Sanchari Chowdhuri
Sohan Shah
Jung Kyu Rhys Lim

MILESTONE 2 – REPORT

RUMOR PREDICTION ON TWITTER

DATA CLEANING

We had our entire dataset cleaned for milestone 2 and there was no additional cleaning required. However, we had to add/remove classes from our dataset to implement binary and multiclass SVMs successfully. Here are the changes we made to each of our dataset in every question –

RQ1: Created 4 classes in our dataset so that we could implement multiclass SVM.

RQ2: There were already 3 classes – 'andy', 'male' and 'female' in the dataset. To implement binary class SVM, we converted the classes into just 'andy' and 'not-andy'.

RQ3: There were already 4 classes in the dataset and we converted them into two classes to implement binary class SVM. The two classes we used were 'city' and 'not-city'.

Lastly, we had to take samples of our dataset in two occasions – 1) To successfully run a neural network, 2) To represent dendrograms and check the result based on samples of the actual data.

These data cleaning/manipulation approaches are explained in detail as they are implement in each research question.

RQ1: Predicted Whether Tweets Are Rumors or Not

SVMs (BINARY CASE) –

The first implementation is using the Vanilladot kernel. Here is some basic information about the model after we trained the classifier –

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 8135

Objective Function Value : -8130.309
Training error : 0.443908
```

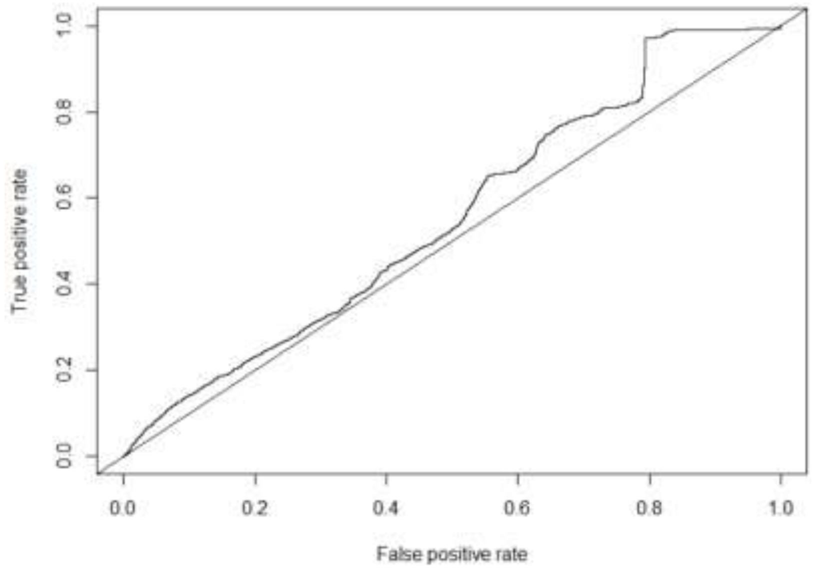
The following is the confusion matrix resulting from this classification –

```
rumor_predictions Non-Rumor Rumor
        Non-Rumor       180    23
        Rumor           925  1046
```

As you can see above, majority of the rumors are classified correctly. However, most of the non-rumors are incorrectly classified as rumors. This reflects on the overall accuracy as seen below –

```
FALSE  TRUE                   FALSE         TRUE
  948  1226            0.4360625575 0.5639374425
```

ROCR Curve:



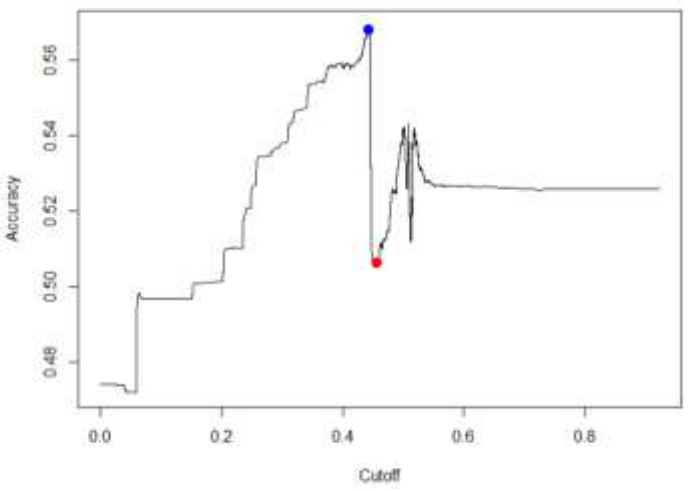
Ideally, we would want the false positive rate to be as low as possible and the corresponding true positive rates to be high. However, at the least all the points should be above the ab line as seen in the plot above. We can see that as the true positive rate is increasing, the false positive rates increase as well.

Performance Measures –

Sensitivity - 0.6507659492

Specificity - 0.4460381319

Plot of Accuracy vs Cutoff –



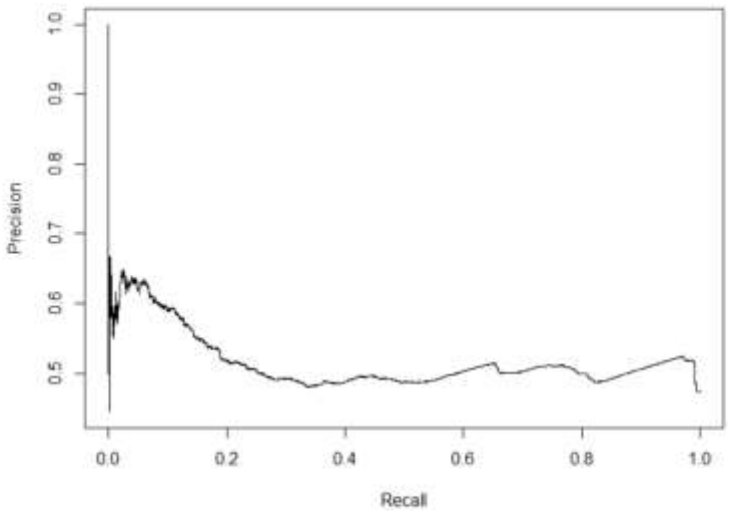
The red dot marks the cutoff value and the blue dot marks the accuracy. Computing it mathematically in R, we get the same results –

Cutoff - 0.4453048731

Accuracy - 0.5690638220

Just to test further, we calculated accuracy using Area Under the Curve and the value was 0.5547698031.

Plot of Precision vs Recall –



There is always a tradeoff between precision and recall. It is hard to get both high precision and recall. The values for precision, recall and F-measure and listed below –

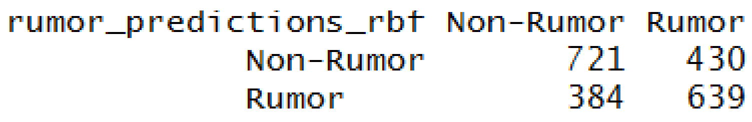
Precision – 0.8866995074,

Recall - 0.1628959276

F1 Measure - 0.2752293578 [2 * Precision * Recall / (Precision + Recall)

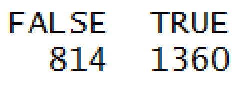
RBF Kernel

We then tried to improve the model using the RBF Kernel. The confusion matrix is below –

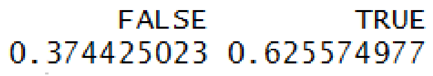


```
rumor_predictions_rbf Non-Rumor Rumor
            Non-Rumor       721   430
            Rumor           384   639
```

Clearly, the model has improved the performance because non-rumors are now being more accurately predicted. Majority of rumors are also predicted correctly. This reflected in the overall accuracy values –



```
FALSE  TRUE
  814  1360
```



```
      FALSE        TRUE
0.374425023 0.625574977
```

Performance Measures –

The values of precision, recall and F-measure are as follows –

Precision - 0.6264118158

Recall - 0.6524886878

F1 Measure - 0.6391843972

<u>SVMs (MULTICLASS CASE)</u> –

The dataset only contains two labels – Rumors and Non-Rumors. Multiclass classification does not apply in such a case. To be able to implement the multiclass case as well, I divided each rumor and non-rumor into two categories. It would be interesting to know whether the URLs in tweets matter in the case of rumors. If a URL is present in a tweet that is a rumor, the user is referencing an existing article on the internet and spreading the rumor. This gets further cascaded when others retweet such information. Hence, I divided the dataset into the following four categories –

Rumor Tweets With URL

Rumor Tweets Without URL

Non-Rumor Tweets With URL

Non-Rumor Tweets Without URL

The first implementation is using the Vanilladot kernel. Here is some basic information about the model after we train it –

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1
```

```
Linear (vanilla) kernel function.

Number of Support Vectors : 7464

Objective Function Value : -3439.353 -5022.721 -1088 -3217.944 -1088 -1087.63
9
Training error : 0.537816
```

The following is the confusion matrix resulting from this classification –

```
rumor_predictions            Non-Rumor Tweet With URL Non-Rumor Tweet Without URL
  Non-Rumor Tweet With URL                         10                          31
  Non-Rumor Tweet Without URL                      33                         105
  Rumor Tweet With URL                            557                         369
  Rumor Tweet Without URL                           0                           0

rumor_predictions            Rumor Tweet With URL Rumor Tweet Without URL
  Non-Rumor Tweet With URL                      2                       1
  Non-Rumor Tweet Without URL                   0                      15
  Rumor Tweet With URL                        898                     153
  Rumor Tweet Without URL                       0                       0
```

Again, the rumor part is predicted better than the non-rumor part. Since the non-rumors are still not predicted correctly, accuracy is hit.

```
FALSE  TRUE                 FALSE         TRUE
 1161  1013          0.5340386385 0.4659613615
```

Performance Measures –

Recall for label – 'Rumor Tweet With URL' = 0.997

Out of all the times this label should have been predicted, 99% of the labels were correctly identified.

Precision for lable – 'Rumor Tweet With URL' = 0.454

Out of the times this label was predicted, 45% of the time the classifier was correct.

F Measure = 0.624

Similarly, the recall, precision and F-measure are calculated for the other 3 labels.

RBF Kernel

Now we try and improve the model using the RBF Kernel. The confusion matrix is below –

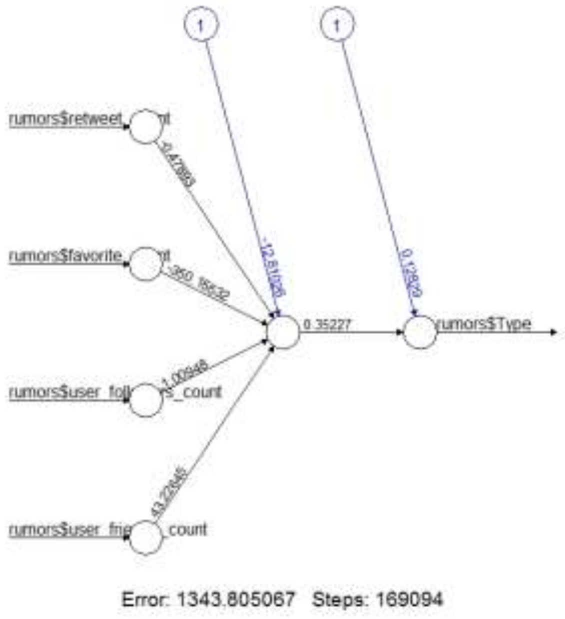
```
rumor_predictions_rbf        Non-Rumor Tweet With URL Non-Rumor Tweet Without URL
  Non-Rumor Tweet With URL                        156                          55
  Non-Rumor Tweet Without URL                      15                         151
  Rumor Tweet With URL                            427                         294
  Rumor Tweet Without URL                           2                           5

rumor_predictions_rbf        Rumor Tweet With URL Rumor Tweet Without URL
  Non-Rumor Tweet With URL                     78                      61
  Non-Rumor Tweet Without URL                   8                       3
  Rumor Tweet With URL                        811                      88
  Rumor Tweet Without URL                       3                      17
```

The model has improved because it now predicts non-rumors better than it did with the vanilladot kernel. Rumors are also predicted fairly well. The result is an increase in accuracy –

```
FALSE   TRUE                    FALSE          TRUE
 1039   1135             0.4779208832 0.5220791168
```

NEURAL NETWORKS

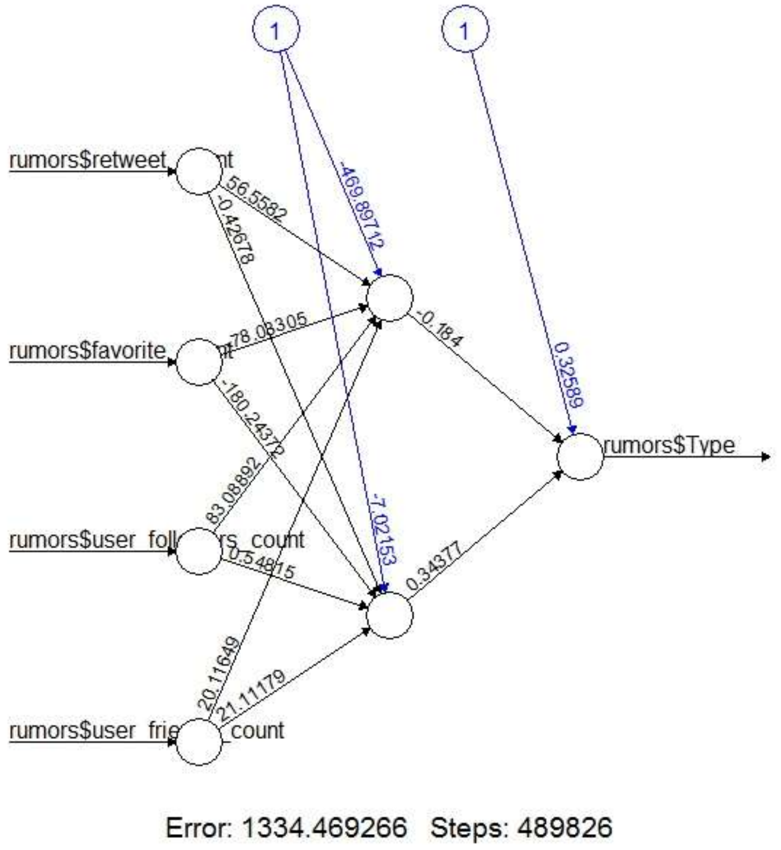
The following is a basic neural network with only 1 hidden layer and 1 node in it.



The four variables on the left side are the independent variables and are the input to the neural network. The model computes the weights which is represented on the arrows. And the node at the right end is the output.

The next node below adds another hidden node in the same layer –

rumor_model2 <- neuralnet(formula = rumors$Type~rumors$retweet_count+rumors$favorite_count+rumors$user_followers_count+rumorsuser_friends_count, data = rumor_train, **hidden = 2, stepmax = 1e6**)

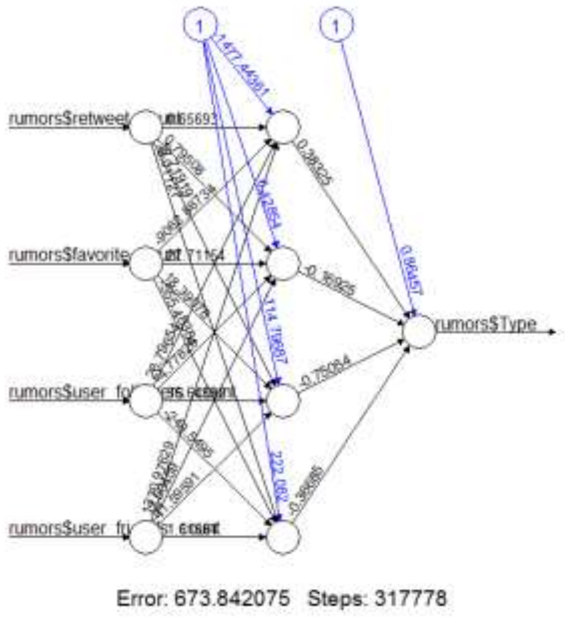


Note that we had to increase the value of stepmax to be able to create this neural network with 2 nodes in the hidden layer. 1e6 = 1 million! Which means that the model can take a maximum of 1 million steps to converge.

As we increase the value of hidden to add more layers and/or nodes, the algorithm does not converge for the given value of stepmax. We increased stepmax to 1e9 (1 billion steps!!!) and the algorithm still did not converge. We believe the reason for this is the size of the dataset. To test the neural network, we then took a random sample from our existing dataset. We computed the means, medians and variance of the different columns in the sample and they were representative of the original dataset. We could then compute neural networks for values of hidden = 3 and 4, with maximum steps = 1 billion. However, any further increment beyond hidden = 4 failed with the multiple trials we performed. The sample dataset was exactly half the size of the original dataset.

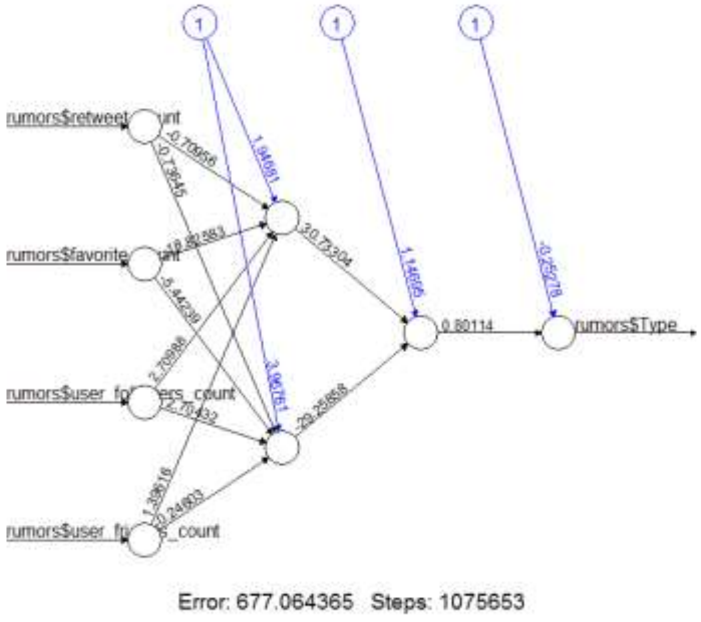
Hidden = 4:

This is the best neural network model.



