Project P:

PINPOINT PERFECT PARISH (Porto City)

Ivan Kisialiou

April – May 2019

1. INTRODUCTION

1.1. Background

Imagine, you want to visit or relocate to Porto City (Portugal). Which district would you choose for a stay? Not far from the ocean? Or, perhaps, you dislike noise and winds, so you prefer to stay neither by the shore, nor in the City Centre? Or, maybe, you are a student, so the best neighbourhood for you is located not far from the University? Anyway, you probably want to know, how districts of the City differ from each other and which are similar. Something that is important for one person may be neglectable for another one. It may take loads of time to get acquainted with the City, and some traits of the districts may stay unobvious.

In these terms, people might be interested in a system showing them – based on their individual tastes – similar areas of the city with detailed description, so that they could make a choice easier. The main target audience is everyone who desires to choose place for living in Porto. It may also be useful for real estate agencies as well as for business desiring to open a new venue.

1.2. Problem

The objective of my capstone Project P, named PINPOINT PERFECT PARISH (Porto City), is to build a system that helps a customer to choose the district in Porto to settle down.

A customer cannot know all types of venues in the City. And we are not expecting them to think out succinct criteria for the segmentation. The thing we really can do here is to ask them to fill simple questionnaire and to rank suggested features in terms of importance for their choice. Given this information we can customise the segmentation individually.

For the demonstration purpose, I'm going to perform analysis of Porto Parishes for a one certain person – say, a student of Porto University – providing that we know their preferences. The output of the cluster analysis will be a map with labelled neighbourhoods (Parishes) on it and detailed descriptions for every parish type.

2. DATA

In order to perform clustering over Porto City districts (Parishes), I consider venues within each parish together with distances from the parish centre to the City Centre ("Remoteness"), to the sea shore, to the Porto University (FEUP), and to the airport.

The customer's preferences will be included to a model as weights.

2.1. Parishes of Porto

First, we need names and coordinates of geographical areas in Porto. Portuguese administrative division is quite complicated and includes 18 districts [1].

Porto District comprises 18 Municipalities, and each Municipalities has several Parishes ("Freguesias") in it. There is no list of Parishes in Porto Agglomeration, so we can select them by their remoteness. The complete list of Portuguese Parishes can be found here on Wikipedia [2].

I carried out parsing of this page and selected Parishes in Municipalities of Porto District.

I used *OpenStreetMap Nominatim* [3] for geocoding.

Cleaning: dropped Parishes that are too far (>10 km) from the City Centre; split several joint Parishes (e.g. "Aldoar, Foz do Douro e Nevogilde") to obtain more points om map; moved one geo-point ("Matosinhos") to its true position.

The resulting dataframe and the map with Parishes are shown in Fig.1 and Fig.2.

	Municipality	Parish	Latitude	Longitude	Remoteness_km
0	Gondomar	Baguim do Monte (Rio Tinto)	41.187473	-8.537759	7.4
1	Gondomar	Fânzeres	41.171488	-8.531211	7.1
2	Gondomar	São Pedro da Cova	41.153457	-8.506091	8.8

Figure 1. Dataframe with geo-data for Porto City Parishes

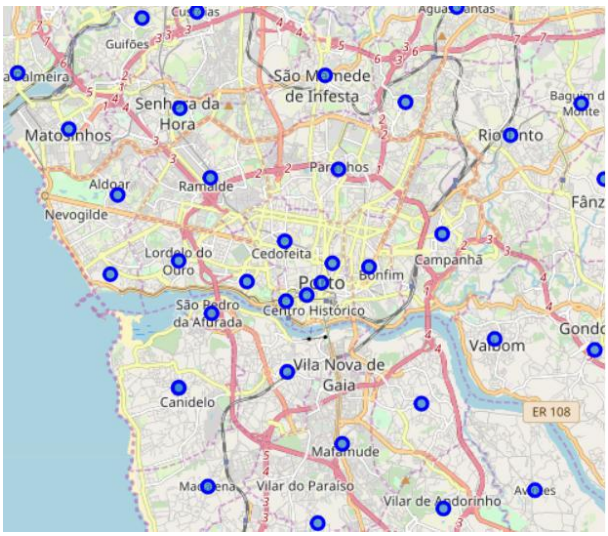

Figure 2. A map with the centres of Porto Parishes on it

2.2. Venues

I applied Foursquare API [4] to obtain list of maximum 100 most popular venues for each Parish. The request form is shown below.

```
# create the API request URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'
.format(
    fs.id, # user's ID
    fs.secret, # user's Secret
    fs.version, # API version
    lat, # latitude
    lng, # longitude
    radius, # radius to explore
    limit)  # number of venues to return
```

