Big Data - Coursework 2

Introduction

Data in the real world is frequently unlabelled and can seem very random. In situations like this, unsupervised learning techniques are a good way to find underlying patterns. This project uses K-means clustering algorithm, which starts with a search using random centres and manages to separate data in groups of points based on their differences in several features. The algorithm in this case is being implemented on the Stack Overflow user base and tries to find ways in which the community is divided and explore causes for these groupings.

The primary implementation chosen for k means clustering algorithm is Hadoop MapReduce. The reason for this choice in this particular group is the fact that we have already gained substantial experience in Hadoop MapReduce. Therefore, the fact that the team is confident with Hadoop could output a much better end result compared to Spark.

Data sets

We have used different inputs from various stack overflow data set such as Users, Posts and Badges. For each data set input of 1 dimension as well as 2 dimensions have been considered.

Project Use Case

Our k means application to the Users stack overflow data set revealed age group insights of stack overflow users. While combining the ‘Reputation’ feature with the ‘Age’ feature the algorithm gave more specific information relating to which age groups are more reputable in the stack overflow community. In addition, applying K-Means on the Posts data set gave us intuition on the post quality posted on stack overflow by taking into account the score in relation with the post views. Finally, by joining the Users data set with the Badges data set information was revealed regarding the competence of different age group users.

Therefore, these clustering applications could potentially assist data analysts, researchers and students to gain valuable insights on stack overflow users, posts and badges.

Data Pre-Processing

This is a crucial step and has been adopted to enhance the clustering quality. Data preprocessing concepts such as calculating minimum, maximum have been used to aid data normalisation (see Normalisation section).

Using the Map Reduce Framework minimum and maximum aggregations where calculated for a selected set features of the following Stack Overflow data sets:

- Users
- Posts
- Badges

Normalisation

It is a process used to standardise the selected features of a particular data set and give equal weight during the distance calculation between points in k-means. This will guarantee unit variance of each feature in the input data set (Deepali Virmani).

K-Means can generate effective cluster groupings if normalisation is applied to the input data set especially when multiple dimensions are involved and ranges of values vary widely.

K means algorithm uses Euclidean distance, that is affected by irregularities in the size of features (Vaishali Rajeev Patel, 2011) and makes K-Means sensitive. Therefore, the range of all features should be normalised so that each feature contributes proportionately to the Euclidean distance (Juszczak & Dui, 2002).

We have used Min-Max normalisation method in our K-means algorithm that performs a linear transformation on the data (Deepali Virmani). In this way, the range of features is rescaled to the standardised range of [0,1]. The formula for the Min-Max normalisation method used can be seen in the figure below.

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Figure 1. Min-Max Normalisation Formula

Choosing K

K represents the number of clusters(centroids) that are gonna be shaped in our dataset. There are many approaches to choosing a K:

1. Visualising the distribution of data and deciding in how many clusters the data can be divided to.
2. Given a visualisation of data we can 'force' the data to be clustered in our desired resolution.
3. Trial and error, trying a number of Ks and deciding after visualising the resulting clusters which K satisfies us.

Implementation – K means with Map reduce

1-Dimensional Clustering

Frequency of Ages Clustering

JAVA Package: 1D Clustering
Input data set from StackOverflow: Users
Dimensions: Age
No of Cluster: 3, because we want the dataset to be clustered into 3 age groups
Clusters: Young, Middle Age, Old

Driver

Iterate through multiple jobs until centroids converge

- Configure job by setting Mapper that classifies tuples, Reducer that calculates means, Input (Users dataset), and Output (new centroids).
- At first iteration, set some values manually as initial centroids and set them accessible to the job configuration
- In every other iteration, get centroids from previous job and set them accessible to the job configuration
- If/When job is completed, compare old centroids with new ones to see if they have converged

If/When centroids converge, chain new job

- Configure job by setting Mapper that classifies tuples, Reducer that accumulates the tuples as Text, Input, Output (new centroids with their tuples) etc.
- Set converged centroids accessible to job configuration to classify tuples

Mapper

-At setup, access centroids set from the job configuration and store them to private fields of the Mapper class.
-At mapping, get data from dataset, transform it from XML and store it to the Map data structure, get relevant features, and classify the tuple of features (in our example, age) to its nearest centroid (from set of centroids created at setup).
-Write/Emit (Key: Nearest Centroid, Value: Tuple of features).

