MILESTONE 3

PREDICTING FUTURE USER RATINGS USING TRIPADVISOR DATSET

INST737: Digging Into Data

Course Instructor:
Venessa Frias-Martinez

Group Members:
Hardik Jhaveri
Ronak Sangani
Sanjna Srivatsa

Background & Introduction

In this day and age more and more people are wanting to travel often. Technology has proliferated all aspects of our life and people want easy solutions. However travel can be tough on the pocket. People rely on user ratings and reviews to make an informed decision. This unregulated review system can be chaotic at times. People use network orchestrating platforms that enable them to find the best fit for their travel needs.

'TripAdvisor, Inc. is an American travel website company providing reviews of travel-related content. It also includes interactive travel forums. TripAdvisor was an early adopter of user-generated content. The website services are free to users, who provide most of the content, and the website is supported by an advertising business model.'[5] TripAdvisor has to constantly innovate and build to improve their recommendation system. In this scenario, having a predictive analysis can go a long way in building a futuristic business model. A predictive rating is indicative of how the business is going to trend. Accordingly, different measures can be taken to correct what users dislike based on the analysis we propose to do.

Research Question

What are the significant independent variables in predicting future ratings for the hotels on TripAdvisor platform?

We will consider individual ratings for factors such as

- service of staff
- room ambiance and comfort
- location of hotel
- value for money
- price range of hotel
- Sentiment analysis of review.

Milestone 1

Network orchestrating platforms like TripAdvisor publish multiple clients on their network. Users tend to like some, and dislike some. The recommendation system of these platforms can do better in recommending the best performing clients. This enables the company to review its operation model and make smart business decisions to have an agile and adaptive environment. This way the users have a better experience and feel the urge to keep coming back for what they like, and only what they like. We are combining quantitative analysis with sentiment analysis, this will strengthen our result.

Balancing dataset and reiterated results

Our dataset was highly skewed towards the overall rating 5. Using ubSMOTE we balanced to get a more equitable distribution.

```
> summary(old)
 Service    Value      Rooms      Overall    Location   price        Sentiment
 1: 2795   1: 2582   1: 1991   1: 1234   1:  576   1: 7050   Min.    :0.0000477
 2: 2711   2: 3485   2: 3045   2: 1776   2: 1197   2:13199   1st Qu.:0.7981865
 3: 6330   3: 8492   3: 7586   3: 4905   3: 4818   3: 9237   Median :0.9804477
 4:13574   4:16805   4:16773   4:13286   4:13437   4:14149   Mean   :0.7973045
 5:32278   5:26324   5:28293   5:36487   5:37660   5:14053   3rd Qu.:0.9948483
                                                           Max.   :0.9999991
>
```

```
> summary(new)
 Service   Value     Rooms     Overall   Location price       Sentiment
 1:1739   1:1707   1:1509   1:1234   1: 377   1:1844   Min.    :0.0000595
 2:1328   2:1675   2:1682   2:1776   2: 532   2:2659   1st Qu.:0.0481860
 3:2123   3:2389   3:2307   3:2377   3:1500   3:1770   Median :0.7491953
 4:2423   4:2546   4:2632   4:2473   4:2696   4:2617   Mean   :0.5607086
 5:3251   5:2547   5:2734   5:3004   5:5759   5:1974   3rd Qu.:0.9861066
                                                      Max.   :0.9996670
  .
```

As we can see from the distribution that our data has now become quite balanced with around 10,800 data points from earlier 57,000.

Milestone 2 - Revisited

LINEAR & MULTIVARIATE REGRESSION:

In milestone 2 the data was unbalanced. Rating 5 had lot more data points than any other rating value. So after balancing the dataset, we ran the regression model again. . In the case of our problem, our dependent variable is Overall rating which an user will give to share experience of the hotel as a whole. Our independent variables are individual rating which a user will give to Service, Value, Rooms, Location along with price of the hotel and values generated by sentiment analysis of the reviews given to the hotel.

In our regression model, we have carried out regression iteration three times with different sets of testing and training data. In one iteration we have computed linear regression for each independent variable with respect to the dependent variable. Then we computed multivariate regression wherein we tried to determine the effect of all independent variable on dependent variable. We also tried different combinations of independent variables to see the effect it has on the dependent variable to determine the best fit model for making predictions.

We have divided the dataset into training and testing sets thrice by selecting rows from entire dataset so that there is no bias in the report. First we data set was divided as 75% training data and 25% testing data. Then the dataset was divided into ratio of 80-20 and 66-34.

To serve the purpose of this milestone we have reported only the correlations from the unbalanced data and balanced dataset. The correlation mentioned below are the correlation of each and every independent variable with respect to Overall rating that an user would give to a hotel suggested by TripAdvisor.

This table consists of all previous values of correlation and mean squared error with unbalanced dataset

	1st Iteration		2nd Iteration		3rd Iteration	
Variable	Correlation	MSE	Correlation	MSE	Correlarion	MSE
Service	0.67	0.475	0.66	0.471	0.66	0.476
Value	0.627	0.529	0.6277	0.5153	0.626	0.518
Rooms	0.741	0.392	0.742	0.381	0.737	0.389
Location	0.447	0.69	0.44	0.68	0.44	0.687

Price	0.02	0.87	0.01	0.85	0.018	0.85
Sentiment	0.61	0.54	0.603	0.538	0.608	0.539

The table given below depicts the value of all independent variables against the dependent variable in a balanced dataset. It can be seen evidently that the value of correlation increases for all variables. The value of mean squared error also increases because the data points have reduced. Hence the probability of an error occurring is high.