Hidden = c(2,1):

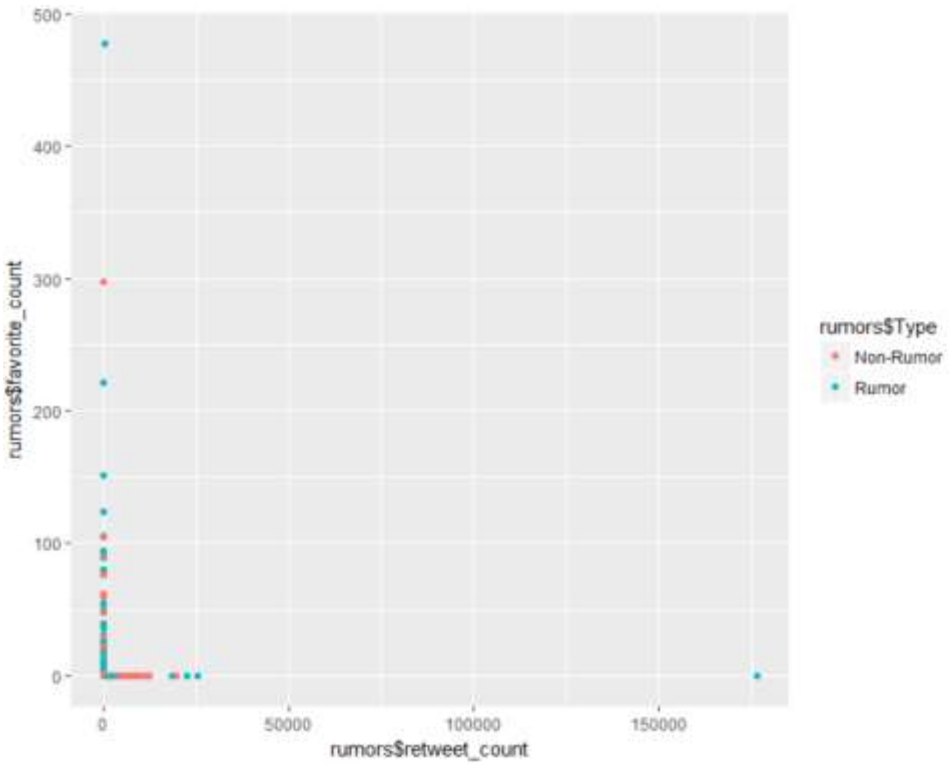
The following neural network contains 2 layers, with 2 nodes in the first layer and 1 node in the second.



The activation function in each model above is the Sigmoid Function.

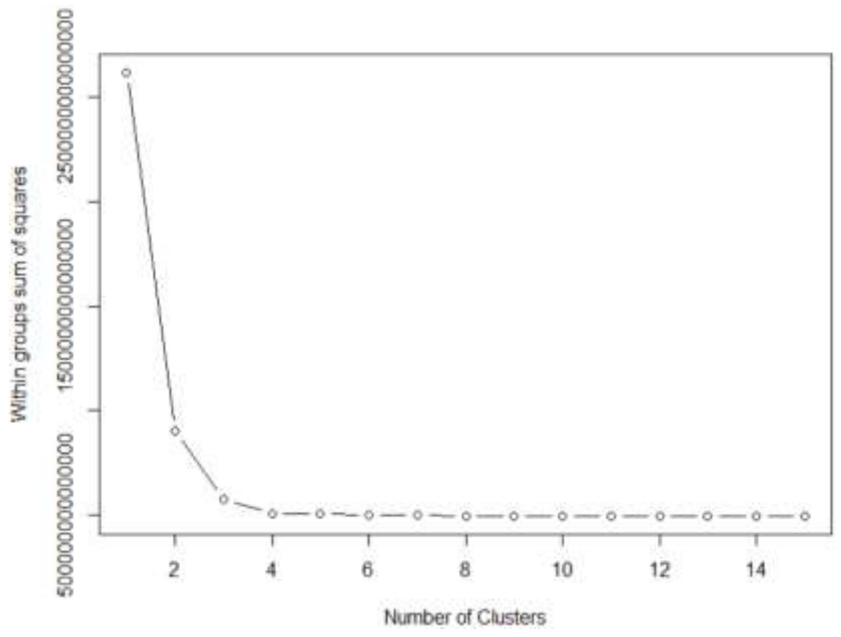
PARTIONAL CLUSTERING (K-MEANS) –

The following is a plot of our current dataset –



As you can see above, it is concentrated in one region. Also, there are a few outliers.

Although I know that I would want 2 clusters, I went ahead and plotted the graph to see what the algorithm returns as part of unsupervised clustering. The plot below shows that we can choose either 2 or 3 clusters. We are going to go ahead with 2 since we know that we want rumors and non-rumors.



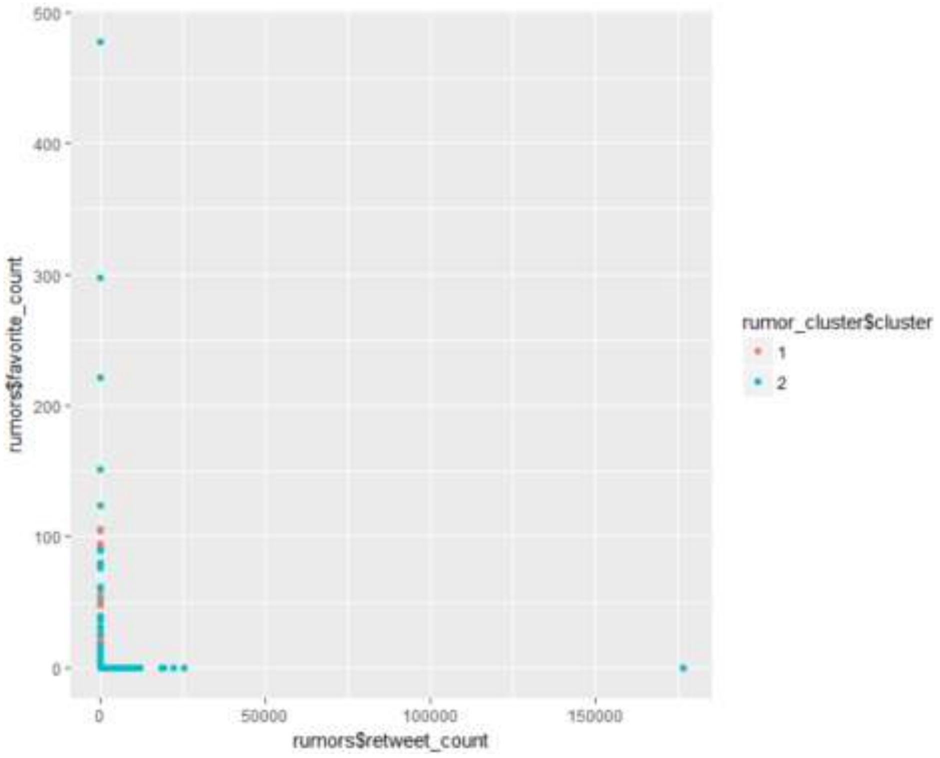
We then ask R to try 50 different random starting assignments and select the one with lowest within cluster violation. The result explains 65.5% variance in the model.

```
K-means clustering with 2 clusters of sizes 12, 10862

Cluster means:
  retweet_count favorite_count user_followers_count user_friends_count
1   27.91666667   44.583333333       11981457.66667        649.0833333
2 1437.95359971    0.373780151          16637.09197       1386.4263487

Within cluster sum of squares by cluster:
[1] 501486856096025 400413225819773
 (between_SS / total_SS =  65.5 %)
```

Comparing the cluster with the Type (Rumor/Non-Rumor), the result we get shows that most of the rumors and non-rumors are in the same cluster. The table is below -

```
  Non-Rumor Rumor
1        10     2
2      5707  5155
```

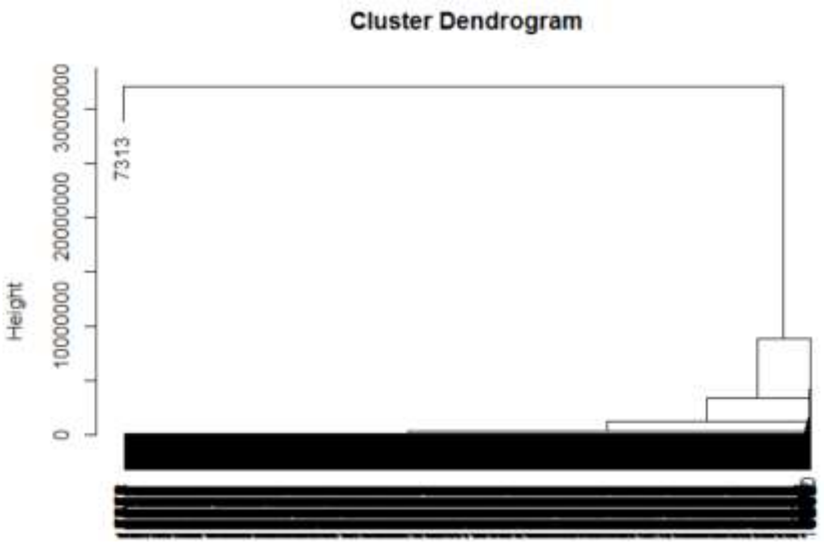
The table above can be represented on a plot as well –



As you can see above, cluster 2 has most of the points, the same as the table.

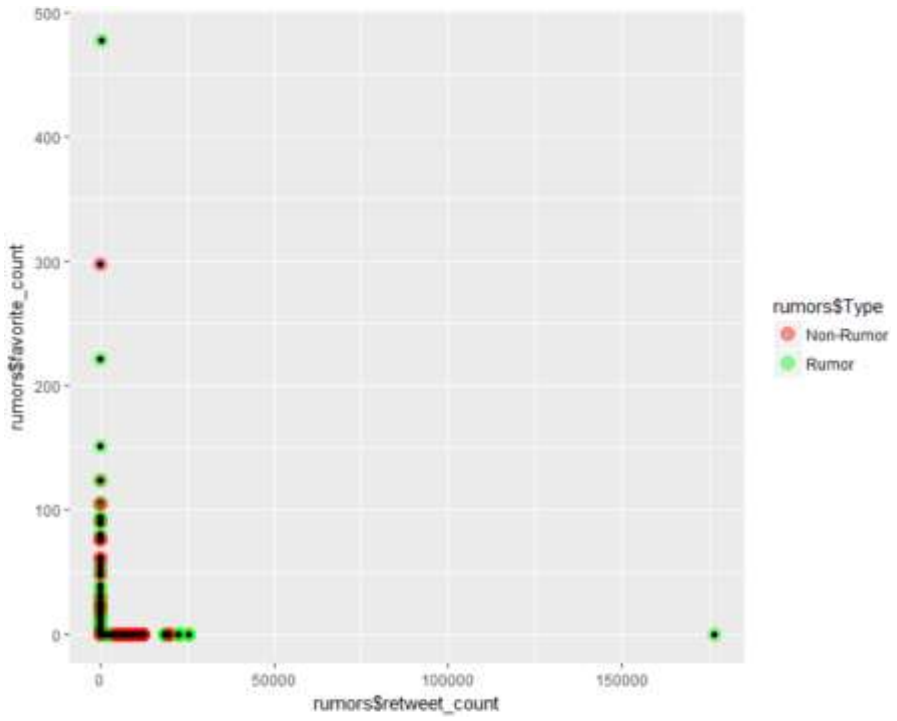
HIERARCHICAL CLUSTERING –

The following is the dendrogram based on our dataset. All the objects are not easily distinguishable and that is because of the size of the dataset. There are over 10,000 rows and the objects are very close to each other.



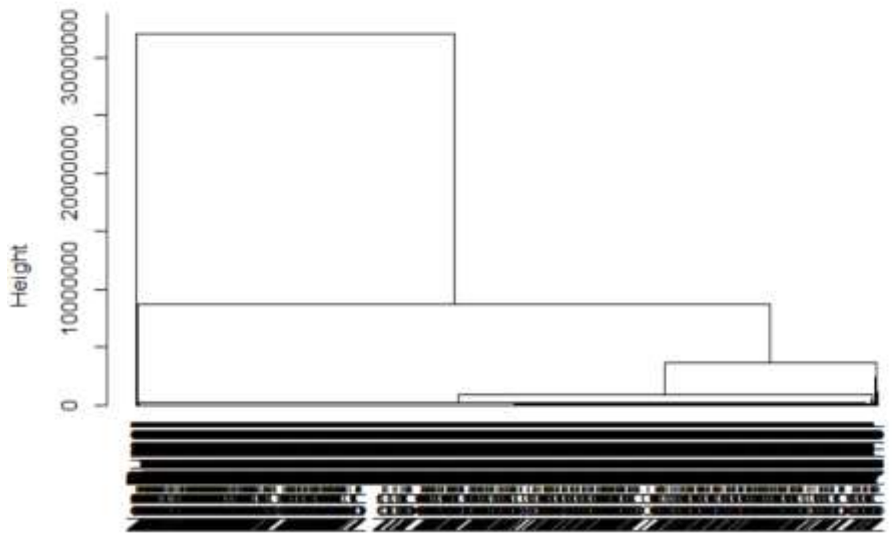
Cutting it at two points, the algorithm again groups rumors and non-rumors in the same cluster -

```
clustercut Non-Rumor Rumor
         1      5716  5157
         2         1     0
```



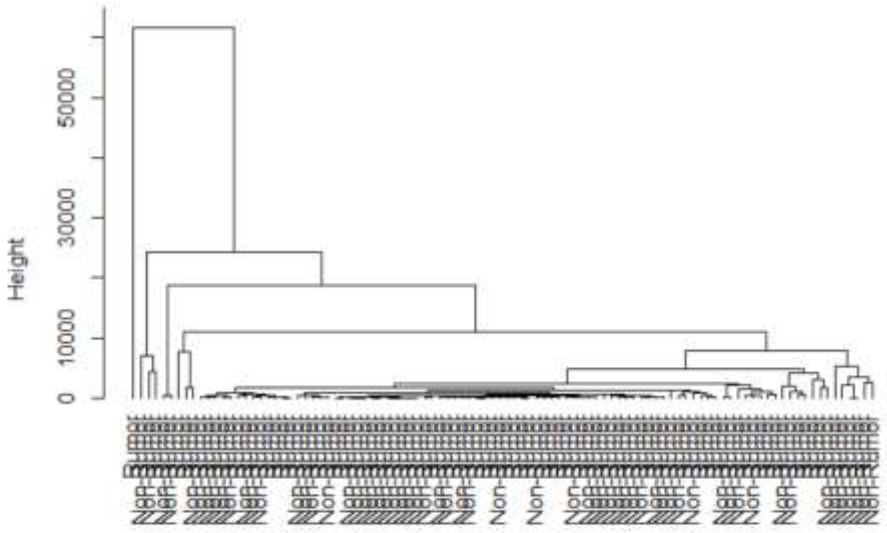
We then took a random sample of our existing dataset. The sample was representative of the original data. The first sample contained 1000 rows and the second contained 100 rows. As the sample size reduces, the objects and cluster mergers are more visible in the dendrograms. However, the clustering results remain the same.

