# Artificial Intelligence – Homework 3: Machine Learning

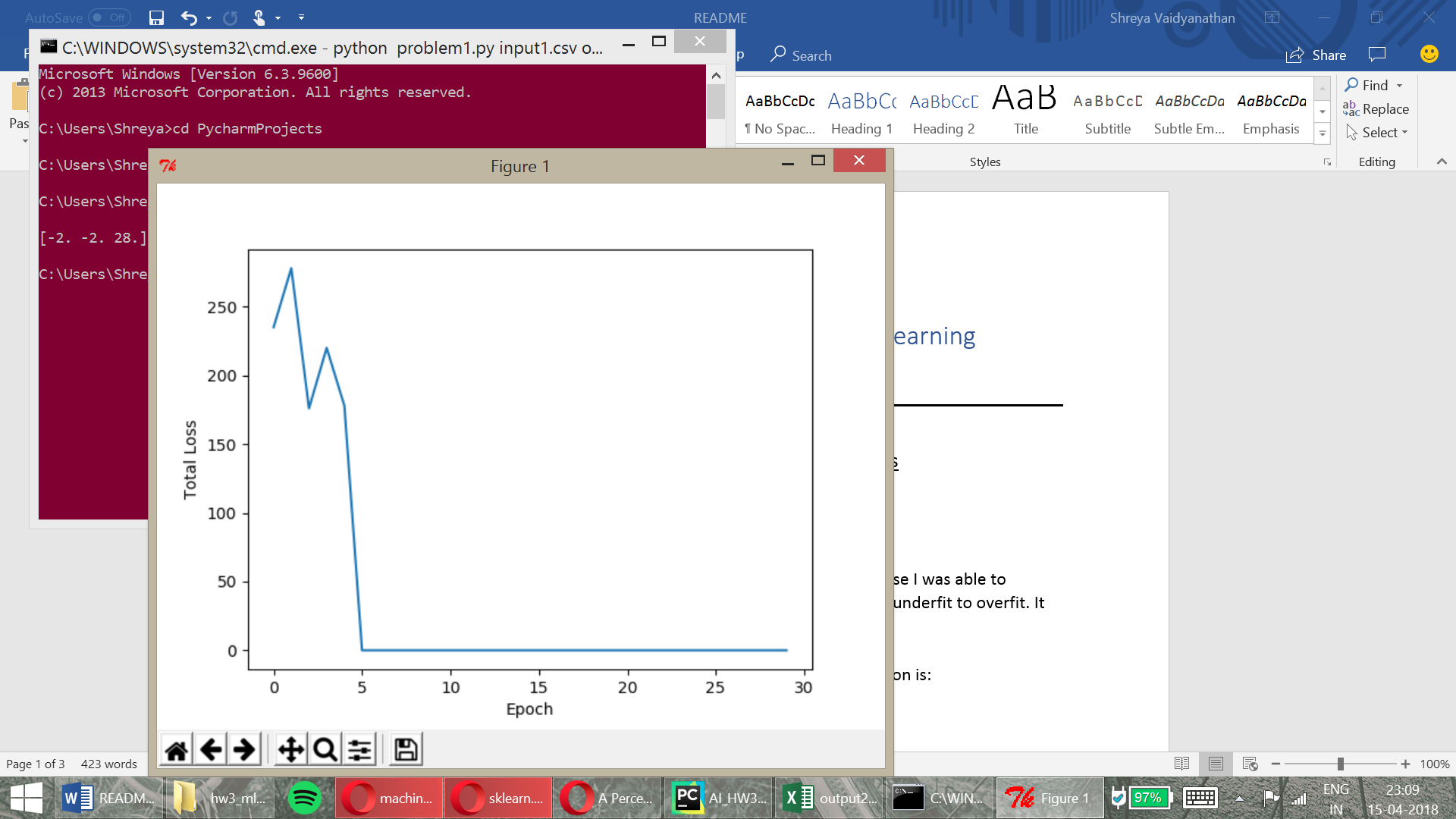
Name: *Shreya Vaidyanathan*

UNI: *sv2525*

README – for PROGRAMMING Questions

1. **PERCEPTRON**

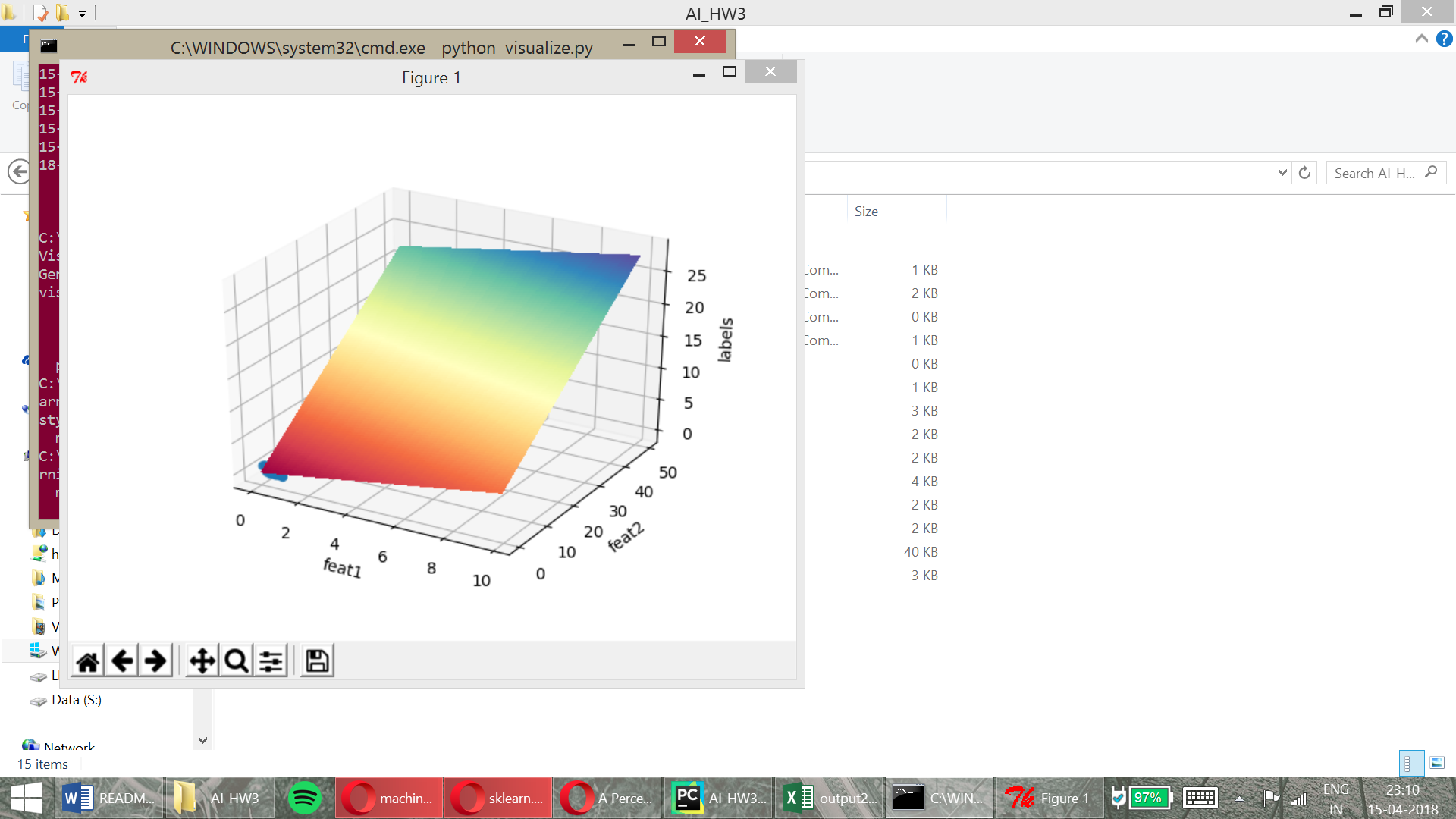
Visualising the total error rate and loss with epoch for the given data set



1. **LINEAR REGRESSION**

10th Alpha value: I picked ‘5’ with an iteration of 150 because I was able to observe that the values from the first 9 were ranging from underfit to overfit. It was normalized around the range of

Final visualisation of the plane in the multivariable regression is:



1. **CLUSTERING**
   1. **K-Means:**
      * Firstly, I tried KMeans with n=3. The clusters were based on very few colours – clearly ***underfit*** as only 3 colours have been chosen for the picture that probably has thousands of colours in the original - and the result was as follows. (Figure 1)

k\_colors = KMeans(n\_clusters=3)  
k\_colors.fit(image\_rescale)