	1st Iteration		2nd Iteration		3rd Iteration	
Variable	Correlation	MSE	Correlation	MSE	Correlation	MSE
Service	0.695	0.9292	0.71	0.891	0.70	0.917
Value	0.691	0.9392	0.707	0.897	0.705	0.904
Rooms	0.794	0.663	0.803	0.637	0.807	0.624
Location	0.473	1.395	0.486	1.372	0.483	1.379
Price	0.047	1.794	0.044	1.794	0.04	1.795
Sentiment	0.680	0.967	0.695	0.927	0.702	0.911

From the new regression model which was run on all independent variables, we can conclude that whatever ratings that the user gives to the rooms has most impact on the Overall rating that the user will give to hotel suggested by TripAdvisor.

	1st Iteration		2nd Iteration		3rd Iteration	
Variable	Correlation old	Correlation new	Correlation old	Correlation new	Correlation old	Correlation new
Overall~ Service+Value+Rooms+Location+price+Sentiment	0.787	0.826	0.786	0.833	0.781	0.839
Overall~ Service+Value	0.707	0.74	0.704	0.755	0.701	0.753
Overall~ Service+Value+Rooms	0.783	0.814	0.782	0.83	0.777	0.833
Overall~ Service+Value+Rooms+Location+price	0.782	0.828	0.785	0.831	0.779	0.834
Overall~ Service+Value+Rooms+Location+Sentiment	0.787	0.829	0.786	0.831	0.781	0.835

From Multivariate regression it is evident that the new values obtained after balancing the data greatly help in increasing the accuracy of the models. The highest value of correlation is 0.84 which has a significant increase from 0.78. The best model is the one which takes into account all independent variables and runs it against the dependent variable. Since all the values of correlation were near to each other for many models we used ANOVA to compare models. It was using ANOVA we derived that the best model to give good and consistent predictions is the one which takes into account all the variables.

LOGISTIC REGRESSION

As documented in Milestone 2 we have used General Linear Model to perform logistic regression. There has been a slight deterioration in correlation after balancing the dataset. We shall summarize work and results from milestone 2 and then show how the results vary on comparison.

BEFORE BALANCING

Model:

Predictor	Coefficient	P value	Significant or not?
Service	0.60481	2e-16	Very significant
Value	0.29481	2e-16	Very significant
Rooms	1.21894	2e-16	Very significant
Location	0.18949	2e-16	Very Significant
Sentiment	0.17041	0.0031	Significant
Price	-0.06237	1.73e-05	Very significant

```
Call:
glm(formula = Overall ~ Service + Value + Rooms + Location +
    Sentiment + price, family = "binomial", data = train_trip,
    maxit = 100)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2052   0.1175   0.1448   0.3064   2.8921

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.51624    0.12821 -50.827  < 2e-16 ***
Service      0.60481    0.02030  29.799  < 2e-16 ***
Value        0.29481    0.02286  12.894  < 2e-16 ***
Rooms        1.21894    0.02390  51.003  < 2e-16 ***
Location     0.18949    0.02216   8.550  < 2e-16 ***
Sentiment    0.17041    0.05767   2.955  0.00313 **
price       -0.06237    0.01451  -4.298 1.73e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34918  on 43265  degrees of freedom
Residual deviance: 17309  on 43259  degrees of freedom
AIC: 17323

Number of Fisher Scoring iterations: 6

> |
```

Intercept:

The intercept for the model is -6.516.

Odds ratio:

ATTRIBUTE	ODD RATIO
LOCATION	1.208638424
SERVICE	1.830901151
SENTIMENT	1.185790050
PRICE	0.939537020
VALUE	1.342868359
ROOMS	1.21894182

Correlation:

There is a positive correlation of 0.83 between the predicted Overall rating and the training Overall rating.

AFTER BALANCING:

Model:

Predictor	Coefficient	P value	Significant or not?
Service	0.33507	e-13	Very significant
Value	0.13859	0.0178	Slightly significant
Rooms	1.16679	2.2e-12	Very significant
Location	0.13508	0.0001	Very Significant
Sentiment	1.02315	3.4e-07	Very Significant
Price	-0.11689	0.0001	Very significant

```
> mlogit<-glm(Overall~Service + Value + Rooms + Location + Sentiment + price, data=train_trip,family="binomial"
,maxit=100)
> summary(mlogit)

Call:
glm(formula = Overall ~ Service + Value + Rooms + Location +
    Sentiment + price, family = "binomial", data = train_trip,
    maxit = 100)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-3.4795876   0.0400582   0.0910580   0.3143276   1.6168259

Coefficients:
              Estimate  Std. Error   z value               Pr(>|z|)
(Intercept) -2.18553197  0.17688215 -12.35587 < 0.000000000000000222 ***
Service      0.33507162  0.04700858   7.12788      0.0000000000010193 ***
Value        0.13859252  0.05850940   2.36872              0.01784966 *
Rooms        1.16679559  0.06207946  18.79519 < 0.000000000000000222 ***
Location     0.13508751  0.03553859   3.80115              0.00014403 ***
Sentiment    1.02315216  0.20083749   5.09443      0.0000003497953100 ***
price       -0.11689760  0.03067678  -3.81062              0.00013862 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5769.9908  on 8147  degrees of freedom
Residual deviance: 3389.2267  on 8141  degrees of freedom
AIC: 3403.2267

Number of Fisher Scoring iterations: 7

>
```

Intercept:

The intercept for the model is -2.1855.

