In this project we will explore the following concepts:

* Use *PCA* to visualize the clustering result.
* Define the number of clusters using *Elbow method* & *Dendogram*.
* Compare the result of *K-Means* vs *Agglomerative Clustering*.
* Do deeper analysis with clustering result.

**Outline**

* Data Exploration
* Data Preparation
* Training Models (*K-Means & Agglomerative Clustering*)
* Cross validate by classification accuracy
* Conclusion

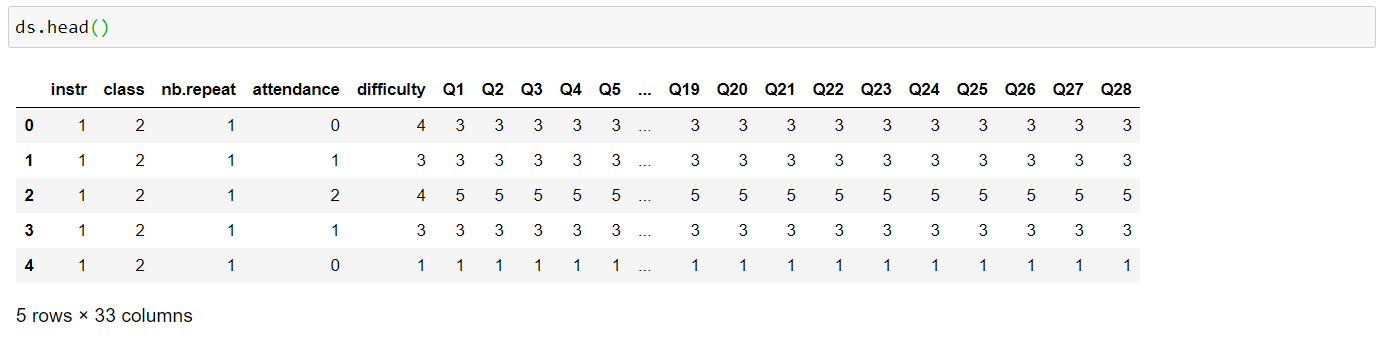
**1.Data Exploration**

First things first, let’s check the dimension of this dataset:

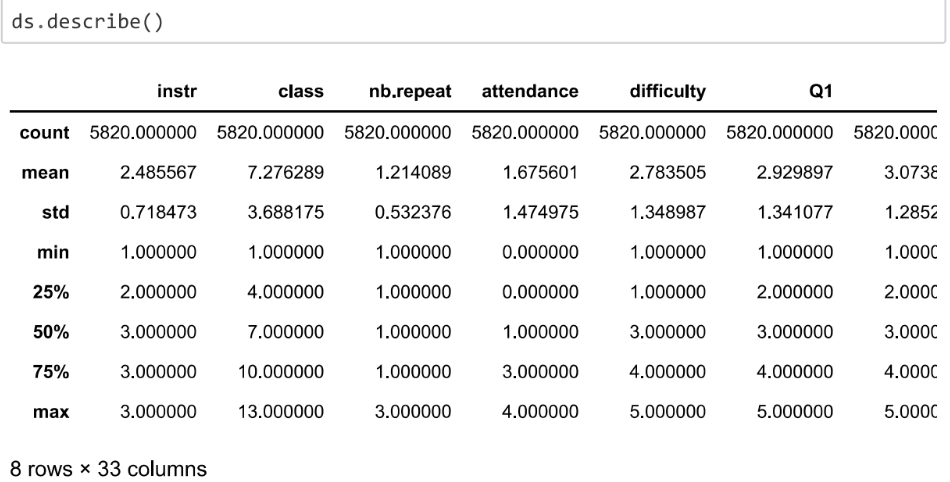


We have **5820** observations (students) with **33** features (28 survey questions and 5 attributes)

