Executive Summary Part 3

March 20, 2020

1 Exploratory component

1.1 Introduction

We are now considering a more in-depth analysis about the data. With all the accessibility-related information that we have, we want to cluster the different neighbourhoods by accessibility and then try to identify the type of neighbourhood that it corresponds to. So the question is: **How should we cluster our data in order to identify the different localities in the city?**

In addition, we will also try to get some more data that will give us more information on the neighborhoods. This data will be collected via either the *Yelp* or the *Google Maps* API. By doing so we will be able to get locations of stores, businesses, restaurants,... and their corresponding reviews given by customers. This will help us compute new features to add to our original dataset and will hopefully help us increasing our classifier accuracy and give us more insights to answer the above question.

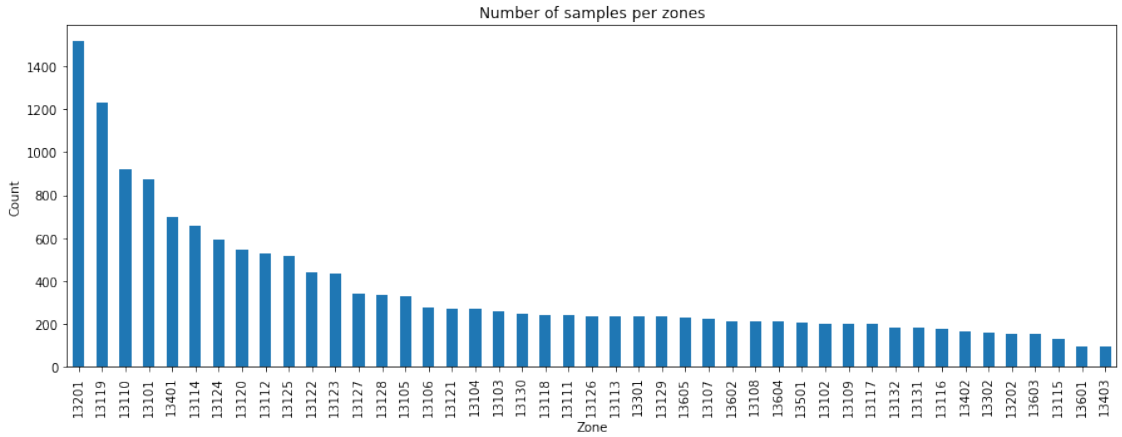
Note: this notebook is run on googlecolab so some of the code reflect this feature by calling an API to collect data.

1.2 Preparation

This part is almost identical to the preparation part in the previous notebooks. Therefore, there is no further explanation of the steps made provided. Some graphs are plotted to make sure the codes and data are correct. Please check that you have the python libraries and the change the file locations if needed.

Let's also see how many entries there are per zone.

[0]:
```
df_formap['zone'].value_counts().plot(kind='bar', figsize=(15,5));
plt.xlabel("Zone");
plt.ylabel("Count");
plt.title('Number of samples per zones');
```

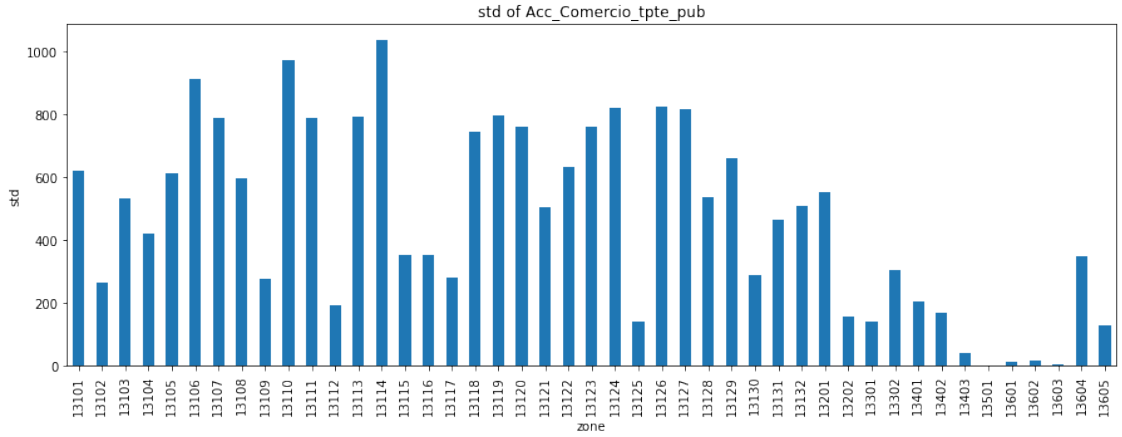


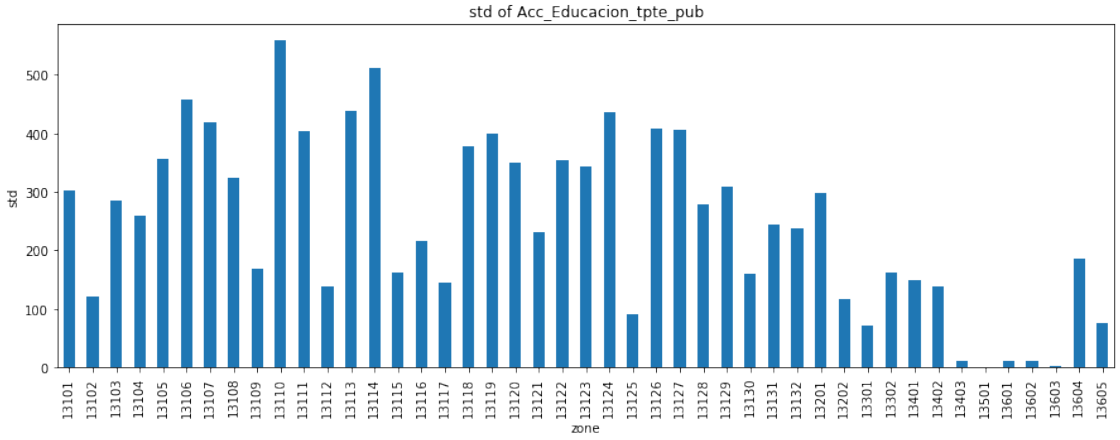
1.3 Plotting different attributes for the zones