We can see that sentiment that was earlier just significant is now very significant and value which was earlier very significant is now only slightly significant. Although there are slight changes, the overall significance of the predictors do not change.

Odds ratio:

ATTRIBUTE	ODD RATIO
LOCATION	1.1446369445
SERVICE	1.3980405117
SENTIMENT	2.7819500968
PRICE	0.8896762907
VALUE	1.1486559469
ROOMS	3.2116845880

Correlation:

There is a positive correlation of 0.71 between the predicted Overall rating and the training Overall rating. We can notice that there is a deterioration of correlation, but even so 0.71 is an acceptable number.

Naïve Bayes

We use the Naïve Bayes classification technique to classify our dependent variable which is overall rating of the hotel and create a probabilistic model. We have divided our dataset into training and testing set and trained the classifier. The proportion in which we divided our dataset is 75% training to 25% testing. We checked the proportion of the overall variable if it is similar in both the training and testing data and found it to be in proportion.

```
> rating_classifier <- naiveBayes(train_data_bayes,train_data_bayes$Overall)
> rating_classifier

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = train_data_bayes, y = train_data_bayes$Overall)

A-priori probabilities:
train_data_bayes$Overall
        1         2         3         4         5
0.1136475 0.1634757 0.2188267 0.2275405 0.2765096

Conditional probabilities:
                         Service
train_data_bayes$Overall     [,1]      [,2]
                       1 1.720302 1.0431544
                       2 2.293544 1.1348666
                       3 2.899607 1.1456818
                       4 3.889428 0.9562284
                       5 4.663560 0.7167677

                         Value
train_data_bayes$Overall     [,1]      [,2]
                       1 1.617711 0.8703336
                       2 2.121622 1.0308164
                       3 2.813236 1.1052389
                       4 3.788026 0.9498187
                       5 4.430093 0.8002663

                         Rooms
train_data_bayes$Overall     [,1]      [,2]
                       1 1.461123 0.8014180
                       2 2.151652 0.9492638
                       3 2.874369 0.9325931
                       4 3.796656 0.7817188
                       5 4.702619 0.5672980
```

```
> rating_classifier <- naiveBayes(train_data_bayes,train_data_bayes$rating)
```

Here we use the naiveBayes command and we pass the training data and the dependent variable. We then use this model to predict the ratings in our testing data. We pass the model and the testing data to the predict function.

```
> rating_classifier_pred <- predict(rating_classifier,test_data_bayes)
```

Here we can see that the model predicted that it is a 5 star hotel and it is truly a 5 star hotel 704 time. Similarly we can see the performance of the model for other ratings, both truly predicted and falsely predicted.

```
Total Observations in Table:  2716


             | actual
   predicted |         1 |         2 |         3 |         4 |         5 | Row Total |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           1 |       301 |         0 |         2 |        13 |         2 |       318 |
             |     0.977 |     0.000 |     0.003 |     0.021 |     0.003 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           2 |         0 |       433 |         0 |        17 |        10 |       460 |
             |     0.000 |     0.975 |     0.000 |     0.027 |     0.013 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           3 |         4 |         0 |       581 |         0 |        32 |       617 |
             |     0.013 |     0.000 |     0.978 |     0.000 |     0.043 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           4 |         2 |         5 |         0 |       589 |         3 |       599 |
             |     0.006 |     0.011 |     0.000 |     0.952 |     0.004 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           5 |         1 |         6 |        11 |         0 |       704 |       722 |
             |     0.003 |     0.014 |     0.019 |     0.000 |     0.937 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
Column Total |       308 |       444 |       594 |       619 |       751 |      2716 |
             |     0.113 |     0.163 |     0.219 |     0.228 |     0.277 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
```

To improve our model we applied a smoothing factor, called the Laplace smoothing which corresponds to a uniform prior over all classes.

```
> rating_classifier_laplace <- naiveBayes(train_data_bayes,
train_data_bayes$rating,laplace = 1)
```

After training the new model, we evaluated it to get the following results. We can see that the accuracy has increased by a small percentage. The model truly predicted the 5 star hotels 708 times. Similarly, we can use the results for other ratings in the figure below.

```
Total Observations in Table:  2716


             | actual
   predicted |         1 |         2 |         3 |         4 |         5 | Row Total |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           1 |       302 |         0 |         2 |        18 |         4 |       326 |
             |     0.981 |     0.000 |     0.003 |     0.029 |     0.005 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           2 |         0 |       433 |         0 |         9 |        15 |       457 |
             |     0.000 |     0.975 |     0.000 |     0.015 |     0.020 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           3 |         3 |         0 |       587 |         0 |        23 |       613 |
             |     0.010 |     0.000 |     0.988 |     0.000 |     0.031 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           4 |         2 |         5 |         0 |       592 |         1 |       600 |
             |     0.006 |     0.011 |     0.000 |     0.956 |     0.001 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           5 |         1 |         6 |         5 |         0 |       708 |       720 |
             |     0.003 |     0.014 |     0.008 |     0.000 |     0.943 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
Column Total |       308 |       444 |       594 |       619 |       751 |      2716 |
             |     0.113 |     0.163 |     0.219 |     0.228 |     0.277 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
```

Conclusion:

We can state that the accuracy given by the Naïve Bayes classifier after balancing the data set is more. The accuracy we get in the first model is 96.02% compared to the accuracy of 96.54% using the laplace smoothing factor.