NRow = 1000



```
clustercut Non-Rumor Rumor
         1       533   466
         2         0     1
```

NRow = 100

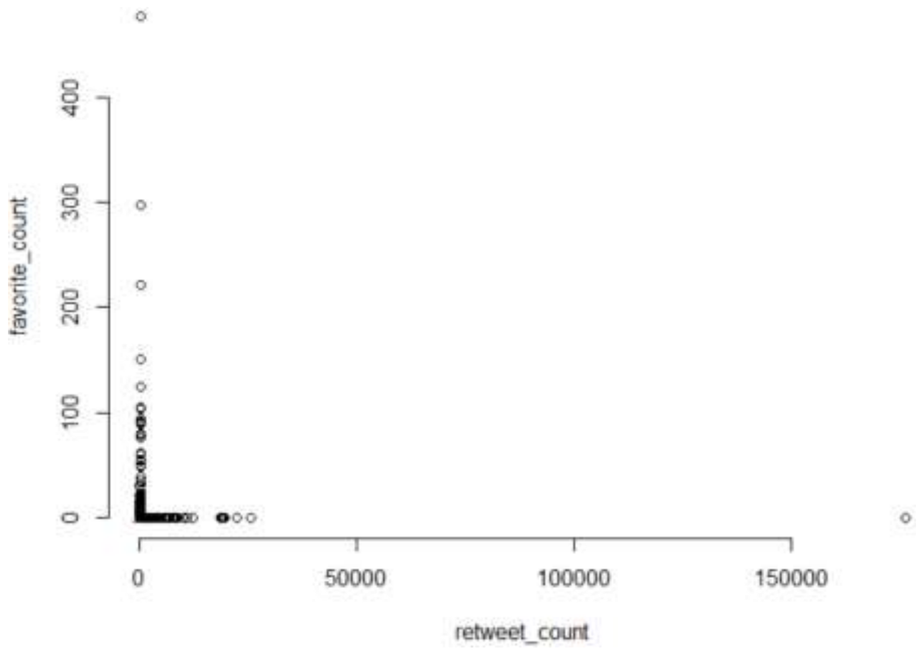


```
clustercut Non-Rumor Rumor
         1        46    53
         2         0     1
```

DENSITY-BASED CLUSTERING –

We used DBScan to perform clustering in this case. We ran dbscan with an epsilon value of 0.2 and minimum points as 20. Considering the size of our dataset, 20 was a good number to use as minimum points. The algorithm created 3 clusters from our dataset. Surely it found certain characteristics that could not be grouped into just 2 clusters. The result is as shown below –

```
dbscan Pts=10874 MinPts=20 eps=0.2
           0  1  2  3
border 10809  0  0  0
seed       0 20 20 25
total  10809 20 20 25
```



<u>COMPARATIVE ANALYSIS</u> –

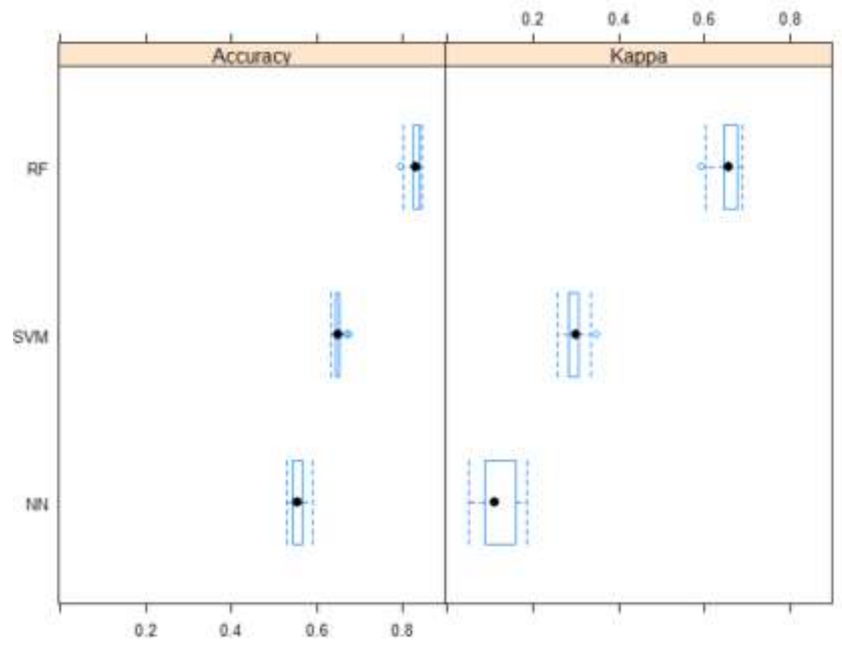
We compared three different models – Random Forest, Support Vector Machines and Neural Networks.

<u>K-Fold Cross-Validation:</u>

Among the three, random forests performed better as they had a higher median accuracy and median kappa values.

```
Call:
summary.resamples(object = results)

Models: NN, RF, SVM
Number of resamples: 10

Accuracy
      Min.  1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.5294  0.5444 0.5540 0.5565  0.5642 0.5879    0
RF  0.7967  0.8243 0.8290 0.8266  0.8382 0.8438    0
SVM 0.6311  0.6435 0.6504 0.6511  0.6530 0.6756    0

Kappa
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.05117 0.09046 0.1094 0.1195  0.1563 0.1860    0
RF  0.59210 0.64740 0.6570 0.6519  0.6752 0.6871    0
SVM 0.25750 0.28450 0.2989 0.2996  0.3047 0.3483    0
```

The following barplot represents the results above –



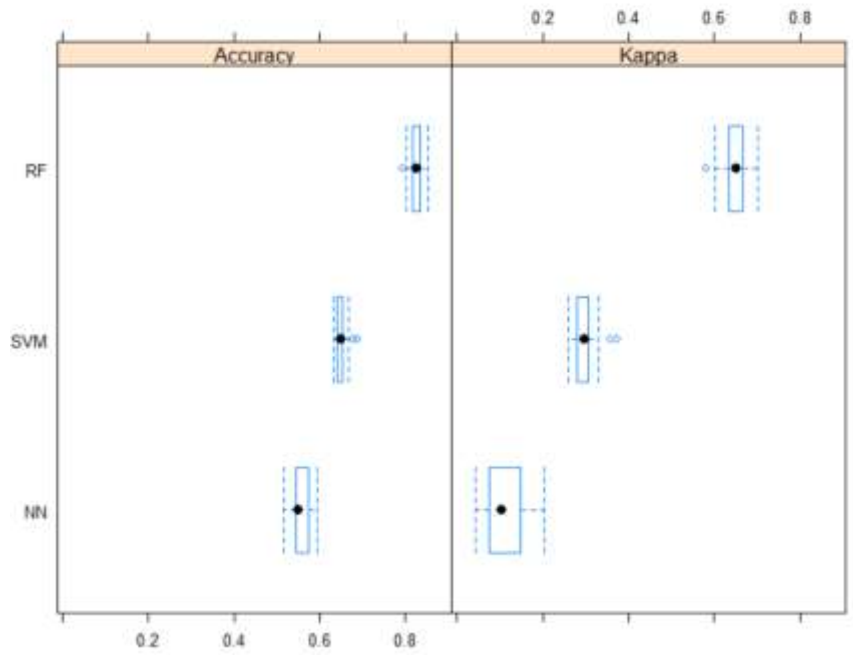
<u>K-Fold Cross Validation (3 Repeats):</u>

In this case too, Random Forests performed better than the other two models.

Note that the number of resamples was 30, which means we used 3 times 10-fold. The result of the comparative analysis is below –

```
Call:
summary.resamples(object = results)

Models: NN, RF, SVM
Number of resamples: 30

Accuracy
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN   0.5147  0.5440 0.5509 0.5546  0.5732 0.5934    0
RF   0.7921  0.8182 0.8265 0.8251  0.8328 0.8511    0
SVM  0.6320  0.6418 0.6492 0.6502  0.6540 0.6875    0

Kappa
        Min. 1st Qu. Median  Mean 3rd Qu.   Max. NA's
NN   0.04413  0.0750 0.1038 0.114  0.1478 0.2033    0
RF   0.58190  0.6345 0.6517 0.649  0.6647 0.7016    0
SVM  0.25930  0.2817 0.2969 0.298  0.3054 0.3735    0
```

This analysis can also be represented as a box plot.

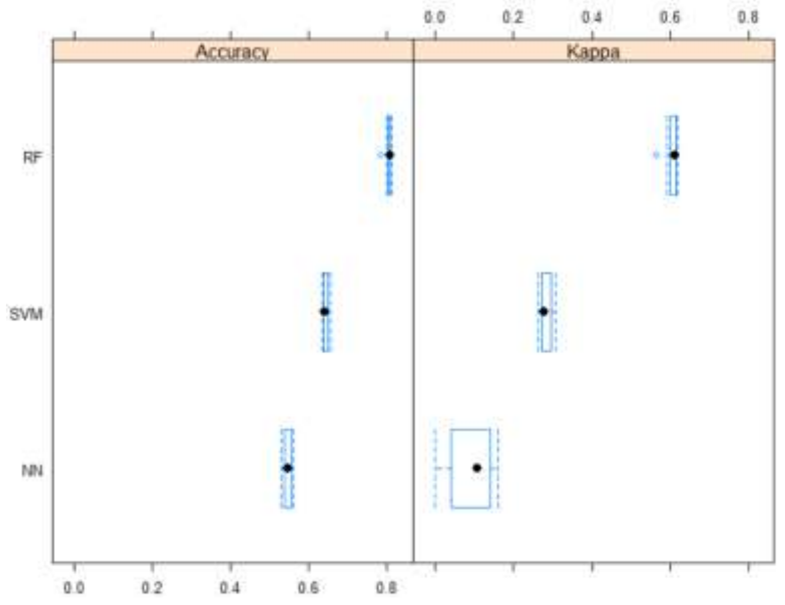


Repeated Bootstrap:

Just as the previous two techniques, random forests perform better.

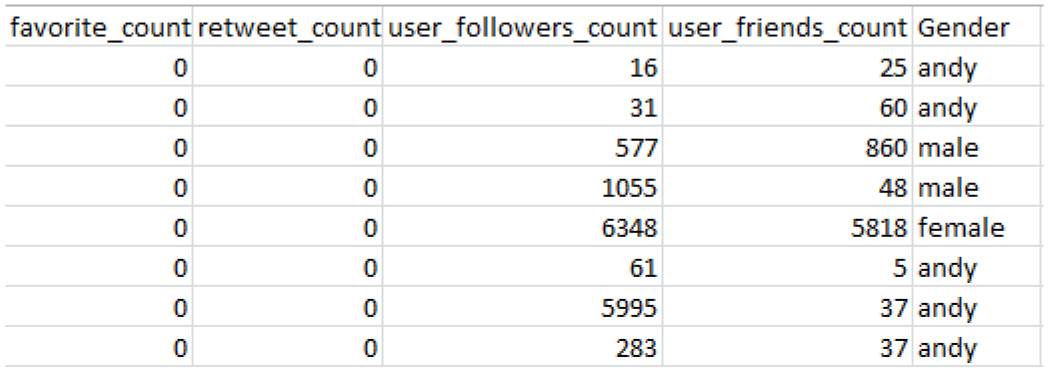
```
Call:
summary.resamples(object = results)

Models: NN, RF, SVM
Number of resamples: 10

Accuracy
      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.5285  0.5384 0.5460 0.5458  0.5533 0.5623    0
RF  0.7832  0.8017 0.8066 0.8036  0.8085 0.8099    0
SVM 0.6356  0.6369 0.6405 0.6430  0.6486 0.6551    0

Kappa
      Min. 1st Qu. Median    Mean 3rd Qu.   Max. NA's
NN  0.0000 0.05123 0.1063 0.09357  0.1400 0.1620    0
RF  0.5660 0.60230 0.6121 0.60590  0.6159 0.6185    0
SVM 0.2644 0.27310 0.2781 0.28400  0.2956 0.3085    0
```

Boxplot for the same –



RQ2: Who Spreads The Rumor?

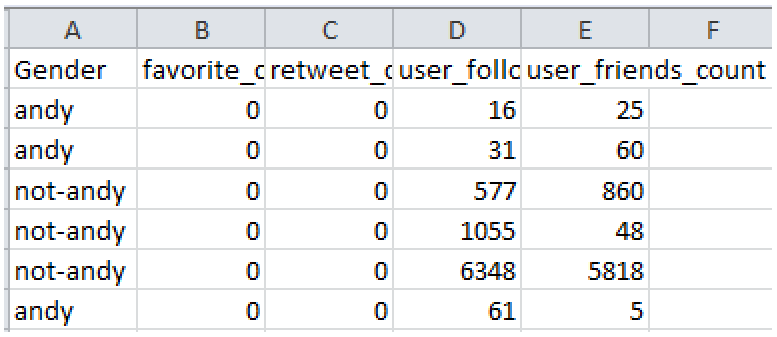
In this question we are trying to build a model which can predict gender of user based of factors like number of retweets, number of favorites, number of friends the user has, number of followers the user have. The dataset looks like the following figure.



favorite_count	retweet_count	user_followers_count	user_friends_count	Gender
0	0	16	25	andy
0	0	31	60	andy
0	0	577	860	male
0	0	1055	48	male
0	0	6348	5818	female
0	0	61	5	andy
0	0	5995	37	andy
0	0	283	37	andy

<u>2.1 SUPPORT VECTOR MACHINE:</u>

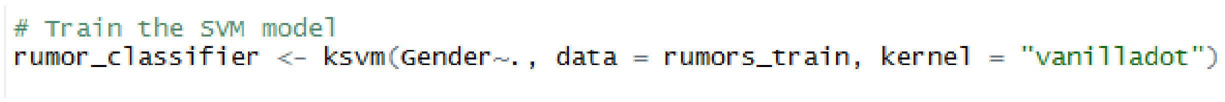
This supervised learning models with associated with learning algorithms that analyze data used for classification and regression analysis. Since there are three classes (andy,male,female) ,this problem is well suited for multiclass approach. However before we proceed with multiclass approach, we are presenting this problem as a binary case by converting two classes (female and male) as Not –Andy. So now the data set shown above becomes as the figure shown below.



A	B	C	D	E	F
Gender	favorite_c	retweet_c	user_follc	user_friends_count	
andy	0	0	16	25	
andy	0	0	31	60	
not-andy	0	0	577	860	
not-andy	0	0	1055	48	
not-andy	0	0	6348	5818	
andy	0	0	61	5	

To start with Binary classification firstly we remove NAs .Then divide the data set in the ratio of 80:20 for training dataset and testing dataset respectively. The kernlab library is used .

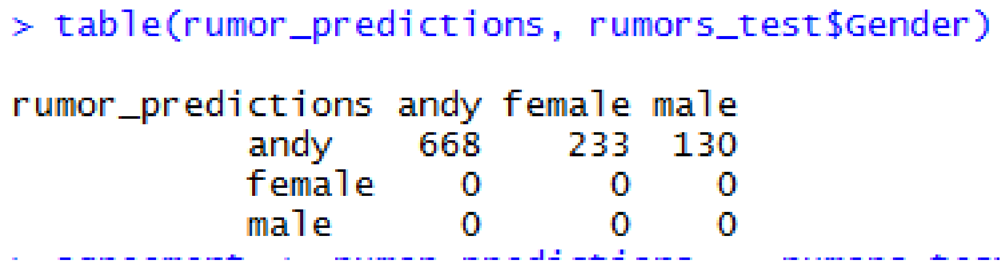
Vanilla Dot kernel



```
# Train the SVM model
rumor_classifier <- ksvm(Gender~., data = rumors_train, kernel = "vanilladot")
```

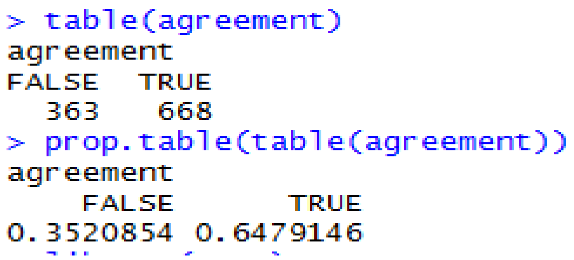
Following are the results for vanilla dot kernel:

Confusion matrix:



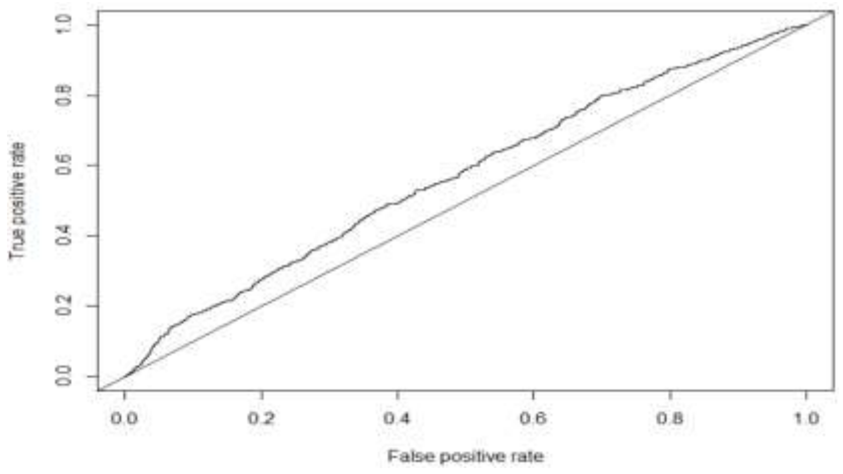
```
> table(rumor_predictions, rumors_test$Gender)

rumor_predictions andy female male
           andy     668    233  130
           female     0      0    0
           male       0      0    0
```

Accuracy:

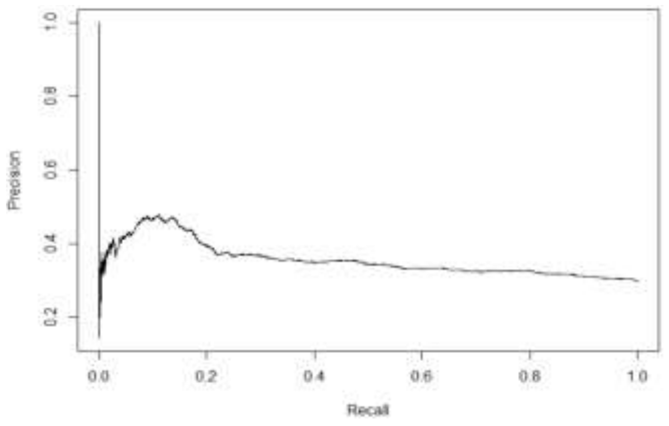


```
> table(agreement)
agreement
FALSE  TRUE
  363   668
> prop.table(table(agreement))
agreement
    FALSE      TRUE
0.3520854 0.6479146
```

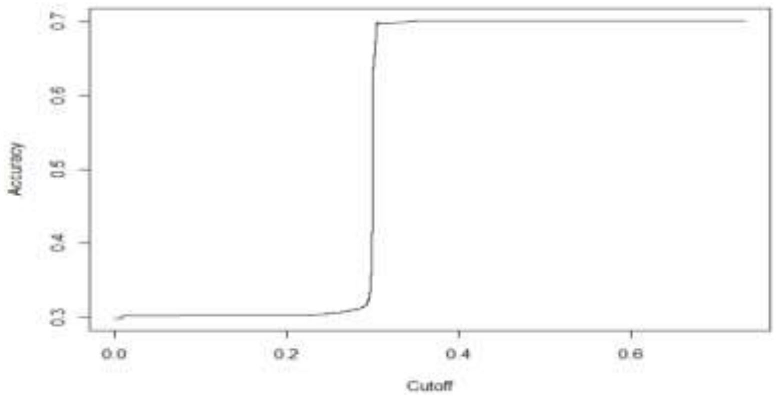
64.7% of the data is correctly classified. The following graph shows the rate of true positive and false positive.