Reducer

We have two reducers for different outputs:
First Reducer (calculating new centroids, used for converging jobs):
-Get key (centroid) with its values (tuples) and calculate the mean of values
-Write/Emit (Key: Mean of Values, Value: Null)
Second Reducer (classifying tuples):
-Get key (centroid) with its values (tuples) and Write/Emit (Key: Converged Centroid, Value: Text of accumulated values)

Data Demonstration

The data used bellow is a distribution of age in the Stack Overflow dataset. This dataset is being used as input in the MapReduce program.

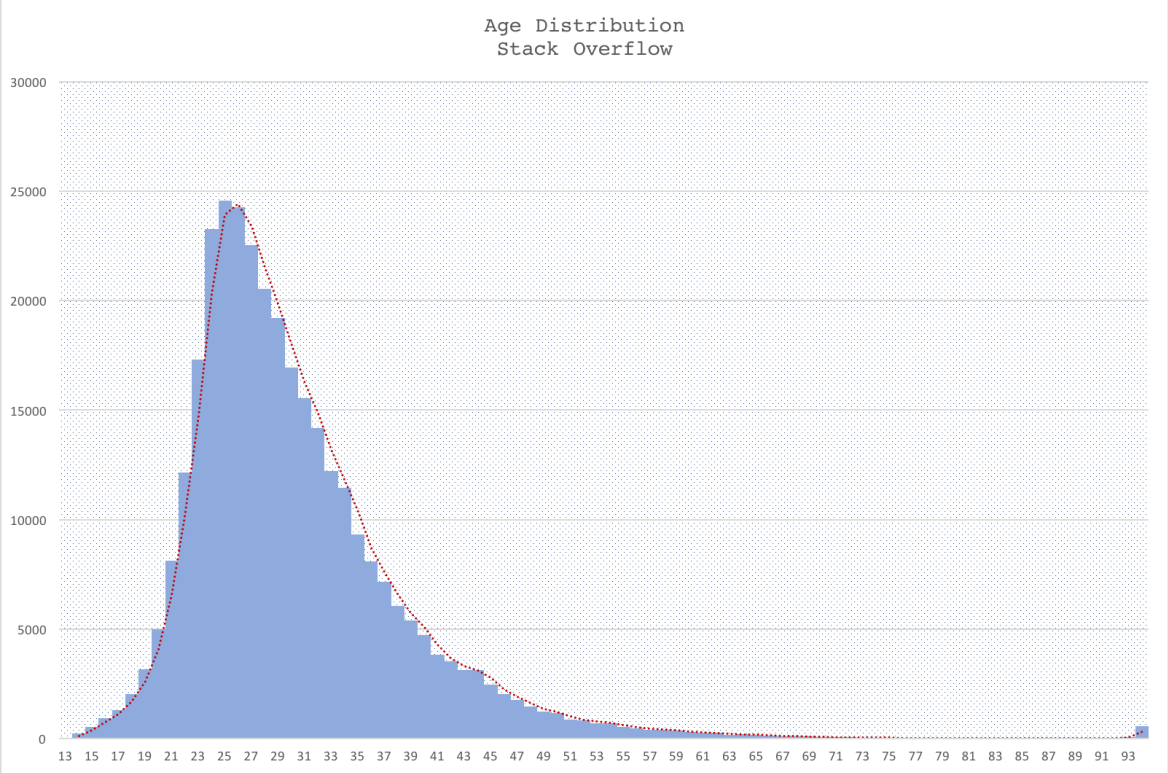

Figure 2

The way the code works is that it adds initial random centroids to help with finding K-means. We can see in Figure 3 how the new centroids are arranged in their relevant clusters.