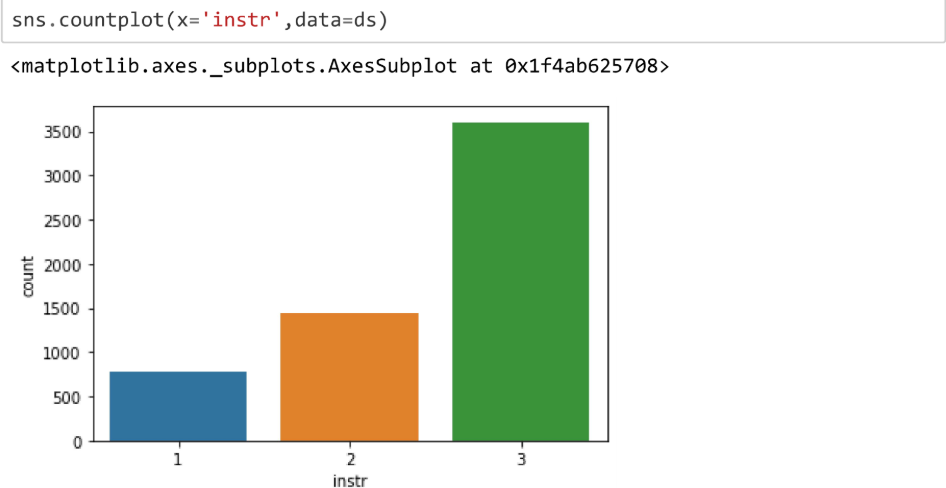
We can check the first 5 features with:



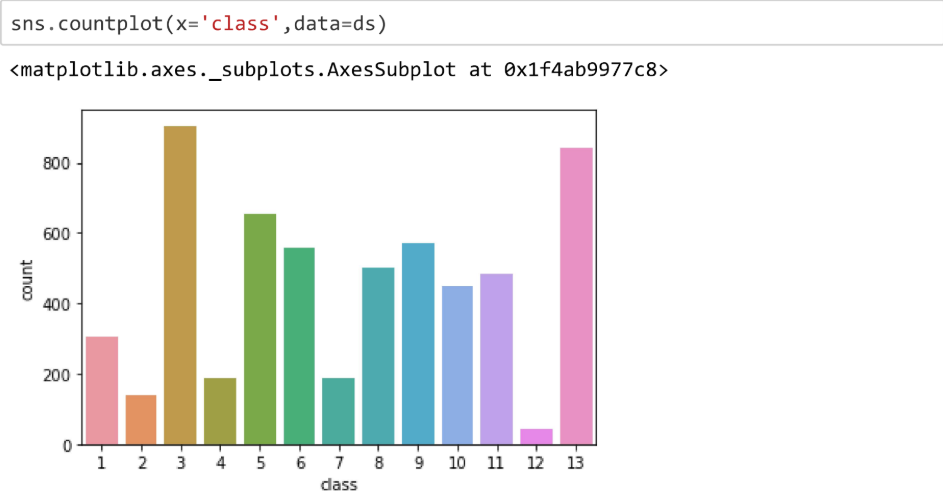
And basic statistical attributes with:



Although we can conveniently do a quick check on the dataset with the methods above, I prefer to open the *csv* directly to do some basic analysis. We can see that this dataset has 3 Instructors, who taught 13 classes.

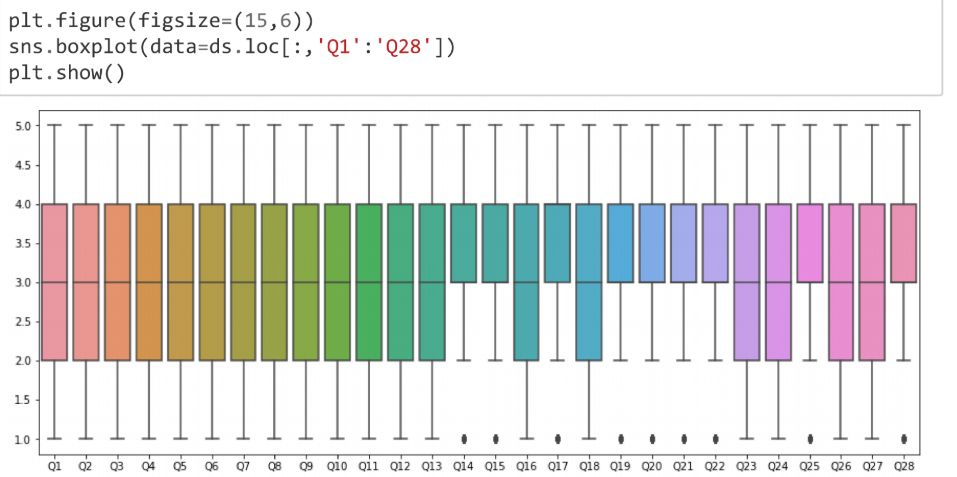


Most student taught by the third Instructor, but this don’t necessarily mean he taught most of the classes. There could be a large gap of the number of students between each class, so let’s check it:

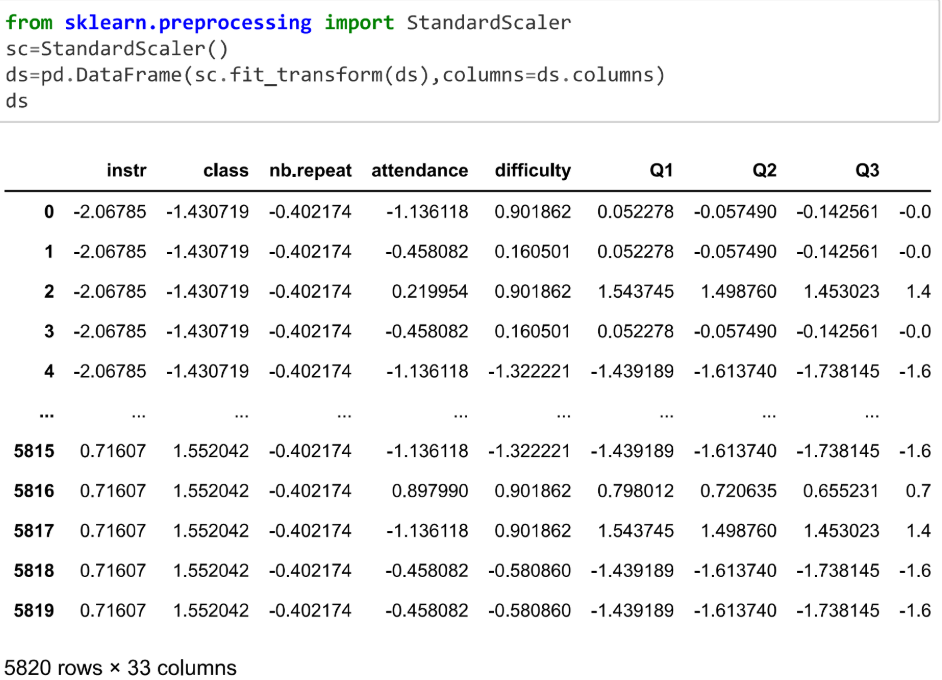


The number of responses of some classes is clearly much lower compared to the rest. Doing a little bit more analysis, we could see that the 3rd instructor taught 7 classes: class no 3, 4, 5, 8, 9, 12, and 13.

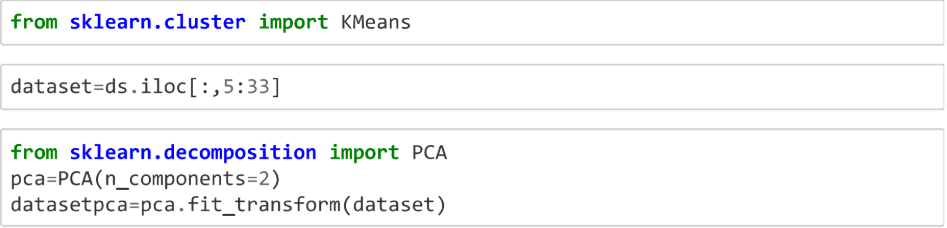
Let's plot the boxplot on questions.



**2. Data Preparation**:

****Scaling the whole dataset

Let’s remove all non-question features because we will group the student based on their answer in the survey:



Recall that we don’t know how many clusters will be on this data and whether the clusters our model generated will be “natural” or not, so we will need to confirm it by using visualization.

However, we have 28 features that will be 28-dimensional space. It will be very hard for us to visualize (and understand) this much dimensions, so people usually use *dimensionality reduction* methods to transform them to 2-dimensional space (*x, y*).

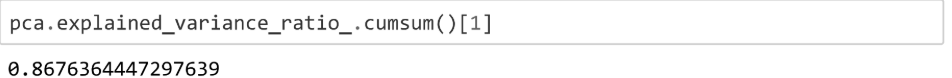


Specifically, we will use *Principal Component Analysis* (*PCA*) to do the dimensionality reduction. *PCA* can help us to identify patterns based on the correlation between features. This algorithm aims to find maximum variance using fewer dimension than the original data.

One drawback of using *PCA* is the new features generated by *PCA* (*x, y* in this case) will have less variance than the original features. This means we might lose some information and lower the predictive power of our model.

Hence, in this article we will use *PCA* to visualize the clustering result and then remove the *PCA* (use default features) and compare the difference outcome between them.

*pca* will have 5820 observations with only 2 features (dimensions). Let’s check the amount of variance we preserved from the original data:



We managed to save around 87% of the variance, which is not very good but not bad either (people tend to prefer *explained variance* ≥ 90%). We don’t have to worry too much about this because we will not apply *PCA* for the final clustering result (and we will compare their result).

