**The battle of the neighbourhoods: Discovering Casablanca**

By: Ahmed KCHIKECHE

# Introduction to the business problem

As the largest city in Morocco, Casablanca is one of the best investment destinations in north Africa. Casablanca is located in the centre of the Casablanca-Settat region who according to the ministry of finance[[1]](#endnote-1) contribute 26.5 per cent to the nation’s GDP.

Casablanca’s strategic location, the availability of undertrial and logistical infrastructure and its attractive business climate makes the city the main destination for startups and large-scale investment project. However, the publicly available microdata on the locations of venues is extremely scarce. This can make it hard to choose the best location for an investment project.

The objective of this project is to facilitate the choice of businesses locations in Casablanca based on the frequency of nearby venues.

The goals of this project are:

* Identify the geographical position of all the neighbourhoods in the city of Casablanca
* Identify the nearby venues to each neighbourhood by frequency
* Cluster the neighbourhoods based on each neighbourhood by frequency

# Data

To achieve the goals of our project, we will need to get the following datasets

1. The names and postal codes of all neighbourhoods in the city of Casablanca;
2. The coordinates of each neighbourhood;
3. The nearby venues data for each neighbourhood.

The data on postal codes in Casablanca was obtained from a webpage[[2]](#endnote-2) from the postal service of morocco website. We scraped the webpage using the .read\_html() pandas method. After processing the data, we obtained a table containing two columns (The neighbourhood name and its corresponding postal code) and 3048 rows.

The second step of our data gathering process was to get the coordinates (Latitude, Longitude) of each neighbourhood. To do so, we used ArcGIS geocoder. After extracting the coordinates of each neighbourhood, we added two new columns to our previous dataset. This version of the dataset was cleaned to fill missing data and remove duplicate and redundant information.

Finally, we used the Foursquare API to get the nearby venues for each neighbourhood in the processed dataset. This data was then processed and made available for clustering. The final dataset consisted of the neighbourhoods names, coordinates and the 10 most frequent nearby venues for each neighbourhood.

# Methodology

## Getting and Cleaning Data

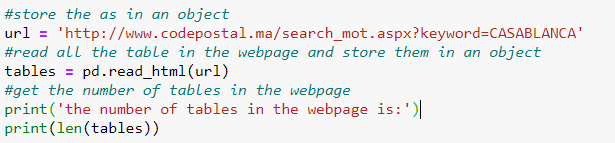
The first part of any data project is Data acquisition and wrangling. In this project, I needed to get the following data

* The names and postal codes of all neighbourhoods in the city of Casablanca;
* The coordinates of each neighbourhood in Casablanca;
* Venues data for each neighbourhood in Casablanca.

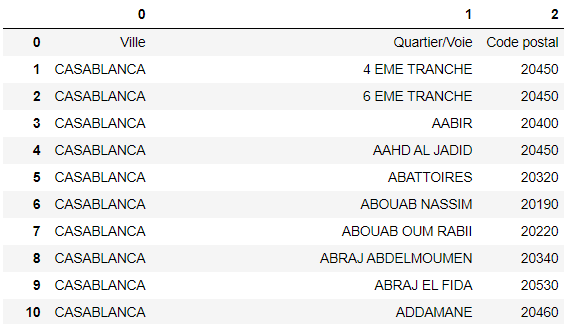
### Neighbourhoods data acquisition

The data on postal codes in Casablanca is obtained from The postal service of Morroco.

To prepare the dataset for modelling, we need first to scrap it and store it in a pandas data frame. To scrap the data from the webpage tables we will use the .read\_html() pandas method. To do so, we used the following code



We found that the data contains 10 tables. We checked each of them and found that the tables that contain all the neighbourhoods in Casablanca are table 5 shown below.



We then Saved the table in a pandas data frame, verified the data in it, dropped non-essential column and translated the column names to English.

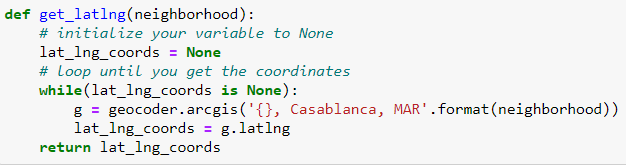
Using the .info() method we can check the processed dataframe main properties

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 3048 entries, 1 to 3048  Data columns (total of 2 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 neighborhood 3048 non-null object  1 postalcode 3048 non-null object  dtypes: object(2)  memory usage: 47.8+ KB |

### Adding the coordinates data to our data frame

This was challenging. After trying to use Google geocoder in vain, The loop took a very long time, we used ArcGIS geocoder to extract the neighbourhoods coordinates.

