**Big Data Project**

# Data Analysis about

# “Individual household electric power consumption”

**Problem Description**

The objective of this project is to identify patterns of consumption behavior based on historical data on electrical energy consumption by applying the machine learning method in order to support making decision in the formulation of public energy policies.

To solve this case, we will be using the most widely used unsupervised machine learning method called k-mean.

K-mean is a very intuitive solution and aims to group the dataset and apply a clustering technique, returning the similarities identified with the patterns within the attributes of the dataset, being extremely useful, because it allows the automatic search for patterns that are not perceived The value of k . We can define the k value considering the business rule and the personal evaluation based on dataset knowledge. The key point to the success of any predictive modeling project is in the pre-processive and treatment phase of dataset.

**Data Set Information**

This archive contains 2075259 measurements gathered in a house located in Sceaux (7km of Paris, France) between December 2006 and November 2010 (47 months).  
Notes:  
1.(global\_active\_power\*1000/60 - sub\_metering\_1 - sub\_metering\_2 - sub\_metering\_3) represents the active energy consumed every minute (in watt hour) in the household by electrical equipment not measured in sub-meterings 1, 2 and 3.  
2.The dataset contains some missing values in the measurements (nearly 1,25% of the rows). All calendar timestamps are present in the dataset but for some timestamps, the measurement values are missing: a missing value is represented by the absence of value between two consecutive semi-colon attribute separators. For instance, the dataset shows missing values on April 28, 2007.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Set Characteristics:** | Multivariate, Time-Series | **Number of Instances:** | 2075259 | **Area:** | Physical |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 9 | **Date Donated** | 2012-08-30 |
| **Associated Tasks:** | Regression, Clustering | **Missing Values?** | Yes | **Number of Web Hits:** | 320862 |

**Attribute Information**

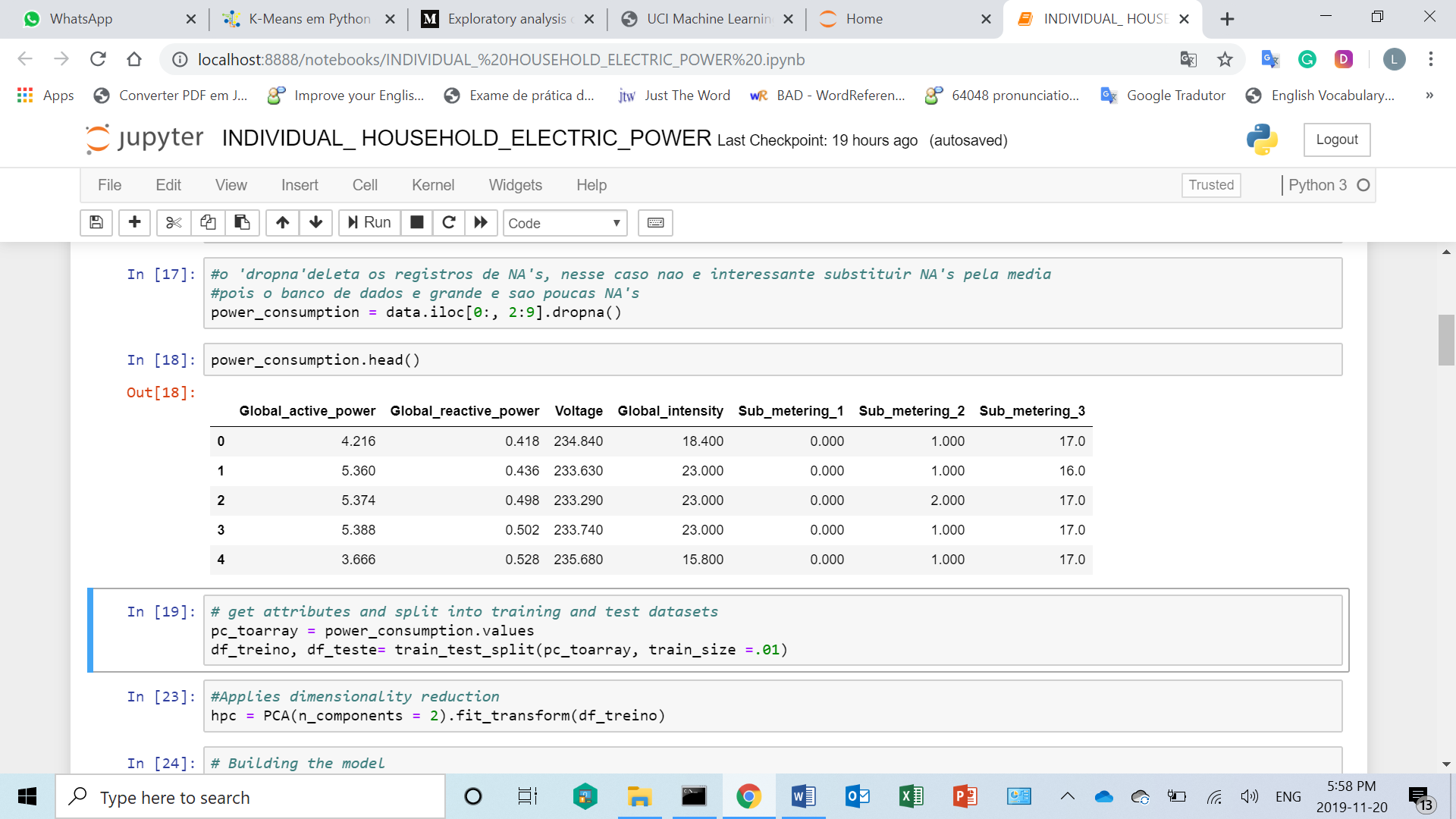
1.date: Date in format dd/mm/yyyy  
2.time: time in format hh:mm:ss  
3.global\_active\_power: household global minute-averaged active power (in kilowatt)  
4.global\_reactive\_power: household global minute-averaged reactive power (in kilowatt)  
5.voltage: minute-averaged voltage (in volt)  
6.global\_intensity: household global minute-averaged current intensity (in ampere)  
7.sub\_metering\_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).  
8.sub\_metering\_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.  
9.sub\_metering\_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

