**MoA: Data Preparation Report**

**Team Members:**

* **Johnny Bou Saab**
* **Hilal Breiss**
* **Anis El Rabaa**
* **Othman Ibrahim**
* **Ahmad Nsouli**

For our project which is concerned about predicting the modes of action (MoA’s) of drugs based on their genetic data and their cell viability data. Our dataset is acquired from Kaggle and it is done by experts in this field. We have a classification problem and in fact, we have a very clear and organized dataset that doesn’t have any missing values and is scaled using z-score standardization. For that reason, and in order to begin working on phase 2 and make sure that we will have work for each part of the phase, we corrupted our data first. We did this by introducing a random amount of missing value to the data and assigned them as NA. We ended up then in a corrupted data set that needs to be pre-processed before applying any ML model to it.

We used co-labs to share code that allow efficient version control. We wrote our code using python for all the parts but EDA, and pattern analysis using kmeans clustering are written in R.

We will be sending you a copy of the original data set (the original Dataset from Kaggle), a copy of the corrupted data set (the one we introduced NA to it), and a copy of a prepared data set (pre-processed).

Below is a description of each of the part of phase 2 along with the method used to perform each task.

**Part 1: Data cleaning**

For the data cleaning part, since our data came clean, we decided to introduce some kind of error to it manually and then clean it. So, what we did was adding NA values to a small percentage of the data, and since we have a lot of data, we chose 1% for this part (230+ NA values introduced). Note that our code is flexible and it's mostly functions so we can change that later if we decided it's better for the purpose of this project depending on the models we try. The way we cleaned the data afterwards was by removing the rows that have NA values only if the variance of the missing column data was higher than some threshold, this helps preventing imputing data that might be significant to our training later. Again, this threshold is a parameter that we can change in our code, and if we find that the variance of the missing data's column is high enough, we can drop the whole row; otherwise, if the variance is not very big, we will replace the missing variance by the mean of that column.

Concerning the outliers, we did some research on our data which is complicated cell information, and we found out that outliers are very important for later model training and we cannot afford to lose any of them, so we decided to leave them as is; not that our problem is multi label classification with more than 200 classes and 800 features so we really cannot lose any data we have as long as it’s feasible to have, and outliers will be very useful in this case because some classes are only detected by these outliers.

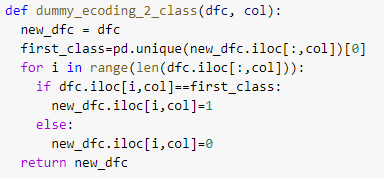
**Part 2: Future selection**

For the feature selection part, we tried 2 methods. We used chi2 scores and f scores to select some ‘k’ features. We used sklearn library, the feature selection part of it. SelectKBest in sklearn helped us do the job. However, we had multiple challenges. The first one is negative values, which should not be allowed when calculating chi2 or f scores, so we had to shift our data values by the minimum value of all values of our data, so we shifted by the absolute value of that minimum in order to have positive values only in our data. After doing that, we realized that since our problem is multi label and not only multi class, we need to calculate the scores of each label alone so we kind of need to break down our feature selection into selecting for each label, making it sum of subproblem of a binary classification problem. So, we calculated the scores, in both methods, for each label, and then we took the mean of chi2 or f scores as the final score to select the most important k features, and that mean should be higher than some threshold in order to be chosen as a feature. This threshold determines the number ‘k’ of chosen features at the end, and this is a parameter we tried. For chi2 we managed to get 500+ features instead of 800+ (exact numbers are in the code comments along with thresholds chosen) and for the f scores we got 300+ features.

**Part 3: Data Transform**

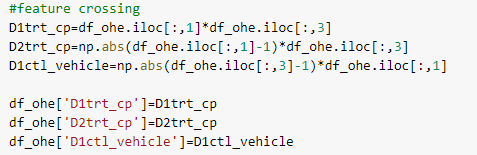
Given that our data, has negative values for each of the numerical features, we can’t perform feature selection methods, given negative values. Hence, we transform our data, so that no negative values appear in the numeric features. The method we use is shifting. For each column we calculated the min values (min\_val) which is a negative. Then we add absolute value of min\_val for each value in the column. By applying this method, we are certain that no negative values appear anymore and we can perform feature selection efficiently.