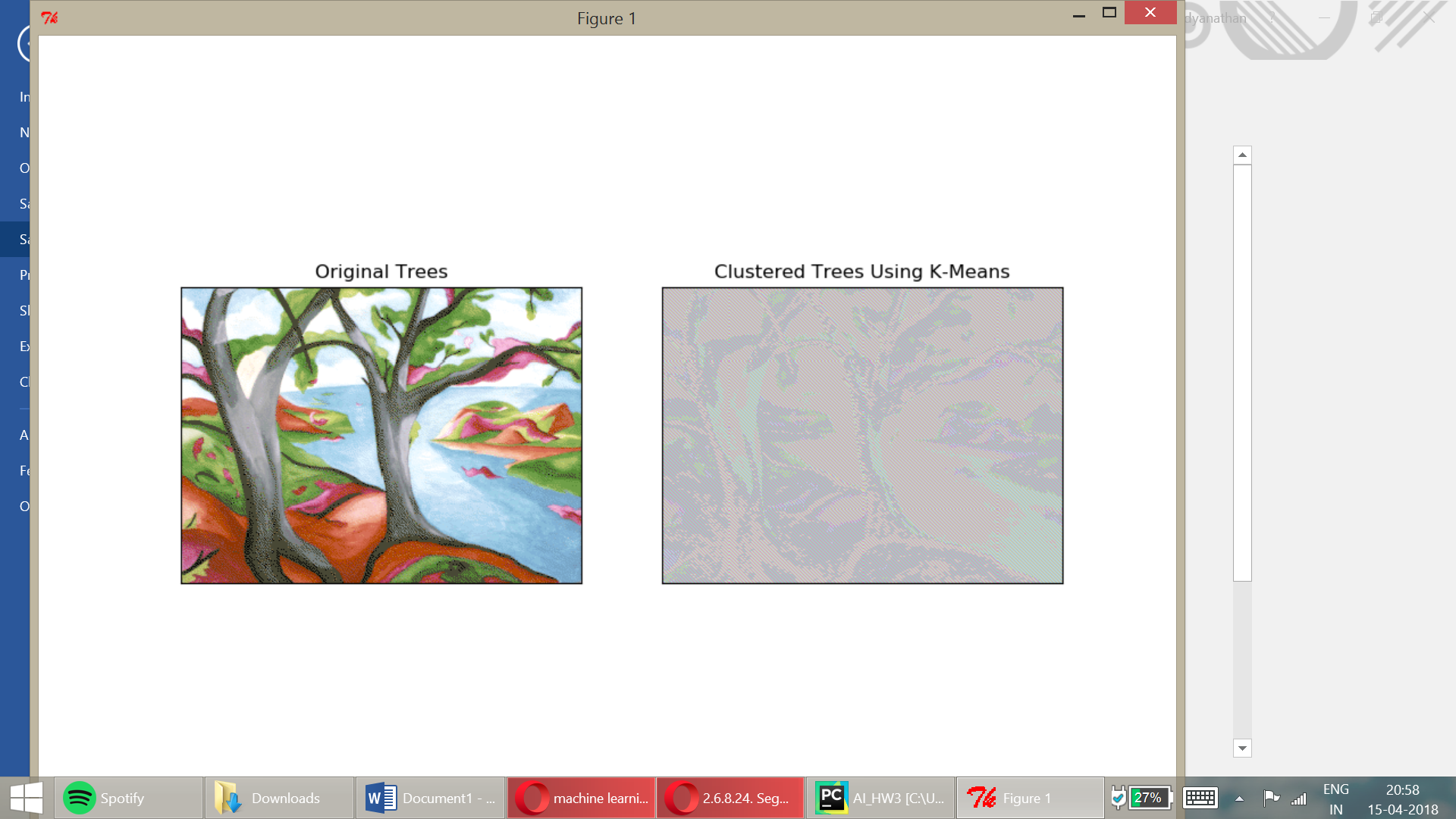


Figure 1 – KMeans Clustering

* + - KMeans with n=5, but also trying to initialise the init and random\_state parameters in order to see if we can make the distribution of selection more effective. Results were more segmented but the many portions were still underfit. Or colored in the same color itself. (Figure 2)

> k\_colors = KMeans(init='k-means++', random\_state=0, n\_clusters=5)