The Python function used the extract the data from the geocoder is the following.



We defined the function, called the function to get the coordinates and store in a new list using list comprehension. Afterwards, we created a temporary data frame to populate the coordinates into Latitude and Longitude, then merge the coordinates into the original data frame.

Finally, we checked the resulting data frame for missing data and found that the geocoder didn’t get the coordinates for one of the neighbourhoods. To solve this problem, we used google map to find the coordinates manually and added them to our data frame.

The resulting dataframe was the following

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  Int64Index: 3048 entries, 0 to 0  Data columns (total of 4 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 neighborhood 3048 non-null object  1 postalcode 3048 non-null object  2 Latitude 3048 non-null object  3 Longitude 3048 non-null object  dtypes: object(4)  memory usage: 119.1+ KB |

### Removing duplicate neighbourhoods

To make sure we don’t have any redundant data (e.i. Duplicate neighbourhoods or neighbourhoods with the same coordinates), we filtered the data to get distinct neighbourhoods.

First, we dropped 67 duplicate neighbourhoods from our dataset. The duplicate neighbourhoods could result from neighbourhoods with more than one postal code. This could be due to inconsistencies in the postal service data or to the fact that different addresses in the same neighbourhoods have different postal codes. The resulting data frame can be seen below.

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  Int64Index: 2981 entries, 0 to 0  Data columns (total of 4 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 neighborhood 2981 non-null object  1 postalcode 2981 non-null object  2 Latitude 2981 non-null object  3 Longitude 2981 non-null object  dtypes: object(4)  memory usage: 116.4+ KB |

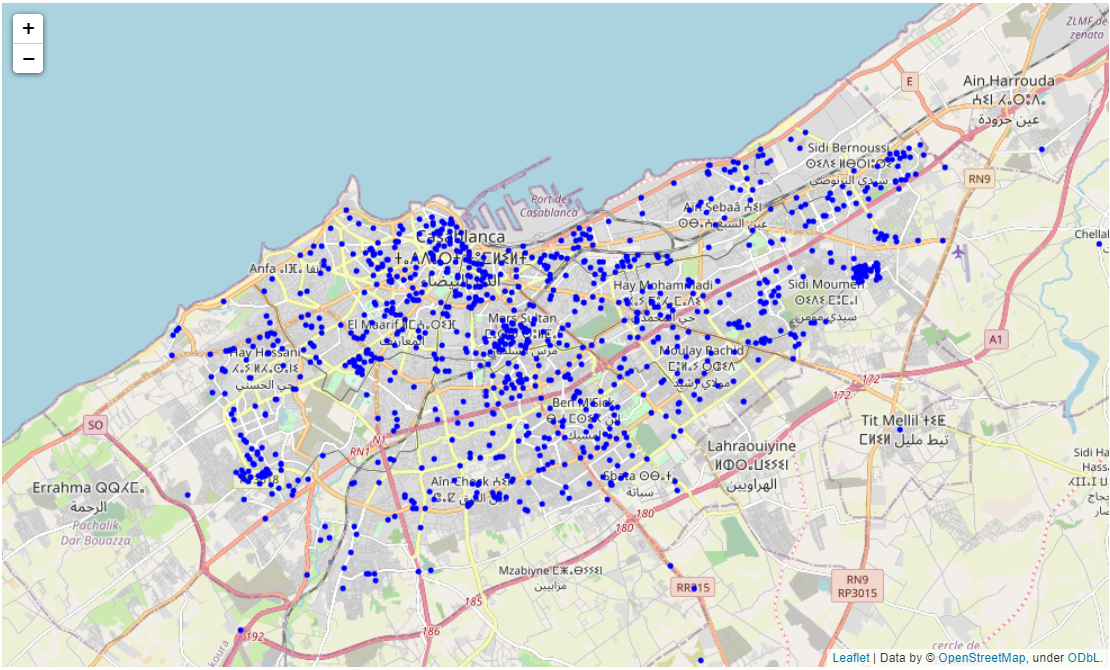
We then dropped neighbourhoods that have the same coordinates. These neighbourhoods could be close enough that the geocoder didn’t have distinct coordinates for each of them. One problem with the postal service data that is collected to give addresses to each building and this is extremely detailed. To achieve this, the same neighbourhood is devised into partitions with different postal codes. We don’t need this level because we will get the nearby venues for each neighbourhood and we don’t want these venues to intersect.