Encoding categorical data is another data transform technique we applied on our dataset. We have a total of 3 categorical features and one feature that is an ordered factor. One of these features is the id which is distinct for all the row, then we removed it from our features list. Two other features are consisting of at most 2 classes, then we used dummy encoding to transform them. For the remaining feature, we keep it as an ordered factor.

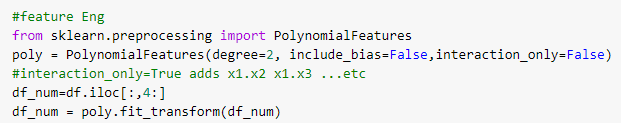


**Part 4: Feature engineering and Feature Crossing**

Since we don’t have enough knowledge about the features in our data (expert needed in this fields), we weren’t that much flexible with doing feature crossing since we don’t know what the new created features will represent. However, given that we have categorical data that we can interpret some meaning from it, we did feature crossing on the categorical features mainly cp\_dose and cp\_type (2 classes each). And this we did it after applying dummy encoding to them. We ended up with a 4 (2 classes crossed by 2 classes) new features that and each has the values 0 or 1 depending on the categories of the given features that created them.

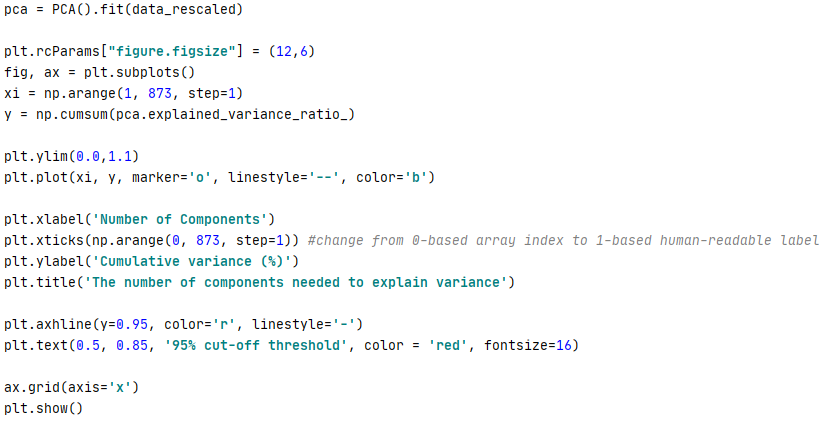


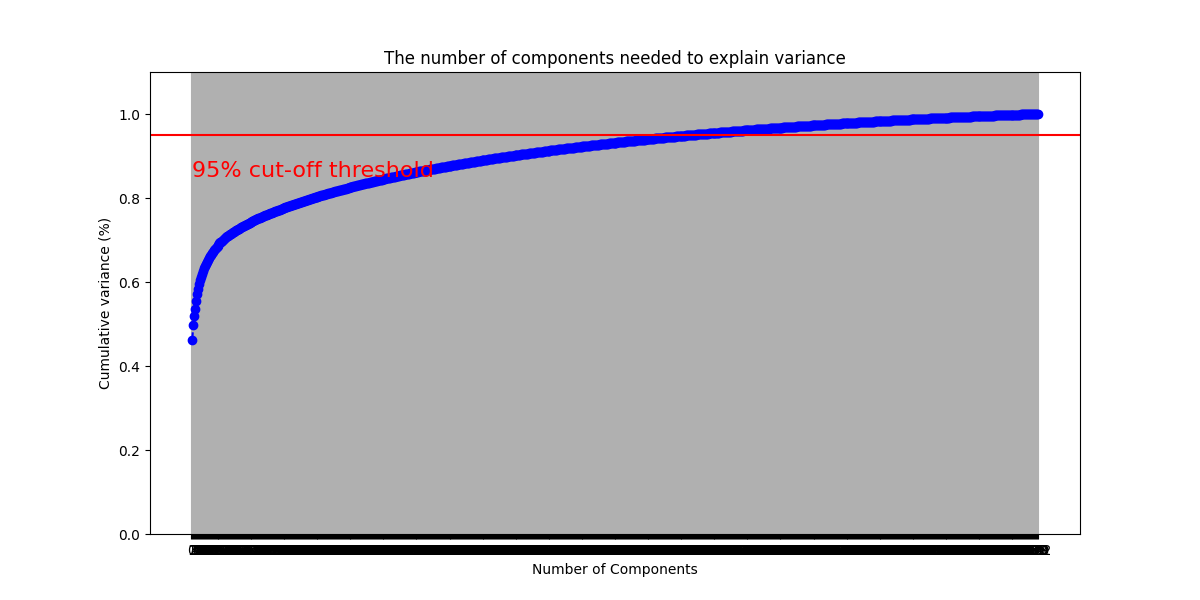
Regarding feature engineering, first we assumed that we did the shifting as described in feature selection. Moreover, we applied polynomial feature method to perform more feature engineering. The way we did this using the interaction only argument. When it is false it will triple the data and when it is true, we don’t do anything since the dataset will become much bigger.



