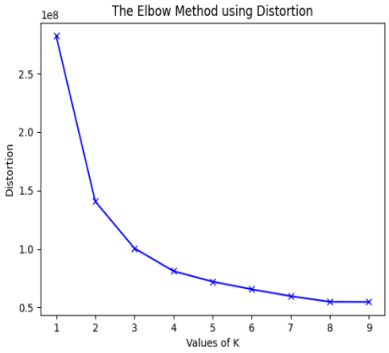
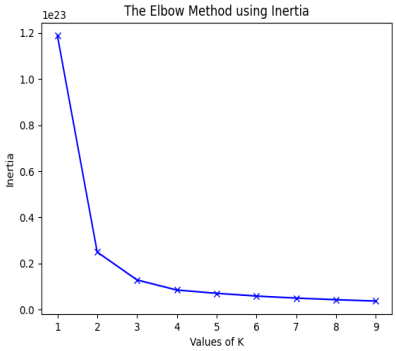
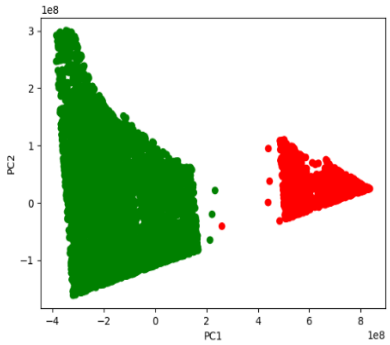
REPORT OF EX.1

In this work in part 1 of the exercise, I used every single numeric data to evaluate the number of clusters appropriate for $steam\_reviews$ except strings. However, I mapped True or False to 0/1 and then considered them as numeric data. Obviously, some of these data are not useful but in order to check the output and performance and compare it with part two, I included everything. plus I needed to handle missing data(NA) with a few lines of code .in this part I found k=2 and clusters are shown as below. (Note the following figures are for the first 1,000,000 rows of the dataset)



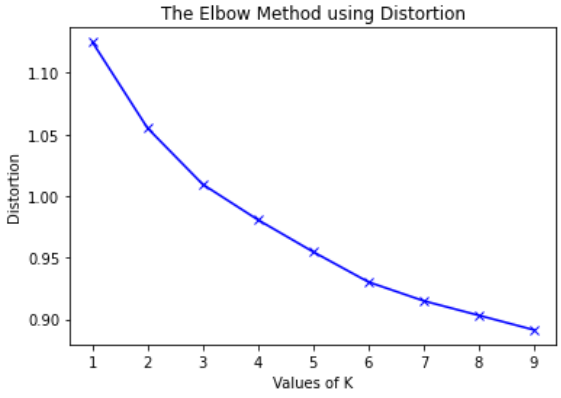


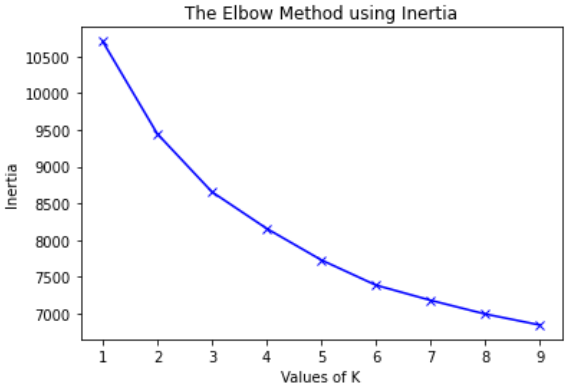


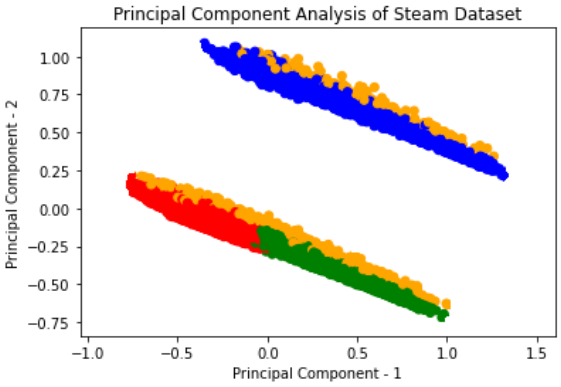
In part 2 we are asked to use feature engineering techniques. I took into account the following steps to gain informative, useful, and discriminant data. Note that though I have written these in order, sometimes I have done these in code alongside another step, but this order mainly means that I encountered these issues first and I felt the need of handling the problem I was facing during debugging procedure.