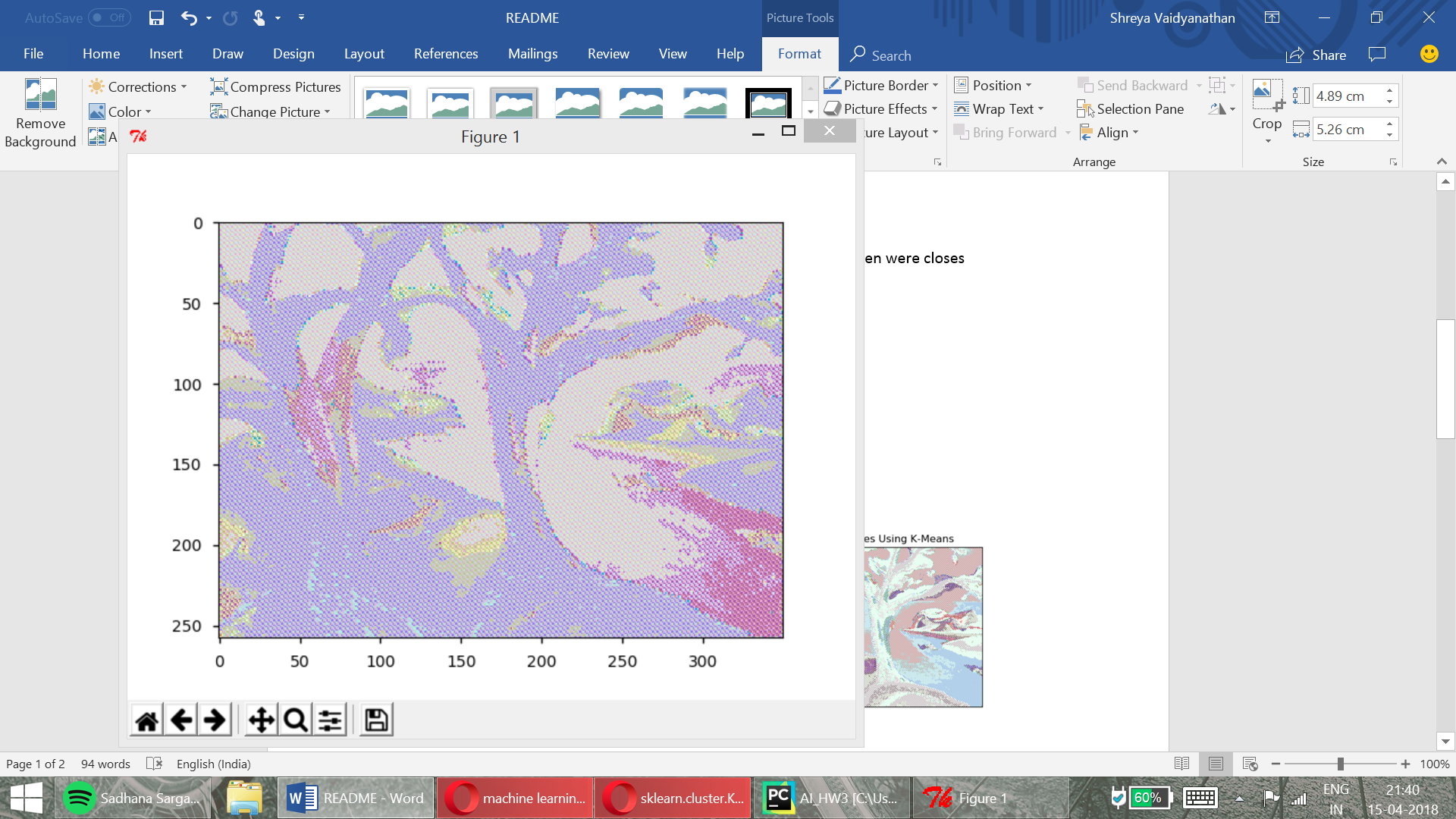


Figure 2 - KMeans Clustering

* + - Next, I tried n=10 for the number of clusters. The segmentation appears much better as it separates even smaller segments of colors in the pictures clearly. Result visualization below (Figure 3)

1. k\_colors = KMeans(random\_state=0, n\_clusters=10)

