Business Problem

For this project, I decided to model the relation between the price of an estate and the type of venues in its neighborhood. The objective is to determine whether or not some venues have a direct impact on the price, and if so how is the price affected by it.

Concretely I will have to:

1. geolocalize a set of real-estate transactions. As I cannot use geocoder API at such a large scale (too much data), I will have to make the analysis at the level of the neighborhoods.
2. Calculate the average price per square meter for all neighborhoods in my data set.
3. List the venues (and the type of venues) for each neighborhood
4. Concatenate my two sources of data into one df
5. Split my data in order to have a train set and a test set
6. Pick and design the right algorithm to determine & predict the price of an estate given the venues in its neighborhood
7. Evaluate the accuracy of the model

Target Audience:

Investors may be interested by the independant variables that have a direct impact on the price of a real estate as their interest is to make the greater return on investment. So for instance, if a type X of venues is currently under construction in an area, one can expect to see the price of a real estate in this neighborhood impacted at the end of this project.

Note:

1. To make this analysis accurate and a bit pertinent I would need to work at the level of each transaction (i.e. geolocalize each real estate), because not only is it the closest neighborhood that has an impact, but also the distance is a fluctuant & quite relevant parameter (e.g. it may be convinent to have a supermarket nearby your place but maybe not to having seen on the building from your living room. You'd rather have a beautiful park).
2. Some other factors will be disregarded to make this study simplier. Yet they probably have some big impacts on the price (e.g. construction date of the building, material used for the construction, ecominical & social indicators of the population living in the neighborhood, etc.)

Data – Presentation

I used 3 different and complementary sources of data:

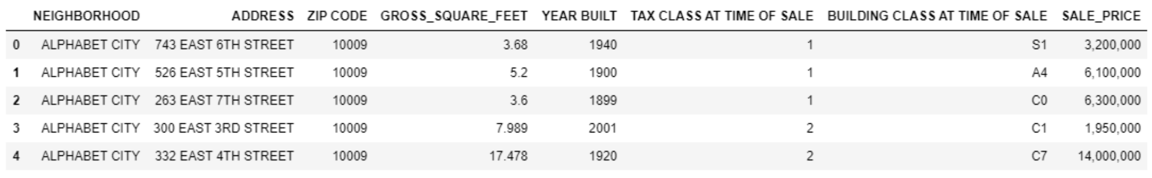
1. A data set that contains all the sales transactions of real estates that occured in Manhattan from 11-18 to 10-19. I will remove the rows with NaN values, transform the dtypes if needed (I will explain why and how below), group my data by neighborhood (I explained the reason in my presentation) and calculate the averige price per neighborhood per square meter.
2. A second data set will help me to extract the geographical coordinates of every neighborhood in Manhattan. Those data are mandatory to use the 4square API. I will then merge my two dataframes.
3. Finally I will extract the venues of each Neighborhood from 4square. I will merge the data and use a binary matrix to list & classify types of venues per neighborhood.

I will first import my main data source as a csv using the following method. Please find below a data set description:

Manhattan Rolling Sales File. All Sales From Oct 2018 - Sep 2019. "For sales prior to the Final, Neighborhood Name and Descriptive Data reflect the Final Roll 2019/20. Sales after the Final Roll, Neighborhood Name and Descriptive Data reflect current data" Building Class Category is based on Building Class at Time of Sale. Note: Condominium and cooperative sales are on the unit level and understood to have a count of one.