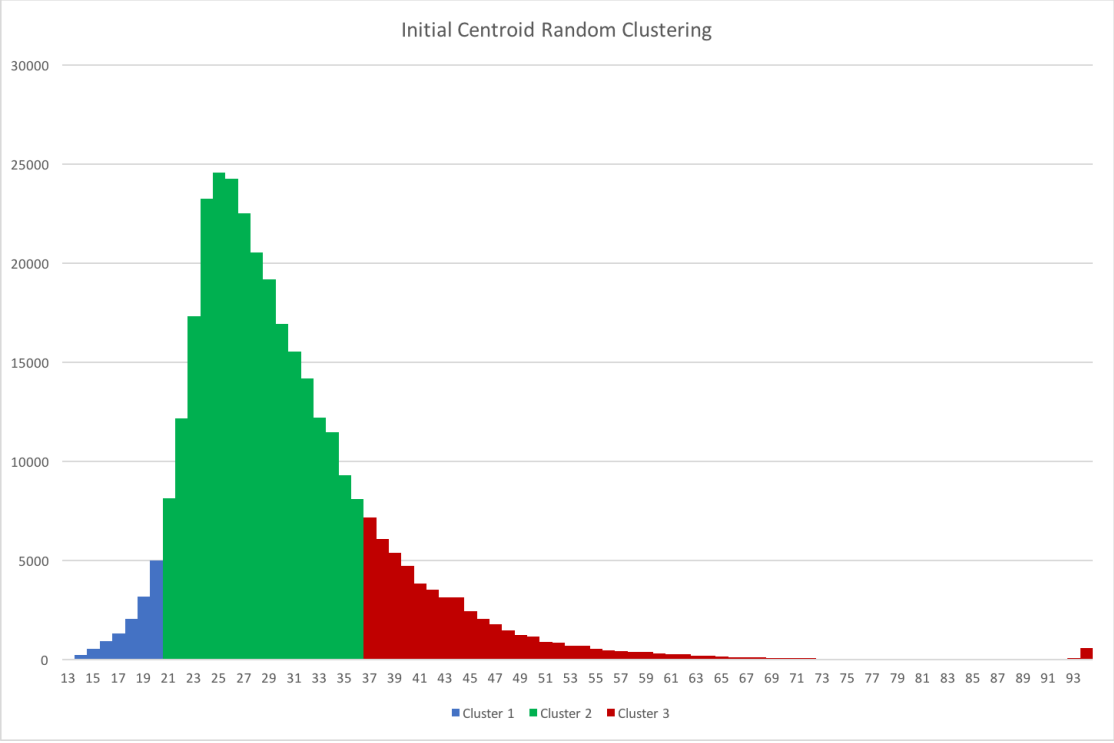

Figure 3

Furthermore, in Figure 4 it the graph shows the clusters after just 2 iterations of the K-means algorithm.

Figure 4

Finally, the algorithm stops the iterations when the centroids converge and the distance for the oder centroids is less that 0.1 points away.