1. Firstly I needed to handle missing data as part 1 due to errors during debugging because of NA. handling missing data could be considered as a step in feature engineering and the appropriate value to be filled instead of missed data. Which by itself is an important topic to be discussed. For sake of simplicity, I filled the missing data with 0. Fortunately, there is no infinite value in the dataset
2. Secondly, I felt the need of taking into account string data and numeric data separately. String data were mainly in the review part. Another string column was app name which has the same identifier in numeric columns called app id(in final steps I will drop some columns and app name is one of those columns, language column is also useful when we want to do tokenization as input of stopword and stemmer while processing reviews
3. Another consideration was mapping True/false to 0-1 as mentioned in part one.
   if I call these steps some basic preprocessing, they are crucial in further steps since only numeric data are understandable by our models
4. Then I decided to do the feature selection. In this part, I dropped the app name since it is probably not a useful column because it has an equivalent column, app_id. #row is also another not useful column. because they are different for every instance. Some other columns have True/False information including recommended, steam_purchase, received_for_free, written_during_early_access. After dropping these columns I realized my clustering is still the same so they are irrelevant or at least have the least impact on my clustering procedure.

After processing string columns and numeric columns and concatenating them, I have now a good data frame, ready to be processed to evaluate the number of clusters for clustering the data and the actual clustering process itself using kmeans++. To find out the number of clusters I used the elbow method the same as part1 which the elbow on distortion or inertia is the approximate appropriate value of K but now using data engineering I can see the plot which I can estimate K better, and also I can see two clusters are not an appropriate number for clustering my data anymore but 4 is a better number. The next step is to visualize my many-dimensional array using principal component analysis. Before a normalization step. As it is obvious now, using feature engineering and considering several points now we can say our data is clustered better. Below are the figures obtained at the end.







Running instructions: after having the csv dataset and source code at the same path, you need to open the source code in any IDE. Then, after installing probably not installed libraries, simply run the code. I divided the procedure into 10 main steps that will be shown in the terminal. One of the heavy steps while running is when the string columns are being processed by removing spam, numbers and punctuations, stopwords, then stemming and tokenizing. At this point, this information as a data frame must be concatenated with numeric columns. Another heavy processing during running the code is when the number of K is going to be estimated using the elbow method. After this step, two figures are going to be shown. The K on the elbow for both figures is the best K for clustering our data. This value is used in kmeans model as the K parameter. Finally, the visualization of our data being clustered is shown which by itself takes time to show up.

CONCLUSION: obviously running time is less when performing the elbow method and doing PCA concerning performing on raw data(for sake of simplicity I did not include strings in my raw data, but if I would, as I did in part two, it is too time-consuming). Also, the shape of the elbow changed in a way that I can say two clusters are not the appropriate size for clustering my data. Feature engineering not only is helping me in finding a better k, but also I can have better distinct clusters

EX2:

Part 1

We know that the k-means algorithm, in general, is NP-Hard. however, in this exercise, I will prove and show an algorithm for K-means for d=1 with O(KN^2) where N is the number of points and K is the number of clusters with the use of dynamic programming.

Let's have n points called 1dimentional space sorted in ascending order called $\{x_1, x_2, \dots x_n\}$ and having K clusters called $C_1, C_2, \dots C_K$. *the goal is* to find a partition of X into nonempty disjoint clusters as the sum of squares of within-cluster distances from each element to its corresponding cluster mean is minimized. Let's call this sum as the within-cluster sum of squares($or\ M_{k=associated\ cluster}$). By the use of dynamic programming then, let's define a sub-problem as finding the minimum $M_k$of clustering *from point 1 to* i into *m* clusters. We record the corresponding minimum M_i in entry *D*[*i*, *m*] of an *n* + 1 by *k* + 1 matrix *D*.(the goal is to fill the "famous" 0-1 knapsack problem algorithm table where here K is the final weight and is increased in every step.). As we know the row and column of this matrix(table) for i=k=0 is 0. And *D*[*n*, *k*] is the minimum M_i value to the original problem.

