural network model can be trained and win in precision to make more accurate predictions.

Bibliography

[1] Sahana K Adyanthaya. *Text Recognition from Images: A Study.* URL: `https://www.ijert.org/text-recognition-from-images-a-study`.

[2] Leo Bouyrac. *Image recognition and Machine learning over house plans.*

[3] Google. *An optical character recognition (OCR) engine.* URL: `https://opensource.google/projects/tesseract`.

[4] IDHN. *Institut des Humanités numériques.* URL: `https://cyidhn.cyu.fr/`.

[5] Keras. *Designed to allow rapid experimentation with deep neural networks.* URL: `https://keras.io/`.

[6] Scikit Learn. *Simple and efficient tools for predictive data analysis.* URL: `https://scikit-learn.org/stable/`.

[7] MatePocs. *How to Collect Text Information from Images with Python and Tesseract.* URL: `https://github.com/MatePocs/floorplan_reader`.

[8] Matplotlib. *Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations.* URL: `https://matplotlib.org/`.

[9] NumPy. *NumPy is the fundamental package for scientific computing in Python.* URL: `https://numpy.org/doc/stable/user/whatisnumpy.html`.

[10] Pandas. *Open source data analysis and manipulation tool.* URL: `https://pandas.pydata.org/docs/`.

[11] Plotly. *Plotly's Python graphing library makes interactive, publication-quality graphs.* URL: `https://plotly.com/python/`.