Another disadvantage of *PCA* is it transforms the features into abstract features. This mean we will lose data interpretability, hence making us unable to answer many business-related questions.

**3. Training Models:**

**a) K-Means**

Let’s roll our sleeves up and start by one of the most popular clustering algorithm: The *K-Means* clustering. In a nutshell, this algorithm will cluster each observation based on their distance to the nearest *centroid*.

*Centroid* is randomly picked from the observations (a sample/row) initially and will be moved on each iteration based on the *mean* (for continuous features) or the *modus* (for categorical features) of the observations that were assigned to it. The number of *centroids* will be the number of the clusters.

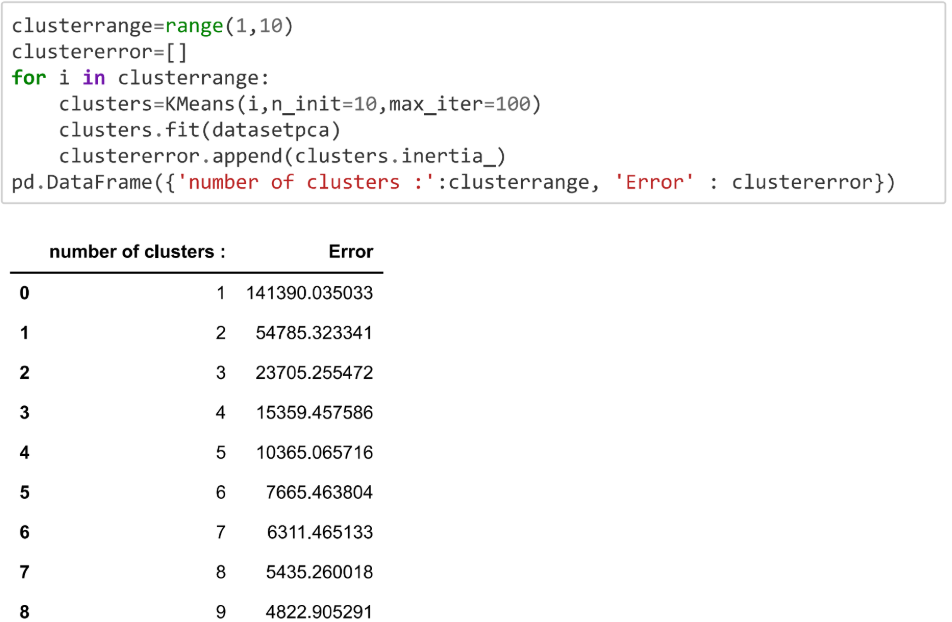
The inner working of this algorithm is summarized by the following steps:

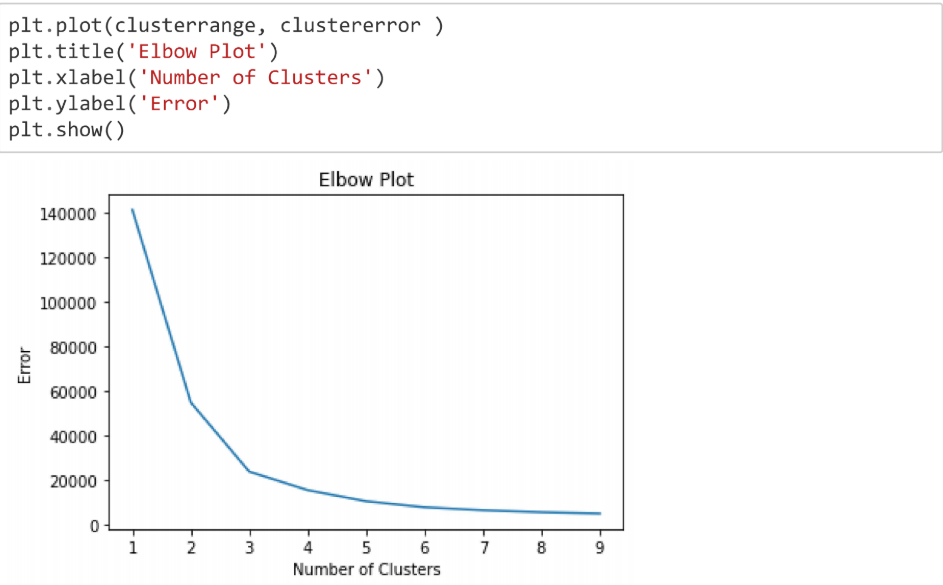
* Pick the number of cluster (we will use *Elbow method*). Let’s call this number *k*.
* Randomly pick *k* observations as initial *centroids.*
* Assign each observation based on the nearest *centroid* (computed by [Euclidean](http://mathworld.wolfram.com/Distance.html) distance).
* Move the centroid to the center (average) of the observations that were assigned to it.
* Repeat step 3 & 4 until the cluster do not change (convergence) or maximum iteration is reached.