Decision Trees, Bagging and Random Forest

After balancing our dataset we recomputed decision trees, bagging and random forest techniques to receive the following results:

Model	Unbalanced Dataset		Balanced Dataset	
	Accuracy	Error	Accuracy	Error
Decision Tress	74.6	25.4	66.4	33.6
Decision Tree (Trials = 10)	74.7	25.3	64.1	33.9
Bagging (Trees = 100)	75.33	24.67	61.97	38.03
Bagging (Trees = 250)	75.54	24.46	61.56	38.44
Bagging (Trees = 500)	75.61	24.39	61.12	38.88
Random Forest (M=3, Trees = 250)	75.83	24.17	64.29	35.71
Random Forest (M=3, Trees = 500)	75.88	24.12	64.32	35.68
Random Forest (M=3, Trees = 750)	75.9	24.1	64.51	35.49

We can see that the accuracy has decreased in all the models. This is because our data was highly skewed when it was unbalanced. After balancing the data, accuracy is observed highest in the decision tree model. We can also notice that in the bagging model as the number of trees increases the accuracy decreases. On the other hand in the random forest model where we have taken m = 3, the accuracy increases by a small amount as the trees are increased.

Milestone 3

For milestone 3 we have balanced our dataset and have received better and consistent results. We have pe rformed Support Vector Machines, Neural Networks and Clustering (k-means, hierarchical, db-scan) as part of this milestone. Comparison has been carried out amongst the models using Caret package.

Support Vector Machine

Support Vectors are discriminative classifiers which are used to define boundaries to separate two or more classes. Our dataset has ratings 1-5 which is why we decided to conduct multiclass classification for our project. The kernels which we have used are VanillaDot, Polydot and RBFdot. VanillaDot is a linear kernel while Polydot and RBFdot are non-linear kernels.

We ran all the models for two sets of dataset. The models were run for binary data as well as multiclass data. For binary data, we converted the data into 0,1. Rating 1,2,3 were 0 and 4,5 were 1.

First we ran SVM for binary dataset.

1. **Results for Binary VanillaDot are as follows:**

Support Vectors: 2563

Confusion Matrix:

```
          Reference
Prediction    0    1
         0 1140  206
         1  206 1164

Value of True and False
FALSE  TRUE
  412  2304
```

Our confusion matrix shows that the number of the error rate is very less. There are only 206 wrong predictions of 1 and 0 which are wrong. Out of the total predicted values, the false values are 412 compared to 2304 which are correct.

The Recall: 0.8470

The Precision: 0.8470
The Accuracy: 0.8483
The Specificity: 0.8496
F1 measure: 0.4235 (calculated by formula (2 * precision * recall) / (precision + recall))

2. **Results for Binary PollyDot kernel are as follows:**

Support Vectors: 2564

Confusion Matrix:

```
          Reference
Prediction    0    1
         0 1140  206
         2  206 1164

Value of True and False
FALSE  TRUE
  412  2304
```

Our confusion matrix shows that the number of the error rate is very less. There are only 206 wrong predictions of 1 and 0 which are wrong. Out of the total predicted values, the false values are 412 compared to 2304 which are correct.

The Recall: 0.8470
The Precision: 0.8470
The Accuracy: 0.8483
The Specificity: 0.8496
F1 measure: 0.4235 (calculated by formula (2 * precision * recall) / (precision + recall))

There was no difference in results obtained after running polydot and vanilladot. Even thought the kernels are different (one is linear and other non-linear) the results were same.

3. **Results for Binary RBFdot are as follows:**

Support Vectors: 2341

Confusion Matrix:

```
          Reference
Prediction    0    1
         0 1178  216
         1  168 1154

Value of True and False
FALSE  TRUE
  384  2332
```

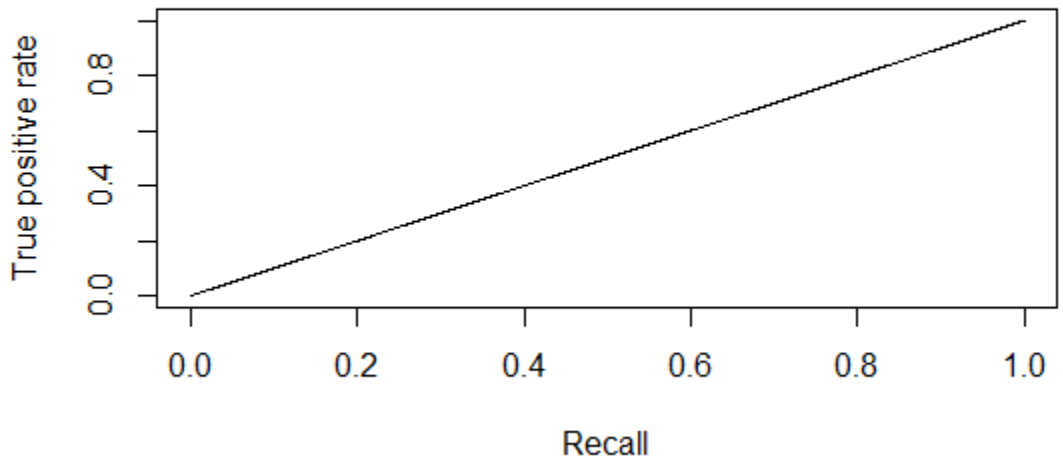
Our confusion matrix shows that the number of the error rate is very less. There are only 216 wrong pre dictions of 0 and 168 predictions for 1 are wrong. Out of the total predicted values, the false values are 3 84 compared to 2332 which are correct. This shows that RBFdot has increased accuracy of the model.