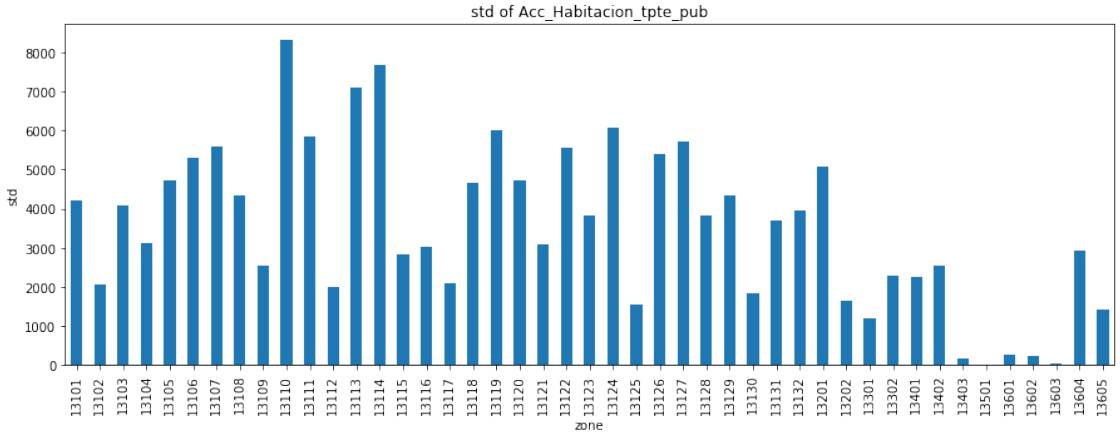
Now lets analyze some parameters of our households dataset on the 'Acc_'. We are especially gonna focus on plotting the different standard deviation to justify that we used a mean over all the samples of a zone in the later.

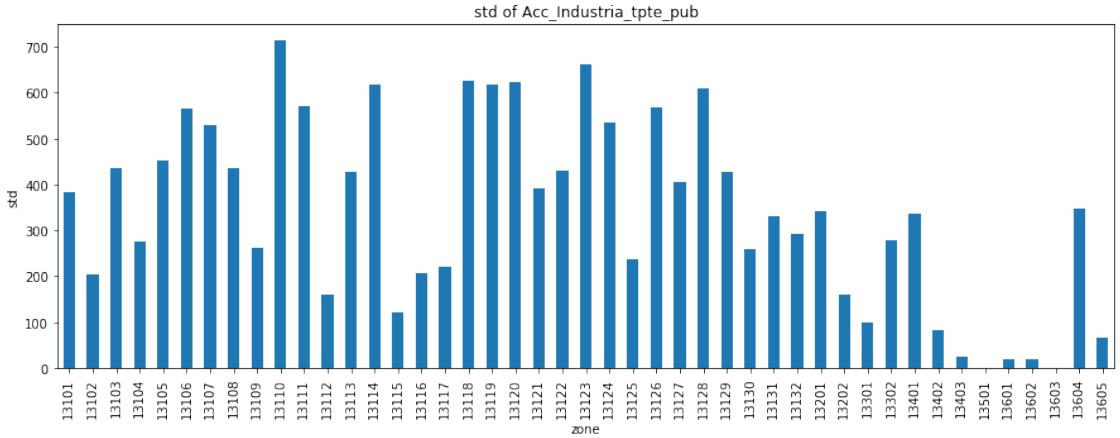
[0]:
```
accs = ['Acc_Comercio_tpte_pub', 'Acc_Educacion_tpte_pub',␣
 ↪'Acc_Habitacion_tpte_pub',
       'Acc_Industria_tpte_pub', 'Acc_Servicios_tpte_pub', 'Acc_Comercio_auto',
       'Acc_Educacion_auto', 'Acc_Habitacion_auto', 'Acc_Industria_auto',
       'Acc_Servicios_auto']
```

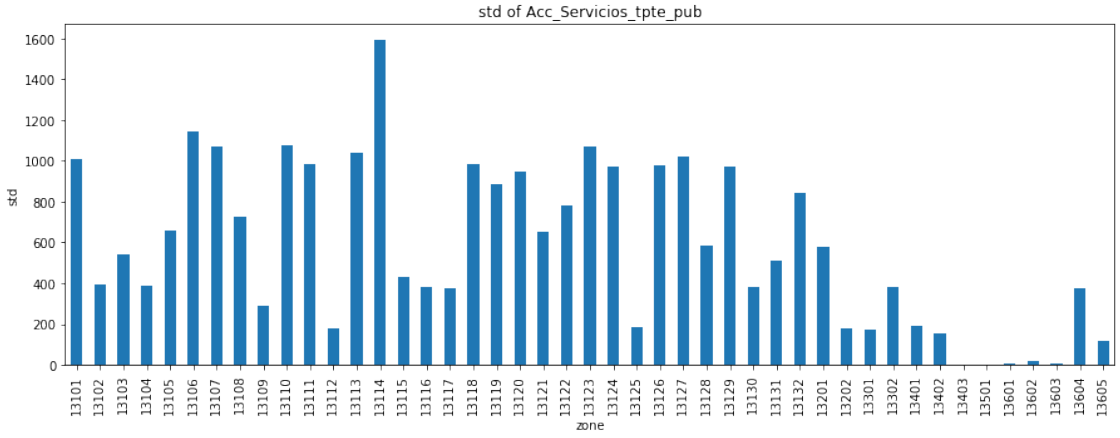
[0]:
```
for acc in accs:
  df_formap.groupby(by='zone')[acc].std().plot(kind='bar', figsize=(15,5));
  plt.xlabel("zone");
  plt.ylabel("std");
  plt.title('std of {}'.format(acc));
  plt.show()
```