Looking at the graph ,Precision = 0.6479146 ,Recall = 1 ,F Measure = 0.7863449



The following graph shows accuracy of 0.7

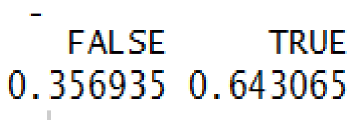


Using RBF kernel

Following is the confusion matrix wherein andy is 662 times correctly classified

```
rumor_predictions_rbf andy not-andy
                 andy      662      362
             not-andy        6        1
```

Accuracy of RBF kernel is 64.3%



```
   FALSE      TRUE
0.356935 0.643065
```

Following are the performance measure

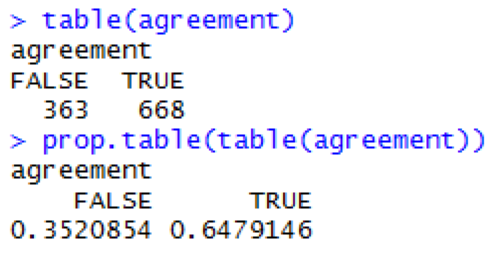
Precision = 0.6464844 ;Recall = 0.991018 ;F1 Measure = 0.7825059

Next we do multiclass code since there are 3 classes.

- **Confusion Matrix-**

rumor_predictions	andy	female	male
andy	668	233	130
female	0	0	0
male	0	0	0

668 Any were correctly classified by the classifier. Following figure shows the accuracy of the table

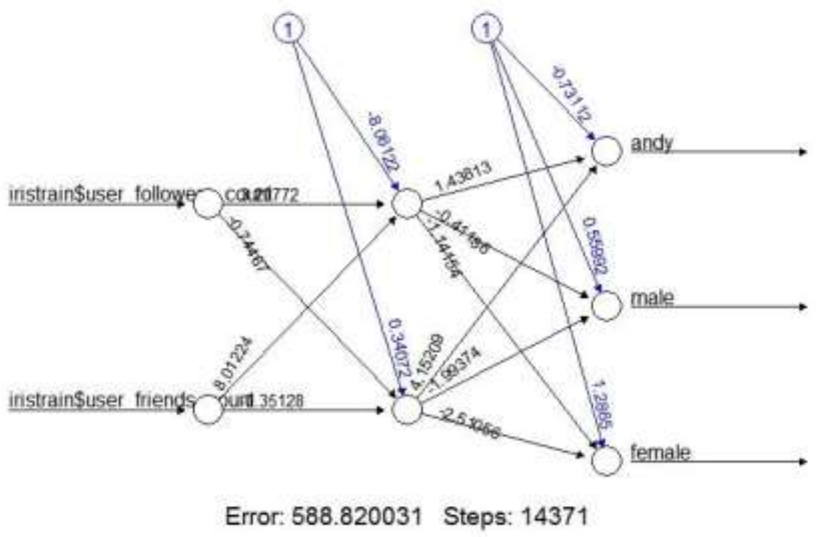


```
> table(agreement)
agreement
FALSE  TRUE
  363   668
> prop.table(table(agreement))
agreement
    FALSE      TRUE
0.3520854 0.6479146
```

35% were wrongly classified.

2.2 NEURAL NETWORK:

Inputs like user followers and users friends are taken into consideration and the output is in form of 3 categorical variables like, male,female, Andy.



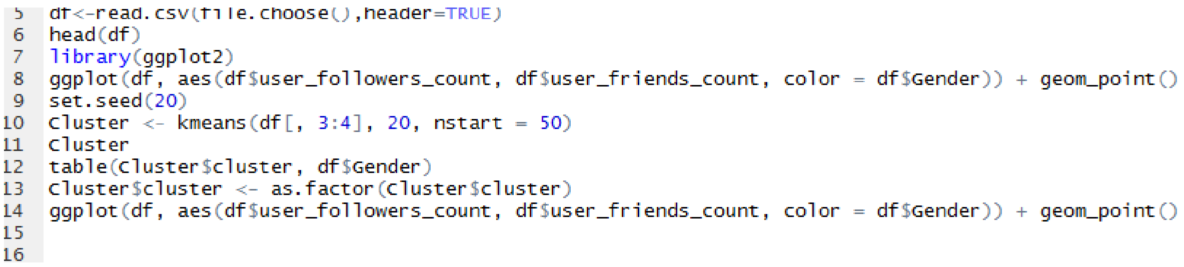
The activation function that could be used is hyperbolic tangent.

2.3.CLUSTERING

3 types of clustering is been done. K mean clustering, Hierarchy clustering and Density based clustering.

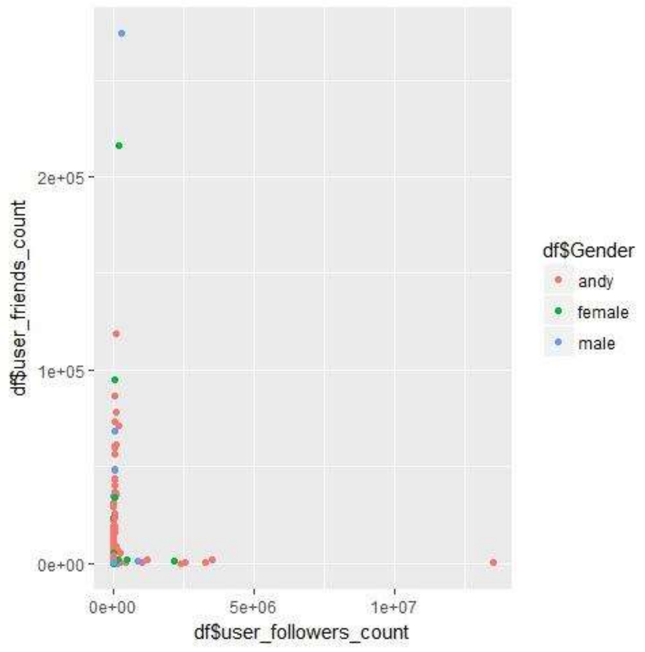
K mean clustering:

The predicted variable is categorical and the predictors are numerical. The figure below shows code snippet.



```
df<-read.csv(file.choose(),header=TRUE)
head(df)
library(ggplot2)
ggplot(df, aes(df$user_followers_count, df$user_friends_count, color = df$Gender)) + geom_point()
set.seed(20)
Cluster <- kmeans(df[, 3:4], 20, nstart = 50)
Cluster
table(Cluster$cluster, df$Gender)
Cluster$cluster <- as.factor(Cluster$cluster)
ggplot(df, aes(df$user_followers_count, df$user_friends_count, color = df$Gender)) + geom_point()
```

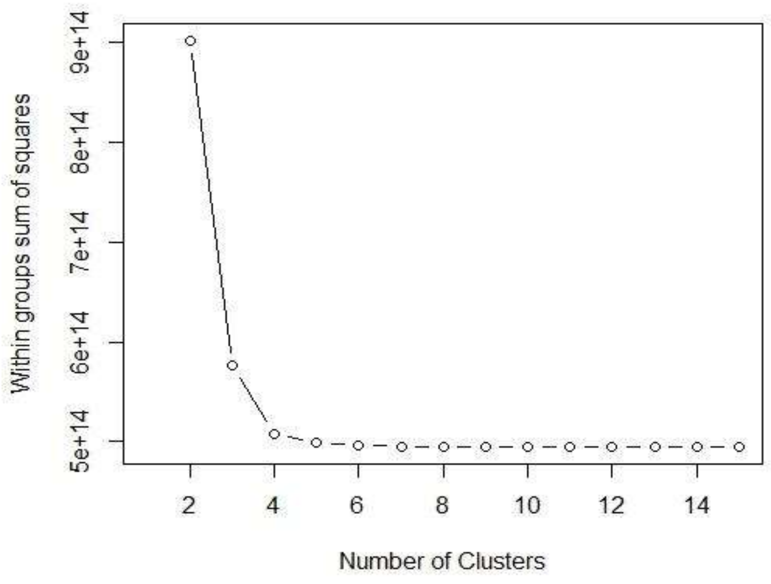
The graph shows distribution of the data in the dataset



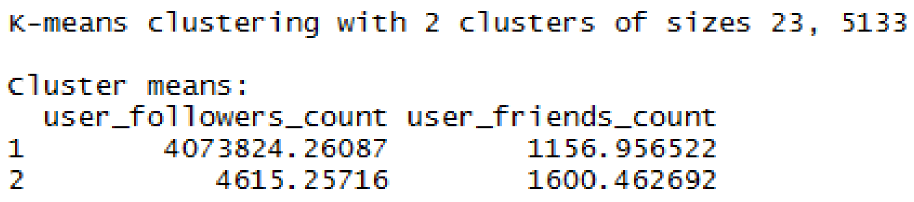
Inorder to find right number of clusters following code is executed

```
# Determine number of clusters
wss <- (nrow(df[,2:5])-1)*sum(apply(df[,2:5],2,var))
for(i in 2:15) wss[i] <- sum(kmeans(rumors[,2:5], centers = i, iter.max = 50)$withinss)

# Bases on the plot we should try our analysis with 2 clusters
plot(1:15, wss, type = "b", xlab = "Number of Clusters", ylab = "Within groups sum of squares")
```

The graph shown below shows ideal number of clusters which is 2 or 4

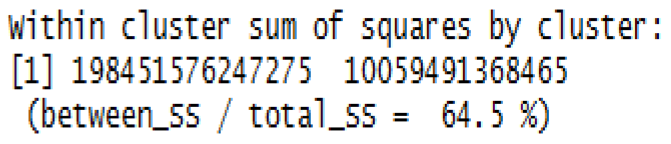


After introducing 2 clusters following figure show means of various predictors and also the number of data points available in each cluster.



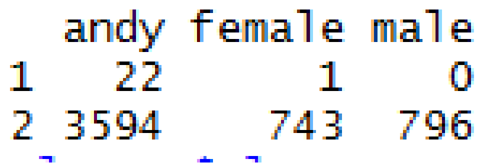
```
K-means clustering with 2 clusters of sizes 23, 5133

Cluster means:
  user_followers_count user_friends_count
1        4073824.26087        1156.956522
2           4615.25716        1600.462692
```

This model explains 64.5% of variance.



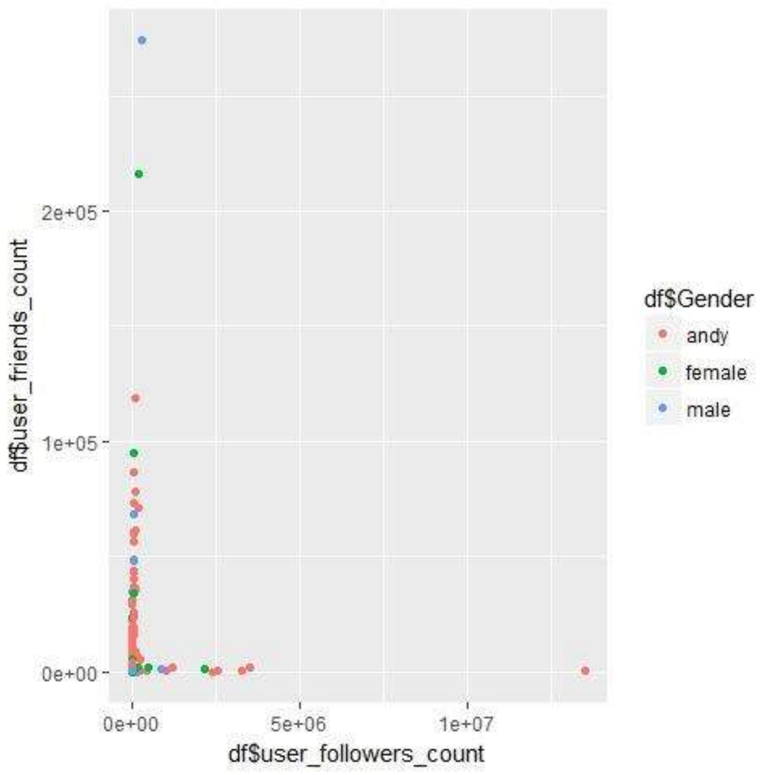
```
Within cluster sum of squares by cluster:
[1] 198451576247275  10059491368465
 (between_SS / total_SS =  64.5 %)
```

Results for k mean



```
   andy female male
 1   22      1    0
 2 3594    743  796
```

After k mean clustering the following figure shows clustered data. The graph doesn’t show any distinctive clustering because the original dataset is biased having exceptionally large number of andy in it. Hence the output is biased.

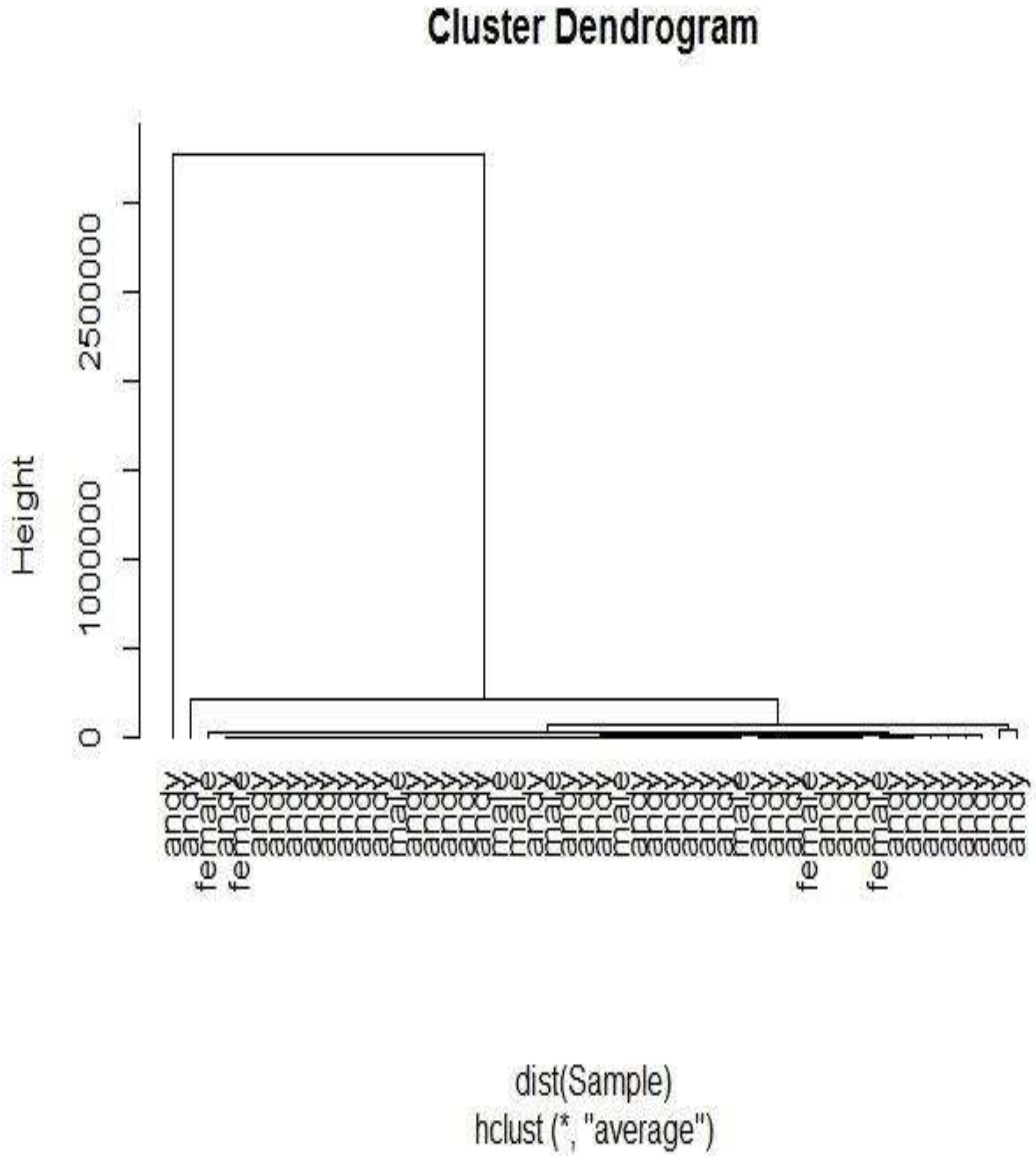


Hierarchical Clustering

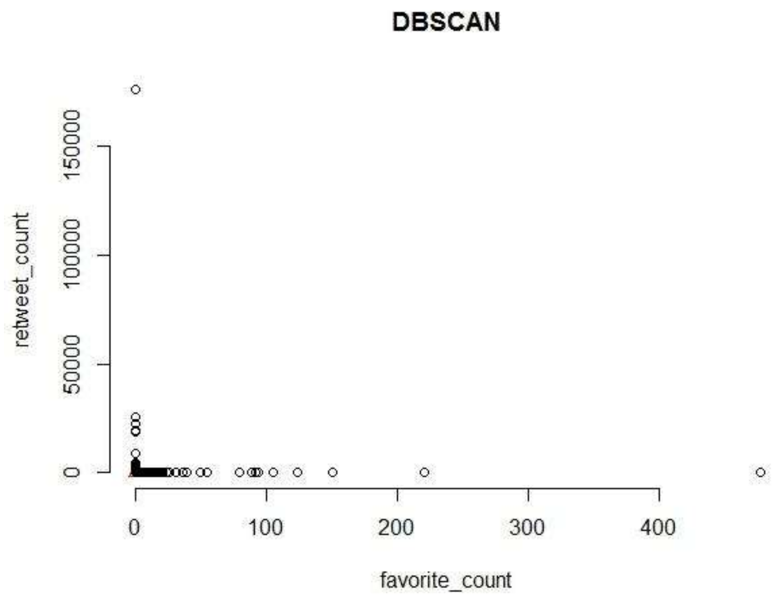
In data mining and statistics, **hierarchical clustering** (also called **hierarchical cluster** analysis or HCA) is a method of **cluster** analysis which seeks to build a **hierarchy** of **clusters**. Hclust function is used. Input : user friends and user followers ,Output: Clusters of Andy, Female,Male