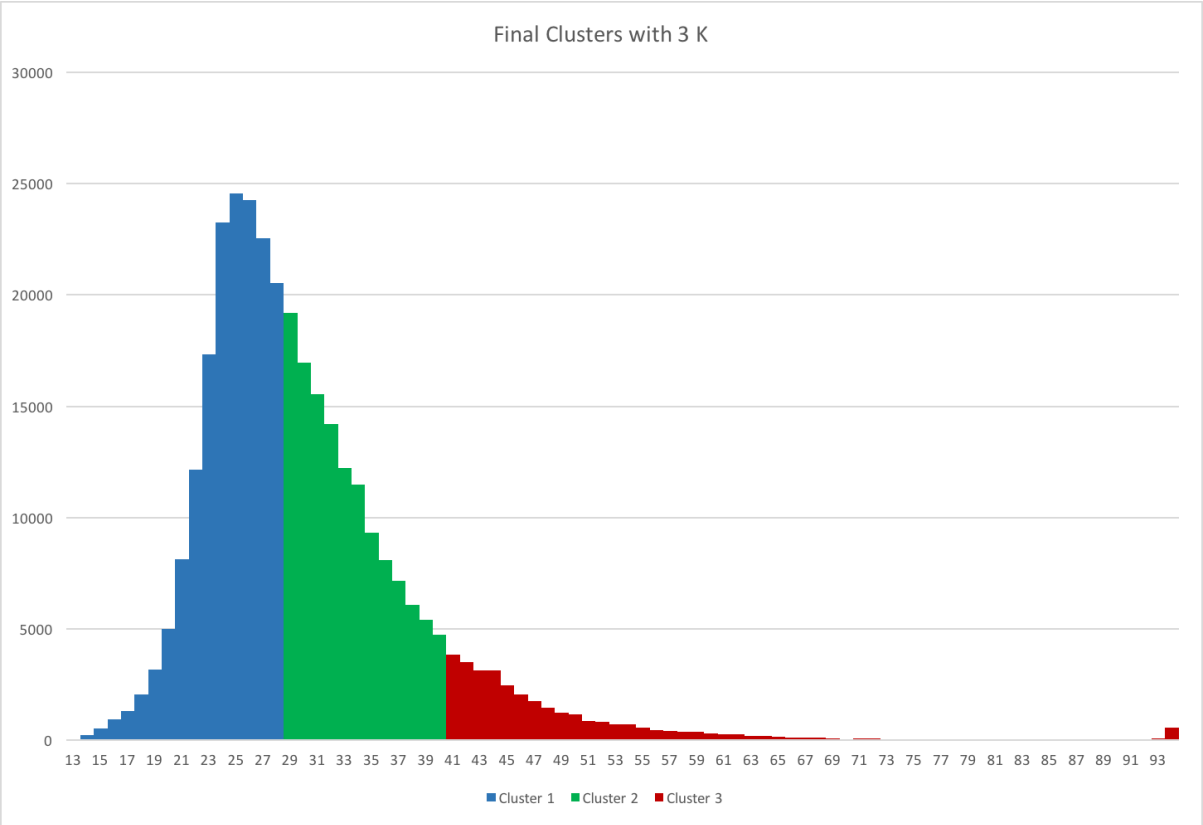

Figure 5

2-Dimensional Clustering

User Age Clustering

JAVA Package: 2D Clustering
Input data set from StackOverflow: Users
Dimensions: Age, Reputation
No of Cluster: 3, because we want the dataset to be clustered into 3 age groups
Clusters: Users with bad reputation, Users with average reputation, Users with good reputation
Clusters: Users with bad reputation, Users with average reputation, Users with good reputation

Data Pre-Processing

A map reduce job (see DataPreProcessing.java) was utilised to calculate minimum and maximum of the following features:

- Age
- Reputation

Feature Scaling – Normalisation

Age values ranged from [0-90] while Reputation values ranged from [0-60000]. However, this unequal range between those features caused the algorithm to give more weight to the Reputation feature which led to inaccurate clusters. To standardise the ranges of the features and give them equal weight data normalisation was applied to convert Age and Reputation data points to a common range of [0-1].

Using the minimum and maximum values gained from the DataPreprocessing.java job for each feature (i.e. Age, Reputation) the following formula was applied to aid normalisation.

$$z = \frac{x \ - \min(x)}{\max(x) \ - \min(x)}$$

Figure 1.0: Min-Max Data Normalisation formula

Driver

- As with previous Driver for 1-Dimensional but we normalise the centroids to get 'fairer' clustering and adjust initial centroids based on input dataset (Users) and its selected features (Age and Reputation).
- Configure job by setting Mapper that classifies tuples (in normalised form), Reducer that calculates means (in normalised form), Input (Users dataset), and Output (new normalised centroids).

Mapper

- As with previous Mapper for 1-Dimensional Clustering but we process data (in this example, Age and Reputation) in their normalized form
- Write/Emit (Key: Nearest Normalized Centroid, Value: Tuple of normalized features)

Reducer

- As with previous Reducer for 1-Dimensional Clustering we have two reducers for different outputs:
- First Reducer (calculating new normalized centroids, used for converging jobs):
- Get key (normalized centroid) with its values (tuples of normalized features) and calculate the mean of values
- Write/Emit (Key: Mean of normalized values, Value: Null)
- Second Reducer (classifying unormalized tuples):
- Get key (normalized centroid) with its values (tuples of unormalized features) and Write/Emit (Key: Previous Key, Value: Text of unormalized accumulated values)