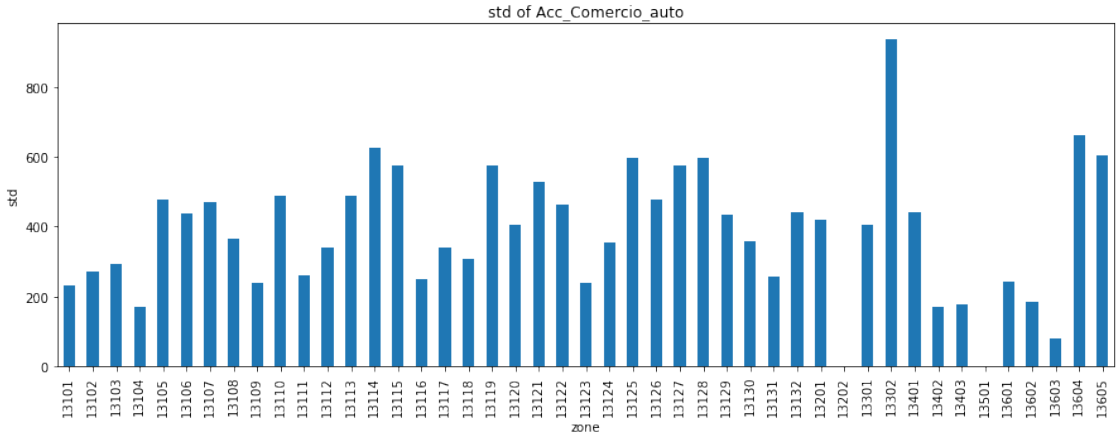


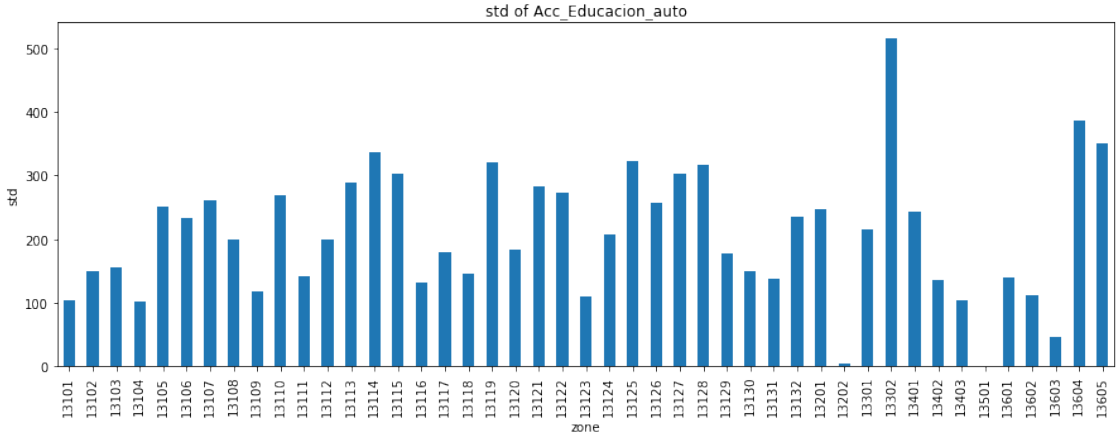

std of Acc_Educacion_tpte_pub
std
zone
0
100
200
300
400
500
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605

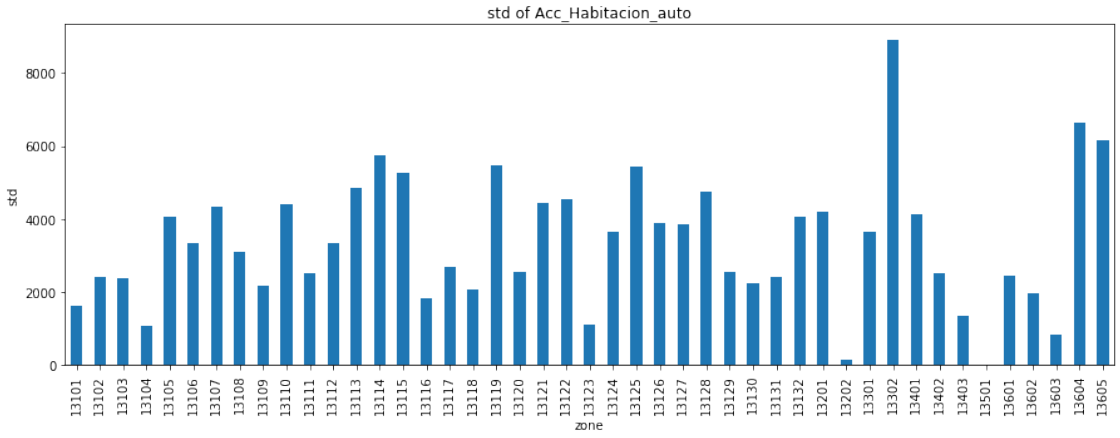

std of Acc_Habitacion_tpte_pub
std
zone
0
1000
2000
3000
4000
5000
6000
7000
8000
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605

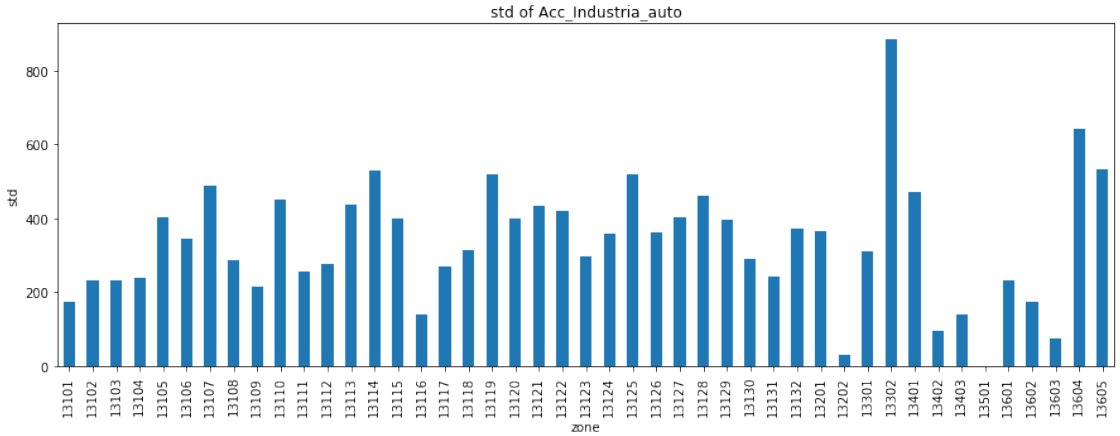

std of Acc_Industria_tpte_pub
std
zone
0
100
200
300
400
500
600
700
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605