If all the rows are used for clustering, it becomes difficult to comprehend the hierarchical tree. Hence we take 50 random sampling of the data

```
# heirarchial clustering
clusters2 <- hclust(dist(df[, 0:1]))
plot(clusters2)

#heirarchy clustering with a small sample
idx <- sample(1:dim(df)[1], 100)
Sample <- df[idx,]
Sample$Gender <- NULL
hc <- hclust(dist(Sample), method="ave")
plot(hc, hang = -1, labels=df$Gender[idx])
```


Cluster Dendrogram
Height
0
1000000
2500000
dist(Sample)
hclust (*, "average")

Density based Clustering

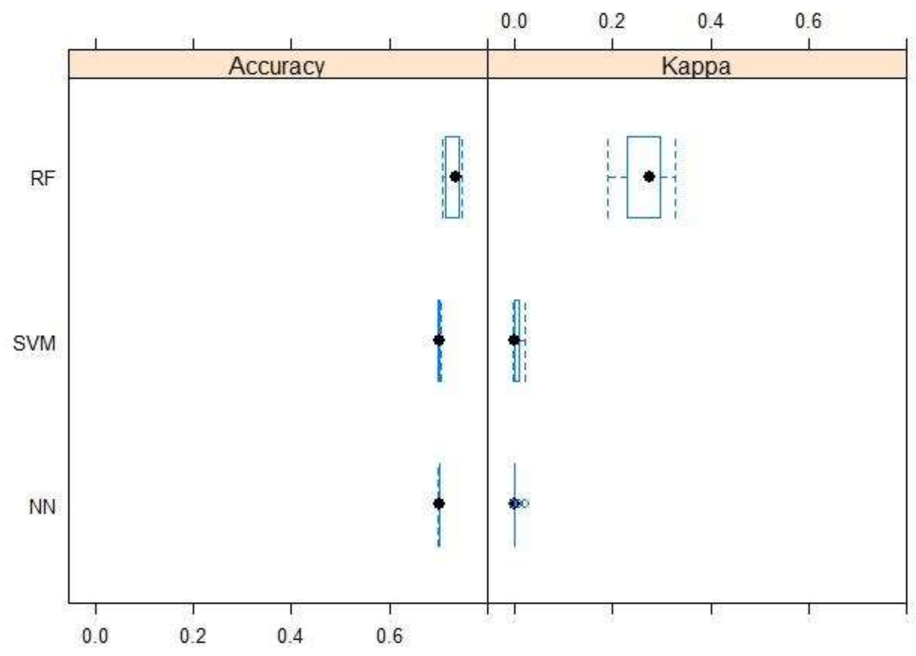


Comparative Analysis

Comparing Support Vector Machine, Neural Networks and Random Forests using the caret library. 3 different CV methods: K-Fold, Bootstrap and Repeated K-Fold are used. Following are the results:

K fold

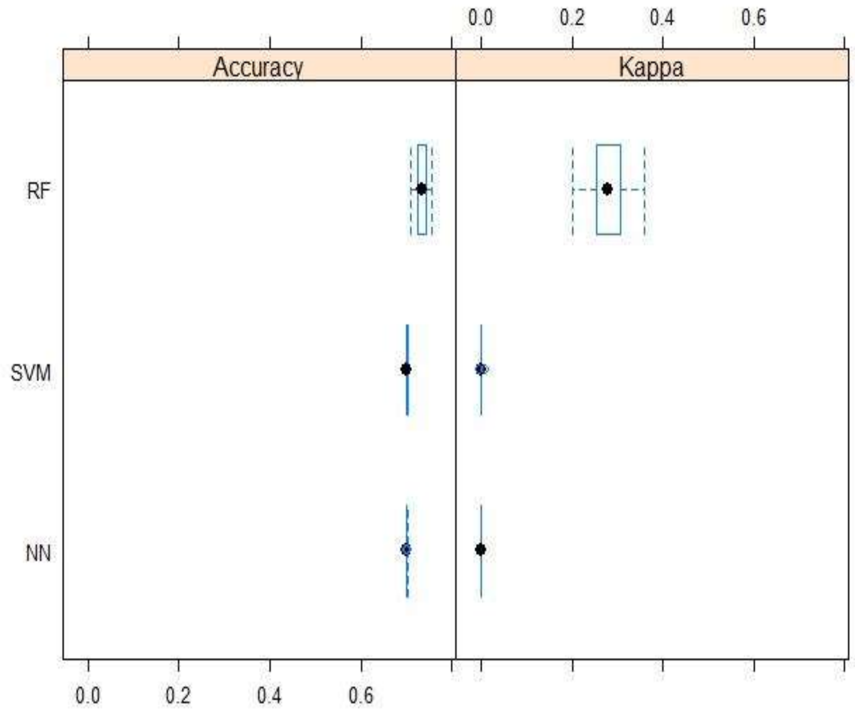
```
Models: NN, RF, SVM
Number of resamples: 10

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.6996124 0.7009709 0.7015504 0.7015137 0.7025721 0.7029126    0
RF  0.7093023 0.7160194 0.7347529 0.7302105 0.7409220 0.7475728    0
SVM 0.6996124 0.7002423 0.7018390 0.7019028 0.7027681 0.7062257    0

Kappa
            Min.   1st Qu.    Median        Mean     3rd Qu.       Max. NA's
NN  -0.002870402 0.0000000 0.0000000 0.002766786 0.000000000 0.02046250    0
RF   0.188883300 0.2370374 0.2765709 0.266870100 0.298020900 0.32879510    0
SVM -0.002939537 0.0000000 0.0000000 0.003730032 0.007497948 0.02016134    0
```



The table and graph shows random forest has higher accuracy.

Repeated K fold

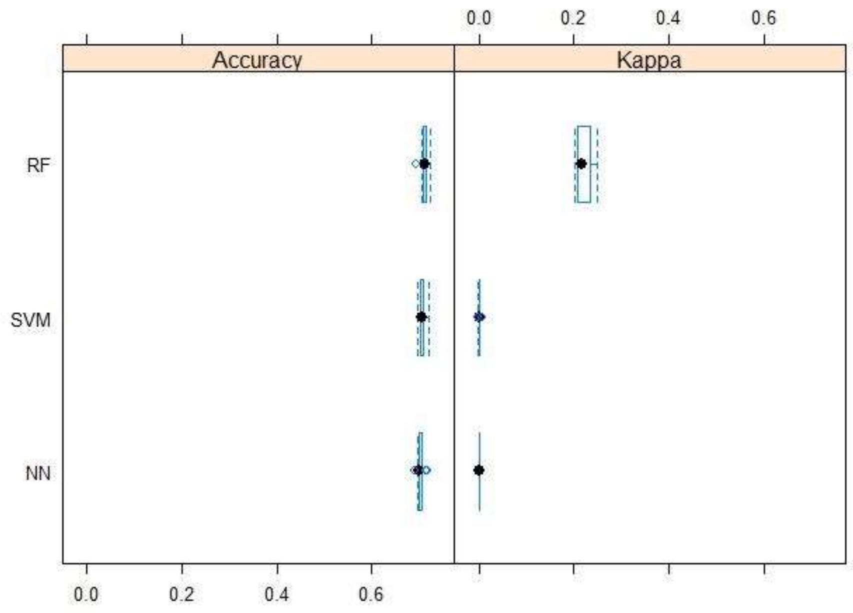
```
Models: NN, RF, SVM
Number of resamples: 30

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.6996124 0.7009709 0.7015504 0.7013196 0.7015504 0.7029126    0
RF  0.7087379 0.7251580 0.7354660 0.7340318 0.7445571 0.7558140    0
SVM 0.6990291 0.7009709 0.7015504 0.7017077 0.7027681 0.7048544    0

Kappa
            Min.   1st Qu.    Median        Mean   3rd Qu.       Max. NA's
NN   0.000000000 0.0000000 0.0000000 0.000000000 0.0000000 0.00000000    0
RF   0.201954600 0.2542852 0.2768109 0.277829000 0.3055226 0.35961230    0
SVM -0.002940031 0.0000000 0.0000000 0.002250185 0.0000000 0.01015389    0
```



The table and graph shows random forest has higher accuracy.

Bootstrap

```
Call:
summary.resamples(object = results)

Models: NN, RF, SVM
Number of resamples: 10

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.6900739 0.6990689 0.7010884 0.7026654 0.7041932 0.7170315    0
RF  0.6947596 0.7081564 0.7117669 0.7115339 0.7148307 0.7231579    0
SVM 0.6979167 0.7016337 0.7044477 0.7058540 0.7088357 0.7208311    0

Kappa
            Min.   1st Qu.    Median         Mean      3rd Qu.        Max. NA's
NN   0.000000000 0.0000000 0.0000000 0.0000000000 0.0000000000 0.000000000    0
RF   0.200992900 0.2080194 0.2166239 0.2201183000 0.2314771000 0.248575000    0
SVM -0.001579574 0.0000000 0.0000000 0.0005507419 0.0007001189 0.005997077    0
```



The table and graph shows random forest has higher accuracy.

RQ3: Predicted Types of Rumors

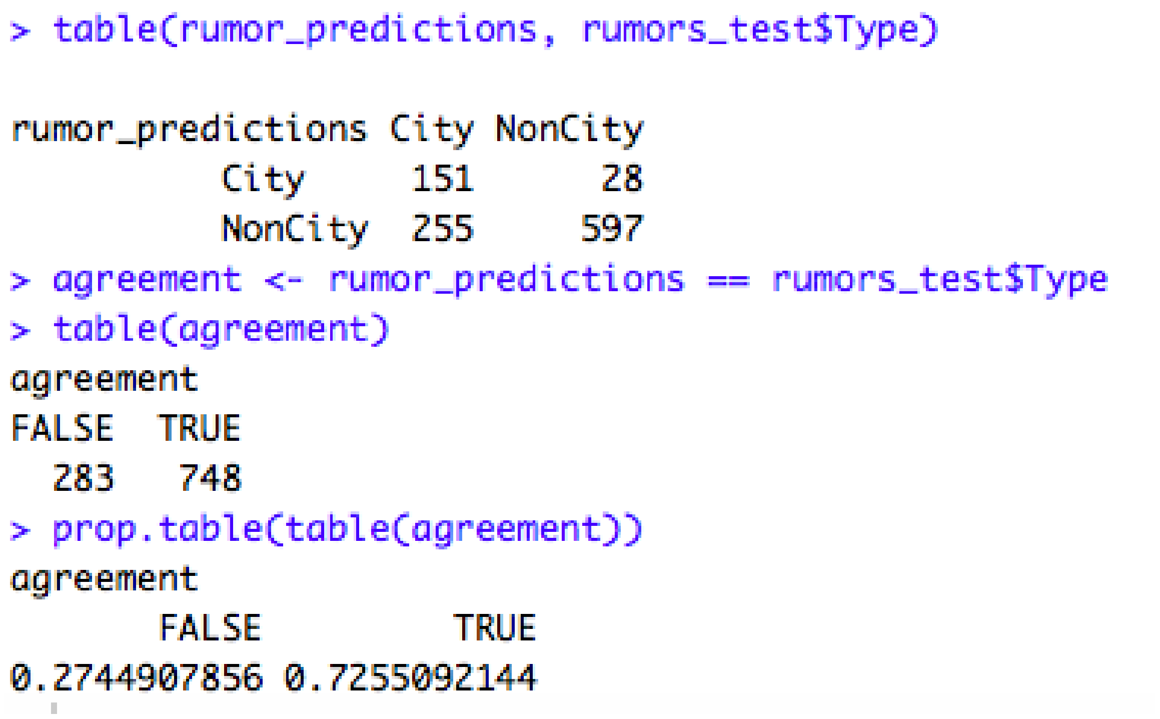
<u>SVMs (BINARY CASE)</u> –

There are 4 types of rumors: city, electronics, transport, and politics. However, since the Milestone 3 requested us to conduct SVM on binary case, we re-coded the data into city and non-city. City is one of the well-predicted rumor types that machine learning techniques that we practiced.

In order to predict the types of rumors, we used the Vanilladot kernel. Please see the basic information for the model after we trained the classifier below.

```
Support Vector Machine object of class "ksvm"


SV type: C-svc  (classification)
 parameter : cost C = 1


Linear (vanilla) kernel function.


Number of Support Vectors : 2977


Objective Function Value : -2966.4735
Training error : 0.245516
```

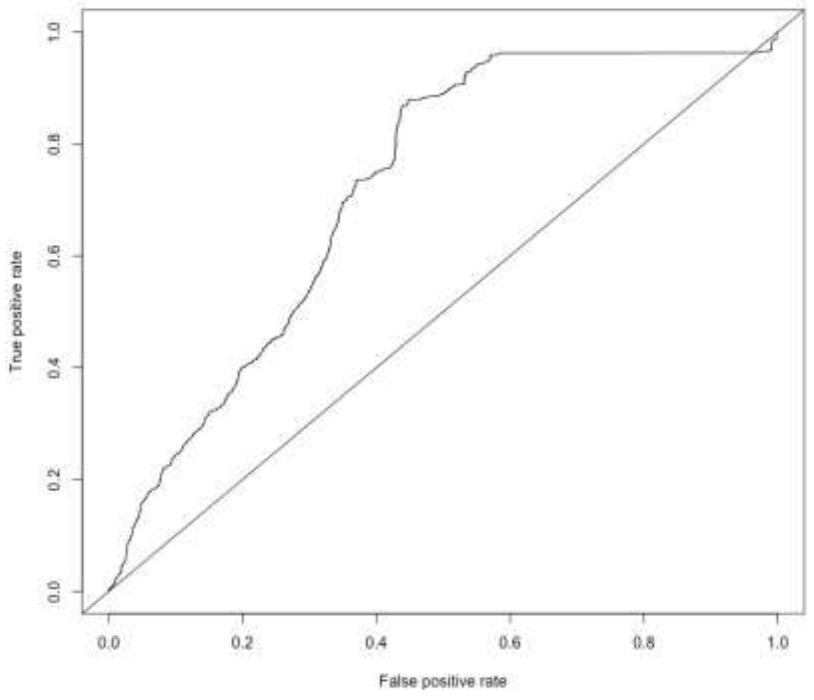
Results revealed that the majority of the noncity was classified correctly, while most of the city were incorrectly classified. Overall accuracy also reflected this. The confusion matrix from this classification and the overall accuracy below.



```
> table(rumor_predictions, rumors_test$Type)

rumor_predictions City NonCity
          City     151      28
          NonCity  255     597
> agreement <- rumor_predictions == rumors_test$Type
> table(agreement)
agreement
FALSE  TRUE
  283   748
> prop.table(table(agreement))
agreement
      FALSE        TRUE
0.2744907856 0.7255092144
```

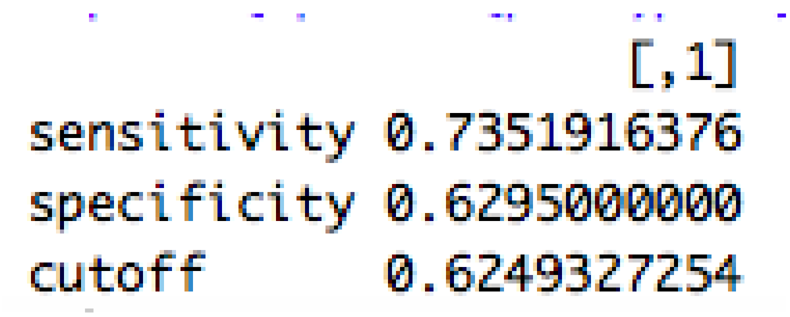
–

ROCR Curve:



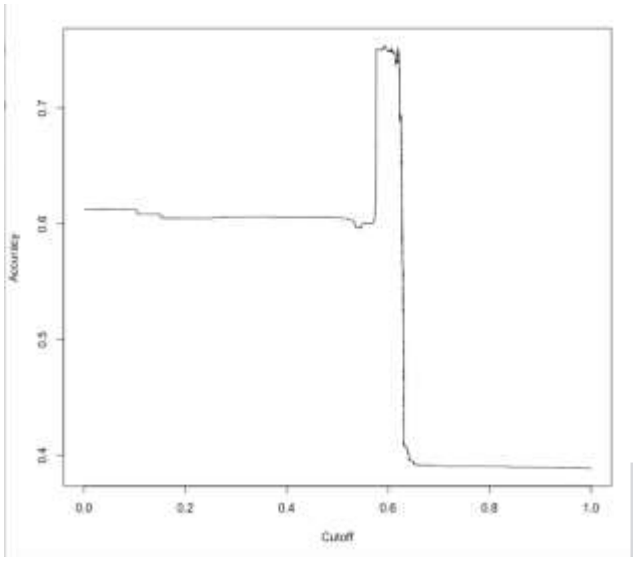
Again, ideally, we would want the false positive rate to be as low as possible and the corresponding true positive rates to be high. However, at the least all the points should be above the ab line as seen in the plot above. We can see that as the true positive rate is increasing, the false positive rates increase as well.