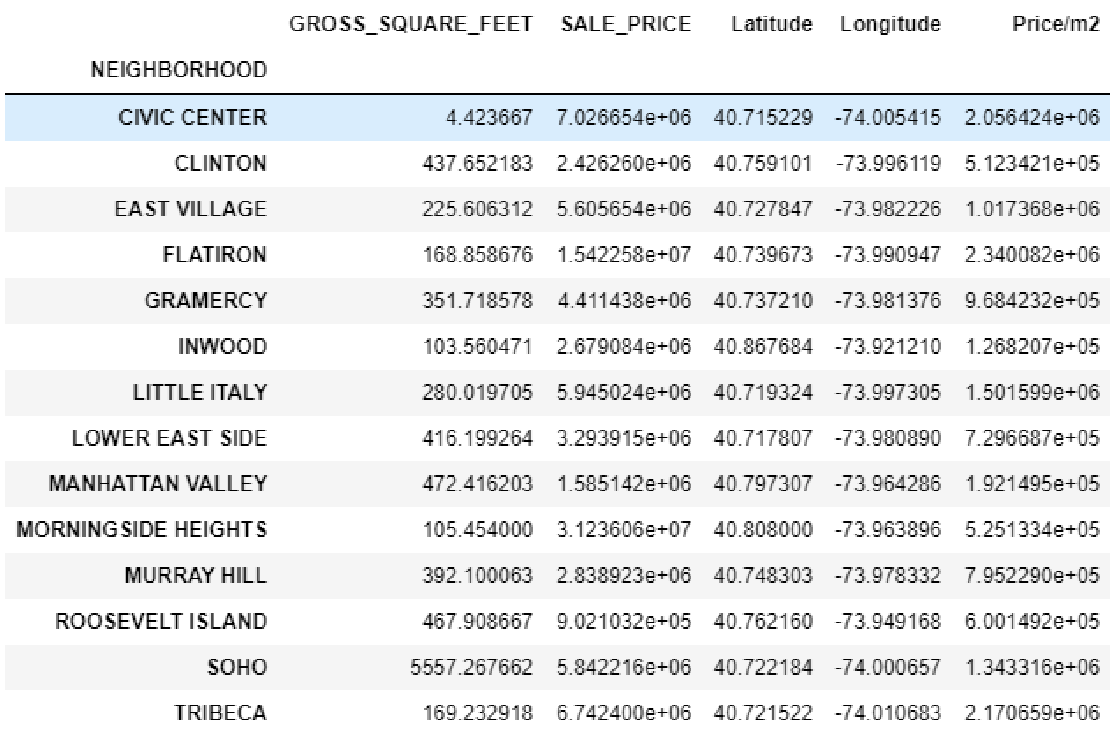
If you are interested by this dataset, you can download it from this url: https://www1.nyc.gov/site/finance/taxes/property-rolling-sales-data.page

My main data source



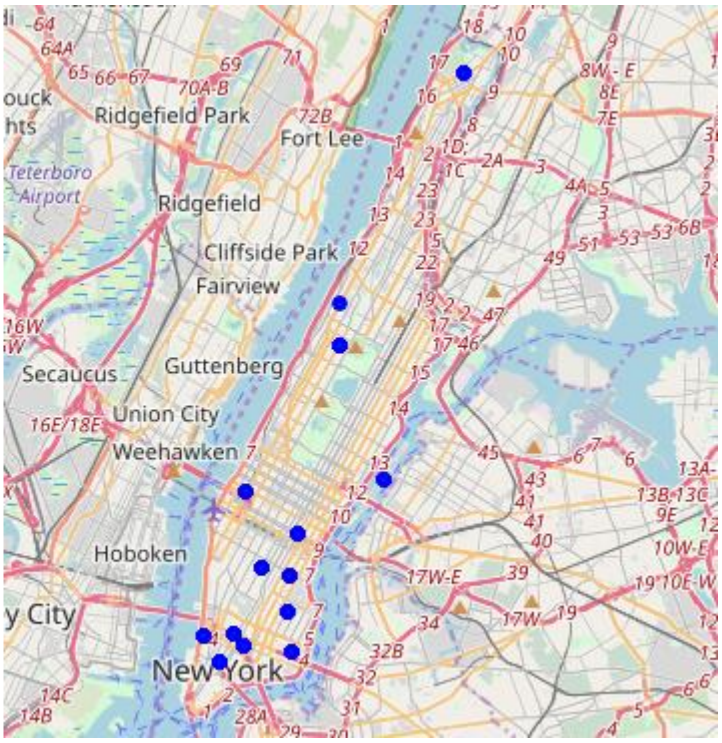
	NEIGHBORHOOD	ADDRESS	ZIP CODE	GROSS_SQUARE_FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE_PRICE
0	ALPHABET CITY	743 EAST 6TH STREET	10009	3.68	1940	1	S1	3,200,000
1	ALPHABET CITY	526 EAST 5TH STREET	10009	5.2	1900	1	A4	6,100,000
2	ALPHABET CITY	263 EAST 7TH STREET	10009	3.6	1899	1	C0	6,300,000
3	ALPHABET CITY	300 EAST 3RD STREET	10009	7.989	2001	2	C1	1,950,000
4	ALPHABET CITY	332 EAST 4TH STREET	10009	17.478	1920	2	C7	14,000,000