Figure 3. Foursquare API request for fetching the most popular venues within the certain radius from given geo-point

The radius parameter for request with endpoint “explore” was set to the half of average distance between two neighbouring Parishes in Porto (1.67 km). To do that I had to write the code calculating this parameter for a set of geo-points.

The resulting dataframe is as follows:

	Municipality	Parish	Parish Latitude	Parish Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Gondomar	Baguim do Monte (Rio Tinto)	41.187473	-8.537759	O "Zé Pacheco"	41.188296	-8.539716	Food
1	Gondomar	Baguim do Monte (Rio Tinto)	41.187473	-8.537759	Lidl	41.182358	-8.537666	Grocery Store
2	Gondomar	Baguim do Monte (Rio Tinto)	41.187473	-8.537759	Tentação das Bifanas	41.184127	-8.539149	Restaurant

Figure 4. Dataframe with the data about venues in Porto City Parishes

2.3. Distances

Besides “Remoteness”, I’m going to consider distances to the Airport, to the seashore, and to the University, as our sample customer is a student of The Porto

University (FEUP). Distances to the Airport and to the University can be calculated in the same way as "Remoteness" – with the help of the method *.distance* from geopy library [6]. The calculation of the distance to the seashore (line shown in Fig. 5) requires the knowledge of cross-track distance concept.

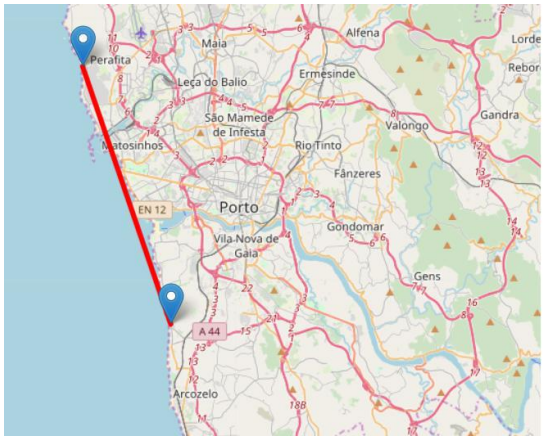

Figure 5. The model line representing the seashore for distance calculation

The cross-track distance can be analytically calculated with the help of mathematical formulas, but fortunately there is n-vector library for Python [7].

2.4. Preferences

We can formalize the customer's preferences about segmentation criteria by asking whether the given feature is important for their choice, or not. Then the answers are transformed into the weight vector, which we apply to the features. The features are described in Feature Engineering section.

3. METHODOLOGY

The goal of the study is to make a tip for a customer, where to live in Porto. One of the ways to do that is to compare different Parishes, find the common groups and underline the differences, so that a customer could make a choice easier. Based on the form of data we have, the best solution, probably, would be a clustering approach. We can group all Parishes of Porto City to several clusters and then analyse their most prominent traits and differences. A customer can pick the cluster they like and choose a Parish there, according to their tastes. Customer's tastes can be also included to the model, so that clustering algorithm considers only important features of Parishes and ignores unimportant ones.

3.1. Feature engineering. Selection of metrics

In order to perform clustering over Porto City districts (Parishes), we need to construct a set of features, representing unique profile of each Parish. The data we have for a Parish is 4 distances and a list of maximum 100 most popular venues. We can encode this venue list to a set of columns, showing the contribution of every venue to the profile. For every type of venues in a Parish, we can use either *frequency* count (percentage of such venues in the Parish), or a total count of such venues normalized by the maximum count of such venues among all Parishes. Frequency works well only when we have approximately equal number of all venues for every Parish. Otherwise, for example, a Parish with a single venue appears “better” than a Parish with a large amount of different venues, which is incorrect.