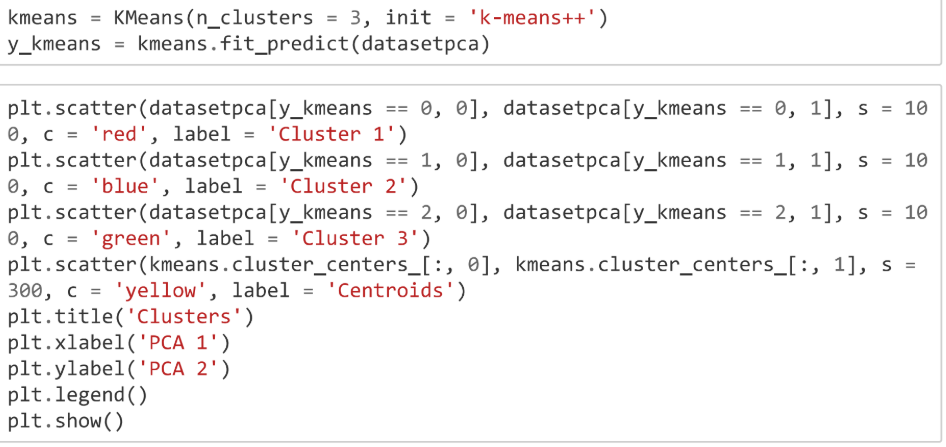
The clustering result of this algorithm depend on the random initial *centroids*. This mean the result might not be the same if we run this algorithm multiple times. Fortunately, the *K-Means* model in *sklearn* have implemented more advanced version of this algorithm, called *K-Means++*.

*K-Means++* will place the initial centroids far away from each other, which lead to faster convergence and more consistent result. This is the default value of *init* parameter.

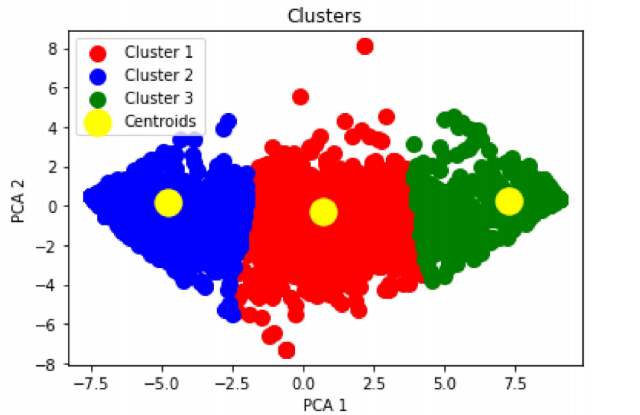
The next thing on our to do list is to perform Elbow method. This method allow us to pick the best number of clusters (*k*) by computing the *Sum of Squared Error* of each cluster (also called *distortion*). Smaller value of *SSE* mean the distance between *centroid* and each observation within the cluster is closer.

Our goal is to identify the *k* number of cluster where the distortion decreases most rapidly. In this article, we will try fitting the *K-Means* model with *k* from 1–9, and compare the change of distortion of each *k:*

Notice that we put the model in the *for* loop and use *n\_clusters=i* to iterate through each value of *K\_to\_try* (*range (1, 10)* will generate a list of integers of 1–9). The code above will generate the following graph:

The “elbow” seems to be located at *k* = 3. This mean the distortion will not be decreased significantly if we tell the algorithm to use a larger number of clusters, hence the clustering result might be too complicated and not “natural”. Let’s train the *K-Means* with 3 clusters:

visualize the result by scatter plot:



We could see that the observations are tightly grouped (this confirmed our analysis in the Data Exploration earlier) and the clusters are reasonably “natural”.