The Recall: 0.8752
The Precision: 0.8451
The Accuracy: 0.8486
The Specificity: 0.8423
F1 measure: 0.4322 (calculated by formula (2 * precision * recall) / (precision + recall))

From all the models that were reported above, it can be very well seen that RBFdot model yields the bes t results. The level accuracy is very high for this model.

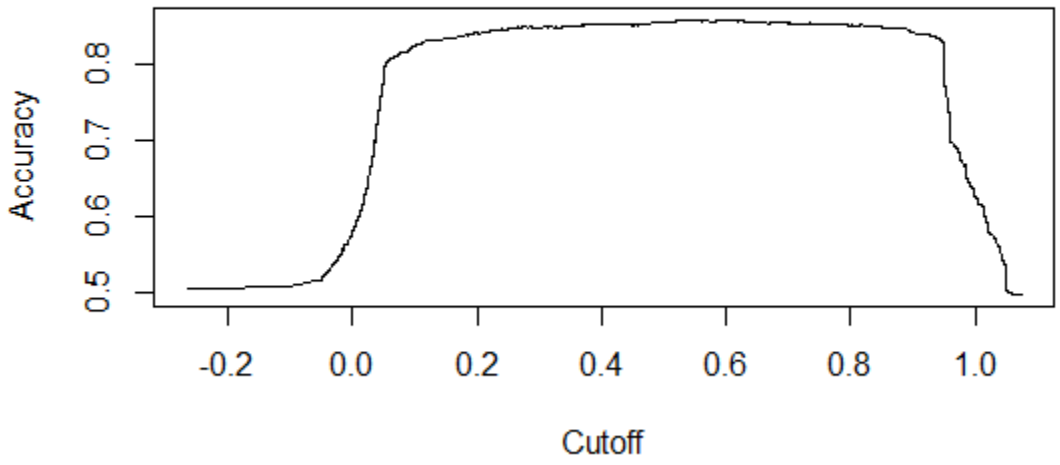
Since RBFdot was the model reported, we plotted the precision recall plot, the accuracy plot and the ROC curve for this model.

Precision Recall plot:



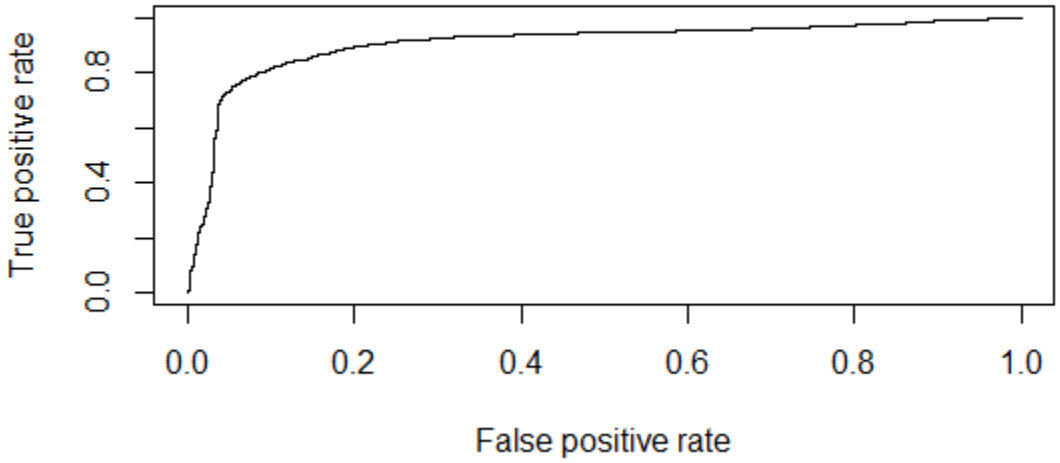
The precision recall plot shows that there is a linear slope which means that the results are accurate and predication rate is good.

Accuracy plot:



The accuracy curve shows that the value of accuracy is near 1 for majority of the plot.

ROC curve:



ROC (Receiver Operating Characteristic) curve is plotted for true positive rate against false positive rate for different possible cutoff points. The close the curve is to left hand corner and top corner, more accurate the test is. In our plot it can be seen that the curve is located close to the left hand corner and near to the top corner.

SVM for Multiclass data. We have used One Vs All approach.

1. **Multiclass SVM using Vanilladot:**

Support Vectors: 6594

Confusion Matrix:

```
                Reference
Prediction      1   2   3   4   5
         1  187 117  39  10   0
         2   72 149  70  18   5
         3   46 143 287 128  41
         4    3  31 182 373 155
         5    0   4  16  90 550
```

```
Value of True and False
FALSE  TRUE
 1170  1546
```

Confusion matrix shows that the error rate is quite high. It can be observed that the error decreases as it goes away from the corresponding rating value. For example for prediction for number 3, most of the values which are not predicted correctly are of 2 and 4. The number of prediction for number 1 and 5 are significantly smaller.

```
Class:                 1       2         3       4       5
Recall           0.60714  0.33559   0.4832  0.6026  0.7324
Specificity      0.93106  0.92738   0.8313  0.8231  0.9440
Precision        0.52975  0.47452   0.4450  0.5013  0.8333
Accuracy         0.76910  0.63148   0.6572  0.7128  0.8382
F1               0.3444   0.1966    0.2316  0.2736  0.3897
```

The Overall Accuracy of this model is 0.5692 which is very small. This small value of accuracy makes this model not acceptable to make predictions.