The k-means application pre requisite is that the entire database to be used is in numerical form, for this purpose follows a small sample of this verification.

Otherwise, we must to convert the dataset for a numeric format before start to apply the k-means model.

Python code:

power\_consumption.head()



**Pre-processing Steps**

**About the Missing Values**

The data does have missing values about **1.25%** of the rows (about 82 days). We can see below some dates where we have missing values.

We can notice that the number of missing values is relatively small compared to the size of the database.  
So, it’s not harmful to the model if we delete these missing values.

Python code :

power\_consumption = data.iloc[0:, 2:9].dropna()

**Training and test the datasets Process**

As we work with datasets, a [**machine learning algorithm**](https://data-flair.training/blogs/machine-learning-algorithm/) works in two stages. We usually split the data around 30%-70% between testing and training stages.

Python code :

pc\_toarray = power\_consumption.values

df\_treino, df\_teste= train\_test\_split(pc\_toarray, train\_size =.01)

**Dimensionality reduction**

Applies dimensionality reduction, we are not reducing the numbers of variables

we only collected all variances from the variables and input it in 2 components

and these components represents the same variables information

so now we can work with only these 2 components to make predictions

Python code :

hpc = PCA(n\_components = 2).fit\_transform(df\_train)

**Building the model**

Now we apply the fit in the object calls hpc which was transformed before, with the reduced dimensionality

Python code :

k\_means = KMeans()

k\_means.fit(hpc)

how we did not choose the parameters for k, the k\_means() chose n\_cluster=8 as a default, how we can see bellow

**Organizing the Data Cluster**

Now we are going to organize the data to get minimum and maximum values and arrange shape

to plot it and to build the graphic

Python code :

x\_min, x\_max = hpc[:, 0].min() - 5, hpc[:, 0].max() - 1

y\_min, y\_max = hpc[:, 1].min(), hpc[:, 1].max() + 5

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, .02), np.arange(y\_min, y\_max, .02))

z = k\_means.predict(np.c\_[xx.ravel(), yy.ravel()])

z = z.reshape(xx.shape)

**Area Clusters Plot**

This second process used k\_means with k = 8.