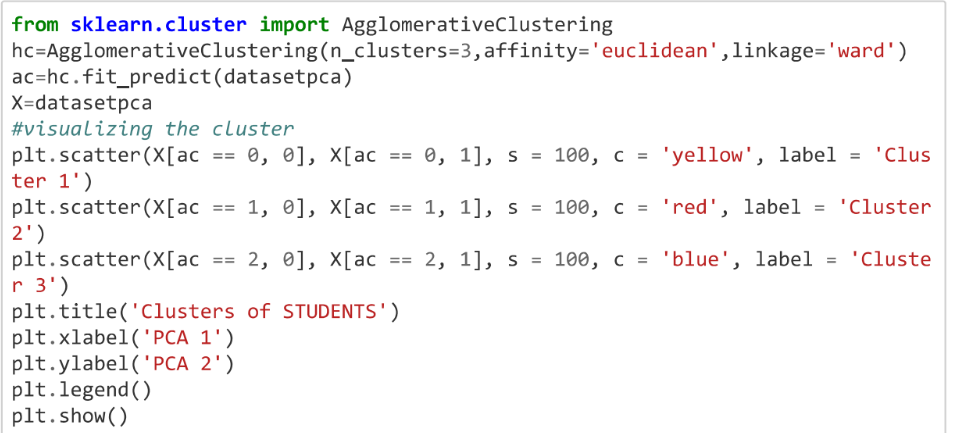
Let’s see what those clusters mean in the context of answers given by those students. Let’s check the number of observations belonged to each cluster:

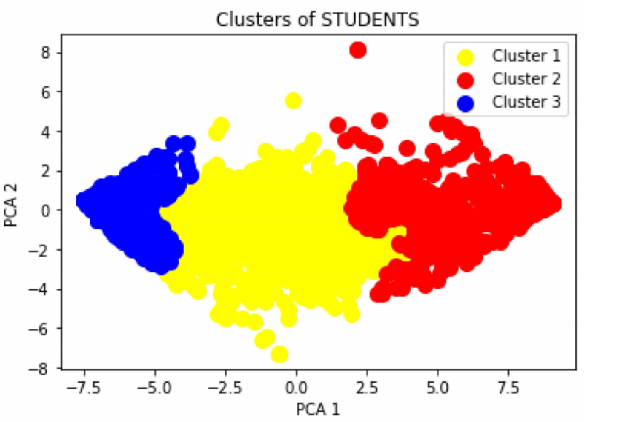
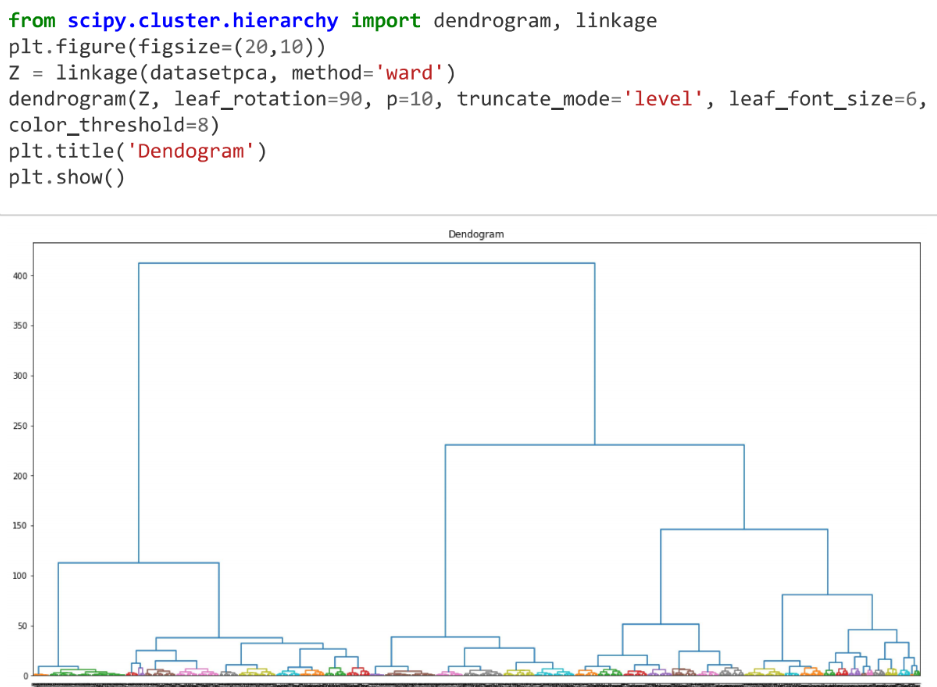
## **b) Agglomerative Clustering (**Hierarchical clustering)

Next, we will explore *Agglomerative Clustering* algorithm. One advantage of using this algorithm is its ability to plot *dendograms* (we will plot it later), which can help us to interpret the clustering result. Another advantage of using *dendogram* is it will tell us the number of best cluster (*k*) hence we don’t need to try out each value of *k* using Elbow method.