In our dataset, there are both Parishes with over 100 venues and with very few venues (see histogram in Fig. 6). Thus, we must use the second metrics (normalized count).

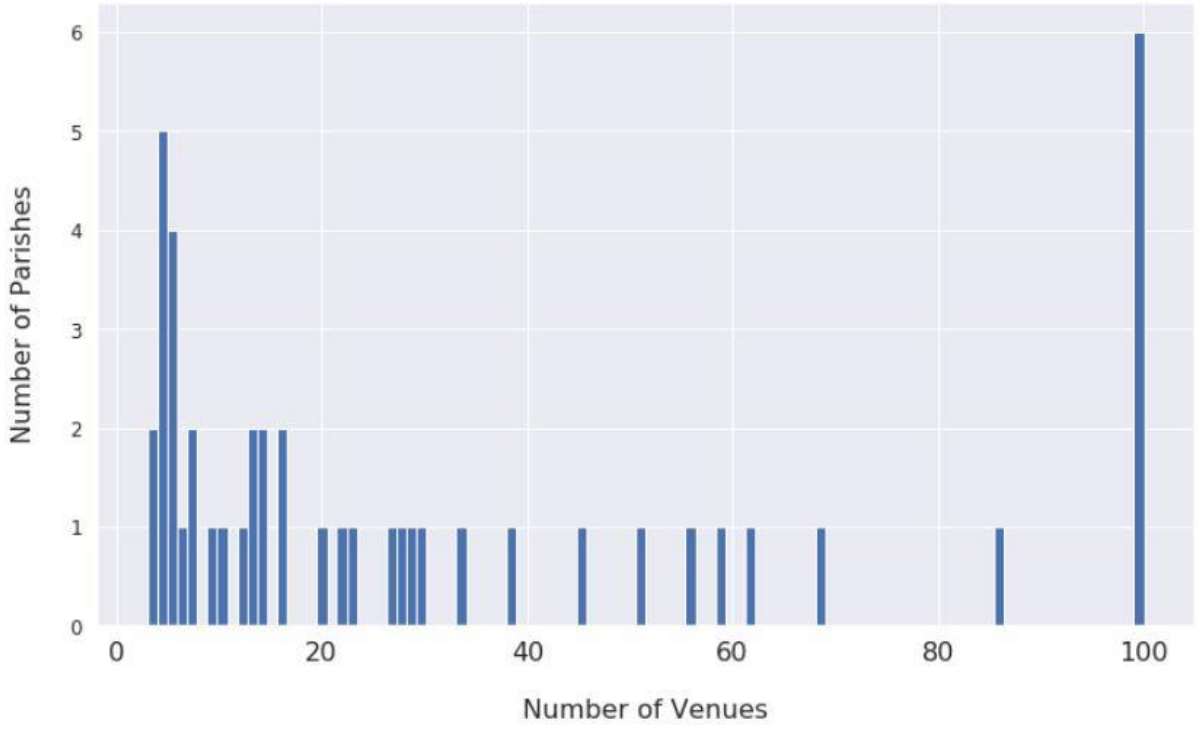

Figure 6. Distribution of venues in Parishes

3.2. Feature engineering. General venue categories and feature index

The request to Foursquare API gives 176 unique types of venues. Some of them are similar, some are too uncommon, some are useless. Anyway, a large portion of these venue types cannot serve as distinguishing criteria for Parishes. We must group and transform these types into more general and more informative categories. To make the process easier, I formed alphabetical list of all 176 categories (see Fig. 7), and sorted them manually into 22 new groups, including group “Waste” for uninformative categories like “Plaza”, “Neighbourhood”, or “Road”.

A	Fish & Chips Shop	Nightclub
American Restaurant	Flea Market	Noodle House
Argentinian Restaurant	Flower Shop	**O**
Art Gallery	Food	Other Great Outdoors
Art Museum	Food & Drink Shop	**P**
Arts & Crafts Store	Food Court	Park
Asian Restaurant	Food Truck	Pastry Shop
Athletics & Sports	Fried Chicken Joint	Pet Store
Auto Garage	Furniture / Home Store	Pharmacy
B	**G**	Pier
BBQ Joint	Garden	Pizza Place
Bagel Shop	Gas Station	Planetarium
Bakery	Gastropub	Playground
Bar	Gift Shop	Plaza
Beach	Gourmet Shop	Pool

Figure 7. A list of all venue categories

Finally, we have a set of 25 features: 4 distances and 21 (without “Waste”) general venue category. Each feature is characterized by its feature index. Feature index takes values from 0 to 1, 1 is the best and 0 is the worst. For general venue categories, feature index equals normalized venue count. For distances, feature index equals one minus normalized distance. I use division by maximum as normalization method.