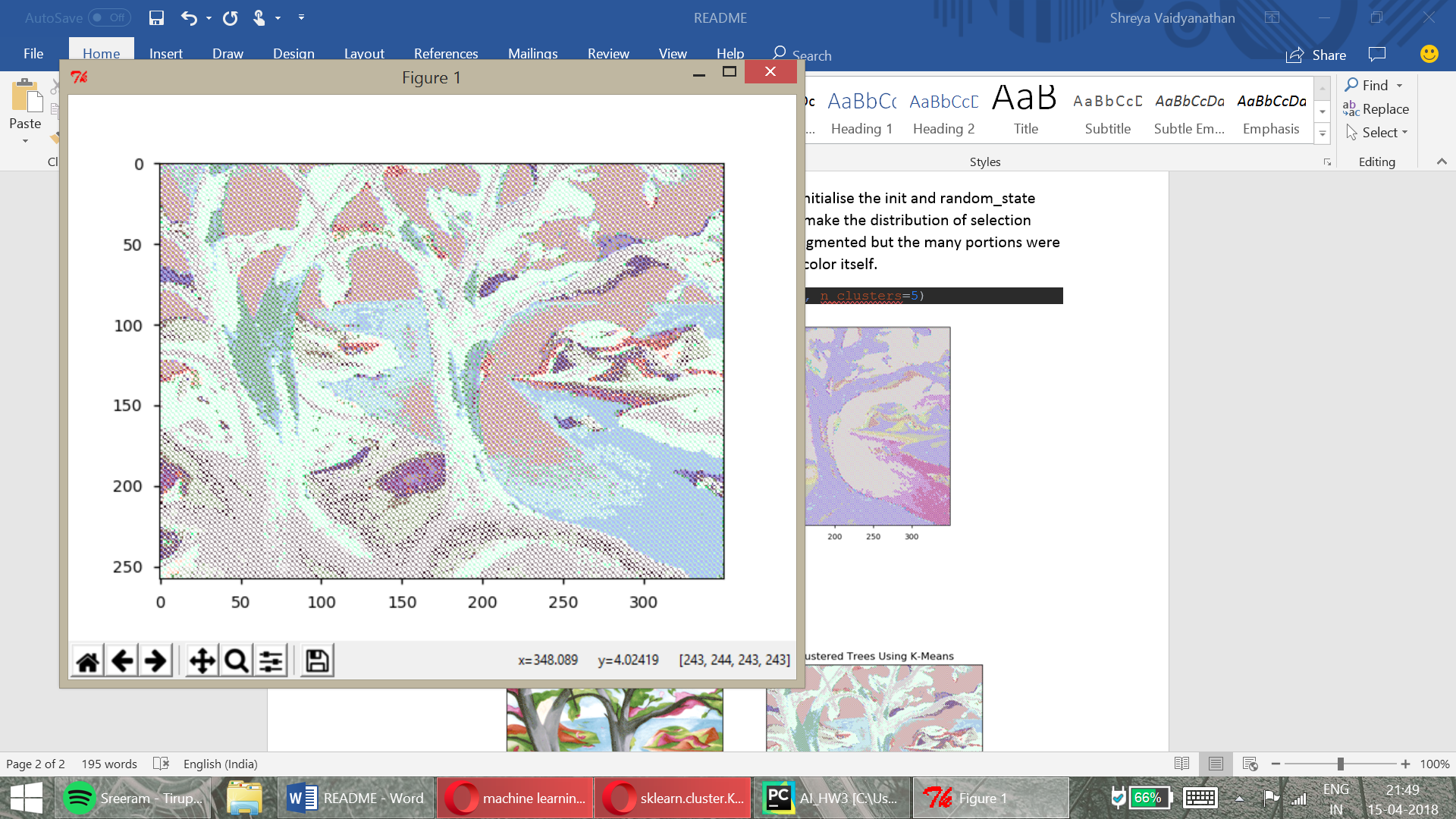


Figure 3 - KMeans Clustering

* Then, I tried to use a higher value of n=16, but the picture quality appeared too **overfit** (especially to the naked eye). The colours were getting segmented more than the limits and it appeared like the algorithm wasn’t as good as the previous times. Visualized below (Figure 4) you can notice how parts of the branches with the same color have been over segmented and that can be confusing as it goes into the detail of numbers of RGB values only and kind of forgets the larger need of color separation.