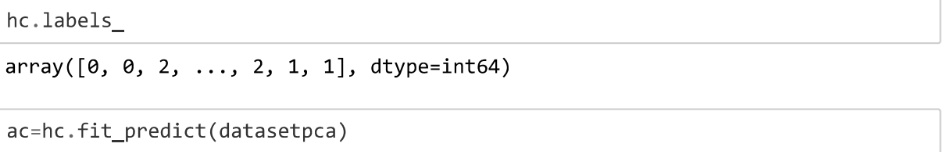
In a nutshell, *Agglomerative Clustering* will assign each observation as individual cluster and merge those clusters based on their distance (similarity) pair by pair, iteratively.

There are many ways to compute the similarity of the clusters, we will focus on *Ward’s linkage* algorithm in this article. *Ward’s linkage* will merge clusters that lead to minimum increase of *Sum of Square Error* (SSE).

Similar with *K-Means* implementation earlier, we will apply *PCA* for visualization purpose. Let’s see the code for Agglomerative Clustering

visualize the result by scatter plot :The code above will generate the following *dendogram*:

Let’s fit the model with the best *k* we found earlier

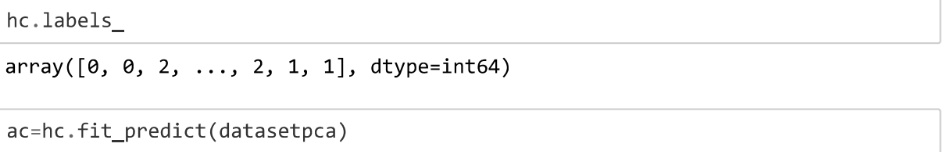


Let’s compare the number of students in each cluster and check the difference:

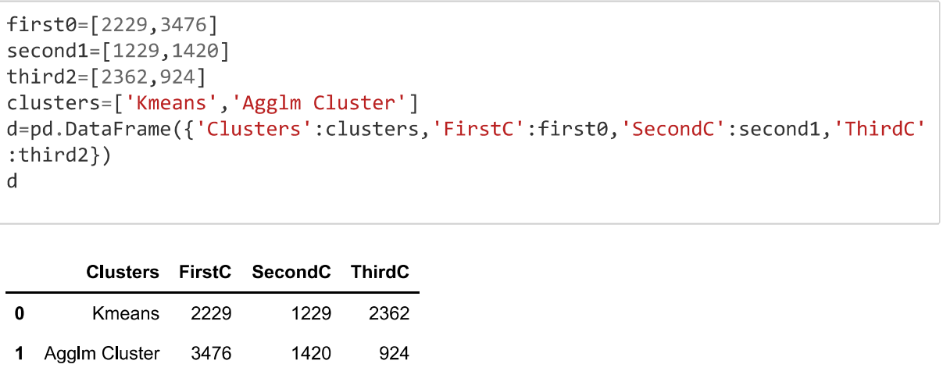
Kmeans :



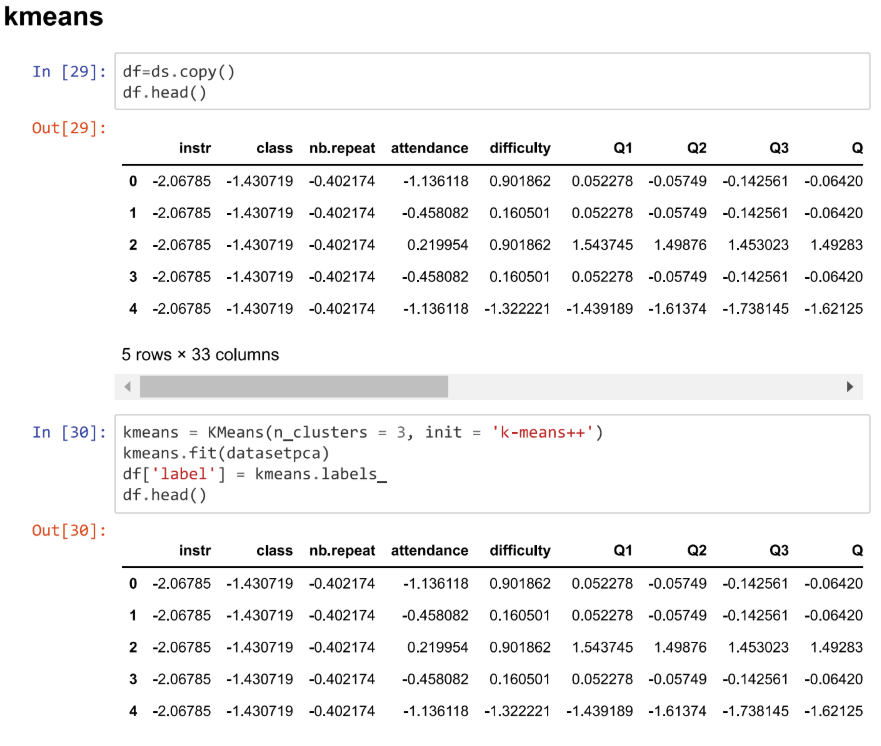
Agglomerative Clustering :