The number 22 is much less than 176, but still hard to interpret. We will discuss it later, when consider the ways to build cluster description. For analysis of parish clusters, it’s better to find 5 most significant categories (i.e. with the highest feature index). But note, that all 25 features participate in machine learning.

3.3. Feature engineering. Customer’s preferences and weighting

Basically, when dealing with normalized data, clustering algorithm equally treats all features. However, we can artificially increase significance of specified features by introducing feature weights. With feature weights some features will contribute to

the model more than others. It's a good moment to take customer's preferences into consideration. We provide a customer with a simple questionnaire form containing the list of all features. We suggest choosing one of three possible options for every feature – "Important", "Normal", or "Not important" (Fig. 8).

Feature	Importance	Description
Remoteness_km	Important	*stance to City Centre*
Airport_km	Important	
Unversity_FEUP_km	Normal Not important	
To_sea_km	Normal	*Distance to seashore*
Alcohol	Normal	
Arts & Museums	Not important	
Basic Shops & Services	Normal	
Café and Desserts	Important	
Car Service	Normal	
Chillout	Not important	*Spa etc.*
Entertainment	Not important	
Fitness & Sports	Important	
Food Places	Normal	

Figure 8. A form for collecting customer's preferences

Then we can assign weight to feature importance, say, "0" for "Not important", "1" for "Normal" and "2" for "Important". Thus, we obtain the weight vector of features. We can modify some weights after that, to make some important features even more significant. Our model customer is a student, so I assigned weight "3" to the feature "University_FEUP_km".

We get weighted features as Hadamard (element-wise) product of features index vectors and the weight vector. In the end of this operation we have a matrix, ready for clustering.

3.4. Model selection

The data we have is numerical and consists of many independent variables. This is a good case for applying K-means clustering algorithm as one of the most effective, fast and robust methods of clustering. While the original method may be sensitive to initial data and to random initialization of clustering seeds, I use K-means++, which returns the best output of several runs.

To define proper number of clusters (k), we can compute clustering error for different values of k in certain range. When too many clusters, it is hard to characterize them. Too few results in large errors and low value of the information.

It is common to apply “elbow method” for choosing number of clusters. In Fig. 9, we can see a plot representing *k-means error* versus *k* number. It can lightly change from run to run, but the common picture stays the same.

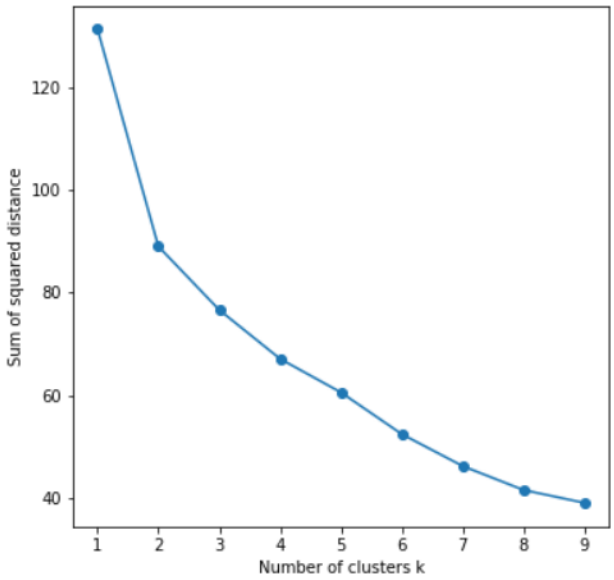

Figure 9. Visualization of cluster number impact on k-means error

As we see, “elbow method” is hardly applicable to this case. I use exponential decay constant as a threshold instead. This means that I choose *k* according to the point when clustering error drops in *e* (2.7...) times. The corresponding value of *k* is 7.