Each color area represent a cluster area

Python code :

plt.figure(1)

plt.clf()

plt.imshow(z,

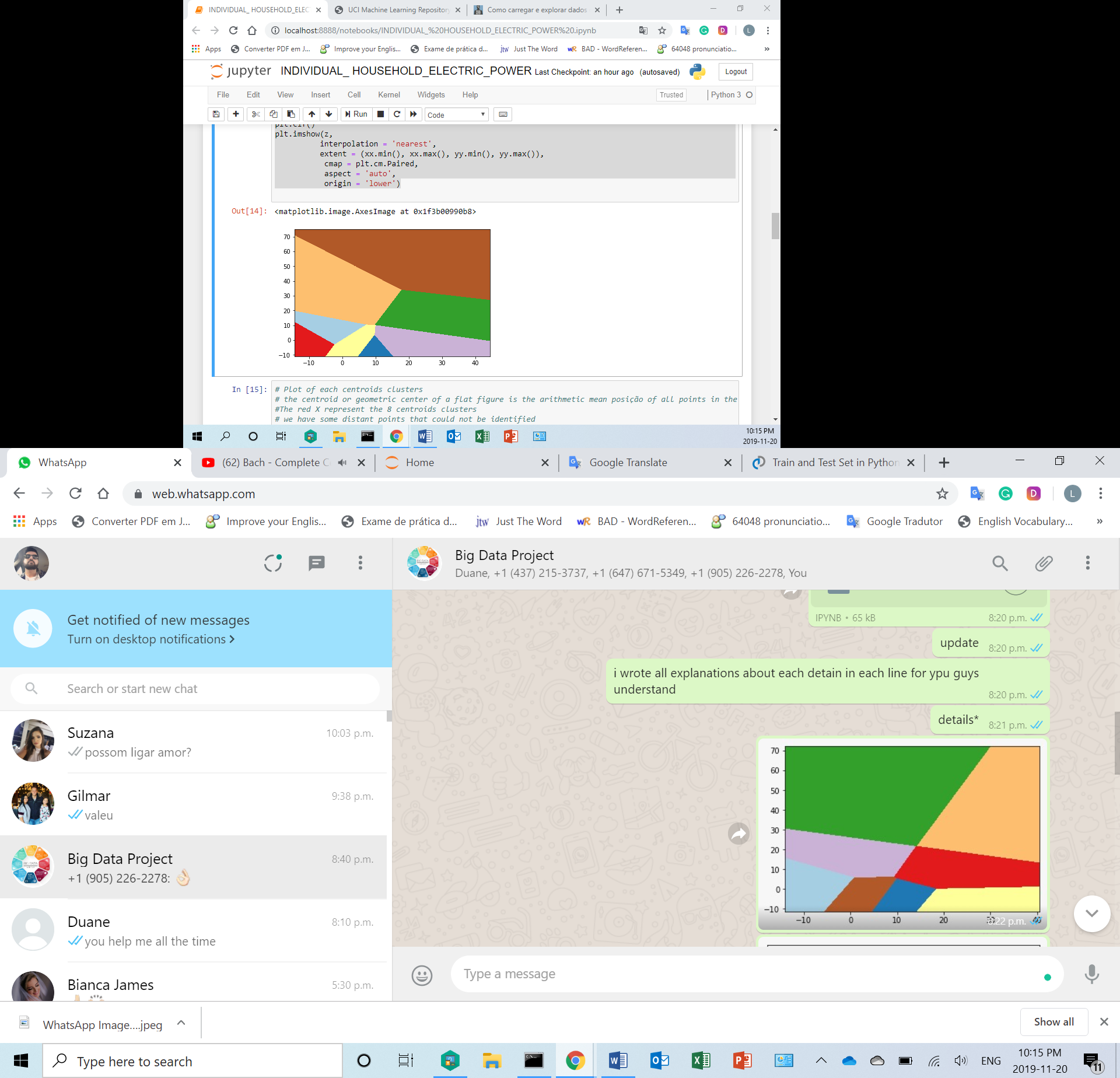
interpolation = 'nearest',

extent = (xx.min(), xx.max(), yy.min(), yy.max()),

cmap = plt.cm.Paired,

aspect = 'auto',

origin = 'lower')



**Plot of each centroid’s clusters**

The centroid or geometric center of a flat figure below is the arithmetic mean position of all points in the figure. The red X represent the 8 centroids clusters and we have some distant points that could not be identified. It is not a clustering mistake but show us that these points do not have similarities with the other groups, so it’s could be out liers or wrong data inserted to dataset.