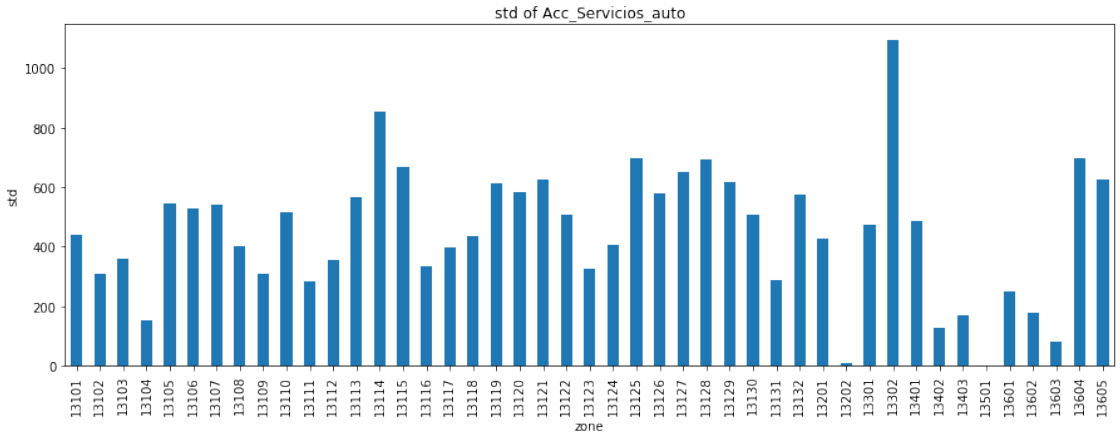

std of Acc_Servicios_tpte_pub
std
zone
0
200
400
600
800
1000
1200
1400
1600
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605


std of Acc_Comercio_auto
std
zone
0
200
400
600
800
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605


std of Acc_Educacion_auto
std
zone
0
100
200
300
400
500
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605


std of Acc_Habitacion_auto
std
zone
0
2000
4000
6000
8000
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605


std of Acc_Industria_auto
std
zone
0
200
400
600
800
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605


std of Acc_Servicios_auto
std
zone
0
200
400
600
800
1000
13101 13102 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 13113 13114 13115 13116 13117 13118 13119 13120 13121 13122 13123 13124 13125 13126 13127 13128 13129 13130 13131 13132 13201 13202 13301 13302 13401 13402 13403 13501 13601 13602 13603 13604 13605

We can see that the standard deviations are quite high for some of the zones. That is why we will then compare a clustering by zones and by samples.

1.4 Clustering the different zones depending on the accessibility

From now on, we want to cluster our data to get some insights about accessibilities and the different zones of the city. First, we are gonna group by zones and averaging to cluster by zones.

[0]:
```
# definition of the min pop in a zone
min_pop = 0

# get a column with the count of pop in zones
zone_counts = df_formap['zone'].value_counts().to_dict()

df_formap['count_pop_zone'] = df_formap['zone'].apply(lambda x: zone_counts[x])
df_zones = df_formap[df_formap['count_pop_zone'] > min_pop]

print('we selected {} zones to cluster out of {}. We only selected the zones␣
 ↪with more than {} datapoints.'.format(
                      df_zones.groupby(by='zone').count().shape[0],
                      len(zone_counts),
                      min_pop,
      ))
```

```
we selected 45 zones to cluster out of 45. We only selected the zones with more
than 0 datapoints.
```

Since we only have 45 zones, we decided to keep them all even if there is a ratio of one tenth in the number of data points per zone. (you can see that in the plot titled: 'Number of samples per zones' above)

[0]:
```
# selection of variables that are going to be useful later
to_keep = ['Acc_Comercio_tpte_pub', 'Acc_Educacion_tpte_pub',␣
 ↪'Acc_Habitacion_tpte_pub',
       'Acc_Industria_tpte_pub', 'Acc_Servicios_tpte_pub', 'Acc_Comercio_auto',
       'Acc_Educacion_auto', 'Acc_Habitacion_auto', 'Acc_Industria_auto',
       'Acc_Servicios_auto', 'zone', 'count_pop_zone']
df_zones = df_zones[to_keep]
```

[0]:
```
# let's try to group by zones in order to get a clustering of the zones
df_zones_mean = df_zones.groupby(by='zone').mean()
print(df_zones_mean.shape)
df_zones_mean.head()
```

```
(45, 11)
```

[0]:
```
        Acc_Comercio_tpte_pub  ...  count_pop_zone
zone                           ...
13101             5158.309126  ...             873
13102             1973.677194  ...             201
13103             1921.998466  ...             259
13104             2179.575178  ...             267
13105             1675.201461  ...             330

[5 rows x 11 columns]
```

Here, we are ready to cluster.

[0]:
```
# Clustering using KMeans
nb_clusters = 6
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=nb_clusters, random_state=0)
kmeans.fit(df_zones_mean.drop(columns=['count_pop_zone']))
df_zones_mean['clusters'] = kmeans.predict(df_zones_mean.
 ↪drop(columns=['count_pop_zone']))
```

[0]:
```
# group cluster will contain the mean of each variable per cluster
group_cluster = df_zones_mean.drop(columns=['count_pop_zone']).
 ↪groupby(by='clusters').mean()
group_cluster
```

[0]:
```
           Acc_Comercio_tpte_pub  ...  Acc_Servicios_auto
clusters                          ...
0                    1687.090633  ...         5988.197999
1                     126.934515  ...         1485.657900
2                    3611.262014  ...         7568.869284
3                     865.069201  ...         4590.143694
4                    2270.298707  ...         7455.506517
5                     498.306962  ...         3293.442515

[6 rows x 10 columns]
```

[0]:
```
# plotting the number of zones in each clusters
print("Clusters repartition:");
df_zones_mean['clusters'].value_counts(sort=False)
```

```
Clusters repartition:
```

[0]:
```
0     9
1     4
2    15
3     4
4     9
5     4
Name: clusters, dtype: int64
```