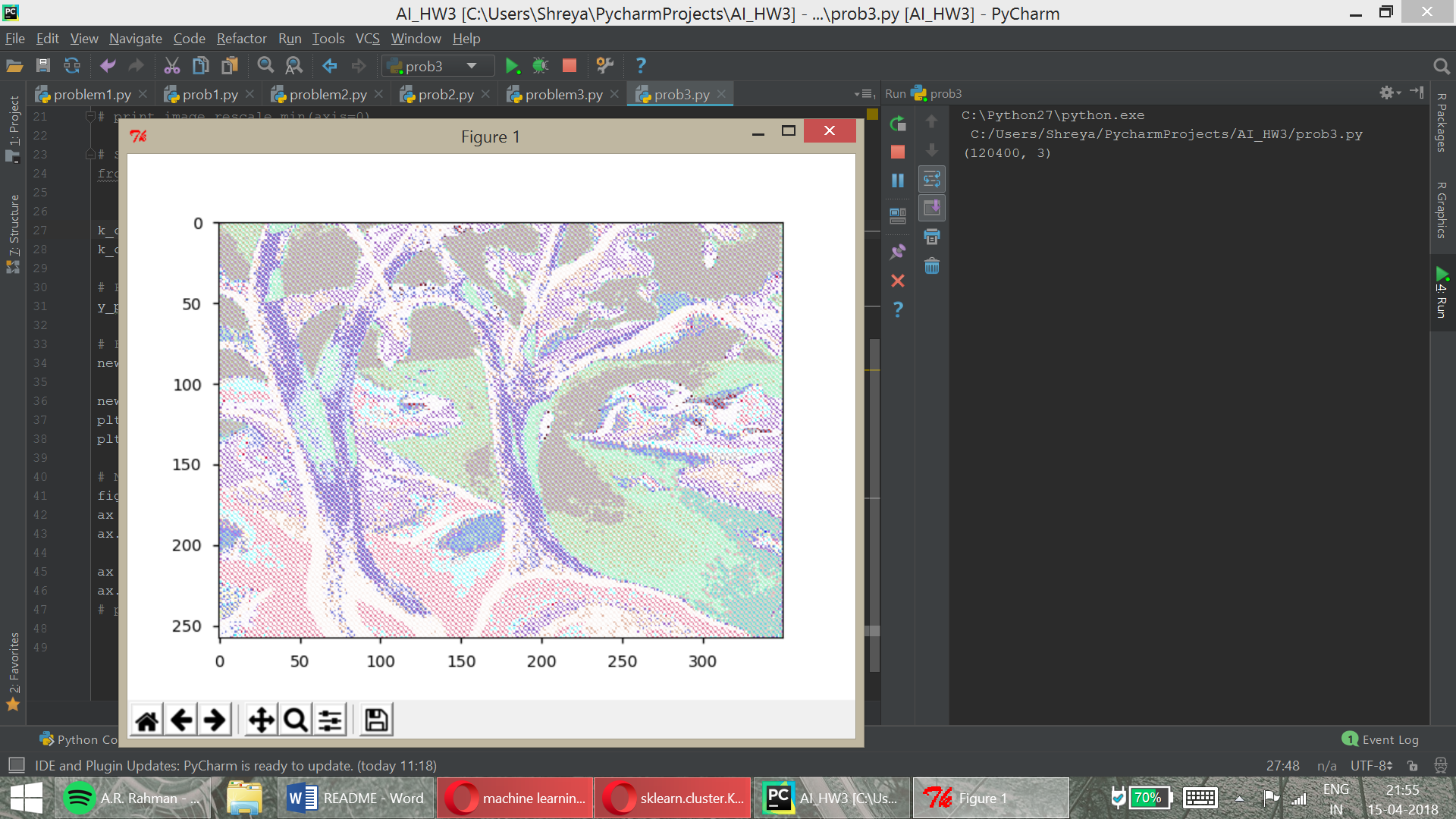


Figure 4 - KMeans (n\_clusters=16)

* I was most happy with n in the range of n=10 to n=15 for my algorithm – using the various parameters in scikitlearn and after shuffling the image array. This appeared to perform the segmentation clearly and would be a good way to perform the clustering technique. (Figure 5)
* My colours were not becoming close to the original as the centers were chosen randomnly. I tried to optimize that to my best effort, but it wasn’t getting recognised effectively – so I have shown the true result with the best segmentation combination I found.