Intermediate Dataframe (after concatenation)



	GROSS_SQUARE_FEET	SALE_PRICE	Latitude	Longitude	Price/m2
NEIGHBORHOOD					
CIVIC CENTER	4.423667	7.026654e+06	40.715229	-74.005415	2.056424e+06
CLINTON	437.652183	2.426260e+06	40.759101	-73.996119	5.123421e+05
EAST VILLAGE	225.606312	5.605654e+06	40.727847	-73.982226	1.017368e+06
FLATIRON	168.858676	1.542258e+07	40.739673	-73.990947	2.340082e+06
GRAMERCY	351.718578	4.411438e+06	40.737210	-73.981376	9.684232e+05
INWOOD	103.560471	2.679084e+06	40.867684	-73.921210	1.268207e+05
LITTLE ITALY	280.019705	5.945024e+06	40.719324	-73.997305	1.501599e+06
LOWER EAST SIDE	416.199264	3.293915e+06	40.717807	-73.980890	7.296687e+05
MANHATTAN VALLEY	472.416203	1.585142e+06	40.797307	-73.964286	1.921495e+05
MORNINGSIDE HEIGHTS	105.454000	3.123606e+07	40.808000	-73.963896	5.251334e+05
MURRAY HILL	392.100063	2.838923e+06	40.748303	-73.978332	7.952290e+05
ROOSEVELT ISLAND	467.908667	9.021032e+05	40.762160	-73.949168	6.001492e+05
SOHO	5557.267662	5.842216e+06	40.722184	-74.000657	1.343316e+06
TRIBECA	169.232918	6.742400e+06	40.721522	-74.010683	2.170659e+06