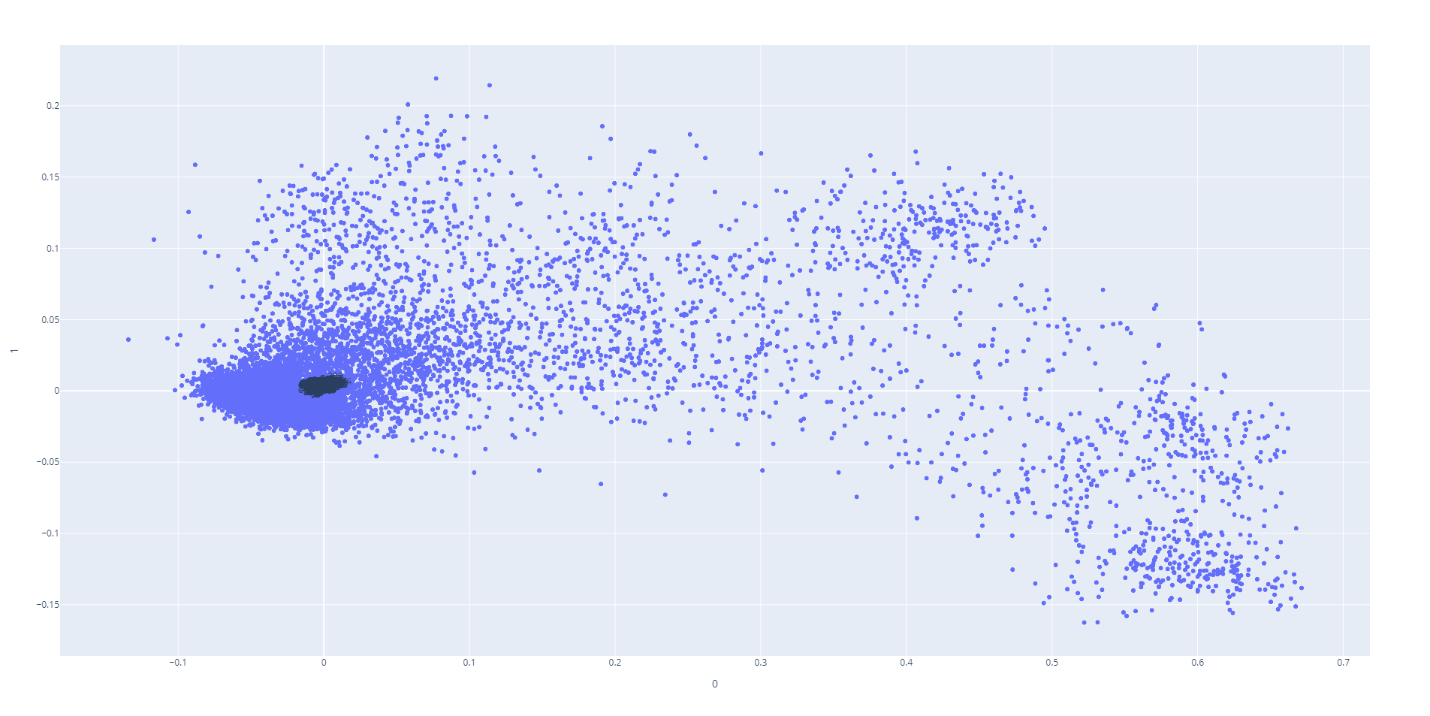
**Part 5: Dimensionality Reduction**

For dimensionality reduction, we used the well know PCA method. We apply PCA on the numeric features using MinMaxScalar function. PCA calculate linear combinations of your variables in such way that they get "summarized" in fewer variables. PCA help in understanding the level of redundancy present in multiple observations, so the power of PCA is that it creates a new dataset with only the essential information. Straight forward implementation is provided.