Data demonstration -Results

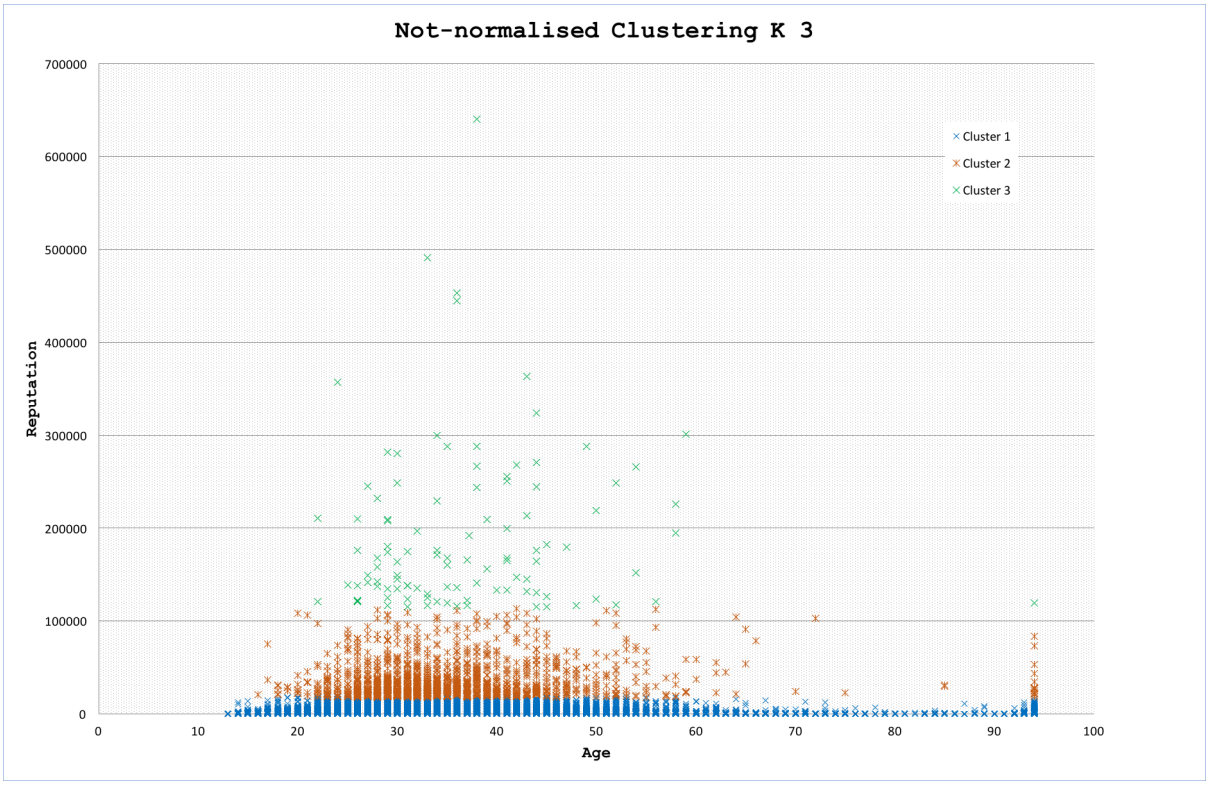

Figure 6

Since the data is not normalized the resulting clusters were not distinguishable logically. The high range of Reputation directed the clusters vertically, overlooking Age.