Map of my Neighborhoods

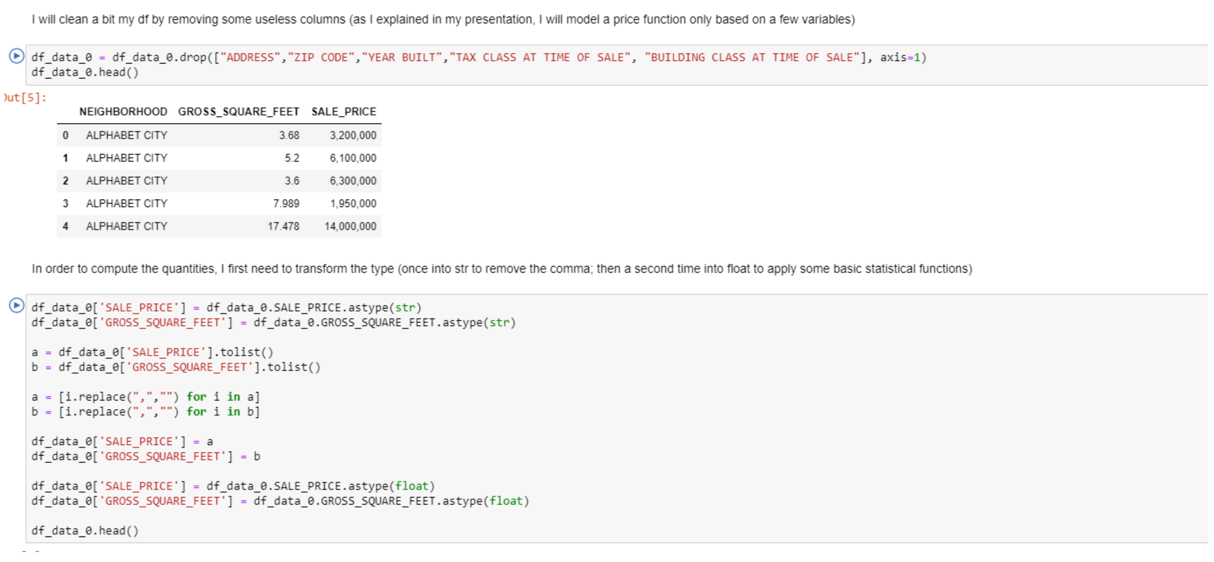


Methodology

Data Cleaning

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.[1] Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.


I will clean a bit my df by removing some useless columns (as I explained in my presentation, I will model a price function only based on a few variables)

```
df_data_0 = df_data_0.drop(["ADDRESS","ZIP CODE","YEAR BUILT","TAX CLASS AT TIME OF SALE", "BUILDING CLASS AT TIME OF SALE"], axis=1)
df_data_0.head()
```

)ut[5]:

	NEIGHBORHOOD	GROSS_SQUARE_FEET	SALE_PRICE
0	ALPHABET CITY	3.68	3,200,000
1	ALPHABET CITY	5.2	6,100,000
2	ALPHABET CITY	3.6	6,300,000
3	ALPHABET CITY	7.989	1,950,000
4	ALPHABET CITY	17.478	14,000,000

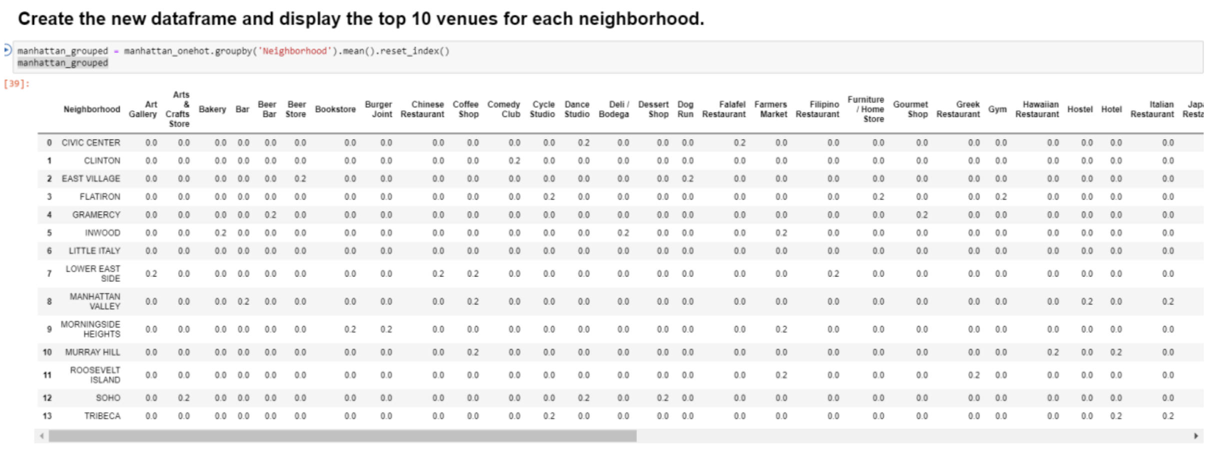
In order to compute the quantities, I first need to transform the type (once into str to remove the comma; then a second time into float to apply some basic statistical functions)

```
df_data_0['SALE_PRICE'] = df_data_0.SALE_PRICE.astype(str)
df_data_0['GROSS_SQUARE_FEET'] = df_data_0.GROSS_SQUARE_FEET.astype(str)

a = df_data_0['SALE_PRICE'].tolist()
b = df_data_0['GROSS_SQUARE_FEET'].tolist()

a = [i.replace(",","") for i in a]
b = [i.replace(",","") for i in b]

df_data_0['SALE_PRICE'] = a
df_data_0['GROSS_SQUARE_FEET'] = b

df_data_0['SALE_PRICE'] = df_data_0.SALE_PRICE.astype(float)
df_data_0['GROSS_SQUARE_FEET'] = df_data_0.GROSS_SQUARE_FEET.astype(float)

df_data_0.head()
```