The second parameter we can adjust is the number of initializations of *k-means++*. We apply can view clustering error versus the number of initializations and pick the minimum value, providing acceptable result. My choice is the value 30 (Fig. 10).

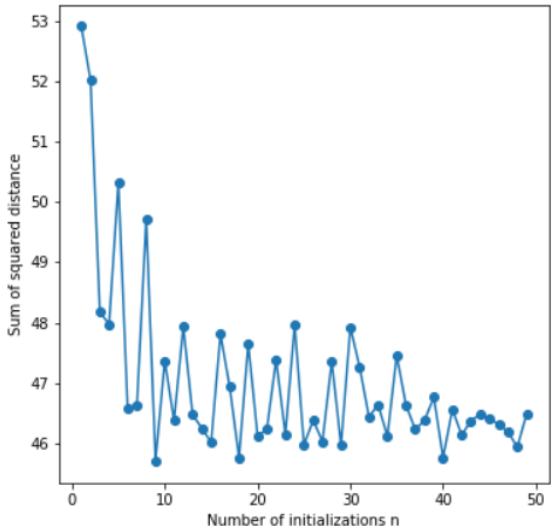

Figure 10. Impact exploration of different numbers of k-means++ initializations

4. RESULTS

The raw results of clustering are 7 numeric cluster labels and 7 cluster centroids (i.e. mean values of all features within every cluster).

First, we must merge back feature index table and Parishes with their coordinates, adding cluster labels. We include centroids into a table as virtual Parishes. We can leave their coordinates blanc, but I put coordinates of City Centre there. The resulting dataframe is as follows:

	Municipality	Parish	Latitude	Longitude	Cluster	Remoteness_km	Unversity_FEUP_km	To_sea_km	Alcohol	Basic Shops & Services	Café and Desserts	Car Service	Fitness & Sports
44	Matosinhos	Matosinhos	41.182060	-8.681925	1	0.456522	1.169492	0.908451	0.041667	0.200000	1.727273	1.000000	0.571429
45		Cluster 0	41.149451	-8.610788	0	0.386473	0.629944	0.567293	0.004630	0.088889	0.111111	0.000000	0.190476
46		Cluster 1	41.149451	-8.610788	1	0.774457	1.242585	0.804577	0.078125	0.125000	0.840909	0.125000	0.678571
47		Cluster 2	41.149451	-8.610788	2	1.560870	2.094915	0.630986	0.225000	0.480000	1.381818	0.000000	1.314286
48		Cluster 3	41.149451	-8.610788	3	0.637681	1.783898	0.342723	0.031250	0.033333	0.325758	0.000000	0.166667
49		Cluster 4	41.149451	-8.610788	4	1.373913	2.049153	0.615493	0.091667	0.360000	0.654545	0.000000	0.685714
50		Cluster 5	41.149451	-8.610788	5	0.536232	1.754237	0.535211	0.013889	0.066667	0.090909	0.333333	0.380952
51		Cluster 6	41.149451	-8.610788	6	1.891304	2.033898	0.629108	0.847222	0.200000	1.303030	0.000000	0.000000

Figure 11. Raw resulting table with added centroids

In order to retrieve useful information, we have to transform this data in something more convenient for interpretation. One of the possible approaches is to show top features for every Parish. The ranking is performed on feature indices, so the 5 top features of the selected Parish are features with the highest – within the Parish – count/absolute maximum ratio, given the weights. To reveal the quantitative characteristic of every feature, I assign rank labels “Top”, “Normal”, and “Outsider” to every feature index in a Parish:

- “Top”: A Parish is in 1st third among all Parishes by selected feature
- “Normal”: A Parish is in the 2nd third among all Parishes by selected feature
- “Outsider”: By this feature, a Parish is in the last third among all Parishes

The rank labels of either all, or top-5 features of a centroid shows the average characteristics of the cluster, like in **Figure 12**:

	Municipality	Parish	Latitude	Longitude	Cluster	1st Feature	2nd Feature	3rd Feature	4th Feature	5th Feature
43	Vila Nova de Gaia	Vilar de Andorinho	41.101582	-8.576659	0	Unversity_FEUP_km | Outsider	Remoteness_km | Normal	To_sea_km | Normal	Food Stores & Markets | Normal	Café and Desserts | Outsider
45		Cluster 0	41.149451	-8.610788	0	Unversity_FEUP_km | Outsider	To_sea_km | Normal	Remoteness_km | Outsider	Food Stores & Markets | Outsider	Fitness & Sports | Outsider

Cluster	Remoteness_km	Unversity_FEUP_km	To_sea_km	Alcohol	Basic Shops & Services	Café and Desserts	Car Service	Fitness & Sports	Food Places	Food Stores & Markets	Nature	Public Transport	Restaurants	S
Cluster 0	Outsider	Outsider	Normal	Normal	Normal	Outsider	Normal	Outsider	Outsider	Outsider	Normal	Normal	Outsider	

Finally, we show the most exciting part of the study – the interactive map.

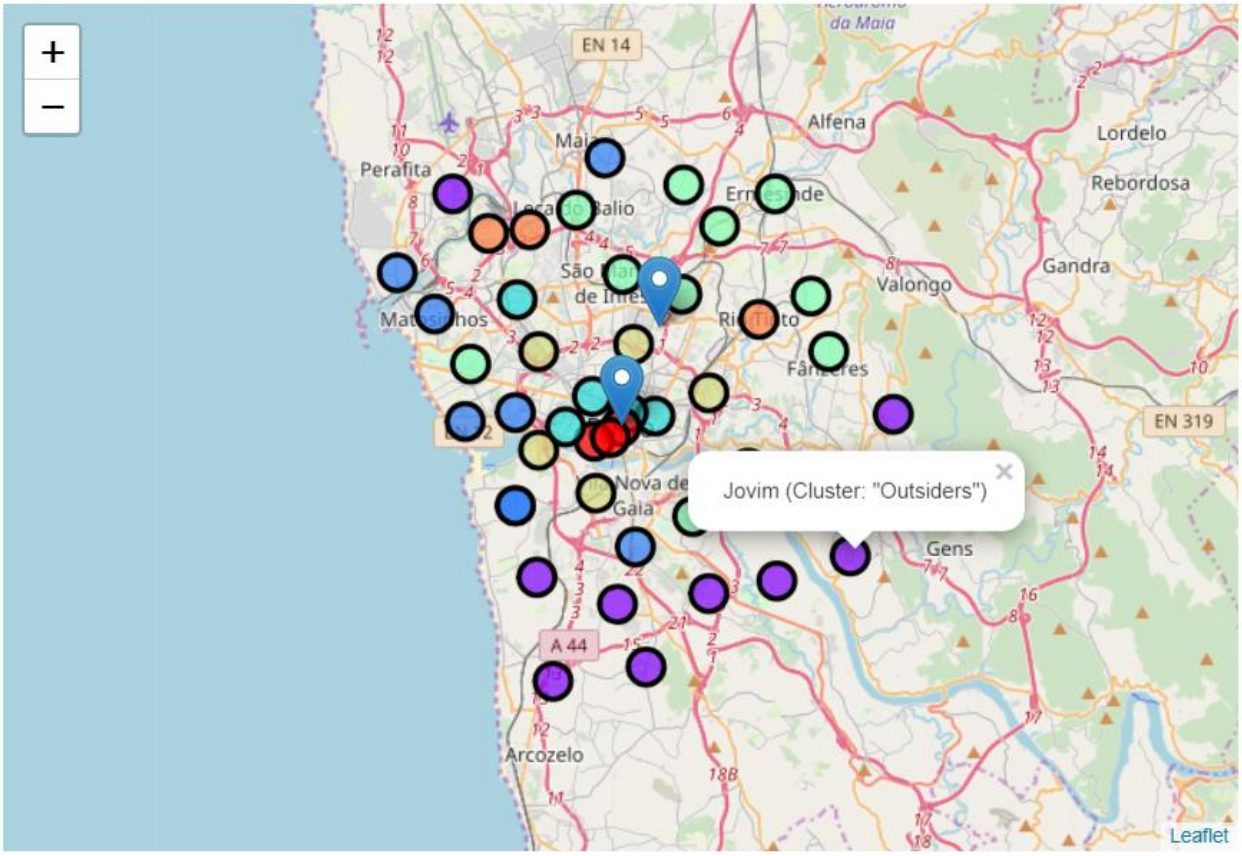

Figure 13. Interactive map of Porto City with clusters on it. Two geo-markers show locations of the City Centre and the University (FEUP)

5. DISCUSSION

In this section I'm going to list obtained clusters and present some observations. All observations are made from tables described above, and from the map.