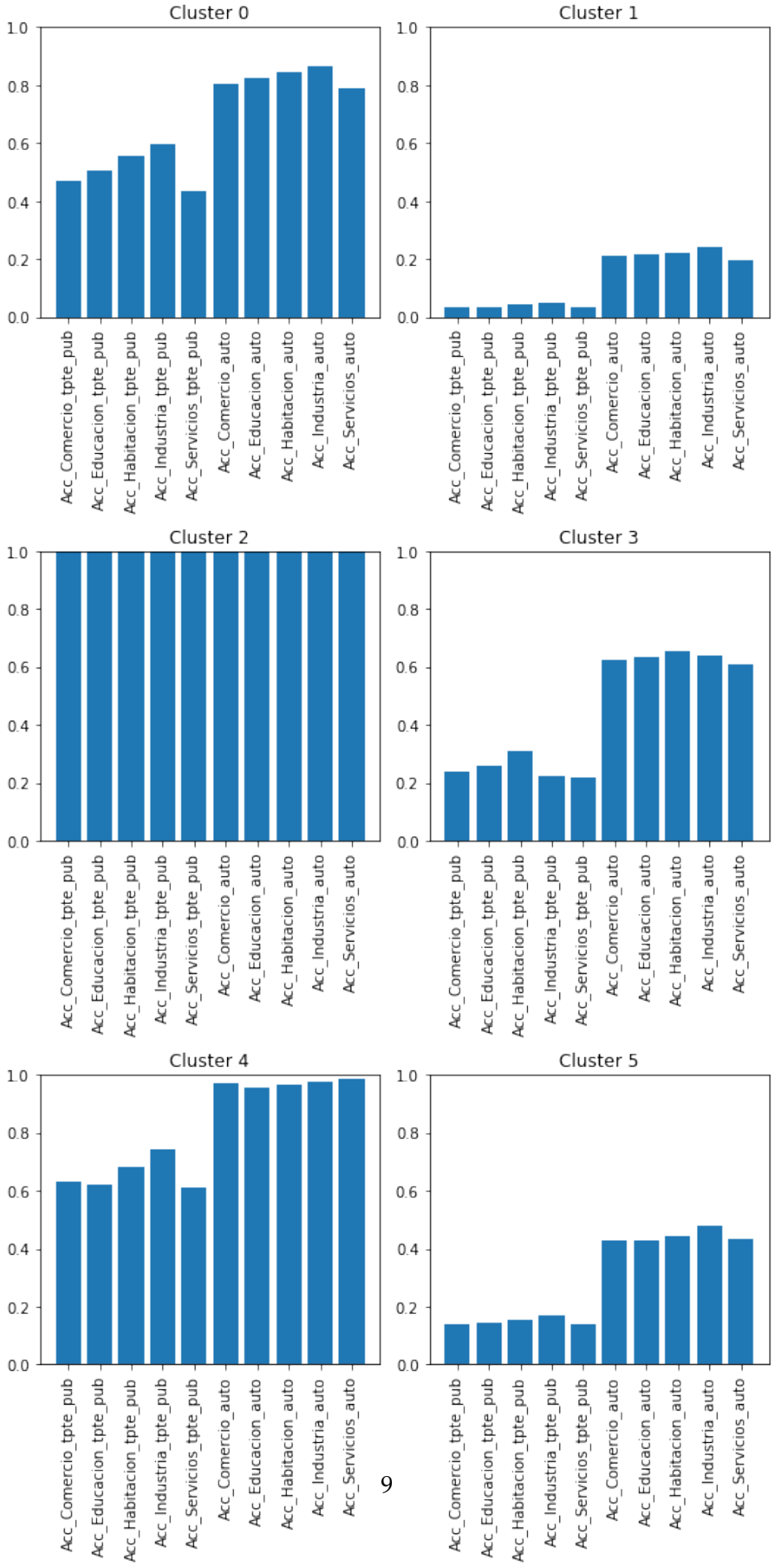
We tried different number of clusters until having at least 4 zones in each cluster.

[0]:
```
# function for plotting
def next_step(previous, limx, limy):
  """given previous tuple, return the next tuple on a defined grid"""
  return_x, return_y = previous
  if previous[1]+1 < limy:
    return_y += 1
    return return_x, return_y
  else:
    return return_x+1, 0
```

[0]:
```
# study of the different clusters
# we will divide by the max value over all clusters for a better visualization
columns = group_cluster.columns
for col in columns:
  max_col = max(group_cluster[col])
  group_cluster[col] = group_cluster[col]/max_col


dict_group_cluster = group_cluster.to_dict()
np_group_cluster = group_cluster.to_numpy()


fig, ax_tot = plt.subplots(3, 2, figsize=(7.5, 15))
grid_loc = (0, 0)
for i in range(nb_clusters):
  ax = ax_tot[grid_loc]
  grid_loc = next_step(grid_loc, 3, 2)
  data = np_group_cluster[i]
  ax.bar(columns, data)
  plt.setp(ax.get_xticklabels(), rotation=90)
  ax.set_title('Cluster {}'.format(i))
  ax.set_ylim(bottom=0, top=1)
plt.tight_layout()
```


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
1.0
0.8
0.6
0.4
0.2
0.0
Acc_Comercio_tpte_pub
Acc_Educacion_tpte_pub
Acc_Habitacion_tpte_pub
Acc_Industria_tpte_pub
Acc_Servicios_tpte_pub
Acc_Comercio_auto
Acc_Educacion_auto
Acc_Habitacion_auto
Acc_Industria_auto
Acc_Servicios_auto

What we see above is that the model is distinguishing the different zones by a global level of accessibility. That is to say, it is not only one type of accessibility that changes between clusters (eg only education) but the overall level. We had the hint of the phenomenon on the first part of the study when we saw that those variables are correlated. We can also see great differences between the zones. For instance, we could imagine that in a cluster where the accessibility is low, people are using the car for everything.

Let see how it actually looks on a map.