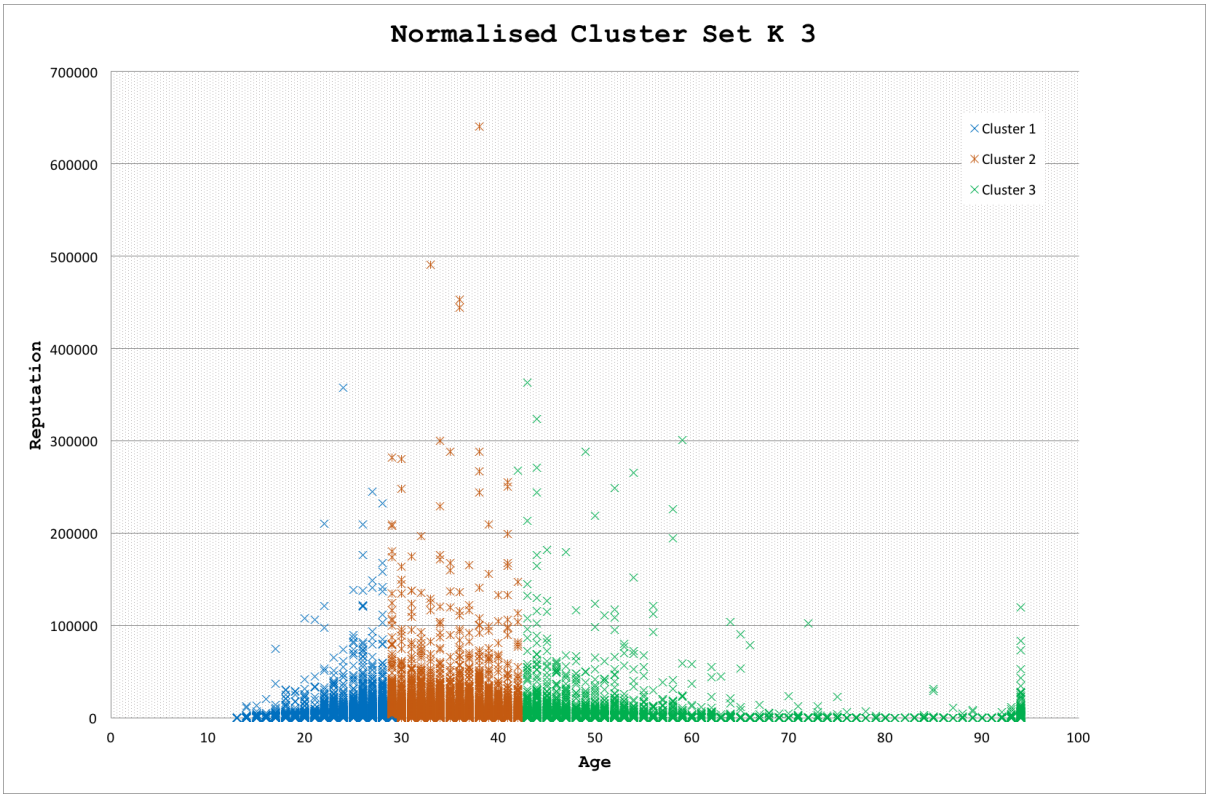

Figure 7

The resulting plot above was a product of KMeans Clustering with Normalized data and showcases a better clustered dataset. Now, Reputation and Age were equally considered during the KMeans Clustering process.

Quality/Popularity of Posts

JAVA Package: 2D Clustering
Input data set from StackOverflow: Posts
Dimensions: Score, ViewCount
No of Cluster: 3, because after visualizing a sample, we noticed that the dataset could be clustered to 3 classes
Clusters: Bad Posts, Moderate Posts, Very Good Posts

Data Pre-Processing

A map reduce job (see DataPreProcessing.java) was utilized to calculate minimum and maximum of the following features:

- Score
- ViewCount

Feature Scaling – Normalization

Score values ranged from [0-3000] while ViewCount values ranged from [0-1000000]. However, this unequal range between those features caused the algorithm to give more weight to the ViewCount feature which led to inaccurate clusters. To standardize the ranges of the features and give them equal weight data normalization was applied to convert Score and ViewCount data points to a common range of [0-1].

Using the minimum and maximum values gained from the DataPreprocessing.java job for each feature (i.e. Score, ViewCount) the following formula was applied to aid normalization.

$$z = \frac{x \text{ - } \min(x)}{\max(x) \text{ - } \min(x)}$$

Figure 1.0: Min-Max Data Normalization formula

Driver
-Same as previous 2D clustering example but with centroids adjusted based on input dataset (Posts) and its selected features (Score and View Count).

Mapper
-Same as previous 2D clustering example but getting the selected features.