So, in this case we should to discuss about it with the business area to be sure why these points are different.

Python code :

plt.plot(hpc[:, 0], hpc[:, 1], 'k.' , markersize = 4)

centroids = k\_means.cluster\_centers\_

inert = k\_means.inertia\_

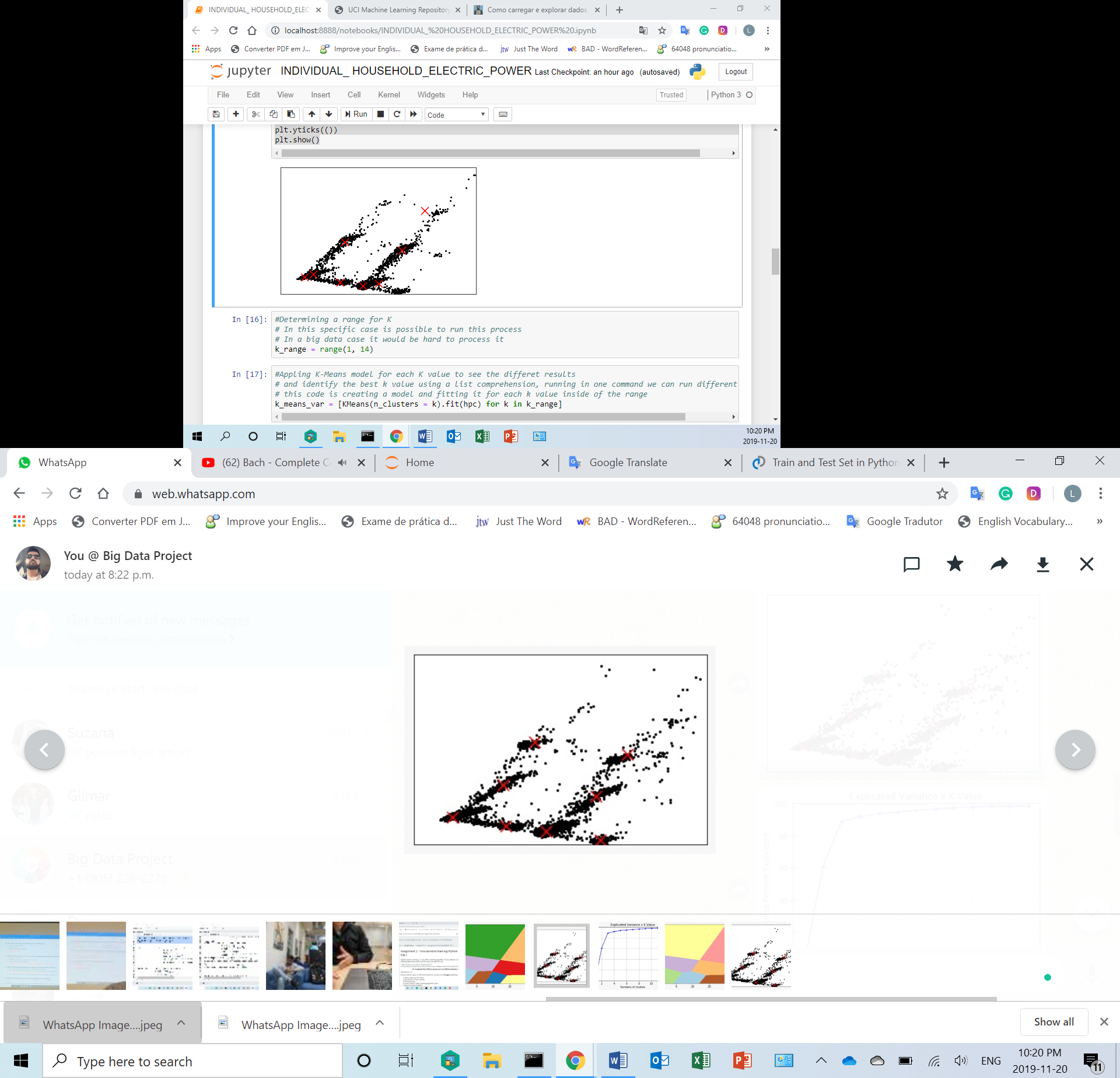
plt.scatter(centroids[:, 0], centroids[:, 1], marker = 'x' , s = 169, linewidths = 3, color ='r', zorder = 8)

plt.xlim(x\_min, x\_max)

plt.ylim(y\_min, y\_max)

plt.xticks(())

plt.yticks(())

plt.show()