[0]:
```
# plot of the clustering on a map:
dict_clus = df_zones_mean['clusters'].to_dict()
new = map.copy()
new['cluster'] = new['Com'].apply(lambda x: dict_clus[x] if x in dict_clus else␣
 ↪None)
new = new.dropna()
```

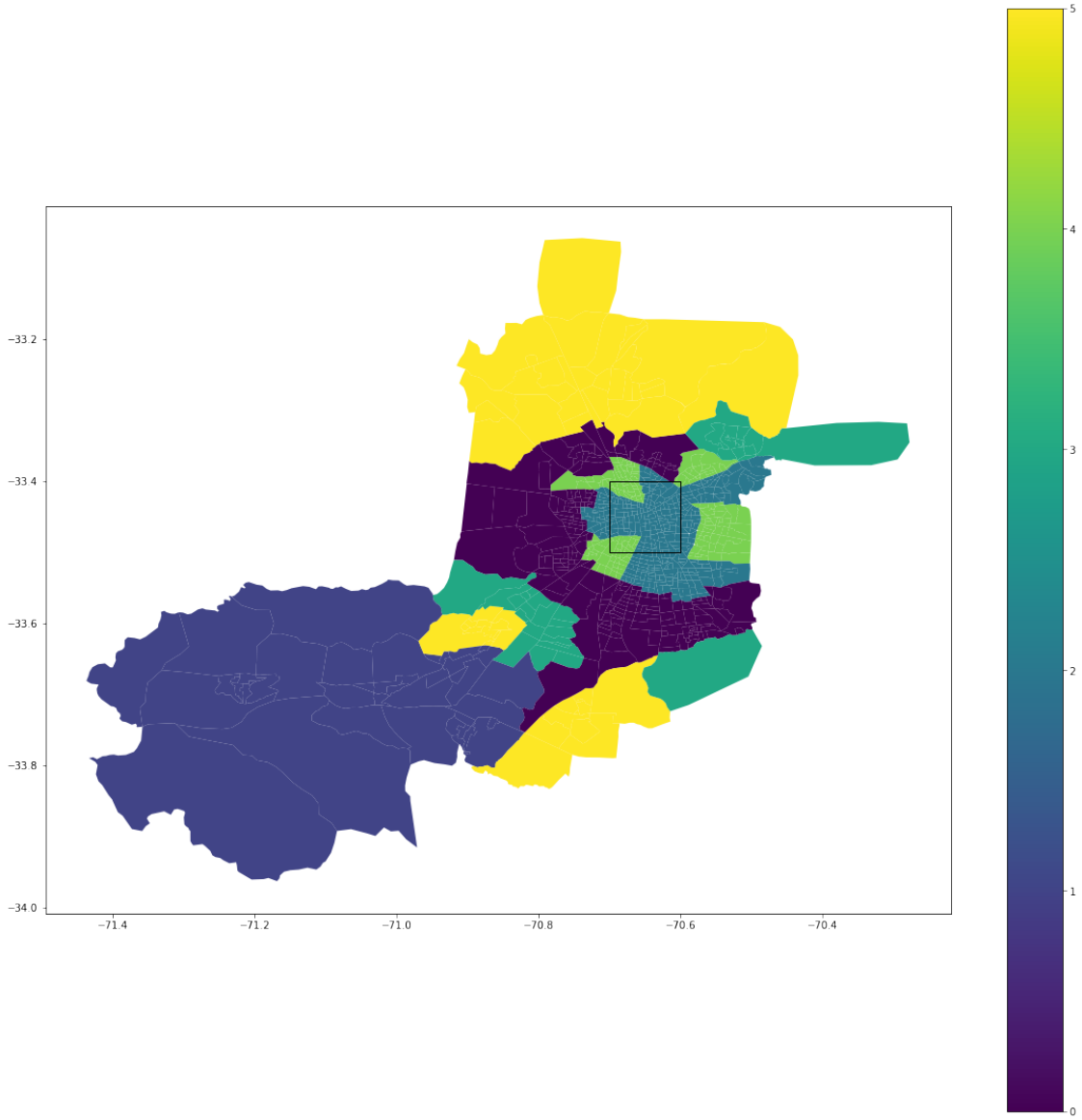
[0]:
```
new = new.to_crs(epsg=4326) #different type of coding (just the coordinates␣
 ↪change)


fig, ax = plt.subplots(1, 1, figsize=(20, 20))
new.plot(column='cluster', ax=ax, legend=True)

# just to see where is located the test dataset from the classification problem
from matplotlib import patches
ax.add_patch(
    patches.Rectangle((-70.7,-33.5), .1, .1, angle=0.0, fill=False))
```

```
/usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning:
'+init=<authority>:<code>' syntax is deprecated. '<authority>:<code>' is the
preferred initialization method.
  return _prepare_from_string(" ".join(pjargs))
```

[0]:
```
<matplotlib.patches.Rectangle at 0x7f4d993af0b8>
```



The insigths drawn from the plot seems to follow perfectly the geography of the city. The farther form the city center, the lower the accessibility. This relatively short analysis confirm that the way of living, the relevance of the data and the pattern we could find depends on the location. That is a factor that could explain the lack of accuracy of the classification. It is difficult to predict the test set (the square on the plot) using a train set with so different characteristics.

1.5 Clustering on the same variable at the house level

We are now gonna cluster on all the samples, without doing a mean on the zones. Since we have quite high standard deviations, this might give us more insigths on the different parts of the city.

[0]:

```
# formatting for the clustering
to_keep = ['Acc_Comercio_tpte_pub', 'Acc_Educacion_tpte_pub',␣
 ↪'Acc_Habitacion_tpte_pub',
           'Acc_Industria_tpte_pub', 'Acc_Servicios_tpte_pub',␣
 ↪'Acc_Comercio_auto',
           'Acc_Educacion_auto', 'Acc_Habitacion_auto', 'Acc_Industria_auto',
           'Acc_Servicios_auto', 'zone', 'DirCoordX', 'DirCoordY']
to_drop_clustering = ['zone', 'DirCoordX', 'DirCoordY']

df_house_acc = df_map[to_keep]
df_house_acc.head()
```

[0]:
```
   Acc_Comercio_tpte_pub  Acc_Educacion_tpte_pub  ...  DirCoordX  DirCoordY
0              704.97642                406.0983  ... -70.779034 -33.729444
1              704.97642                406.0983  ... -70.744339 -33.737278
2              704.97642                406.0983  ... -70.859655 -33.805994
3              704.97642                406.0983  ... -70.859653 -33.805995
4              704.97642                406.0983  ... -70.743269 -33.722055

[5 rows x 13 columns]
```