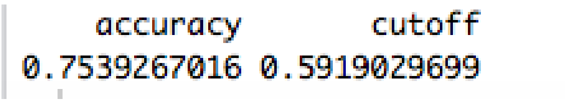
Performance Measures –



```
                  [,1]
sensitivity 0.7351916376
specificity 0.6295000000
cutoff      0.6249327254
```

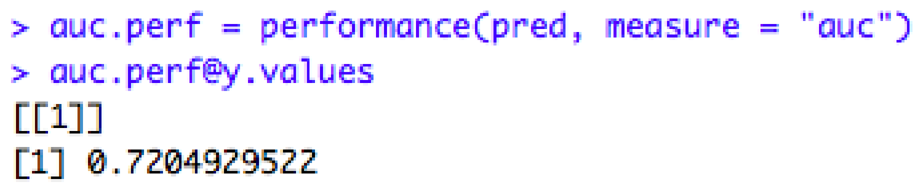
Plot of Accuracy vs Cutoff –





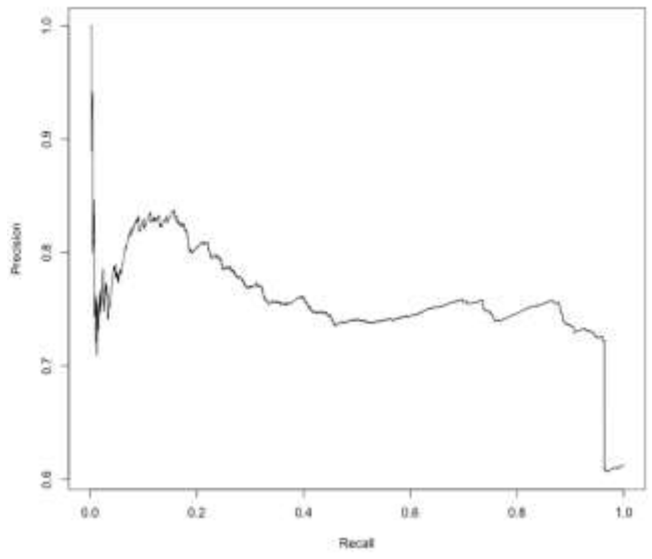
```
    accuracy       cutoff
0.7539267016 0.5919029699
```

Results also showed the cut off and accuracy. R computed mathematically cutoff value and accuracy, where you can see the results and graphics above. We also calculated accuracy using Area Under the Curve and the value was 0.7204.

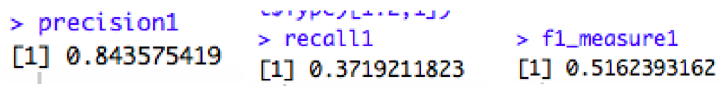


```
> auc.perf = performance(pred, measure = "auc")
> auc.perf@y.values
[[1]]
[1] 0.7204929522
```

Plot of Precision vs Recall –



Again, precision and recall are in trade-off relationship. It is hard to get both high precision and recall. The values for precision, recall and F-measure and listed below –



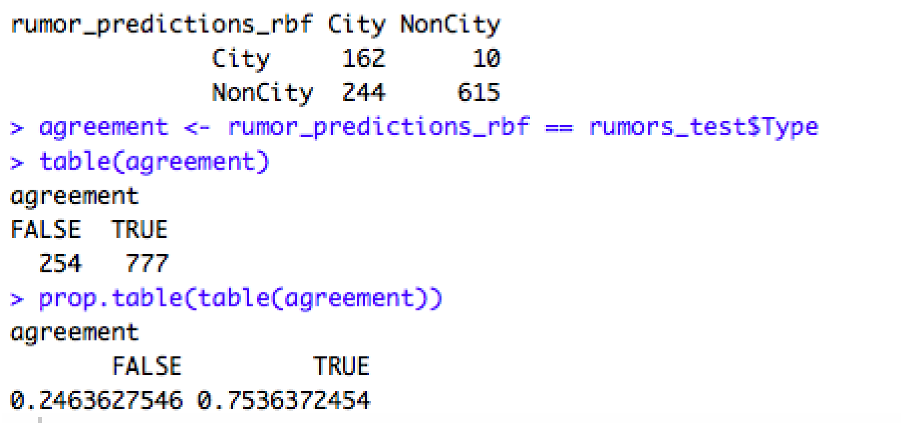
```
> precision1          > recall1             > f1_measure1
[1] 0.843575419       [1] 0.3719211823      [1] 0.5162393162
```

F1 Measure = [2 * Precision * Recall / (Precision + Recall)

RBF Kernel

The team conducted RBF Kernel to improve the model. Clearly, the model has improved the performance, although it was not a huge leap.

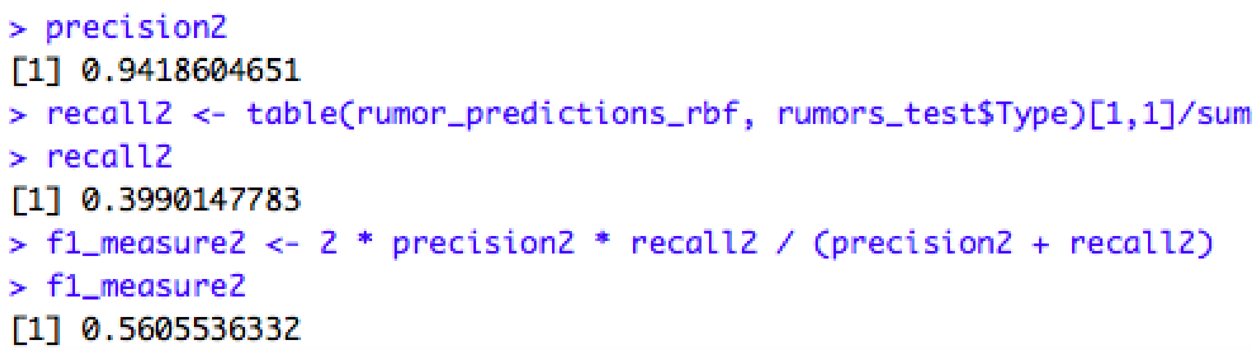
The confusion matrix and the overall accuracy rate–



```
rumor_predictions_rbf City NonCity
              City     162      10
              NonCity  244     615
> agreement <- rumor_predictions_rbf == rumors_test$Type
> table(agreement)
agreement
FALSE  TRUE
  254   777
> prop.table(table(agreement))
agreement
       FALSE         TRUE
0.2463627546 0.7536372454
```

Performance Measures –

The values of precision, recall and F-measure are as follows –



```
> precision2
[1] 0.9418604651
> recall2 <- table(rumor_predictions_rbf, rumors_test$Type)[1,1]/sum
> recall2
[1] 0.3990147783
> f1_measure2 <- 2 * precision2 * recall2 / (precision2 + recall2)
> f1_measure2
[1] 0.5605536332
```

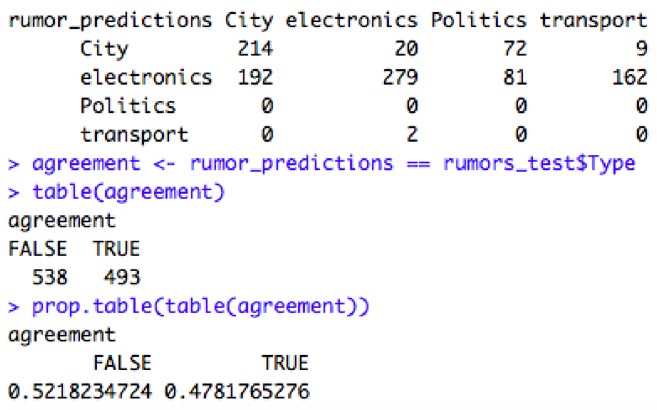
SVMs (MULTICLASS CASE) –

In order to conduct SVM on multiclass case, we use the original data throughout 4 types of rumors: city, electronics, transport, and politics.

The first implementation is using the Vanilladot kernel. Here is some basic information about the model after we train it .

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)

 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 3391

Objective Function Value : -1545.0686 -1286 -1217.1015 -898.6255 -1212.8706 -
812.8882

Training error : 0.480368
```

The following is the confusion matrix resulting from this classification. The model predicted city and electronics very well, while the model predicted politics and transport poorly. This is also reflected in the overall agreement.



```
rumor_predictions City electronics Politics transport
      City         214          20       72         9
      electronics  192         279       81       162
      Politics       0           0        0         0
      transport      0           2        0         0
> agreement <- rumor_predictions == rumors_test$Type
> table(agreement)
agreement
FALSE  TRUE
  538   493
> prop.table(table(agreement))
agreement
       FALSE         TRUE
0.5218234724 0.4781765276
```

Performance Measures –

Recall for city - 0.9145299145

Out of all the times this label should have been predicted, 91% of the labels were correctly identified.

Precision for city - 0.5270935961

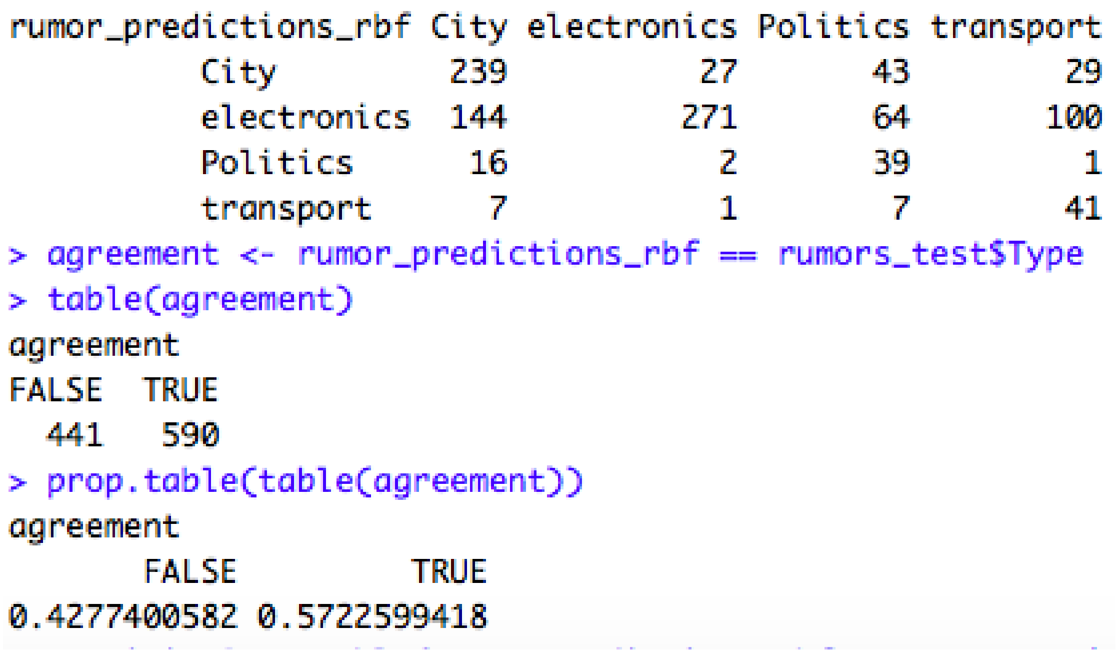
Out of the times this label was predicted, 52% of the time the classifier was correct.

F Measure = 0.66875

Similarly, the recall, precision and F-measure are calculated for the other 3 labels.

RBF Kernel

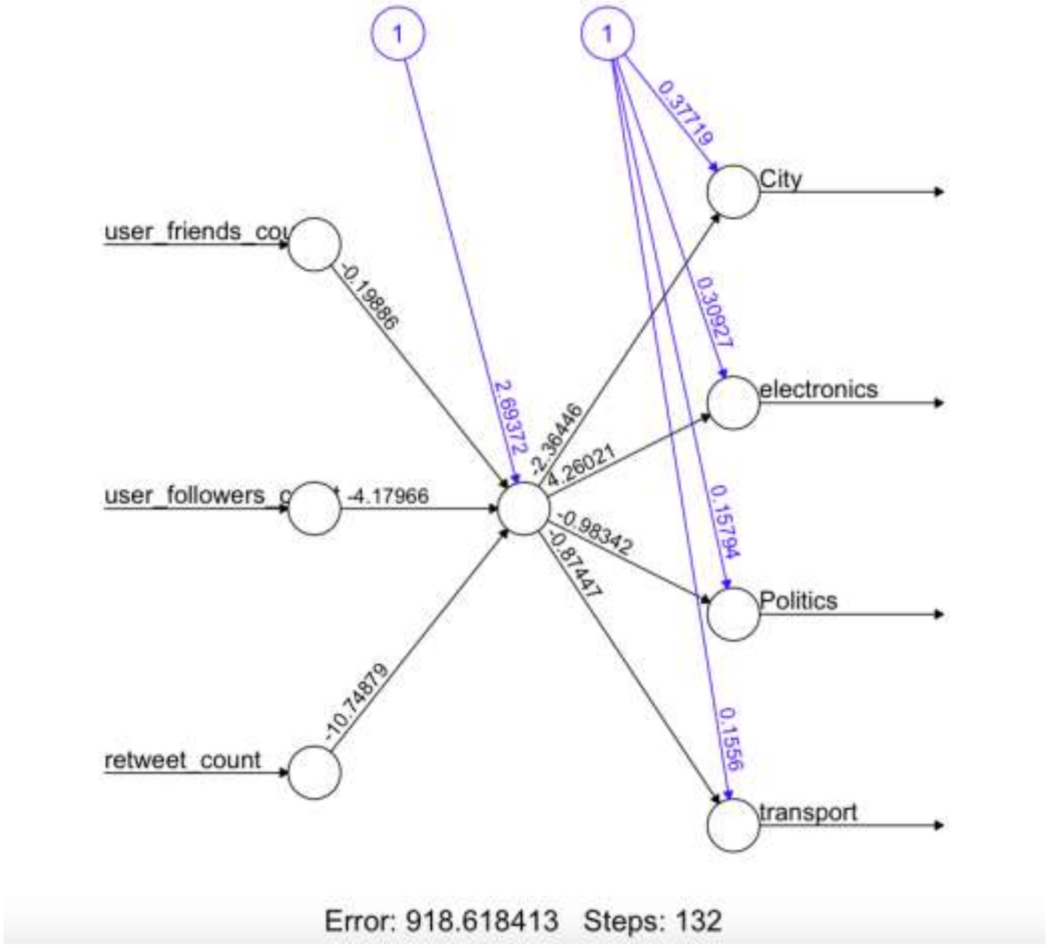
We also conducted RBF Kernel to improve the model. The results showed the improved results. Please find the confusion matrix and overall accuracy.