**Determining a range for K**

In this specific case is possible to run this process, but in a big data case it would be hard to process it because it will spend to much time to run different k values.

Python code :

k\_range = range(1, 14)

**Appling K-Means model for each K value**

The objective is to see the different results and identify the best k value using a list comprehension, running in one command different operations, creating a model and fitting it for each k value inside of the range.

Python code :

k\_means\_var = [KMeans(n\_clusters = k).fit(hpc) for k in k\_range]

**Adjusting the centroids cluster for each model**

The objective in this code using a list comprehension is collect the centroids for each model with different k values

Python code :

centroids = [X.cluster\_centers\_ for X in k\_means\_var]

**Calculating the Euclidean distance**

For each centroids value listed before, we calculate the Euclidean distance (or metric distance ). The Euclidean distance is the distance between two points which can be proved by repeated application of the Pythagorean theorem.

Python code :

k\_euclid = [cdist(hpc, cent, 'euclidean') for cent in centroids]

dist = [np.min(ke,axis=1) for ke in k\_euclid]

**Sum of the squares of distances inside of cluster**

Python code :

wcss = [sum(d\*\*2) for d in dist]

Total Sum of the squares

tss = sum(pdist(hpc)\*\*2)/hpc.shape[0]

**Total Sum of the Squares - Sum of the Squares of distances inside of clusters**

Python code :

bss = tss - wcss

**Elbow Curv**

The next step is calculate the variance resulted of the Total Sum of the Squares - Sum of the Squares of distances inside of clusters. From this result we know that for each k value we can understand the variance value

Python code :

fig = plt.figure()

ax = fig.add\_subplot(111)

ax.plot(k\_range, bss/tss\*100, 'b\*-')