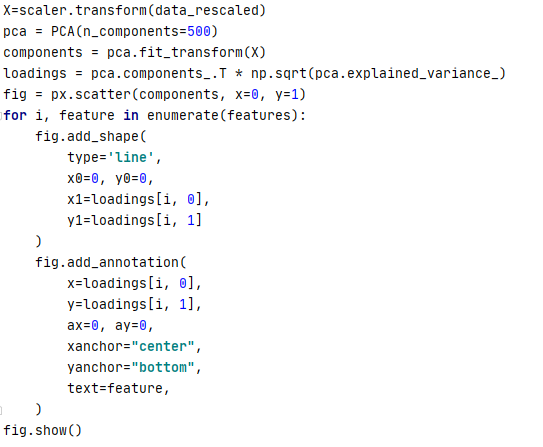


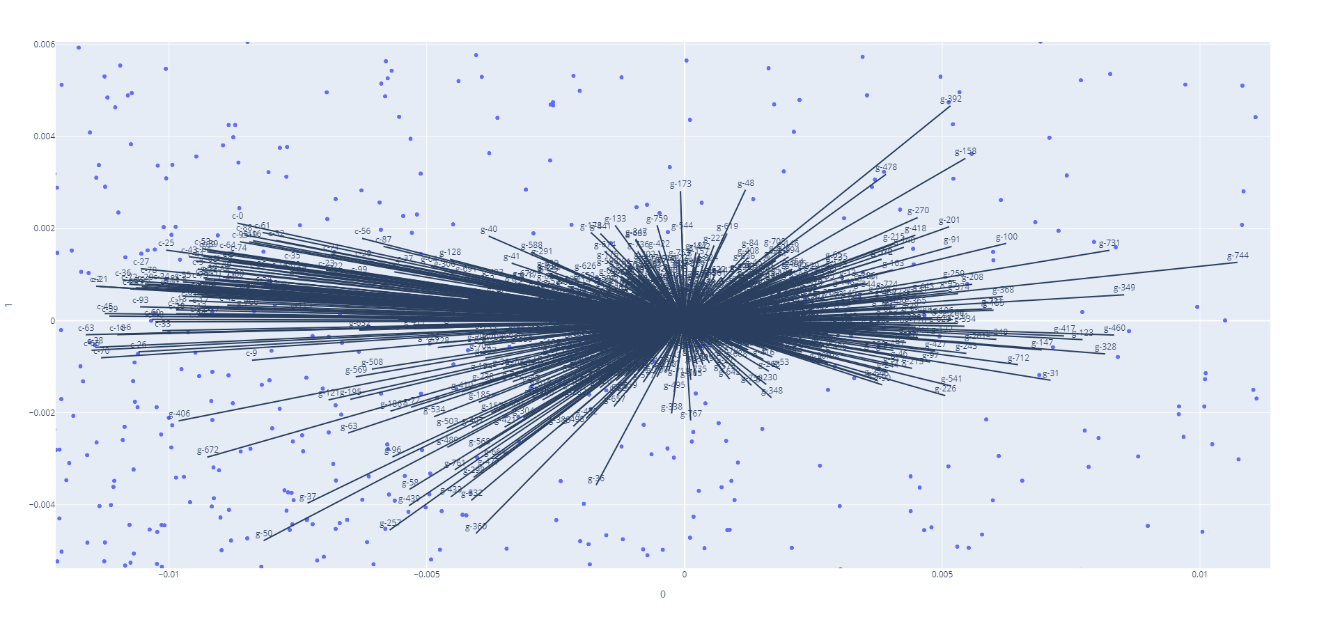


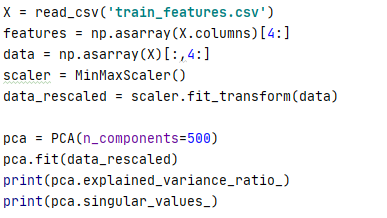
This curve quantifies how much of the total, 208-dimensional variance is contained within the first N components. In order to not lose a lot of information, we see that with 500 components we can approximately get 95% of the variance which still a reliable number and this would reduce the dimensionality in more than 350.In addition.



In this picture we see individual samples as well as vectors that correspond to the loadings. Loading describe how much each variable contributes to a particular principal component. Large loadings indicate that a particular variable has a strong relationship to a principal component. The axes are arrows originating from the center point. In here we can visualize how the data points relate to the axes. Although it is not very informative without knowing which point corresponds to which MOA since we have multi-label classification and it is hard to visualize it, as general we see that some feature relates more than the others.