After dropping neighbourhoods with similar coordinates we got the following data frame. This operation reduced the number of distinct neighbourhoods to 1038 dropping 1943 neighbourhoods.

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  Int64Index: 1038 entries, 0 to 0  Data columns (total of 4 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 neighborhood 1038 non-null object  1 postalcode 1038 non-null object  2 Latitude 1038 non-null object  3 Longitude 1038 non-null object  dtypes: object(4)  memory usage: 40.5+ KB |

Finally, we decided to drop neighbourhoods with the same postal code and got our final neighbourhoods positions data frame.

### Exploring Casablanca and verifying data usability

To show a visualisation of the neighbourhoods in Casablanca. we created a map using the coordinates that we got before and added markers of the neighbourhoods on top.

Unfortunately, Some data points are outside Casablanca.

Ideally, We should only take into account neighbourhoods in ‘Prefecture of Casablanca’. To do so we need to get the coordinates of the city's administrative polygon boundary.

This is done by following these steps

1. use OpenStreetMap https://nominatim.openstreetmap.org/ to search for the city
2. Get the city details <https://nominatim.openstreetmap.org/ui/details.html?osmtype=R&osmid=2523504&class=boundary>
3. Copy the OSM ID. 2523504
4. use http://polygons.openstreetmap.fr/index.py to search for the polygon using the OSM ID.
5. Get the polygon from <http://polygons.openstreetmap.fr/index.py?id=2523504>
6. Calculate if a point is inside the polygon or not and filter out the points outside the polygon.

\*All the credit for the method to get the polygon goes to https://gis.stackexchange.com/a/192298\*

However, we could not implement this solution with our current knowledge and experience with python programming and due to time restrictions. the number of data points outside Casablanca is very low and can be tolerated.

### Getting Venues information

The last step of data collection is getting nearby venues to each neighbourhood in our dataset using the Foursquare API using a developer account. To do so, we used a python function to get nearby venues and store them in a data frame. The resulting data frame information is shown bellow

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 7995 entries, 0 to 7994  Data columns (total of 7 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Neighborhood 7995 non-null object  1 Neighbourhood Latitude 7995 non-null float64  2 Neighbourhood Longitude 7995 non-null float64  3 Venue 7995 non-null object  4 Venue Latitude 7995 non-null float64  5 Venue Longitude 7995 non-null float64  6 Venue Category 7995 non-null object  dtypes: float64(4), object(3)  memory usage: 437.4+ KB |

The data frame contains 4 new columns containing information about nearby venues like their position and category.

## Modeling

### Feature engeneering

To apply the clustering algorithm to our dataset, we did the following

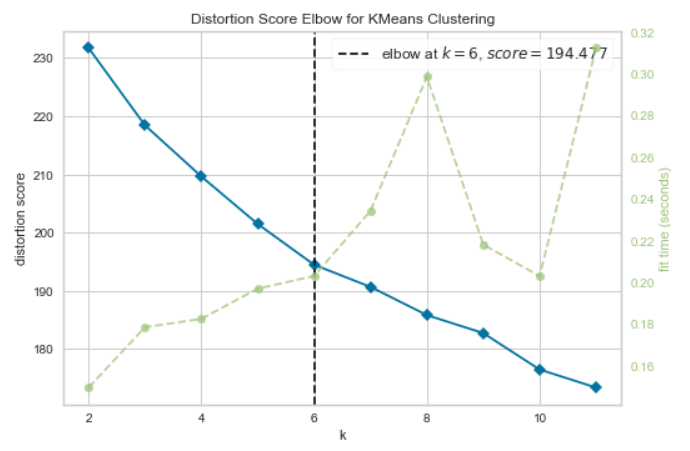
* Used one-hot encoding to get dummies for our categorical variables;
* Grouped rows by neighbourhood and by taking the mean of the frequency of occurrence of each category;
* Sorted the venues in descending order of venues categories frequency;
* Created a new data frame displaying the top 10 venues for each neighbourhood.

The resulting data frame is shown below.