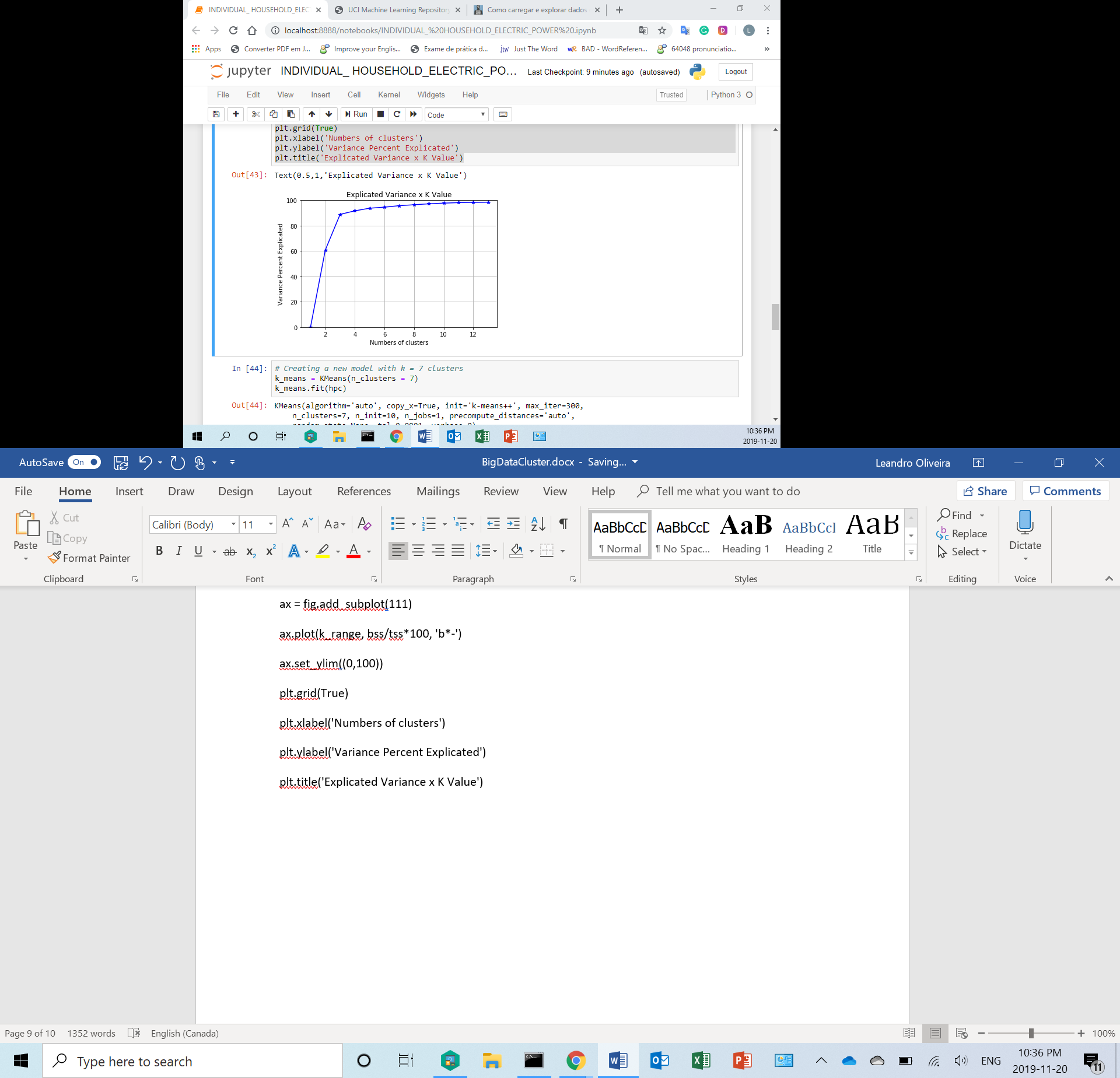
ax.set\_ylim((0,100))

plt.grid(True)

plt.xlabel('Numbers of clusters')

plt.ylabel('Variance Percent Explicated')

plt.title('Explicated Variance x K Value')



The *Explicated Variance x K Value* graphic show us that for k value >= 4 the variance could be more explicated

**Creating a new clustering model with k = 7**

Python code :

k\_means = KMeans(n\_clusters = 7)

k\_means.fit(hpc)

**Get the Minimum and Maximum Value and arrange the Shape**

Python code :

x\_min, x\_max = hpc[:, 0].min() - 5, hpc[:, 0].max() - 1

y\_min, y\_max = hpc[:, 1].min(), hpc[:, 1].max() + 5

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, .02), np.arange(y\_min, y\_max, .02))

z = k\_means.predict(np.c\_[xx.ravel(), yy.ravel()])

z = z.reshape(xx.shape)

**Area Clusters Plot**

Python code :

plt.figure(1)

plt.clf()