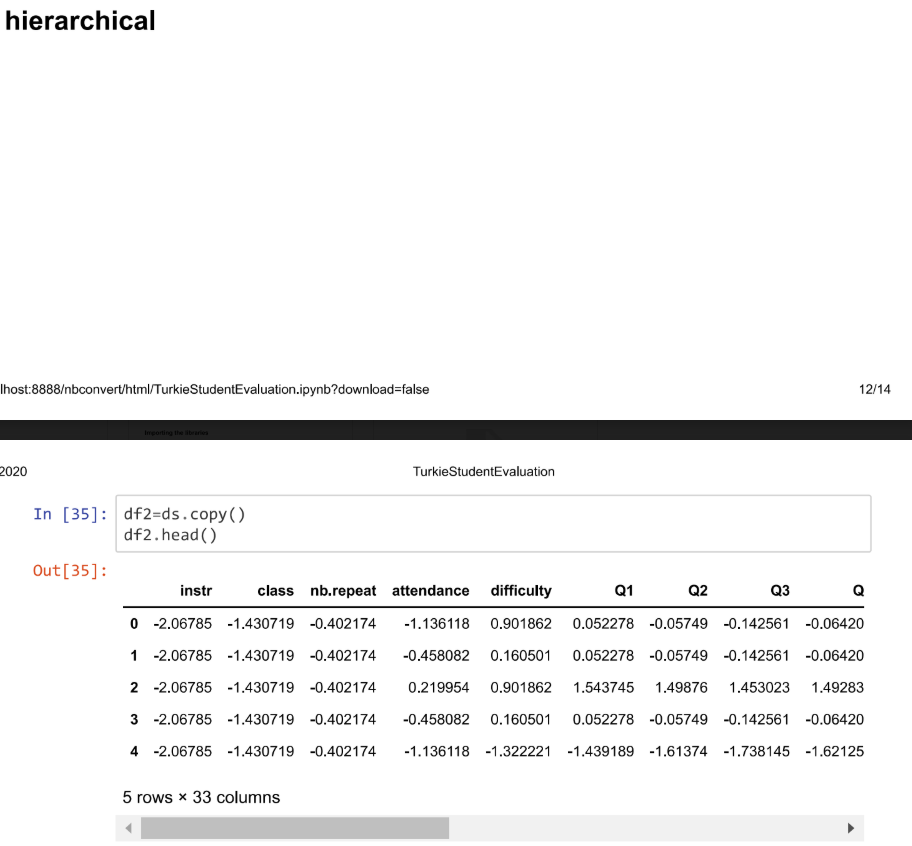
**Cross validate by classification accuracy:**

Comparing the k-means and hierarchical clustering classes 

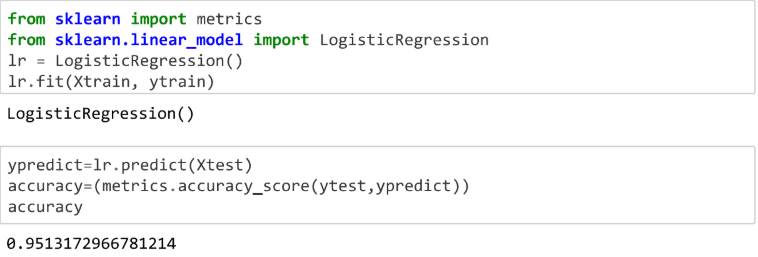
Let us cross validation the dataset by appending clusters group to original data and apply logistic regression to find accuracy.



We got 99% accuracy by using kmeans clustering algorithm.





We got 95% accuracy by using hierarchical clustering algorithm

**Conclusion**

We got 99% accuracy on applying kmeans and 95% accuracy on applying hierarchical clustering so we conclude that k-means gives the best clustering groups.