[0]:
```
# Clustering using KMeans
nb_clusters = 6
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=nb_clusters, random_state=0)
kmeans.fit(df_house_acc.drop(columns=to_drop_clustering))
df_house_acc['clusters'] = kmeans.predict(df_house_acc.
 ↪drop(columns=to_drop_clustering))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

[0]:
```
# plotting the number of zones in each clusters
df_house_acc['clusters'].value_counts(sort=False)
```

[0]:
```
0    3326
1    2859
2     778
3    4065
4    4304
5     740
Name: clusters, dtype: int64
```

[0]:
```
# group cluster will contain the mean of each variable per cluster
group_cluster_house = df_house_acc.drop(columns=to_drop_clustering).
 ↪groupby(by='clusters').mean()
group_cluster_house
```

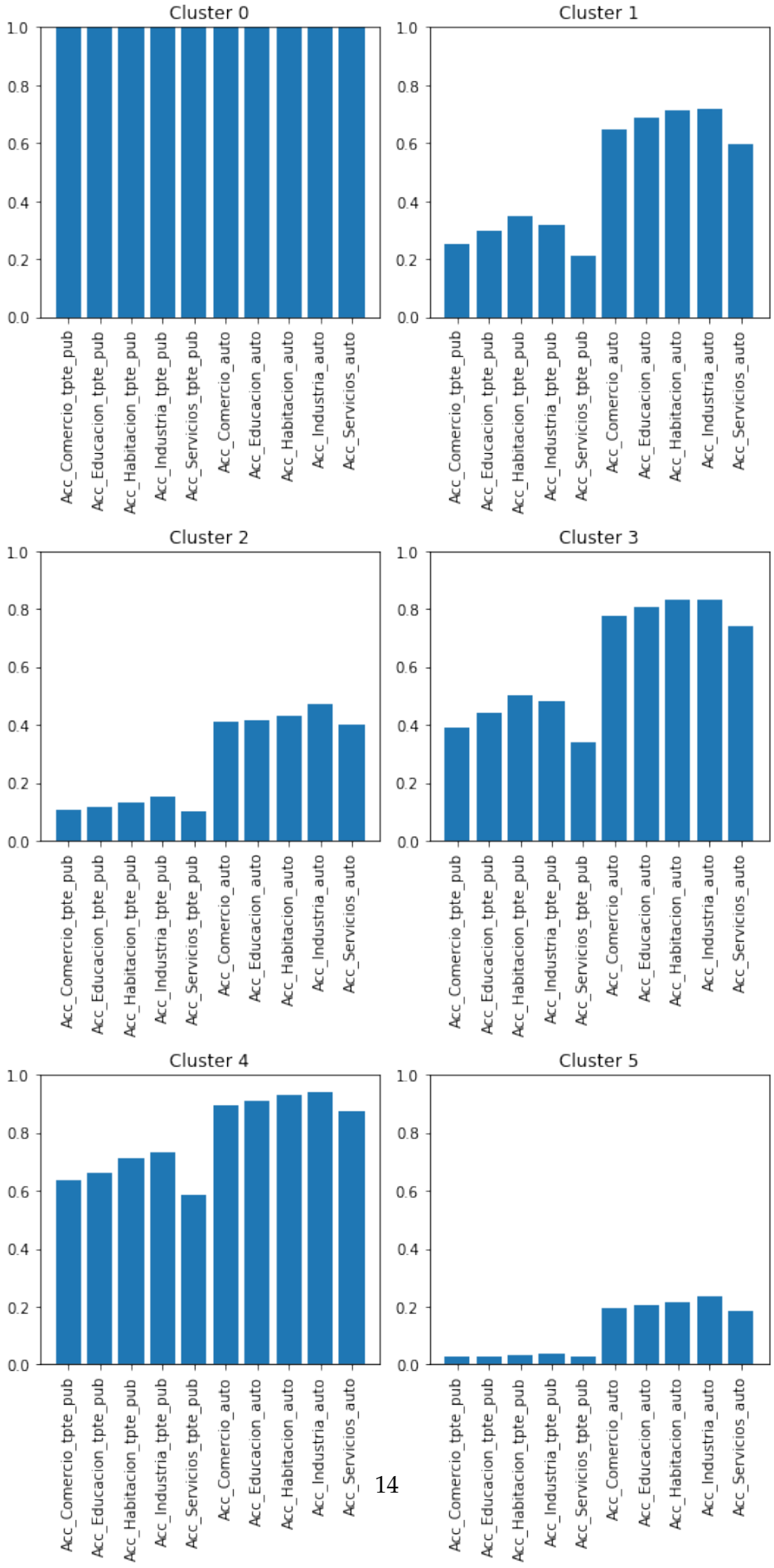
[0]:
```
          Acc_Comercio_tpte_pub  ...  Acc_Servicios_auto
clusters                         ...
0                   4401.526607  ...         8247.959245
1                   1099.243975  ...         4933.121098
2                    476.160723  ...         3289.730527
3                   1718.960604  ...         6096.009909
4                   2808.545459  ...         7209.690446
5                    124.340414  ...         1529.746932

[6 rows x 10 columns]
```

[0]:
```
# study of the different clusters
# we will divide by the max value over all clusters for a better visualization
columns = group_cluster_house.columns
for col in columns:
  max_col = max(group_cluster_house[col])
  group_cluster_house[col] = group_cluster_house[col]/max_col


dict_group_cluster = group_cluster_house.to_dict()
np_group_cluster = group_cluster_house.to_numpy()