plt.imshow(z,

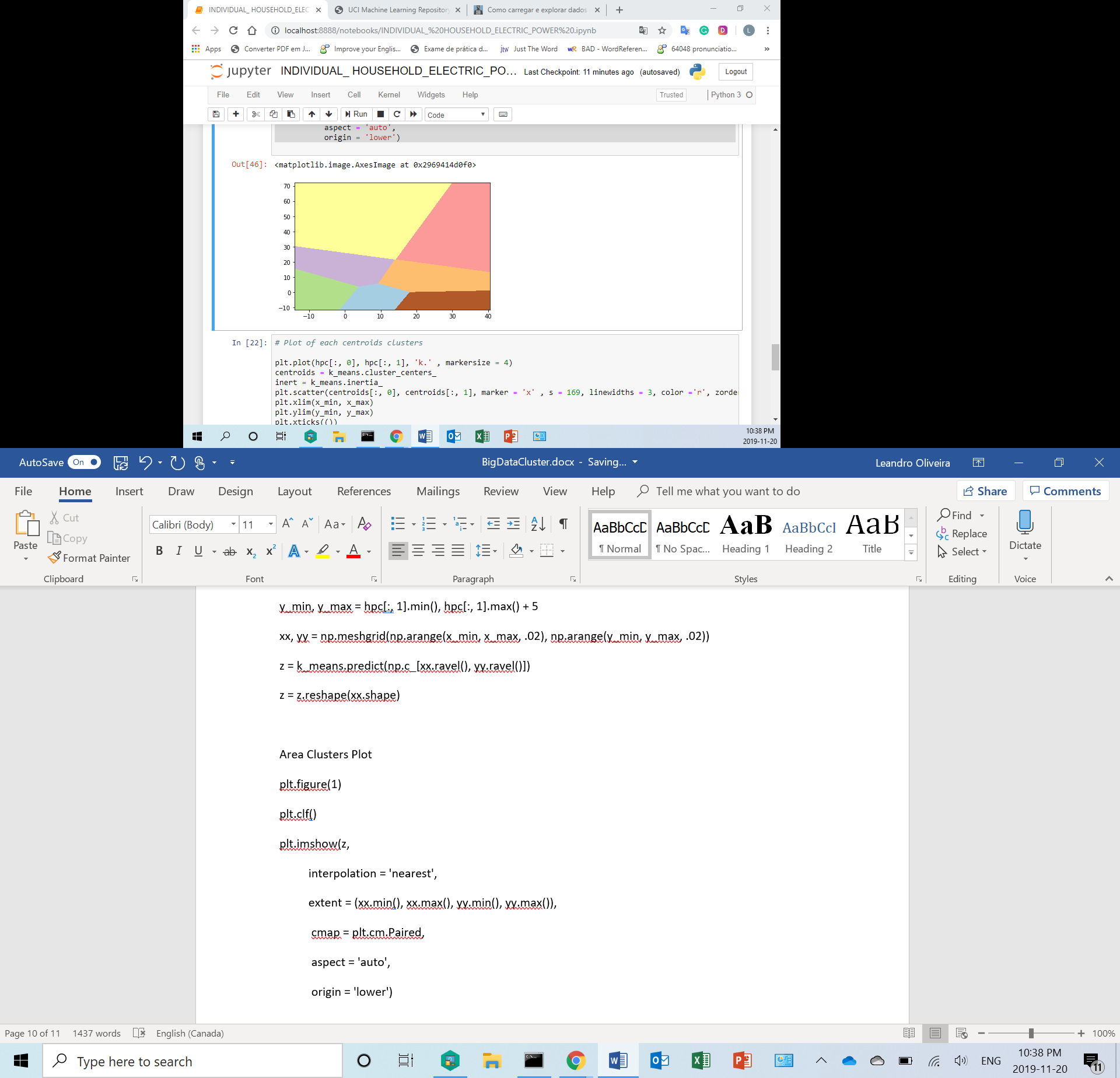
interpolation = 'nearest',

extent = (xx.min(), xx.max(), yy.min(), yy.max()),

cmap = plt.cm.Paired,

aspect = 'auto',

origin = 'lower')



**Plot of each centroids clusters**

Python code :

plt.plot(hpc[:, 0], hpc[:, 1], 'k.' , markersize = 4)

centroids = k\_means.cluster\_centers\_

inert = k\_means.inertia\_

plt.scatter(centroids[:, 0], centroids[:, 1], marker = 'x' , s = 169, linewidths = 3, color ='r', zorder = 8)