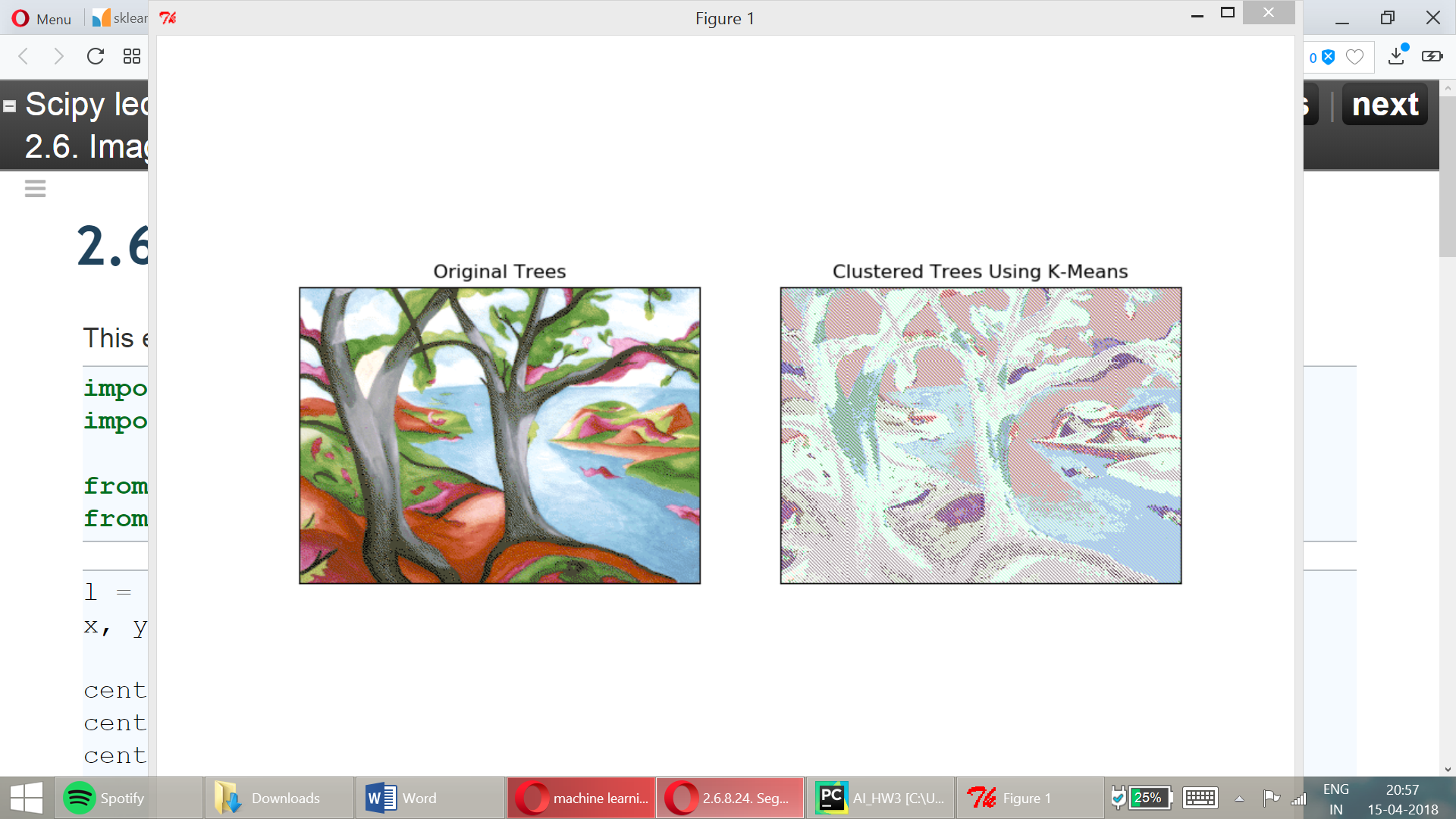
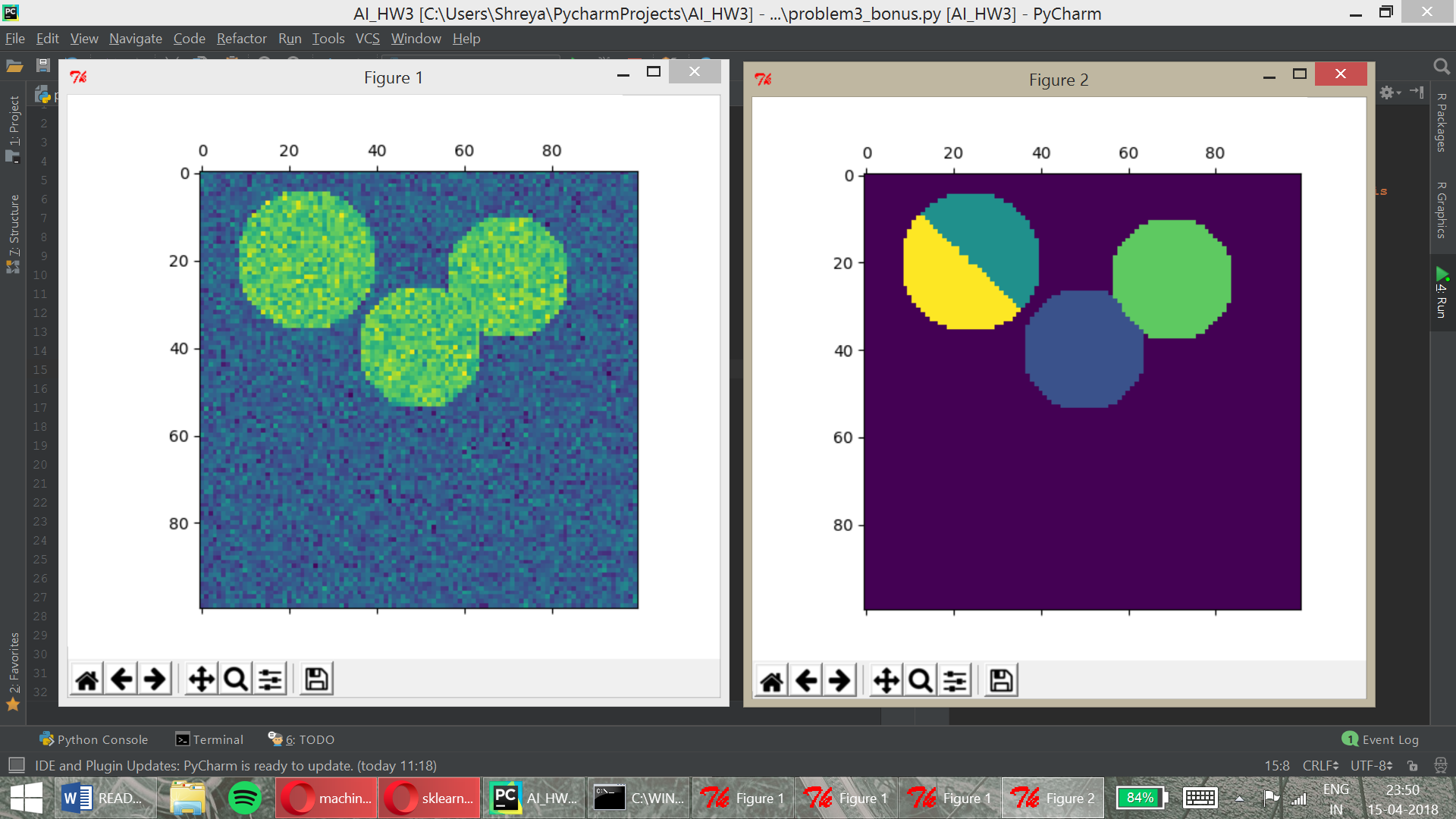


Figure 5 - Final Result

**2. Spectral Clustering:**

Tried to implement spectral clustering with the scikit.learn function available for spectral clustering. Went through several tutorials and options to finally land on a simple circles example to illustrate the best way to implement the algorithm. In the below circles image the spectral clustering works much better than KMeans.

Visualization of my example:



* The algorithm is used for medical imaging and various cases where in the spectral clustering can separate to greater minute detailing. Which KMeans does not do well – it approximates and plays around with the value of K and n\_cluster size.
* But on the other hand, this algorithm is suited for cases wherein the cluster centres are known or computed easily. – Ideal cases would be that of circles of well-defined objects with centers of the clusters being quite prominent.
* It is also not very effective for larger images. I tried to see if the given ‘trees.png’ could be used as a dataset, but it was not working at all.