fig, ax_tot = plt.subplots(3, 2, figsize=(7.5, 15))
grid_loc = (0,0)
for i in range(nb_clusters):
  ax = ax_tot[grid_loc]
  grid_loc = next_step(grid_loc, 3, 2)
  data = np_group_cluster[i]
  ax.bar(columns, data)
  plt.setp(ax.get_xticklabels(), rotation=90)
  ax.set_title('Cluster {}'.format(i))
  ax.set_ylim(bottom=0, top=1)
plt.tight_layout()
```


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
1.0
0.8
0.6
0.4
0.2
0.0
Acc_Comercio_tpte_pub
Acc_Educacion_tpte_pub
Acc_Habitacion_tpte_pub
Acc_Industria_tpte_pub
Acc_Servicios_tpte_pub
Acc_Comercio_auto
Acc_Educacion_auto
Acc_Habitacion_auto
Acc_Industria_auto
Acc_Servicios_auto

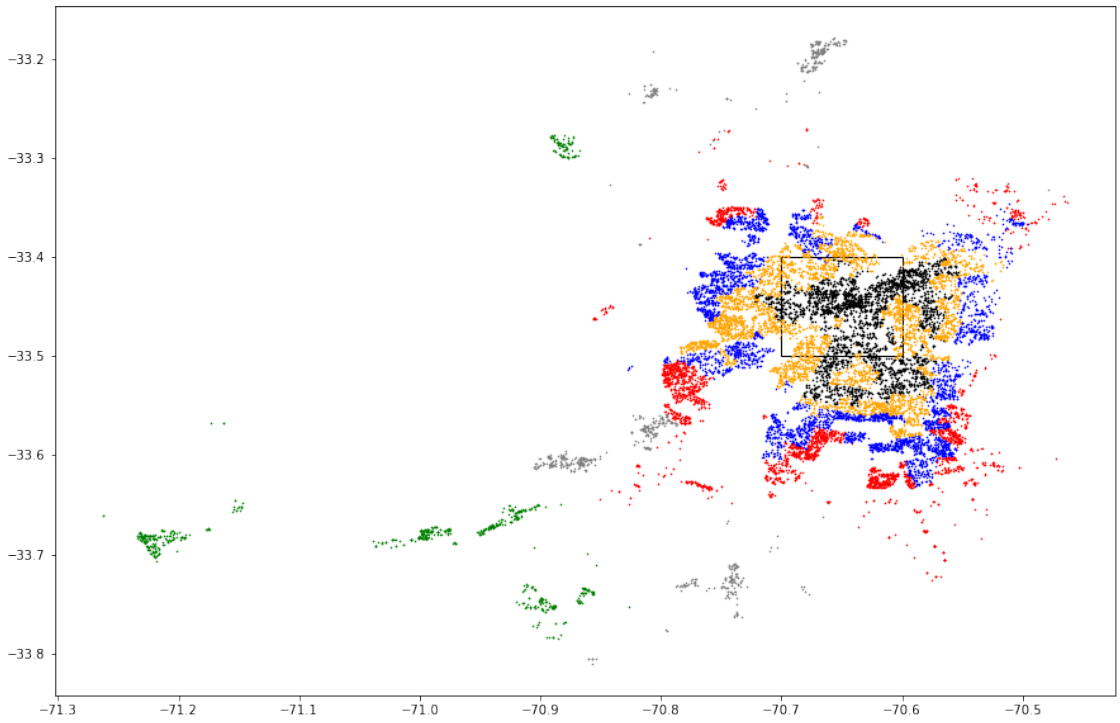
We can see pretty much the same patterns than before with the zones. That is to say that, the accessibilities are correlated and the clusters define a global level of accessibility.

[0]:
```
fig =plt.figure(figsize=(15,10))
plt.plot(df_house_acc.loc[df_house_acc['clusters']==0]['DirCoordX'],␣
 ↪df_house_acc.loc[df_house_acc['clusters']==0]['DirCoordY'], '*',␣
 ↪color='black', markersize=1);
plt.plot(df_house_acc.loc[df_house_acc['clusters']==1]['DirCoordX'],␣
 ↪df_house_acc.loc[df_house_acc['clusters']==1]['DirCoordY'], '*', color='r',␣
 ↪markersize=1);
plt.plot(df_house_acc.loc[df_house_acc['clusters']==2]['DirCoordX'],␣
 ↪df_house_acc.loc[df_house_acc['clusters']==2]['DirCoordY'], '*',␣
 ↪color='gray', markersize=1);
plt.plot(df_house_acc.loc[df_house_acc['clusters']==3]['DirCoordX'],␣
 ↪df_house_acc.loc[df_house_acc['clusters']==3]['DirCoordY'], '*',␣
 ↪color='blue', markersize=1);
plt.plot(df_house_acc.loc[df_house_acc['clusters']==4]['DirCoordX'],␣
 ↪df_house_acc.loc[df_house_acc['clusters']==4]['DirCoordY'], '*',␣
 ↪color='orange', markersize=1);
plt.plot(df_house_acc.loc[df_house_acc['clusters']==5]['DirCoordX'],␣
 ↪df_house_acc.loc[df_house_acc['clusters']==5]['DirCoordY'], '*',␣
 ↪color='green', markersize=1);

from matplotlib import patches
ax = fig.add_subplot(1, 1, 1)
ax.add_patch(
    patches.Rectangle((-70.7,-33.5), .1, .1, angle=0.0, fill=False))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  # Remove the CWD from sys.path while we load stuff.
```

[0]:
```
<matplotlib.patches.Rectangle at 0x7f4d912883c8>
```



We can see a more precise clustering but the idea seems to be still the same. That is to say, the center concentrates the activities and elsewhere it is difficult to access them. It also corresponds to different density of population. However, we can see more differences between neighbourhoods if we focus on the 'black' and 'orange' clusters. The 'black' cluster is the hyper center and corresponds to the maximum accessibility. The 'orange' zones might correspond to residential areas still close to the center.

1.6 Conclusion

To conclude this explanatory part, the city is organised like concentric circles. The city center concentrates the activities and the distances are really short. The farther from the center, the greater the distances are to access anything. It seemed like taking zones or all the samples did not really changed this observation. Also, we can see that all the accessibility variables are really correlated since they depend mostly on the location. Nevertheless, a more in-depth study could investigate the differences between the 'black' and the 'orange' cluster to identify different type of neighbourhoods around the city center.

When considering the classification challenge and the way the training and test sets are divided, we can wonder if training on the outskirt would give us good prediction on the center. Indeed, it seems that life styles are quite different depending on where people live. Hence, we might have patterns that are not in the training set but in the test set.

2 Conclusion to the Challenge

The first part was dedicated to a preliminary analysis. We could apprehend the dataset and the different variables. It was insightful especially when it came to select our variables. It is also at

this moment that we got a hint of the uneven geographic location of the train and test set.

In the second part, we used the learnings from to first part to build a classifier. Our classifier ended up performing at 57% accuracy (compared to approx 50% for any model at the beginning). To achieve that we used a PCA on some variables that were correlated. We also added some data from Yelp API to increase the amount of information.

The last part constitutes a small study of the accessibility in the city. We tried to cluster by zones and then samples depending on the values of the variables concerning accessibility. The result is that we can clearly see how the activities are all concentrated in the city center. The farther from the city center, the scarcer it becomes.

This challenge was not a straight forward 100% accuracy project. However, it was interesting to try to make the best of it. Although, in real life, this type of project would benefit so much from domain knowledge. It would be really useful to work with someone from the city that can give some insights and common representations of Santiago de Chile.