```
rumor_predictions_rbf City electronics Politics transport
          City         239           27       43        29
          electronics  144          271       64       100
          Politics      16            2       39         1
          transport      7            1        7        41
> agreement <- rumor_predictions_rbf == rumors_test$Type
> table(agreement)
agreement
FALSE  TRUE 
  441   590 
> prop.table(table(agreement))
agreement
       FALSE         TRUE 
0.4277400582 0.5722599418 
```

<u>NEURAL NETWORKS</u>

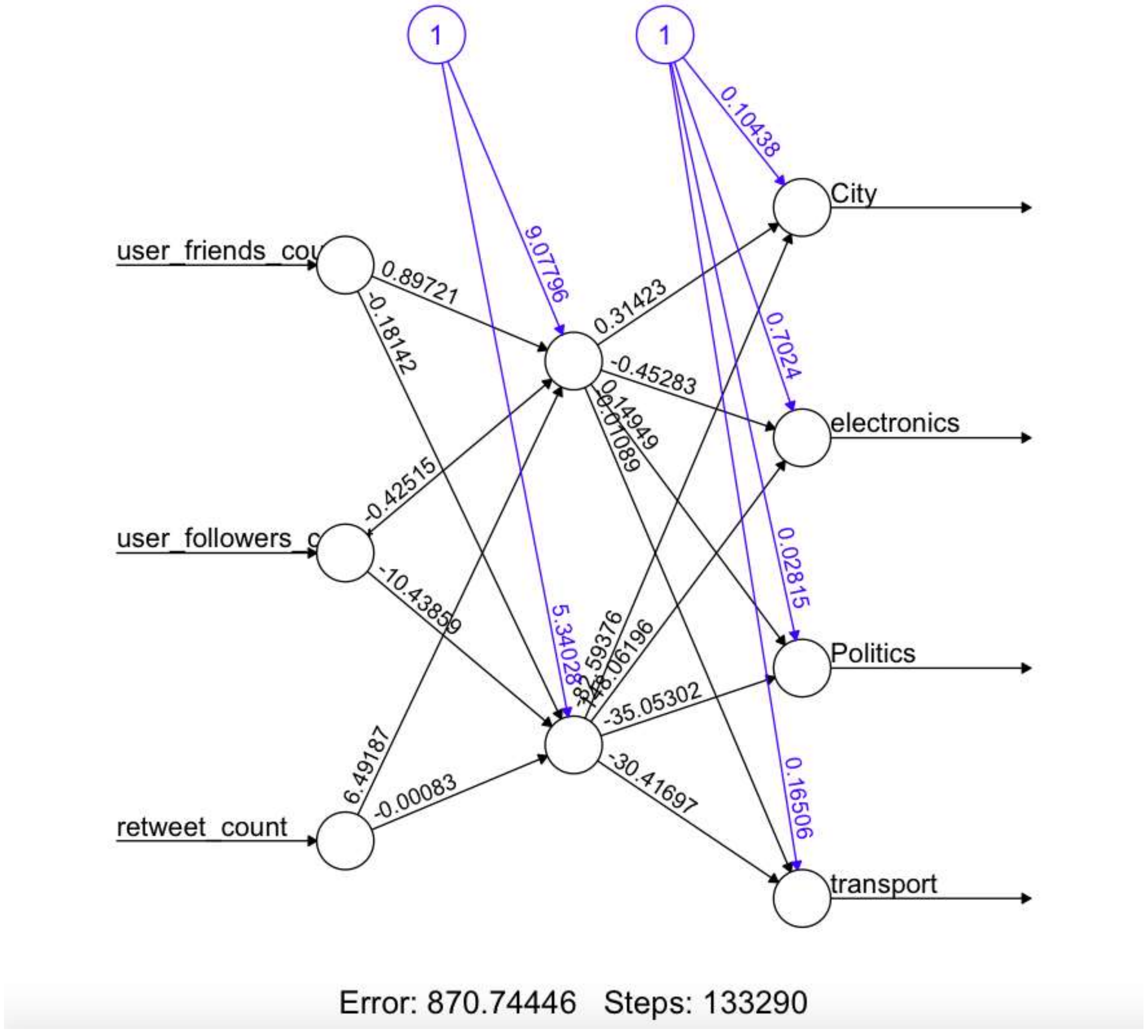
The team conducted Neural Networks with different approaches. The following is a basic neural network with only 1 hidden layer and 1 node in it.



The three variables on the left side are the independent variables, or the input to the neural network. The model computes the weights which is represented on the arrows. And the node at the right end is the output.

Again, we added another hidden node in the same layer. –

nn2 <- neuralnet(City+electronics+Politics+transport ~ user_friends_count + user_followers_count + retweet_count, data=iristrain, **hidden = 2, stepmax = 1e6**)

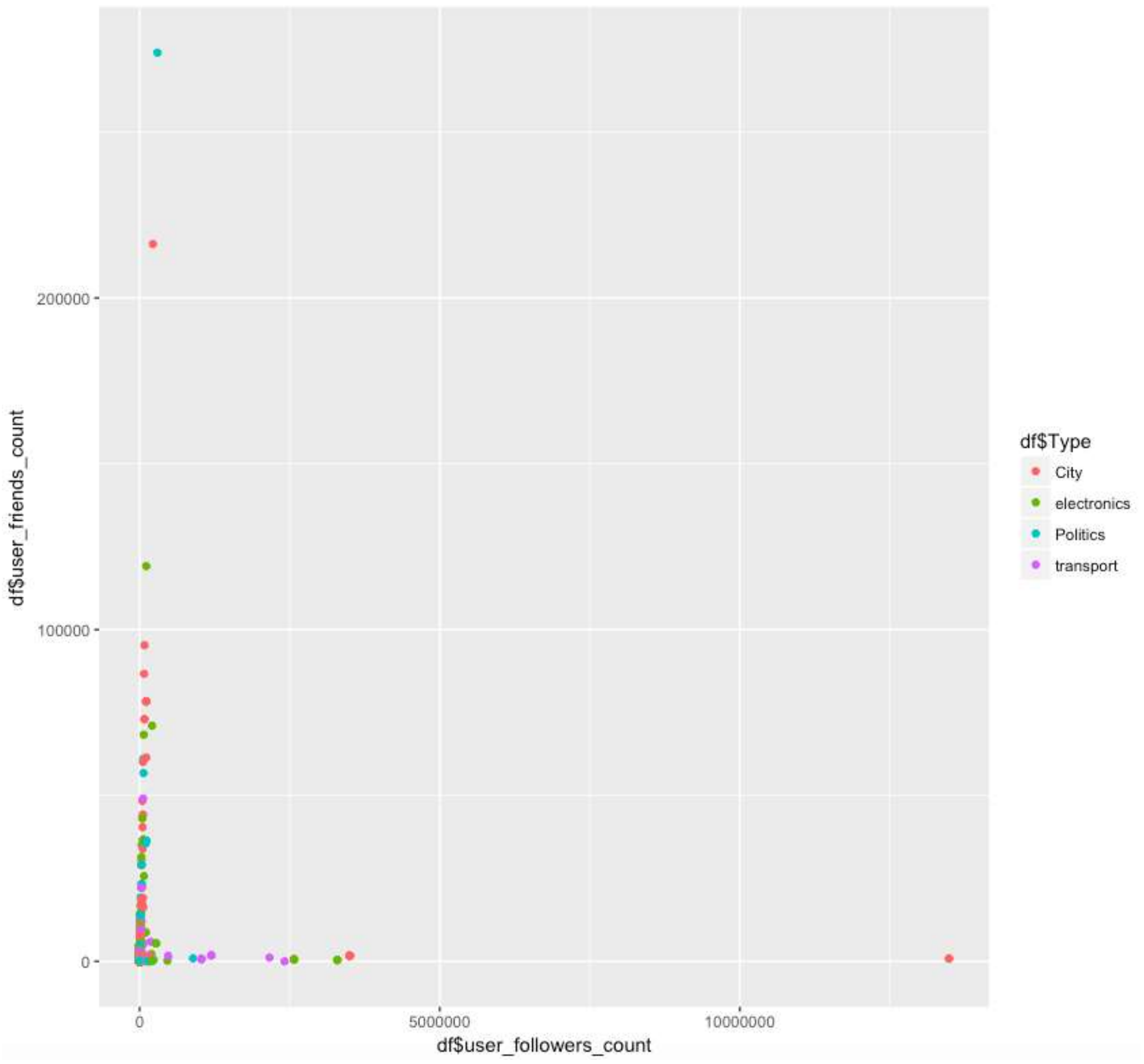


Results revealed much better and improved model here. Please note that we had to increase the value of stepmax to be able to create this neural network with 2 nodes in the hidden layer. 1e6 = 1 million, meaning that the model can take a maximum of 1 million steps to converge.

As we increase the value of hidden to add more layers and/or nodes, the algorithm does not converge for the given value of stepmax. We increased stepmax to 1e9 (1 billion steps) and the algorithm still did not converge. We believe the reason for this is the size of the dataset. Also, any further increment beyond hidden = 3 failed with the multiple trials we performed.

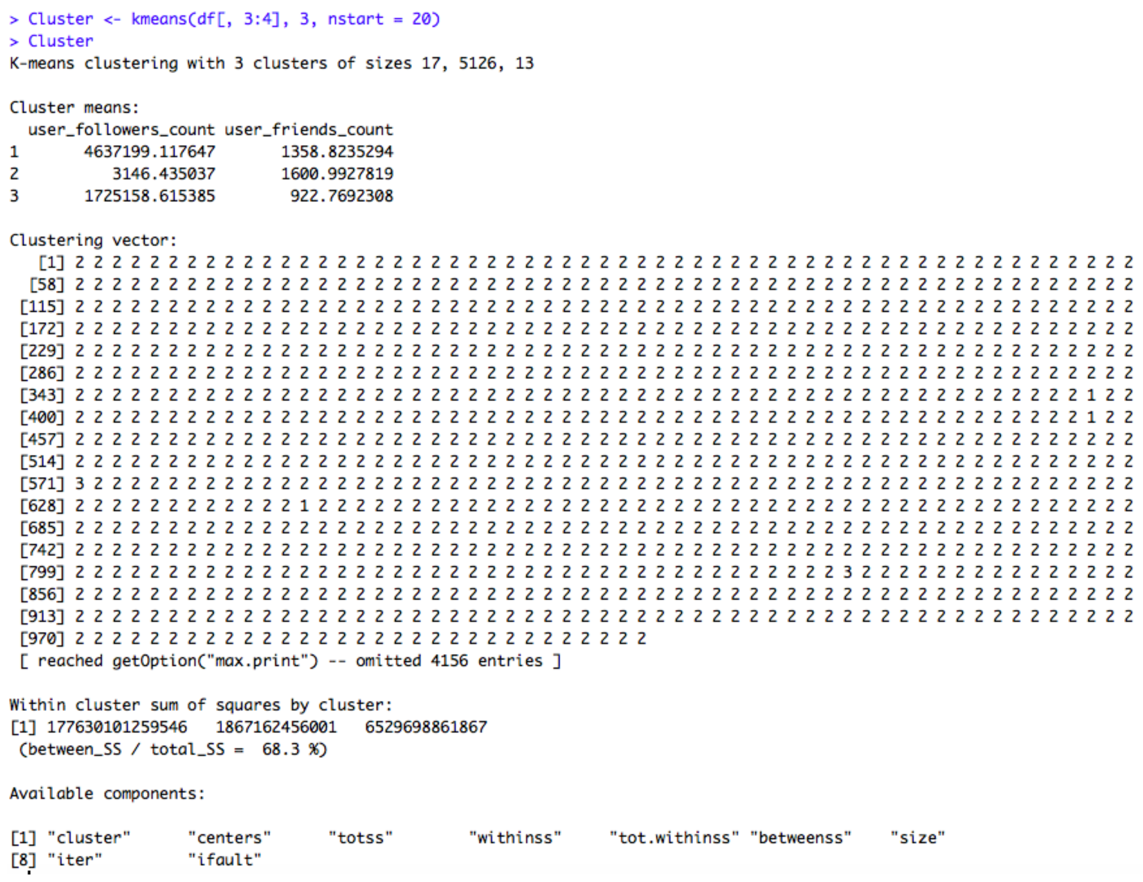
<u>PARTIONAL CLUSTERING (K-MEANS)</u> –

The following is a plot of our current dataset –



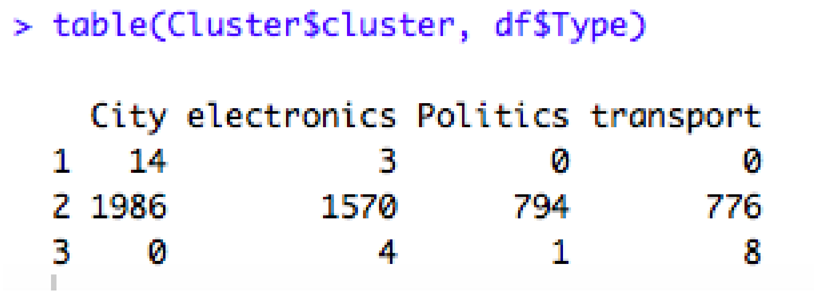
The data is concentrated in one region with only a few outliers (See Figure above). The team conducted random starting assignments and select the one with lowest within cluster violation.

The result revealed 3 clusters, and explained 68.3% variance in the model.



```
> Cluster <- kmeans(df[, 3:4], 3, nstart = 20)
> Cluster
K-means clustering with 3 clusters of sizes 17, 5126, 13

Cluster means:
  user_followers_count user_friends_count
1       4637199.117647       1358.8235294
2          3146.435037       1600.9927819
3       1725158.615385        922.7692308

Clustering vector:
   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  [58] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [115] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [172] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [229] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [286] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [343] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2
 [400] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2
 [457] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [514] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [571] 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [628] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [685] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [742] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [799] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [856] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [913] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [970] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 4156 entries ]

Within cluster sum of squares by cluster:
[1] 177630101259546   1867162456001   6529698861867
 (between_SS / total_SS =  68.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

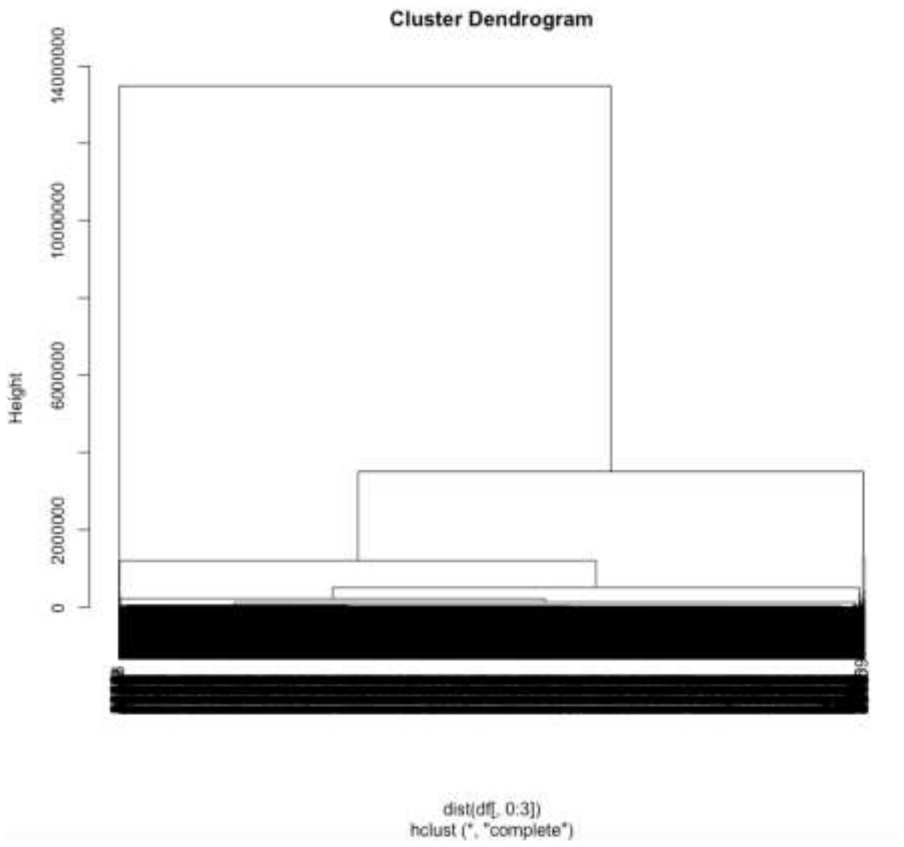
The results above looked promising, yet the closer look revealed that most of the data across 4 rumor types were located in 2nd cluster, meaning it had no differentiating power.



```
> table(Cluster$cluster, df$Type)

    City electronics Politics transport
  1   14           3        0         0
  2 1986        1570      794       776
  3    0           4        1         8
```

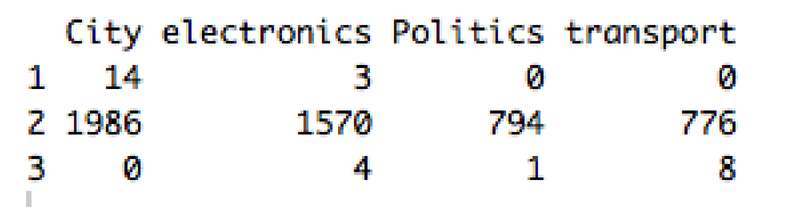
HIERARCHICAL CLUSTERING –

The following is the dendrogram based on our dataset. All the objects are not easily distinguishable and that is because of the size of the dataset. There are over 10,000 rows and the objects are very close to each other.

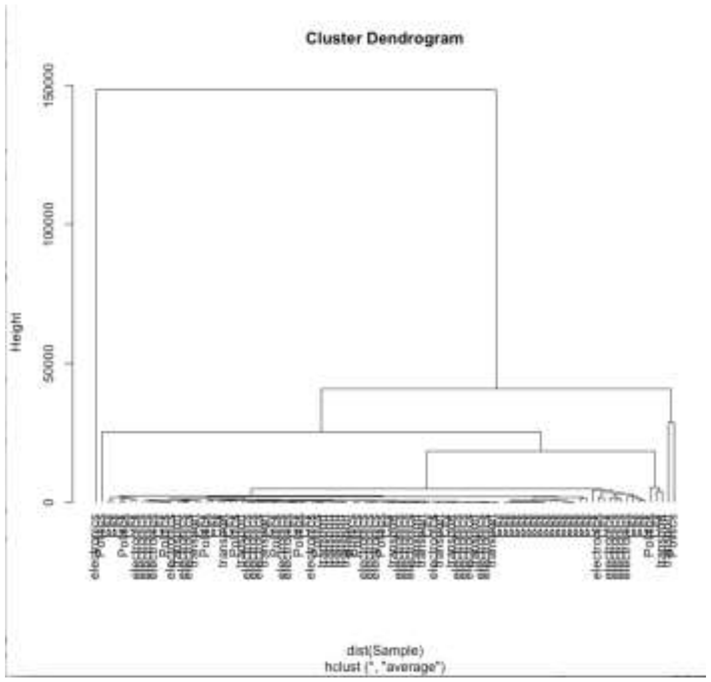


Cutting it at two points, the algorithm again groups rumors and non-rumors in the same cluster -

We then took a random sample of our existing dataset. The sample was representative of the original data. As the sample size reduces, the objects and cluster mergers are more visible in the dendrograms. However, the clustering results remain the same.