Calculate the frequency of each venue per neighborhood using the basic function *.mean()*

In Python, we usually do this by dividing the sum of given numbers with the count of the number present. Python mean function can be used to calculate the mean/average of the given list of numbers. It returns the mean of the data set passed as parameters.

In Python, we usually do the dividing of the sum of given numbers with the count of number present inside the list, tuple or dictionary. First, we need to import the Python statistics module and then we can use the mean function to return the mean of the given list. See the following example.


Create the new dataframe and display the top 10 venues for each neighborhood.

```
manhattan_grouped = manhattan_onehot.groupby('Neighborhood').mean().reset_index()
manhattan_grouped
```

[39]:

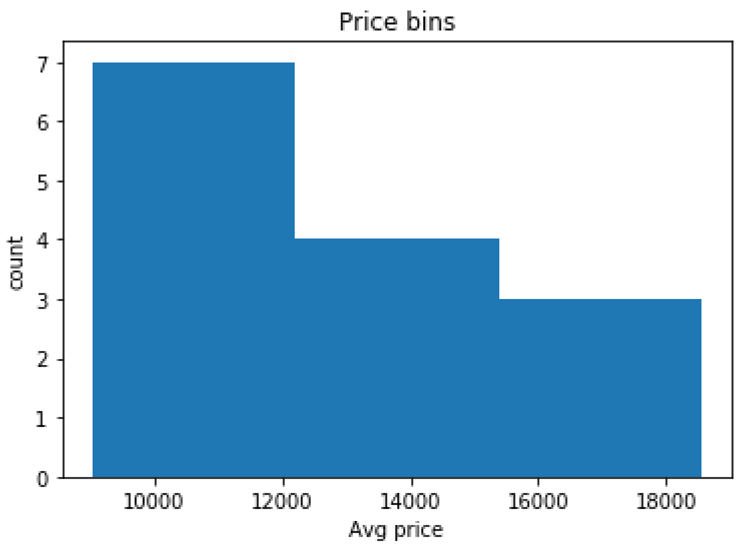
	Neighborhood	Art Gallery	Arts & Crafts Store	Bakery	Bar	Beer Bar	Beer Store	Bookstore	Burger Joint	Chinese Restaurant	Coffee Shop	Comedy Club	Cycle Studio	Dance Studio	Deli / Bodega	Dessert Shop	Dog Run	Falafel Restaurant	Farmers Market	Filipino Restaurant	Furniture / Home Store	Gourmet Shop	Greek Restaurant	Gym	Hawaiian Restaurant	Hostel	Hotel	Italian Restaurant	Japa Resta
0	CIVIC CENTER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	CLINTON	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	EAST VILLAGE	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	FLATIRON	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.2	0.0	0.0	0.0	0.0	
4	GRAMERCY	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	
5	INWOOD	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
6	LITTLE ITALY	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	LOWER EAST SIDE	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
8	MANHATTAN VALLEY	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.2	
9	MORNINGSIDE HEIGHTS	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10	MURRAY HILL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.2	0.0	
11	ROOSEVELT ISLAND	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	
12	SOHO	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13	TRIBECA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	

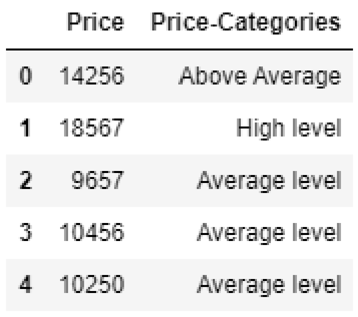
Binning

When dealing with continuous numeric data, it is often helpful to bin the data into multiple buckets for further analysis. There are several different terms for binning including bucketing, discrete binning, discretization or quantization. Pandas supports these approaches using the cut and qcut functions.