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 971 entries, 0 to 970  Data columns (total of 11 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Neighborhood 971 non-null object  1 1st Most Common Venue 971 non-null object  2 2nd Most Common Venue 971 non-null object  3 3rd Most Common Venue 971 non-null object  4 4th Most Common Venue 971 non-null object  5 5th Most Common Venue 971 non-null object  6 6th Most Common Venue 971 non-null object  7 7th Most Common Venue 971 non-null object  8 8th Most Common Venue 971 non-null object  9 9th Most Common Venue 971 non-null object  10 10th Most Common Venue 971 non-null object  dtypes: object(11)  memory usage: 83.6+ KB |

### Clustering

To apply k-mean clustering, we used the Elbow method to choose the optimal k. The figure below shows a visualization of the optimal k.

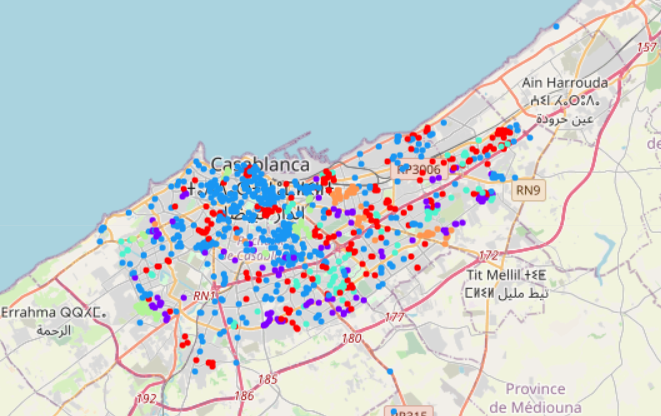


As we can see. The k-mean algorithm should be initialized with 6 clusters.

After applying the clustering algorithm, we added the cluster labels and the coordinates to the previous data frame, the results are shown below

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  Int64Index: 971 entries, 0 to 970  Data columns (total of 15 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Neighborhood 971 non-null object  1 postalcode 971 non-null object  2 Latitude 971 non-null object  3 Longitude 971 non-null object  4 Cluster Labels 971 non-null int32  5 1st Most Common Venue 971 non-null object  6 2nd Most Common Venue 971 non-null object  7 3rd Most Common Venue 971 non-null object  8 4th Most Common Venue 971 non-null object  9 5th Most Common Venue 971 non-null object  10 6th Most Common Venue 971 non-null object  11 7th Most Common Venue 971 non-null object  12 8th Most Common Venue 971 non-null object  13 9th Most Common Venue 971 non-null object  14 10th Most Common Venue 971 non-null object  dtypes: int32(1), object(14)  memory usage: 117.6+ KB |

Using this new data frame, we can visualize the clusters on the map.



# Results

After applying the clustering algorithm, we can show the number of neighbourhoods in each cluster. Furthermore, we aggregated the data by calculating the frequency of occurrence of venues categories as the 1st, 2nd and 3rd most common venue category in each cluster.

This information could be used by potential investors to analyse the neighbourhoods according to the frequency of venues. The results of our analysis are shown below.

Let’s first see the number of neighbourhoods in each cluster :

|  |  |
| --- | --- |
| The cluster | The number of neighbourhoods |
| 0 | 216 |
| 1 | 121 |
| 2 | 463 |
| 3 | 76 |
| 4 | 54 |
| 5 | 41 |

Lets now see the frequency of the most common venues in each cluster

|  |  |  |
| --- | --- | --- |
| The most common venues in cluster 0 | Venues categories | frequency |
| 1st Most Common Venue | Coffee Shop | 74.5% |
| Zoo Exhibit | 3.7% |
| Fast Food Restaurant | 2.3% |
| 2nd Most Common Venue | Coffee Shop | 34.3% |
| Fast Food Restaurant | 7.4% |
| Hotel | 3.2% |
| 3rd Most Common Venue | Yoga Studio | 13.0% |
| Coffee Shop | 16.7% |
| Fast Food Restaurant | 6.9% |

|  |  |  |
| --- | --- | --- |
| The most common venues in cluster 1 | Venues categories | frequency |
| 1st Most Common Venue | Coffee Shop | 61.2% |
| Zoo Exhibit | 18.2% |
| Convenience Store | 4.1% |
| 2nd Most Common Venue | Coffee Shop | 44.6% |
| Fast Food Restaurant | 9.9% |
| Snack Place | 5.0% |
| 3rd Most Common Venue | Coffee Shop | 24.0% |
| Yoga Studio | 9.9% |
| Pizza Place | 9.9% |