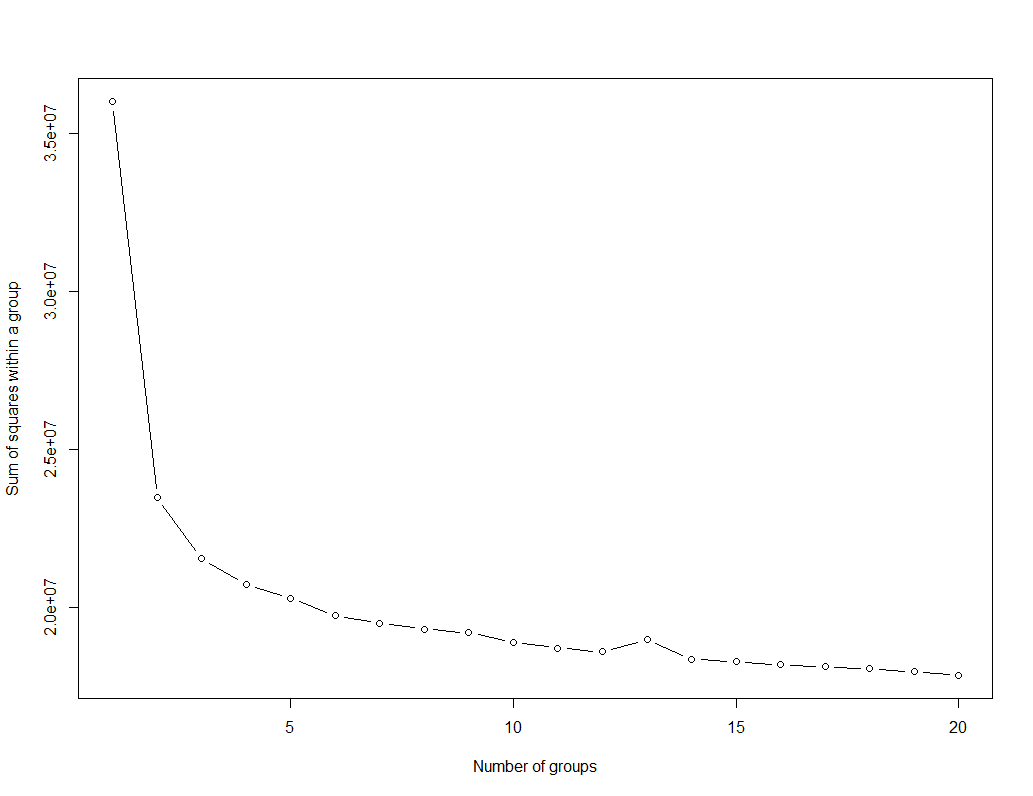




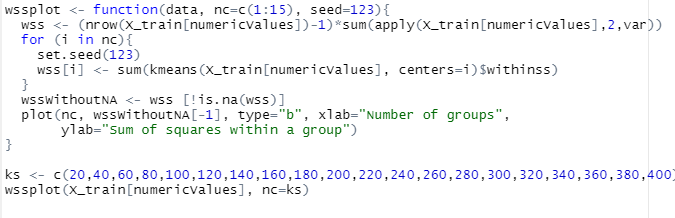
**Part 6: EDA**

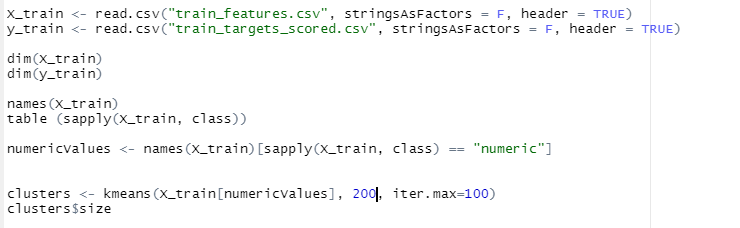
We used EDA to check the uniqueness and type of the categorical data. We also draw some charts, plots, calculated variance to check dependencies. Comments and interpretation are available with the code

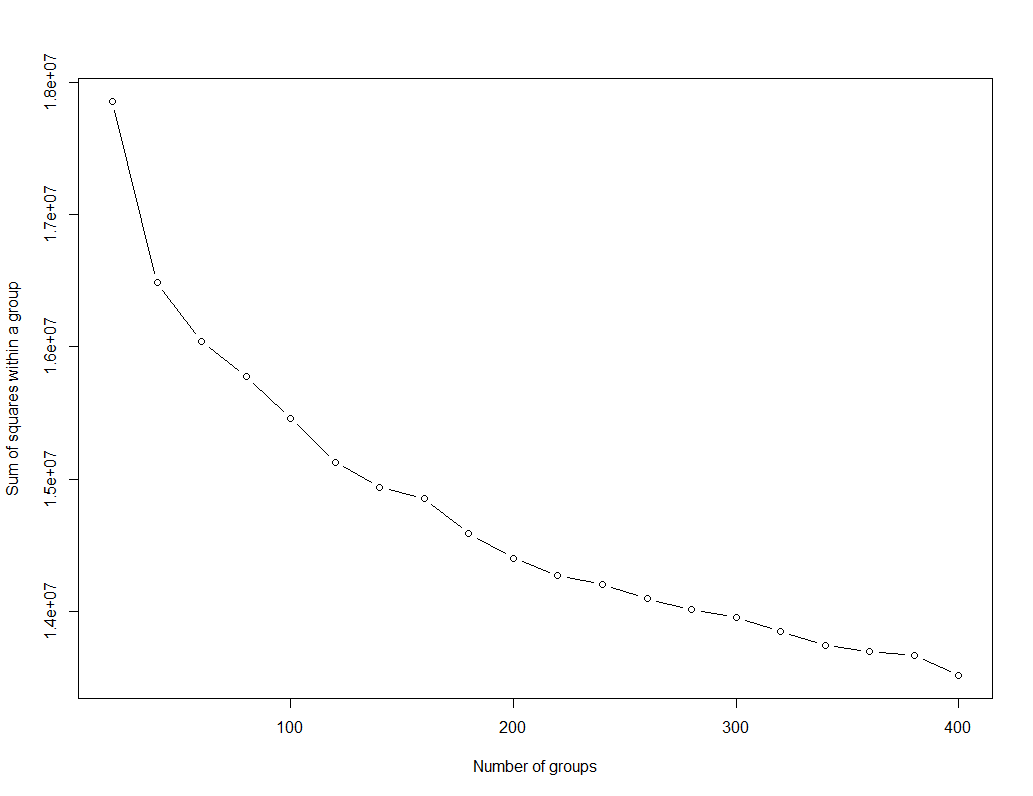
**Part 7: Pattern Analysis Using kmeans Clustering**



The main idea behind cluster partitioning methods k-means is to define clusters such that the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square) is minimized. The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible. This method is called the elbow method and we applied it first at small number of cluster 1-20, but since our data is multi-label classification, we applied it on high number of clusters that vary between 20 and 400 since the number of features is around 208 and, in our data, so far very few have multiple labels. It seems that the best result to choose the elbow which is around 200 clusters.