Reducer
-Same as previous 2D clustering example.
Data Demonstration

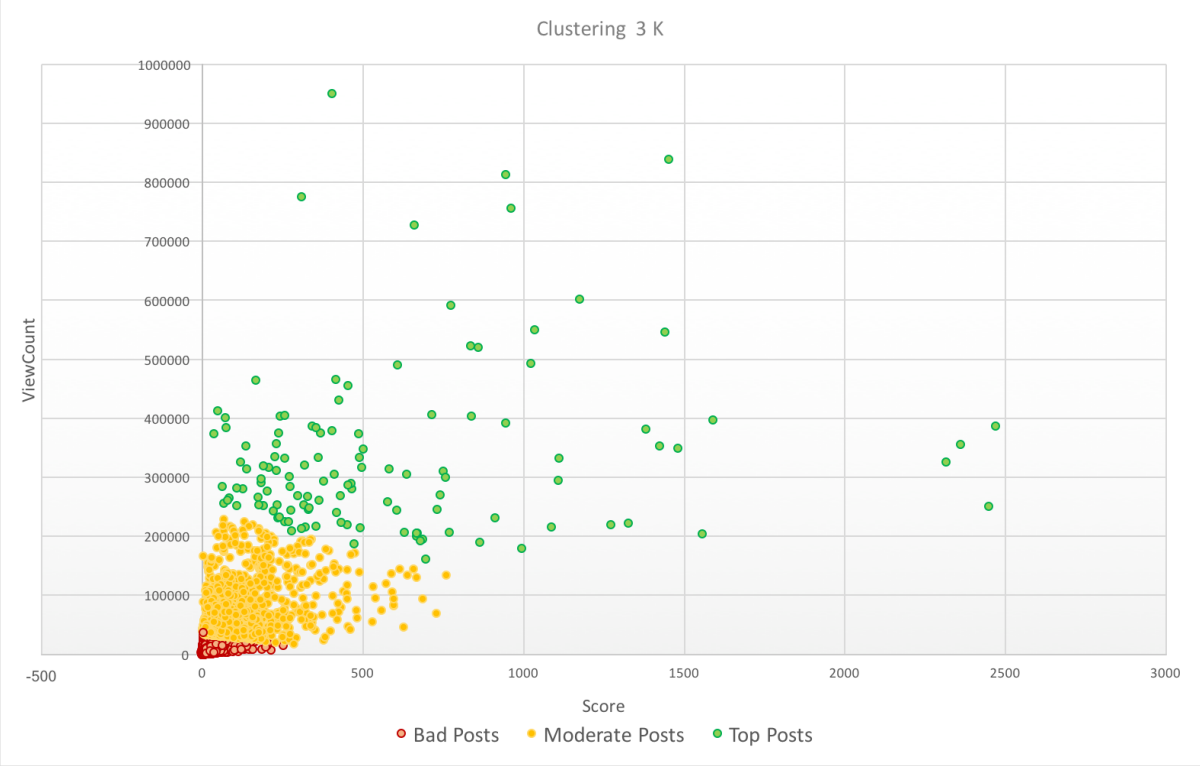

Figure 8

The above plot is a sample of the Posts dataset including only 100 000 records instead of 13 million that were considered during the KMeans Clustering process. We can distinguish 3 classes in the plot: Bad Posts, Moderate Posts and Top Posts.

Badges per Age Clustering

JAVA Package: Name From Athan
Input data set from StackOverflow: Users, Badges
Dimensions: Age, Number of Badges (per Age)
No of Cluster: 3
Clusters: Low competent age group, Moderate competent age group, Very Competent age Group

Data Pre-Processing
A join map reduce job was used to find count the Badges per User Age.
Later, a map reduce job (see DataPreProcessing.java) was utilized to calculate minimum and maximum of the following features:

- Age
- Number of Badges per Age

Feature Scaling – Normalization

Age values ranged from [0-90] while Number of Badges per Age values ranged from [0-XXX]. However, this unequal range between those features caused the algorithm to give more weight to the Number of Badges per Age feature which led to inaccurate clusters. To standardize the ranges of the features and give them equal weight data normalization was applied to convert Age and Number of Badges per Age data points to a common range of [0-1].
Using the minimum and maximum values gained from the DataPreprocessing.java job for each feature (i.e. Age, Number of Badges per Age) the following formula was applied to aid normalization.

$$z = \frac{x \text{ - } \min(x)}{\max(x) \text{ - } \min(x)}$$

Figure 1.0: Min-Max Data Normalization formula

Driver
-Same with 2D Clustering's Driver.

Mapper
-Like 2D Clustering's Mapper but split the values coming as data in Mapper on tab (\t) to construct the tuple of the selected features(Age and Number of Badges from Aggregation of User and Badges).