|  |  |  |
| --- | --- | --- |
| The most common venues in cluster 2 | Venues categories | Frequency |
| 1st Most Common Venue | Coffee Shop | 24.2% |
| Fast Food Restaurant | 11.2% |
| Hotel | 6.0% |
| 2nd Most Common Venue | Yoga Studio | 10.8% |
| Fast Food Restaurant | 8.9% |
| Hotel | 7.3% |
| 3rd Most Common Venue | Coffee Shop | 5.4% |
| Pizza Place | 4.5% |
| Bakery | 4.1% |

|  |  |  |
| --- | --- | --- |
| The most common venues in cluster 3 | Venues categories | Frequency |
| 1st Most Common Venue | Coffee Shop | 100.0% |
|  |  |
|  |  |
| 2nd Most Common Venue | Yoga Studio | 51.3% |
| Ice Cream Shop | 6.6% |
| Bistro | 5.3% |
| 3rd Most Common Venue | Donut Shop | 59.2% |
| Yoga Studio | 32.9% |
| Doner Restaurant | 3.9% |

|  |  |  |
| --- | --- | --- |
| The most common venues in cluster 4 | Venues categories | Frequency |
| 1st Most Common Venue | Shopping Mall | 27.8% |
| Snack Place | 13.0% |
| Coffee Shop | 18.5% |
| 2nd Most Common Venue | Yoga Studio | 10.8% |
| Fast Food Restaurant | 8.9% |
| Hotel | 7.3% |
| 3rd Most Common Venue | Shopping Mall | 35.2% |
| Yoga Studio | 11.1% |
| Pool Hall | 11.1% |

|  |  |  |
| --- | --- | --- |
| The most common venues in cluster 5 | Venues categories | Frequency |
| 1st Most Common Venue | Tram Station | 46.3% |
| Coffee Shop | 31.7% |
| Hot Dog Joint | 12.2% |
| 2nd Most Common Venue | Coffee Shop | 51.3% |
| Tram Station | 6.6% |
| Pizza Place | 5.3% |
| 3rd Most Common Venue | Donut Shop | 31.7% |
| Yoga Studio | 24.4% |
| Doner Restaurant | 12.2% |

# Discussion

The objective of this project was to provide potential investor in Casablanca with an easy way to access location data neighbourhoods locations and the frequency of venues categories in each neighbourhood. This task however was complicated by various factors related to data availability and accuracy. This is a very incomplete project; however, we hope that it could be a starting point to more extensive analysis.

This section of the report will discuss various problems that we encountered in the various stages of this project.

First of all, the only exhaustive list of neighbourhood names and their postal codes was found on a website of the postal code of Morocco. We found that some neighbourhoods have the same postal code and thus could be close to each other and thus redundant. Removing neighbourhoods with duplicate postal code would reduce the number of neighbourhoods dramatically (to less than 100 neighbourhood) we did not remove these neighbourhoods because we are not sure to which degree neighbourhoods with the same postal code are close to each other. This needs more testing and could have a damaging effect on our data quality.

The second source of data instability was the ArcGIS geocoder that we used to get the coordinates for our neighbourhoods. The geocoder did assign wrong coordinates to some of the neighbourhoods (coordinates outside Casablanca). This could result from the fact that the geocoder uses different naming conventions for neighbourhoods or due to some error in our code. This problem could be solved partially if we can exclude the neighbourhoods that are outside Casablanca. We couldn’t however implement this solution due to inexperience with Python.

We could also improve our results using cross-validation to choose the optimal k or by using multiple clustering algorithms and comparing the results.

# Conclusion

The objective of this project is to facilitate the choice of businesses locations in Casablanca based on the frequency of nearby venues. Using various data source we compiled an exhaustive list of all the neighbourhoods in Casablanca, their location and the most common venue categories in each neighbourhood. We then used k-mean clustering to make 6 clusters and we aggregated the data for each cluster.

1. https://www.finances.gov.ma/Publication/depf/2019/profils-regionaux.pdf [↑](#endnote-ref-1)
2. http://www.codepostal.ma/search\_mot.aspx?keyword=CASABLANCA [↑](#endnote-ref-2)