Having this basic knowledge about the general problem now if we define *j* to be the index of the smallest number in cluster *m* in an optimal solution to *D*[*i*, *m*]. It is evident that *D*[*j* − 1, *m* − 1] must be the optimal M_i for the first *j* − 1 points in *m* − 1 cluster, for otherwise, one would have a better solution to *D*[*i*, *m*]. if we have *d*(x_j, . . . , x_i) the sum of squared distances from x_j, . . . , x_i to their mean. then we can formulate the table with all these considerations as below :

$$D[i, m]=\min_{m \le j \le i}\{D[j-1, m-1] + \mathrm{d}(x_j, \dots x_i)\} \quad where\ 1 \le i \le n\ and\ 1 \le m \le K \qquad (2-1)$$

and $D[i, m] = 0\ for\ i = m = 0$

also, it must be taken into account that

$$\mathrm{d}(x_1, \dots x_i) = \mathrm{d}(x_1, \dots x_{i-1}) + \frac{i-1}{i}(x_i - \mu_{i-1})\ where\ \mu_i = \frac{x_i + (i-1)\mu_{i-1}}{i}$$

so the algorithm for clustering one-dimensional data could be suggested as next page :

Note that iteration over B for finding the smallest number in each cluster is computed in O(K) and iteration in the matrix is O(KN^3) but computing d is done in constant time so total complexity can be reduced to O(KN^2)

Polynomial Kmean algorithm in 1d:

```
create a k+1 by n+1 matrix and call it D

init val(D)=0 for all column =row=0

sort input and call it X= {x_1,x_2,… x_n } in ascending order

for i and k < n and K do:

   fill the cells using formula (2-1)

   record the index of the smallest number in cluster m as B[I,m]

   backtrack from B[n, k] to obtain the starting and ending indices for all
   clusters     // indicator of each cluster is the smallest number in that
   cluster(each cluster can be identified by the smallest number in it)

return the smallest number in each cluster

return D[n, K] as  minimum  achievable M_i by a clustering of the given data.
```

Part 2

Generally, we know that the cost function of the k-means algorithm for a set of points in 1d array sorted in ascending order called $\{x_1, x_2, \dots x_n\}$ =X can be calculated as

Cost=$\sum_{x_i \in X} |x_i - \mu|^2 = \sum_{x_i \in X} |x_i|^2 = ||X||^2$=norm of X if μ =0

Now two points are added with distance l from μ =0 and let's call them $x_{\pm l}$.Then the cost function would change as $\sum_{x_i \in X>0} |x_i - l|^2$.

If I define $l = \frac{\sum_{x_i \in X>0} x_i}{m}$ Now the question is that for which value of l the new cost function is less than the previous one, or as the question asks as below :

$$\sum_{x_i \in X>0} |x_i - l|^2 = \sum_{x_i \in X>0} |x_i - \frac{\sum_{x_i \in X>0} x_i}{m}|^2 = \sum_{x_i \in X>0} \left(\sum_{x_i \in X>0} {x_i}^2 + \sum_{x_i \in X>0} (\frac{\sum_{x_i \in X>0} x_i}{m})^2 - \frac{2}{m} x_i \sum_{x_i \in X>0} x_i \right)$$
$$\leq 3/4 \sum_{x_i \in X>0} {x_i}^2$$

Or

$$\sum_{x_i \in X>0} \left(\sum_{x_i \in X>0} (\frac{\sum_{x_i \in X>0} x_i}{m})^2 - \frac{2}{m} x_i \sum_{x_i \in X>0} x_i \right) \geq 1/4 \sum_{x_i \in X>0} {x_i}^2$$

Which equals to

$$\sum_{\sum_{x_i \in X>0} x_i} \left(\frac{2}{m} x_i \sum_{x_i \in X>0} x_i - \frac{1}{m^2} \left(\sum_{x_i \in X>0} x_i \right)^2 \right) \geq 1/4 \sum_{x_i \in X>0} {x_i}^2$$

I can bring $\sum_{x_i \in X>0} x_i$ out of the main sigma then it will be :

$$\frac{2}{m} \left(\sum_{x_i \in X>0} x_i \right)^2 - \frac{1}{m^2} \left(\sum_{x_i \in X>0} x_i \right)^2 \geq 1/4 \sum_{x_i \in X>0} {x_i}^2$$

$$2l \left(\sum_{x_i \in X>0} x_i \right) - l^2 \geq 1/4 \sum_{x_i \in X>0} {x_i}^2$$

Now let's evaluate the position of l with on the above formula to check where it can be true let's evaluate only for points greater than 0 and not include critical point zero

$$\begin{cases} l = 0 \ \ then\ 0 \geq sum\ of\ square\ of\ some\ points \ : is\ not\ acceptable (if\ we\ consider\ excludeing\ zero) \\ l = 1/2 \\ l = 1 \\ l = ? \\ l \to \infty : \ is\ not\ acceptable : \ speed\ of\ increasing\ l^2\ would\ be\ more\ negative\ than\ sum\ of\ square\ of\ some\ positive \end{cases}$$

We want to show for which value of l this is true. Obviously, we can't take too large values

But obviously, if I want to increase my cluster from 1 to 2 the best cluster would be the one equal to $l = mean$ of positive points(and the same for negative points)