2. **Multiclass SVM using Polydot:** (Non-linear) - This model gives exact same results like the Vanilla dot model except for number of support vectors

Support Vectors: 6577

Confusion Matrix:

```
                Reference
Prediction      1   2   3   4   5
         1  187 117  39  10   0
         2   72 149  70  18   5
         3   46 143 287 128  41
         4    3  31 182 373 155
         5    0   4  16  90 550
```

```
Value of True and False
FALSE  TRUE
 1170  1546
```

Confusion matrix shows that the error rate is quite high. It can be observed that the error decreases as it goes away from the corresponding rating value. For example for prediction for number 3, most of the values which are not predicted correctly are of 2 and 4. The number of prediction for number 1 and 5 are significantly smaller.

Class:	1	2	3	4	5
Recall	0.60714	0.33559	0.4832	0.6026	0.7324
Specificity	0.93106	0.92738	0.8313	0.8231	0.9440
Precision	0.52975	0.47452	0.4450	0.5013	0.8333
Accuracy	0.76910	0.63148	0.6572	0.7128	0.8382
F1	0.3444	0.1966	0.2316	0.2736	0.3897

The Overall Accuracy of this model is 0.5692 which is very small. This small value of accuracy ma kes this model not acceptable to make predictions.

3. Multiclass SVM using RBFdot:

Support Vectors: 5938

Confusion Matrix:

Prediction \ Reference	1	2	3	4	5
1	183	109	38	7	0
2	83	160	82	27	9
3	38	145	320	115	42
4	3	26	131	382	120
5	1	4	23	88	580

Value of True and False

FALSE	TRUE
1091	1625

Confusion matrix shows that the error rate is quite high. It can be observed that the error decreases as it goes away from the corresponding rating value. For example for prediction for number 3, most of the values which are not predicted correctly are of 2 and 4. The number of prediction for number 1 and 5 are significantly smaller.

Class:	1	2	3	4	5
Recall	0.59416	0.36036	0.5387	0.6171	0.7723
Specificity	0.93605	0.91153	0.8398	0.8665	0.9410
Precision	0.54303	0.44321	0.4848	0.5770	0.8333
Accuracy	0.76510	0.63595	0.6892	0.7418	0.8566
F1	0.2837	0.1987	0.2551	0.2981	0.4008

The Overall Accuracy of this model is 0.5983 which is small. This accuracy is better than the accuracy giv en by previous two models. This small value of accuracy makes this model not acceptable to make predi ctions.

By running all the models in SVM, for binary as well as multiclass approach, the model with best accuracy is rbfdot kernel (radial kernel). If accuracy of each and every class is compared then class 5 gives the highest accuracy followed by class 1. Class 2 and class 3 give almost same accuracy.

Neural networks

To perform neural network operations we have used the package neuralnet. This package is used to execute the models. To simplify plots and complicated representations of neural networks we have used the package devtools. Devtools takes away the unnecessary information and gives clear representation of only the neural net.

1) To begin with we use a normalization function to normalize IVs:

```
normalize <- function(x){
 return((x-min(x))/(max(x)-min(x)))
}
```

We got an error when we tried to normalize all IVs at one shot, but that was rectified when we individually normalized each IV.

2) The following summary confirms that the range lies between 0 and 1:

```
> summary(datafinal$Overall)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000000 0.2500000 0.7500000 0.5975009 1.0000000 1.0000000
```

Since our dataset was ordered according to the overall rating, we randomly divided our dataset to training and testing dataset.

3) Now we will build a multi-layer perceptron

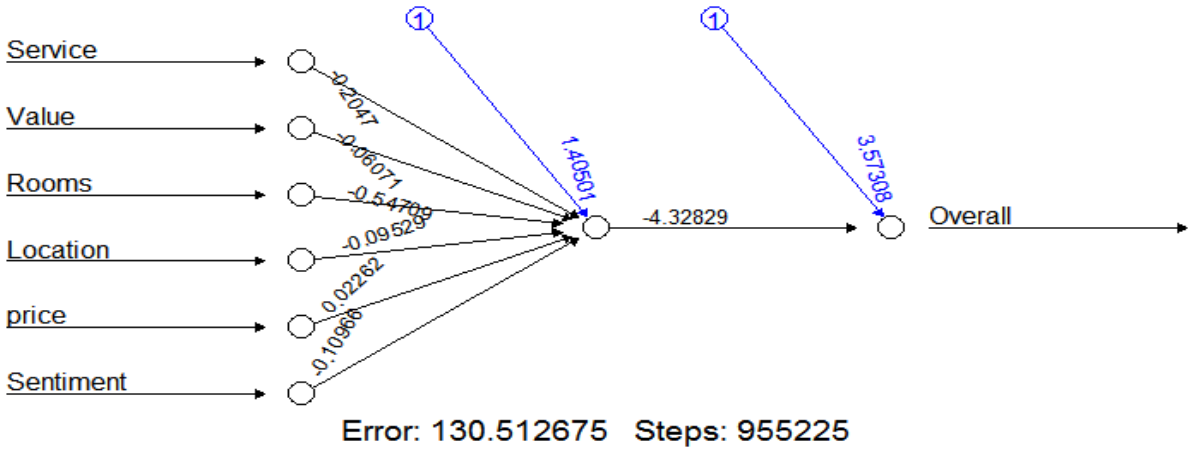
3.1) Our Model 1 is a neural net with one hidden node as default.