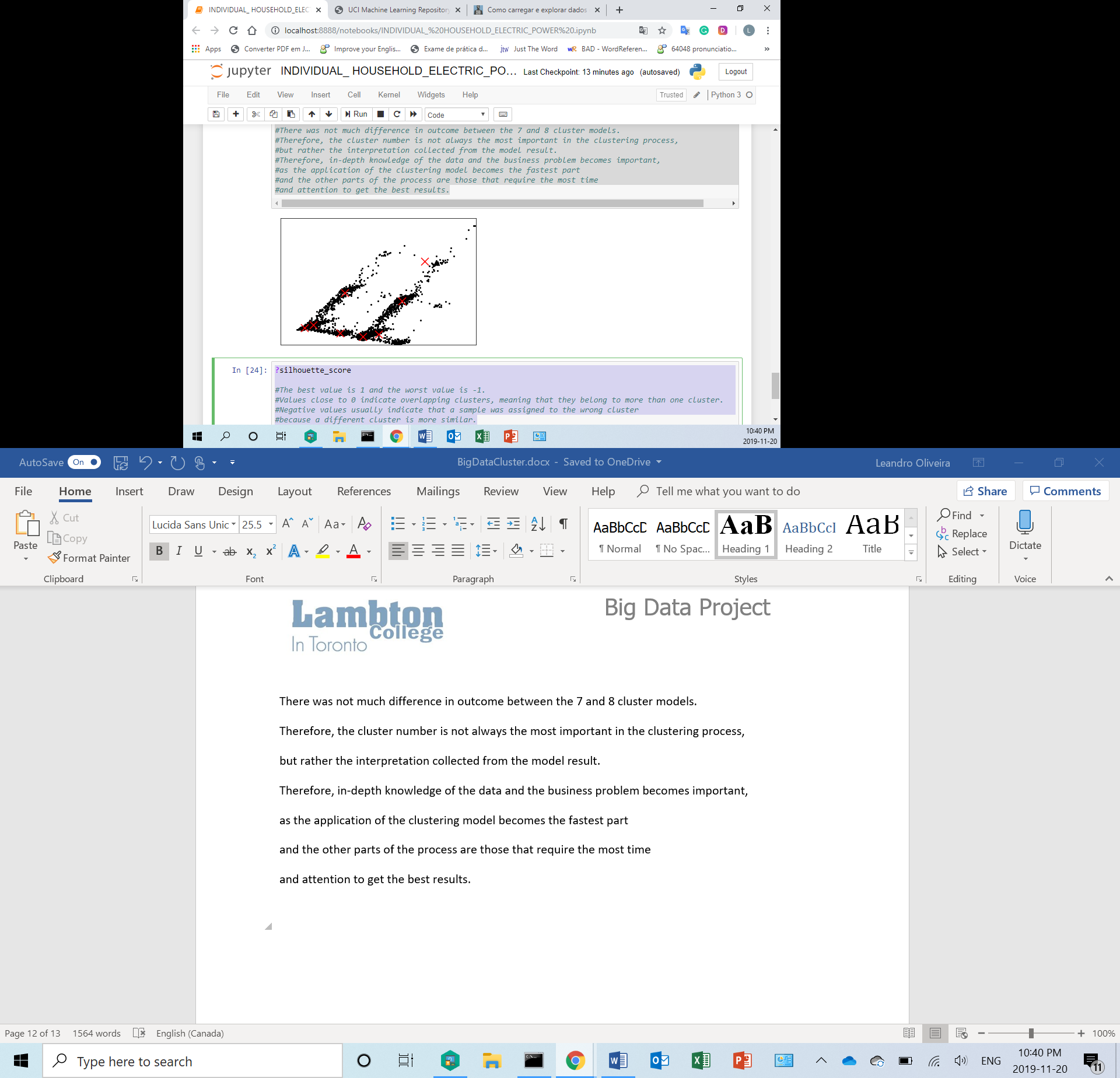
plt.xlim(x\_min, x\_max)

plt.ylim(y\_min, y\_max)

plt.xticks(())

plt.yticks(())

plt.show()



There was not much difference in outcome between the 7 and 8 cluster models.

Therefore, the cluster number is not always the most important in the clustering process,

but rather the interpretation collected from the model result.

Therefore, in-depth knowledge of the data and the business problem becomes important.

In this case, the clustering model application becomes the fastest part process

and the other parts of the process are those that require the most time

and attention to get the best results.

Python code :

# ?silhouette\_score

# The best value is 1 and the worst value is -1.

# Values close to 0 indicate overlapping clusters, meaning that they belong to more than one cluster.

# Negative values usually indicate that a sample was assigned to the wrong cluster

# because a different cluster is more similar.

# Silhouette\_score

Python code :

# labels = k\_means.labels\_

# silhouette\_score(hpc, labels, metric = 'euclidean')

# Summary:

We conclude that the final value above 80% is satisfactory to consider that the similarity classification performed by the clustering method is acceptable to assist in the decision making of the business problem