Data binning, which is also known as bucketing or discretization, is a technique used in data processing and statistics. Binning can be used for example, if there are more possible data points than observed data points. An example is to bin the body heights of people into intervals or categories. Let us assume, we take the heights of 30 people. The length values can be between - roughly guessing - 1.30 metres to 2.50 metres. Theoretically, there are 120 different cm values possible, but we can have at most 30 different values from our sample group. One way to group them could be to put the measured values into bins ranging from 1.30 - 1.50 metres, 1.50 - 1.70 metres, 1.70 - 1.90 metres and so on. This means that the original data values, will be assigned to a bin into wich they fit according to their size. The original values will be replaced by values representing the corresponding intervals. Binning is a form of quantization.

In the graph below we can see that I cut my continuous data into 3 discrete labels: **Average Price**, **Above Average** and **High Price**.





	Price	Price-Categories
0	14256	Above Average
1	18567	High level
2	9657	Average level
3	10456	Average level
4	10250	Average level

Test-train Data set

Creation of a train and test dataset

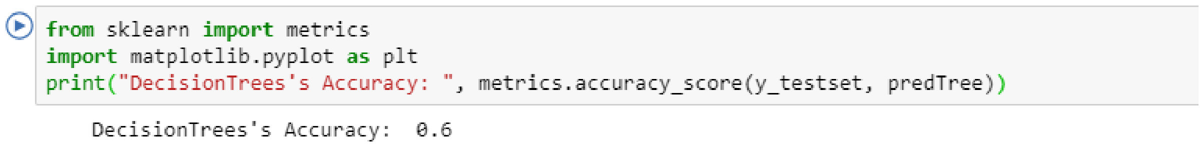
Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set.

This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the data. It is more realistic for real world problems.

This means that we know the outcome of each data point in this dataset, making it great to test with! And since this data has not been used to train the model, the model has no knowledge of the outcome of these data points. So, in essence, it’s truly an out-of-sample testing.