Reducer
-Same with 2D Clustering's Reducer.

Data Demonstration

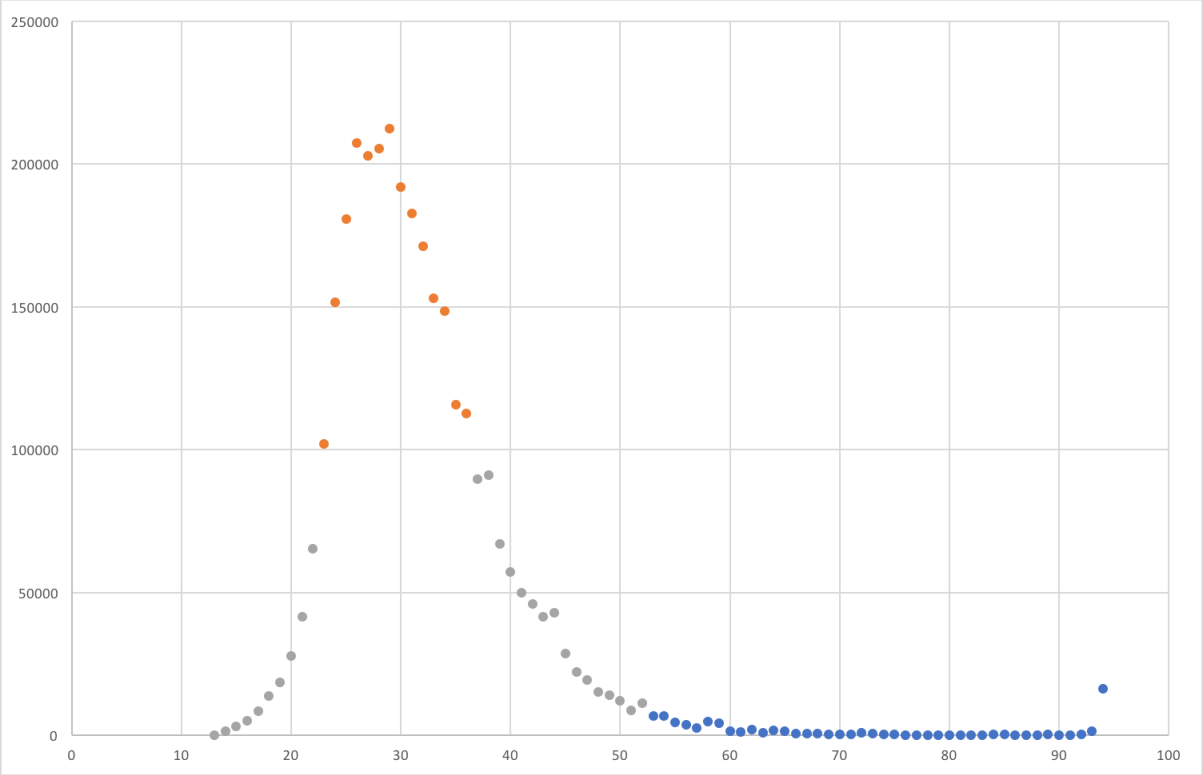

Figure 9.

As we can from the above plot the old users did not have many badges, the age ranges of 10-20 and 35-50 had medium-high quantity of badges and the 20-35 age range had very high amounts of badges.

Extra Features Included:

- Exploration and discussion of hyperparameter tuning (e.g. the number of k groups to cluster the data into) [up to 10 marks]
- Bringing in additional datasets from stackoverflow, such as user badges, to aid in clustering [up to 5 marks]
- Cluster additional datasets (such as posts) [up to 10 marks]
- Delivered on initial deadline.

Bibliography

Deepali Virmani, S. T. (n.d.). *Normalisation based K means Clustering Algorithm.* New Delhi: Pashuram Institute of Technology.

Juszczak, P. D., & Dui, R. P. (2002). *Feature Scaling in support vector data descriptions.* Prox 8th Conf. Adv. School of Computing and Imaging.

Vaishali Rajeev Patel, R. G. (2011). *Performance Analysis of MK-means Clustering Algorithm with Normailisation Approach.* World Congress on Information and Communication Technologies.