RED: Central Core

This cluster is the **Central Core**. Its main features are all in the top. There is a **lack in** *Car Service and Fitness & Sports venues.* We can expect the highest prices here.

Normal-level traits: *Food Stores & Markets, Basic Shops & Services, To-Sea distance*
High-level traits: *Alcohol Venues, Food Places, Nature (due to city gardens), Public Transport.*

Main features in the top-list: *Good proximity to University, the very City Center, Café and Desserts, Shopping venues, Restaurants*

ORANGE: "Rail stations only"

This cluster is normal almost in all parameters. Its **main peculiarity** is that its Parishes are good in Public Transport. However, this is mostly because of their location not far from railway stations.
Also, they are not very far from the University and well-connected with it by rail lines.
Drawbacks: The cluster is outsider in food places such as *fast food*, *restaurants* and *cafés*
High-level traits: *Car Service, Public Transport*.
Main top-level features: *Public Transport*

YELLOW: "Optimum | Daily Life"

This cluster looks like one of the best. It is spread along the perimeter of City Centre. It isn't far from the University and it is in the top-list in the most venues.

On average:

Normal-level traits: *Distance to the sea, Car Service, Café and Desserts*.
High-level traits: *all the others.*
The closest Parish to the University is Paranhos, the most remote is Santa Marinha.

Main top-level features: *Good proximities to the University and to the City Centre, Food Stores & Markets, Fitness & Sports venues, Public Transport infrastructure*

GREEN: "Mediocrity"

This cluster is the Mediocrity. And outsider in *Distance to the sea* (except Aldoar), *Fitness & Sports venues* and in *Food Stores & Markets.*

Distance to the University is different and is normal on average. But there are both Parishes very close to FEUP and distant.

Main top-level features: *No top-level features*

LIGHT BLUE: "Optimum | City Centre"

This cluster might be one of the best. It isn't far from the University and it is in the top-list in many venues.

On average:

Normal-level traits: *Distance to the sea, Car Service, Nature, Public Transport.*
High-level traits: *Alcohol venues, Restaurants, Food Places, Fitness & Sports, Basic shops & Services*

Main top-level features: Good (but not the best) *proximity to the University, Shopping venues, City Centre, Food Stores & Markets, Café and Desserts*

BLUE: "Good but far"

This cluster has a lot of advantages but is quite **remote from the University**. Its Parishes located mostly along the sea.

Normal-level traits: *Remoteness, Alcohol venues, Basic shops & Services, Shopping, Transport, Food places.*
High-level traits: *Restaurants, Nature, Fitness & Sports, Car Service*

Main top-level features: *Food Stores & Markets, Café and Desserts, Distance to the sea.*

VIOLET: "Outsiders"

This cluster is the **worst** in most features, though on average it has several **normal-level traits**: *Distance to the sea, Nature,* and some *Basic shops & Services.*

No top features in cluster.

6. CONCLUSION

The cluster analysis, presented in this study, may be useful for anyone, who is going to relocate in Porto or just to explore its Parishes in detail.

Of course, there is much space for improvements. We can use more dense geo-points, change geo-data provider, gather more info about local venues, implement more interesting visualizations.

This study has been made in educational purpose only and has its limits. However, I tried to work out and apply a number of new ideas for building such a system, and I believe it can be scaled and/or incorporated into useful web-based application.

REFERENCES:

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

[2] https://pt.wikipedia.org/wiki/Lista_de_freguesias_de_Portugal

[3] Geocoding service: https://nominatim.openstreetmap.org/

[4] Foursquare API: https://developer.foursquare.com/docs

[5] https://geopy.readthedocs.io/en/stable/#module-geopy.distance

[6] Cross-track distance: https://pypi.org/project/nvector/#description