Results

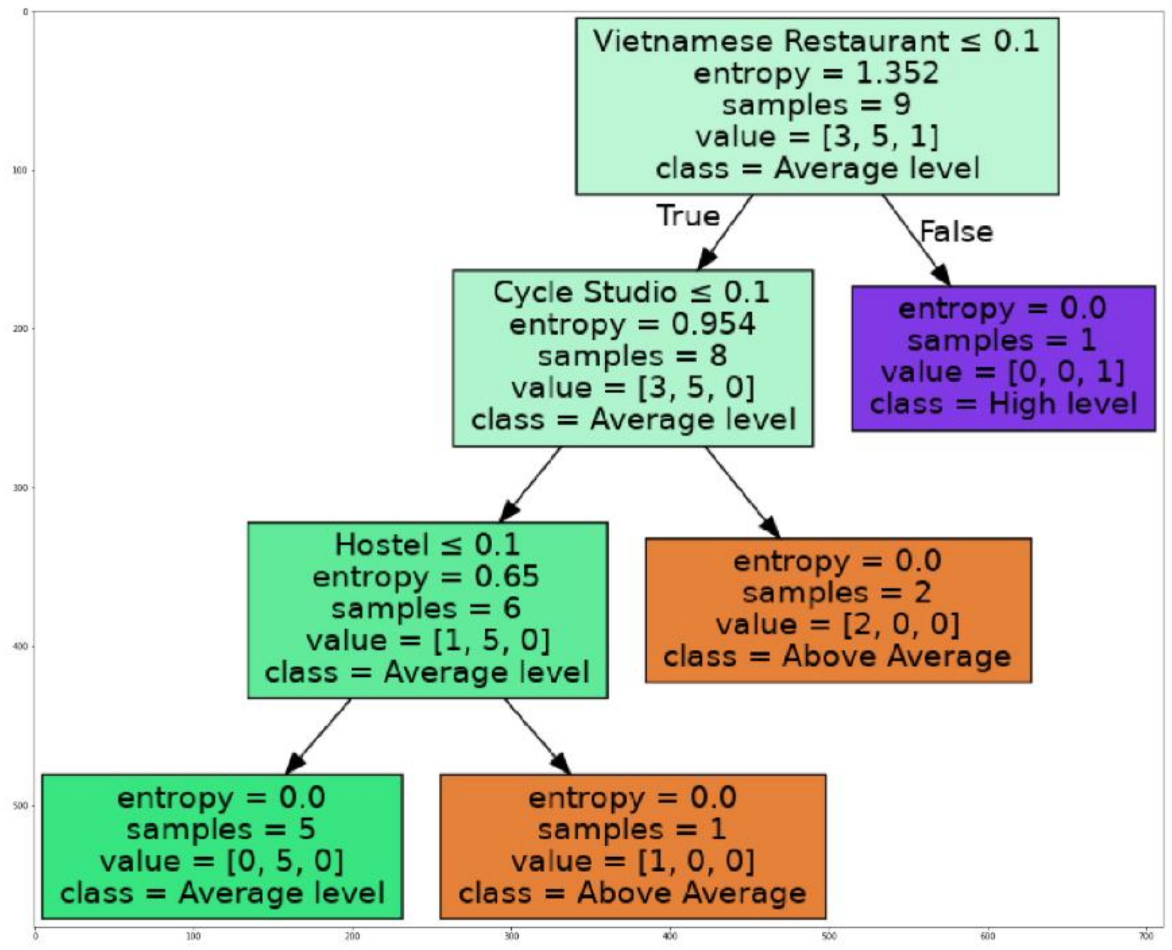
Accuracy-Score of the model



```
from sklearn import metrics
import matplotlib.pyplot as plt
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))
```

```
DecisionTrees's Accuracy:  0.6
```

Decision Tree – Visualization



Remarks

Problem using API

One of my biggest problem was the restriction on both API and IBM platform services. As I was facing many issues with my code, I had to re-run multiple times my code which made me consume way too quickly my free services.

The scarcity of good data

You can build the best comprehensive model for a given problem, but ultimately you will rely on the fuel: your data. As the data is the new modern gold, you will have to pay for better data.

Moreover, it is a discovery and a learning by itself to evaluate the sources of your data: which company, institution or web-communities provide good or bad data is a major skill that needs to be build on the way.

The complexity of cleaning data & the hidden code

Most of my problems were due to the datatypes, the shape of my dataframe. When you follow the labs, everything looks so easy because the data are ready-made. But in reality, you need to spend a lot of time to try understanding your data. The thing is that Python libraries are full of hidden functions. One of my biggest challenge will be to better understand the hidden code (i.e. how the functions work in details).

Conclusions

Overfitting

Overfitting means that model we trained has trained “too well” and is now, well, fit too closely to the training dataset. This usually happens when the model is too complex (i.e. too many features/variables compared to the number of observations). This model will be very accurate on the training data but will probably be very not accurate on untrained or new data. It is because this model is not generalized (or not AS generalized), meaning you can generalize the results and can’t make any inferences on other data, which is, ultimately, what you are trying to do. Basically, when this happens, the model learns or describes the “noise” in the training data instead of the actual relationships between variables in the data. This noise, obviously, isn’t part in of any new dataset, and cannot be applied to it.

We can quite spontaneously anticipate the reasons why my model clearly suffers from overfitting. First of all the binning realized is absolutely specific to my data-set, and the few data used to create it make it even less accurate.

Secondly, the price of an estate cannot be precisely estimated just by using the type of venues in a closed neighborhood. Many more factors should be added to the model to build a stronger and more precise model.