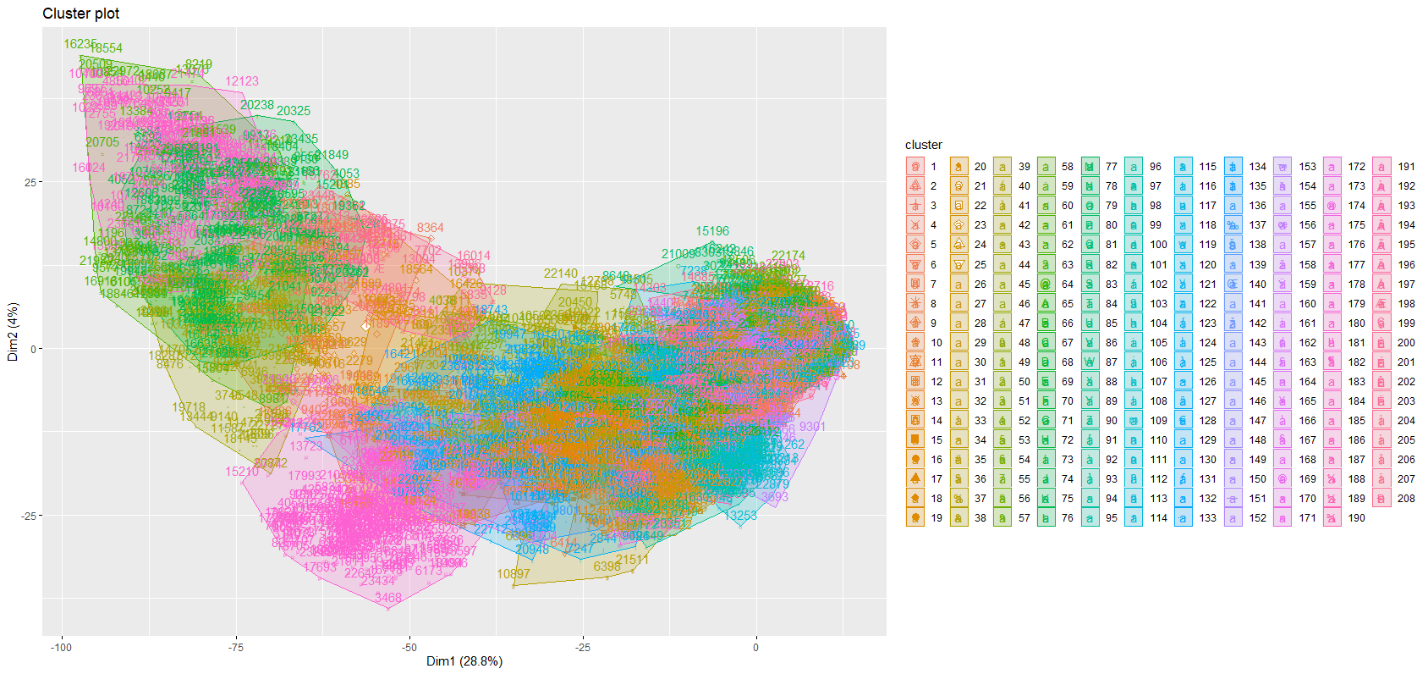






We can also view our results by using fviz\_cluster. This provides a nice illustration of the clusters. If there are more than two dimensions (variables) fviz\_cluster will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain the majority of the variance.





One last remarque is that the data we gave is after feature selection that we did (using f scores). We did not have time and resources to share it every time we apply a modification on the dataset since it’s big and we’re working remotely (using version control) and the internet is not at the best. However, we do have local versions of each modification we applied and every modification is apparent in the code and can be run and tested