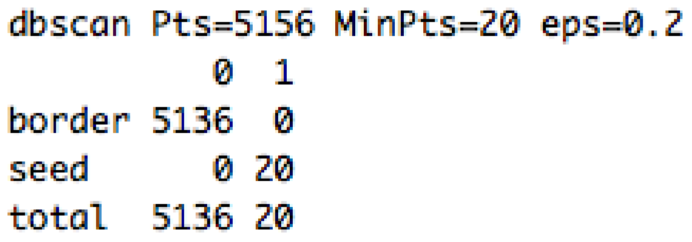


```
  City electronics Politics transport
1   14           3        0         0
2 1986        1570      794       776
3    0           4        1         8
```

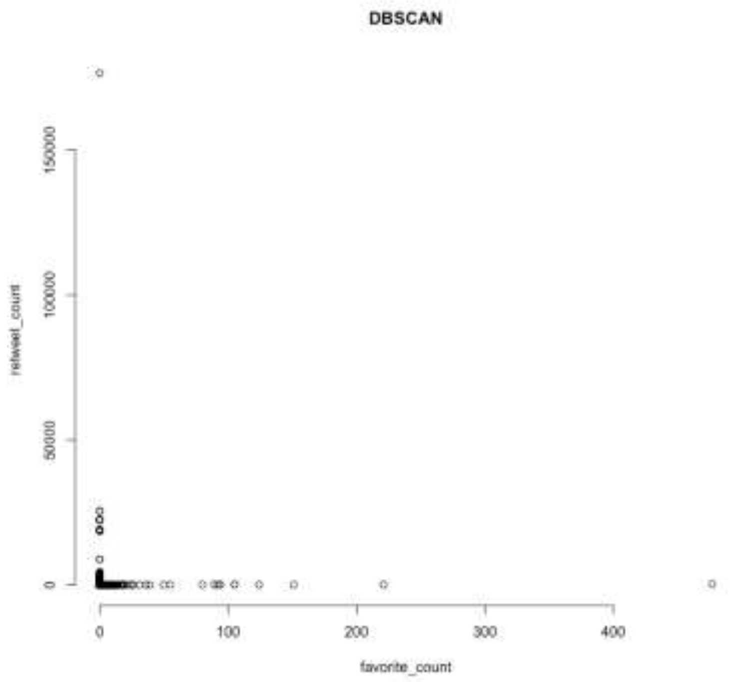


DENSITY-BASED CLUSTERING –

Lastly, the team conducted DBScan to perform clustering. We ran dbscan with an epsilon value of 0.2 and minimum points as 20. Again, considering the size of our dataset, 20 was a good number to use as minimum points. The algorithm created 1 cluster from our dataset. For more information, please find the below.



```
dbscan Pts=5156 MinPts=20 eps=0.2
          0  1
border 5136  0
seed      0 20
total  5136 20
```



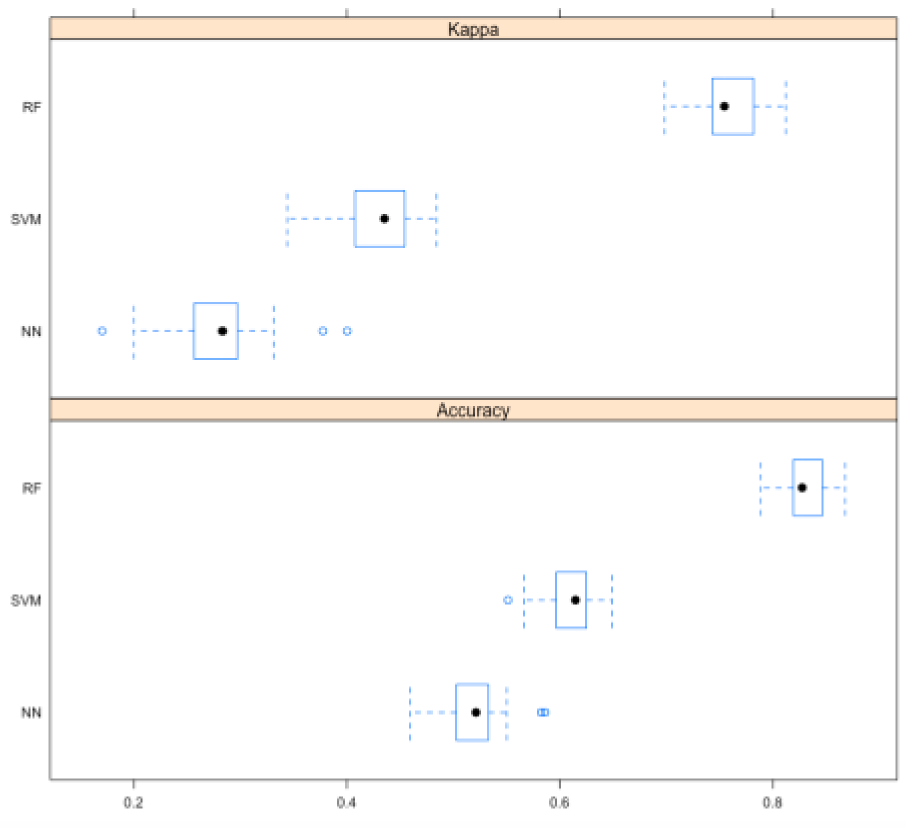
COMPARATIVE ANALYSIS –

Lastly, the team conducted comparative analysis in order to compare three different models: Random Forest, Support Vector Machines and Neural Networks.

K-Fold Cross-Validation (3 repeats):

Fist, the team conducted comparative analysis using K-Fold Cross-Validation with 3 repeats. Among the three, random forests showed the best performance. Specifically, random forest showed higher median accuracy and median kappa values. For more information, please find the result and figure below.

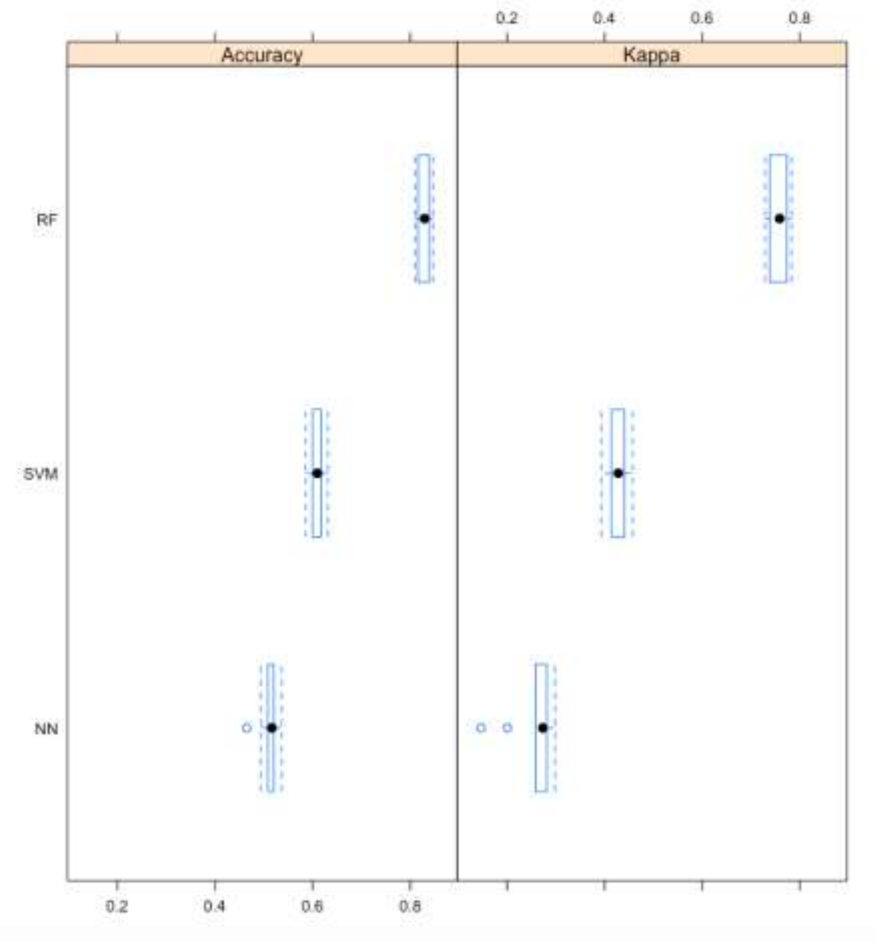
```
Models: NN, RF, SVM
Number of resamples: 30

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.4593023 0.5029042 0.5213178 0.5177036 0.5327190 0.5864078    0
RF  0.7883495 0.8195050 0.8276862 0.8316519 0.8465273 0.8679612    0
SVM 0.5514563 0.5971890 0.6149375 0.6097726 0.6249398 0.6492248    0

Kappa
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
NN  0.1701938 0.2568817 0.2831596 0.2773393 0.2966537 0.4001575    0
RF  0.6986364 0.7433535 0.7545272 0.7605831 0.7818068 0.8124424    0
SVM 0.3438115 0.4092892 0.4352384 0.4294179 0.4540021 0.4844045    0
```



K-Fold Cross-Validation:

The team also conducted the comparative analysis with K-Fold Cross-Validation without any repetition. Among the three, random forests showed the best performance. For more information, please find the result and figure below.

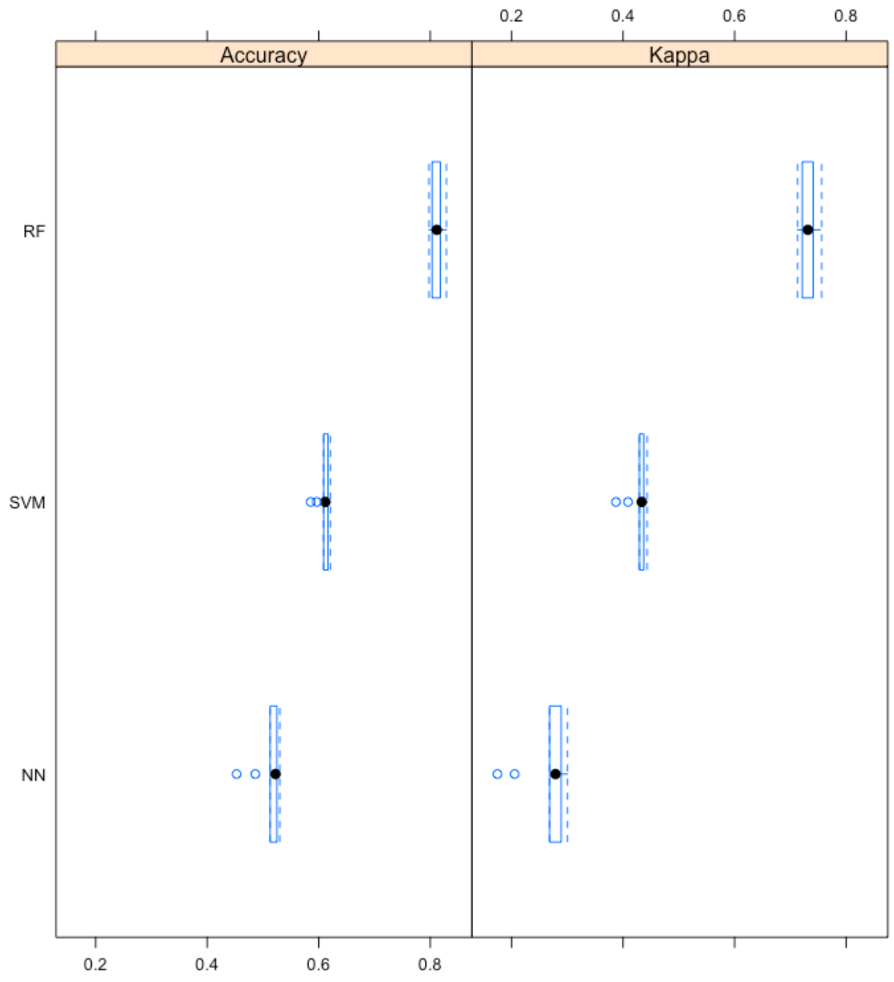
```
Call:
summary.resamples(object = results0)

Models: NN, RF, SVM
Number of resamples: 10

Accuracy
      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.4651  0.5104 0.5165 0.5116  0.5201 0.5368    0
RF  0.8097  0.8180 0.8295 0.8286  0.8379 0.8469    0
SVM 0.5853  0.6002 0.6097 0.6084  0.6164 0.6311    0

Kappa
      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.1470  0.2612 0.2734 0.2565  0.2809 0.2983    0
RF  0.7288  0.7407 0.7584 0.7565  0.7711 0.7824    0
SVM 0.3934  0.4148 0.4278 0.4271  0.4400 0.4573    0
```



<u>Bootstrap:</u>

Lastly, the team conducted the comparative analysis with bootstrap. Among the three, random forests showed the best performance. For more information, please find the result and figure below.

```
Call:
summary.resamples(object = resultsB)

Models: NN, RF, SVM
Number of resamples: 10

Accuracy
      Min.  1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.4527  0.5147 0.5225 0.5122  0.5250 0.5300    0
RF  0.7978  0.8038 0.8116 0.8116  0.8172 0.8289    0
SVM 0.5855  0.6088 0.6118 0.6095  0.6165 0.6208    0

Kappa
      Min.  1st Qu. Median   Mean 3rd Qu.   Max. NA's
NN  0.1747  0.2697 0.2788 0.2645  0.2883 0.3004    0
RF  0.7130  0.7223 0.7314 0.7323  0.7400 0.7562    0
SVM 0.3873  0.4293 0.4335 0.4277  0.4370 0.4432    0
```


Accuracy
Kappa
RF
SVM
NN
0.2
0.4
0.6
0.8
0.2
0.4
0.6
0.8

PROJECT FEEDBACK by Team 2

<u>Group 1:</u> **(Predictive Analysis of Airbnb Rentals)**

1. From their milestone 2 recap we saw that decision trees gave good accuracy of 83% for predicting if a listing would give them review.
2. Neural networks are very well explained .It can be evidently seen from neural networks as the number of hidden layers increase the accuracy also keep increasing.
3. K mean clustering gave good results for them.
4. It would have been really great had there been a screenshot of their dataset attached as it would have helped us understand the attributes of each of the dataset.

<u>Group 3:</u> **(Detecting Cyberbullying On Twitter)**

1. The topic is very interesting. It would have been great had they mentioned what a disease based bullying means.
2. Also they have only 13% of bullying tweets in their dataset .
3. Significant features for each type of classification of tweet are very clearly depicted.
4. I really liked how this group has used graphs to represent accuracy and misclassification in their classifiers.

<u>Group 4:</u> **(Predictable Consumer Complaints)**

1. Very interesting research questions are formed. Each question along with its analysis method is very clearly depicted.
2. Neural networks did not give very good results and the group mentioned the reason for it.
3. The clustering gave good results.
4. It would have been great had they evenly paced out the presentation. But overall very clear analysis of the results.

<u>Group 5:</u> **(Super Bowl babies)**

1. Very Interesting dataset and the RQs are very well chosen.
2. All the classification techniques were used for every research question.
3. A screenshot of dataset would have been useful while initially going through the research questions. Overall very good analysis of the result and project.

<u>Group 6:</u> **(Predicting the Performance of Crowdfunding Campaigns)**

1. Very clear picture given, explaining what the project is about and what their research goals are.
2. SVM for RQ1 had values for precision and recall as 0 and you were able to infer that it is because the model classified all campaigns as successful.
3. Rightly said that normalization impacted the clustering techniques used. Although another reason why clustering could have been impacted is a smaller sample size. While it was not discussed in the presentation, we hope that the smaller sample size was at least tested to be a good representation of the original dataset.

4. The DBScan noise detection was particularly interesting. Since DBScan works fairly well with outliers in the original dataset, looking at the what percentage of the outliers was noise helped enhance the reasoning behind the results.

Group 7: (Pokémon Go - Predict'em All)

1. The variables being used were not clear so it was difficult to review their analysis approach. A quick summary of the project and dataset at the start of the presentation would have been beneficial.
2. Based on what we could remember from the in-class presentations, it does seem like all the techniques were attempted accurately.
3. Results for RQ1 were pretty good and I think that is also because they knew exactly which columns in the data set had to cleaned for the model to work well.
4. In RQ4 the team could have used the entire dataset to build the neural network model instead of being content with a subset. They could have first tested by increasing the value of "stepmax" in the code to allow more steps for the algorithm to converge.

Group 8: (Airbnb Host Beneficiary System)

1. Graphs were explained really well for SVMs. This helped clearly understand the relationship between the IVs and price (DV). It was good that the student referenced the results of milestones 1 and 2, and that provided clarity in the reasoning behind the results.
2. Interesting to see how PAM clustering changed the results in the sense that although they were being clustered better, they were not as insightful. But trying PAM clustering was a good attempt.
3. It would have been useful to discuss the limitations of K-means clustering and evaluate which of those limitations were brought in by the dataset. For example, the dataset had different densities and this might have contributed to K-means not being as accurate.

Group 9: (College Scorecard Data)

1. The prediction accuracy was great for SVM in the first research question. Although it would have been interesting to know why in certain cases the linear kernel was better and why in others the non-linear one performed well.
2. The same applies to clustering. Techniques seemed to be used accurately and the team should have evaluated why the clustering results were not the best. For example, talk about outliers, different sizes, different densities and different shapes.
3. Great work done in RQ2 and the model accuracies were really good too. A probable explanation was given in the clustering section where the accuracy was not the best.
4. Great work explaining clustering in RQ3 and all the aspects of clustering were covered.
5. Accurate explanation in RQ4 around why each kernel produces different results. Same applies to clustering, the reasons behind the results were very well explained.