a) Building the model:

```
NNmodel <- neuralnet(formula = Overall~Service+Value+Rooms+Location+price+Sentiment, data = train_trip, act.fct = "logistic", stepmax = 1e6)
```

Here we have considered all IVs (as all of them are significant) and used it on the training dataset. We have implemented stepmax as 1e6 to avoid the function call ending before the entire model was executed.

b) Plotting the model:



We also get a list of neurons in the model:

```
6912    1 0.6358650244
6916    1 0.6685722473
6920    1 0.6322676027
6924    1 0.6545323327
6928    1 0.6530969007
6932    1 0.6357687822
6936    1 0.6358669623
6940    1 0.6683900743
6944    1 0.6322676027
6948    1 0.6544067710
6952    1 0.6530969007
6956    1 0.6357687822
6960    1 0.6358809910
6964    1 0.6685722448
6968    1 0.6322676027
6972    1 0.6543405790
6976    1 0.6530969007
6980    1 0.7144233607
6984    1 0.6509829397
6988    1 0.6814276788
6992    1 0.6416721192
6996    1 0.6537765691
7000    1 0.6531560513
7004    1 0.6447364126
7008    1 0.6678196491
7012    1 0.6908048254
7016    1 0.6395745929
7020    1 0.6585130371
7024    1 0.6865798042
7028    1 0.6490513095
7032    1 0.6321577816
7036    1 0.6360145715
7040    1 0.6481312932
7044    1 0.6528815327
7048    1 0.6529345527
7052    1 0.6475500192
7056    1 0.6919132953
7060    1 0.6628236006
7064    1 0.6479788591
7068    1 0.6494096788
7072    1 0.6565545726
7076    1 0.6447868083
7080    1 0.6796548229
7084    1 0.6323431153
7088    1 0.6326491292
7092    1 0.6377313612
7096    1 0.6870906589
7100    1 0.6600379799
7104    1 0.6446271783
7108    1 0.6485864754
```

c) Predicting the strength of the model using testing set:

We use the following code to validate our model and test it on our testing set:

NNmodel_results <- compute(NNmodel,test_trip[-3])

NNmodel_results$neurons

We have not considered the final label column 3 to avoid the error of neurons and weights not conforming.

d) Correlation:

We find the predicted strength of values by:

predicted_rep11 <- NNmodel_results$net.result

And now we can use these values to find the correlation between results of training and testing module:

cor(predicted_rep11,datafinal$Overall)

Our correlation for this particular model is 0.818

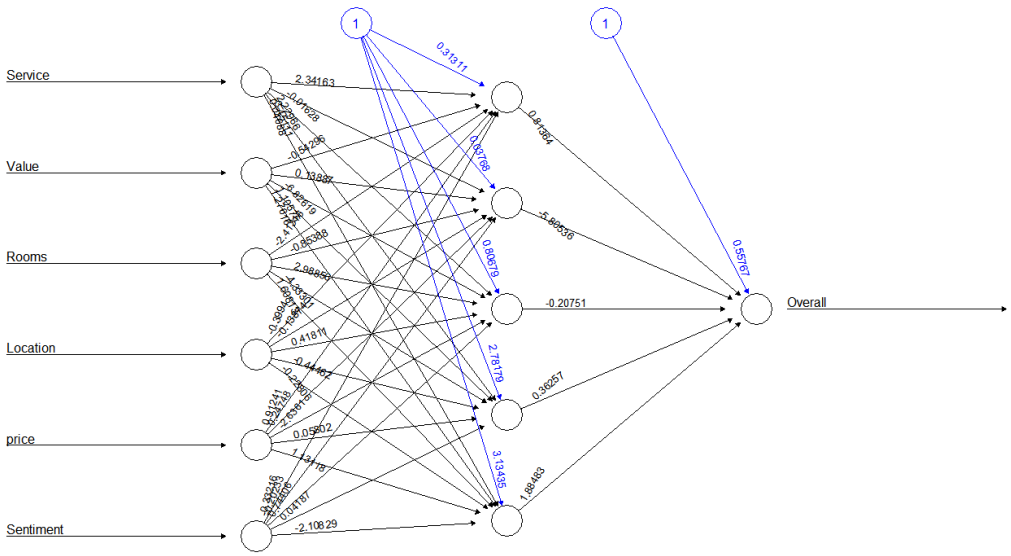
We can now repeat the following steps for all models to find the best model and improve results

3.2) Model 2 – with 5 hidden nodes instead of 1

a) Building the model:

NNmodel12 <- neuralnet(formula = Overall~Service+Value+Rooms+Location+price+Sentiment, data = train_trip, hidden = 5, act.fct = "logistic", stepmax = 1e6)

b) Plotting the model:

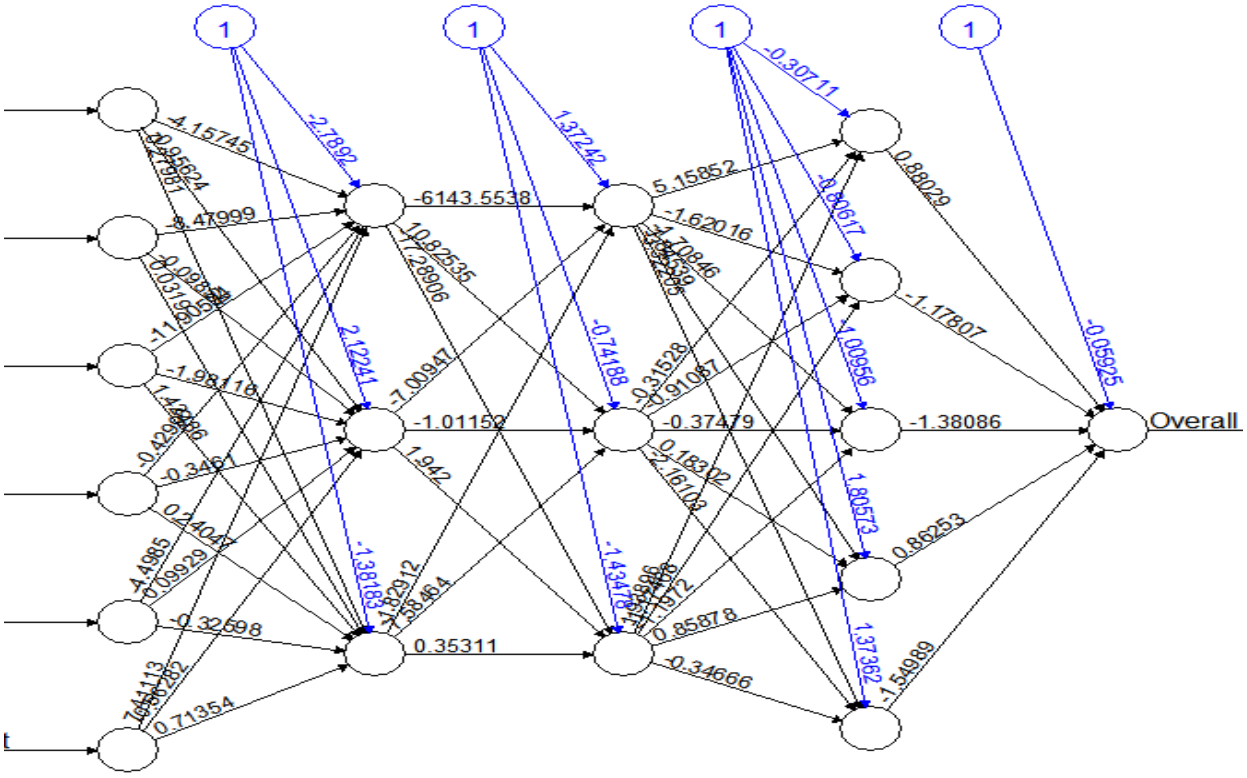


3.3) Model 3 – neural net with 3 hidden layers, 1st and 2nd with 3 nodes, 3rd with 5 neurons/nodes.

a) Building a model

Q1NNmodel13 <- neuralnet(formula = Overall~Service+Value+Rooms+Location+price+Sentiment, data = train_trip, hidden = c(3,3,5), act.fct = "logistic")

b) Plotting the model



Error: 127.515544 Steps: 76455

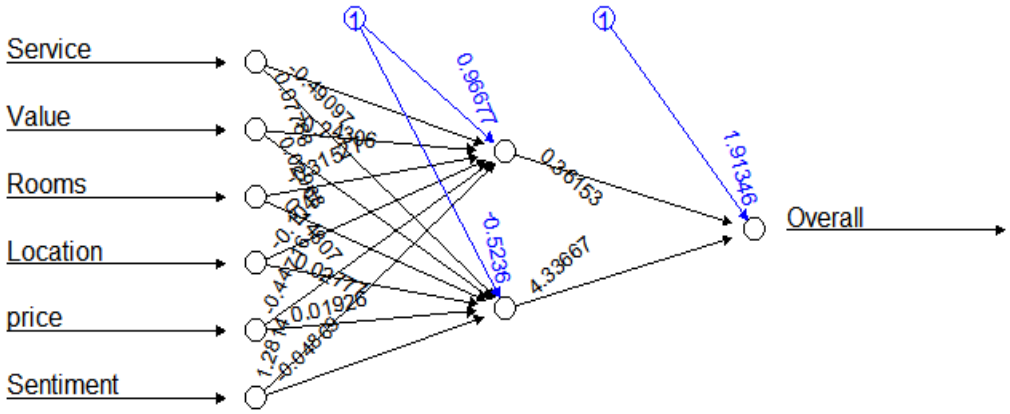
As we can see the model above is very hard to read and decipher simple results, therefore using devtools package we obtain a cleaner model.

3.4) Model 4 – neural net with 2 hidden nodes and tangent hyperbolicus activation function

a) Building the model

Q1NNmodel14 <- neuralnet(formula = Overall~Service+Value+Rooms+Location+price+Sentiment, data = train_trip, hidden = 2, act.fct = "tanh",stepmax = 1e8)

b) plotting the model:

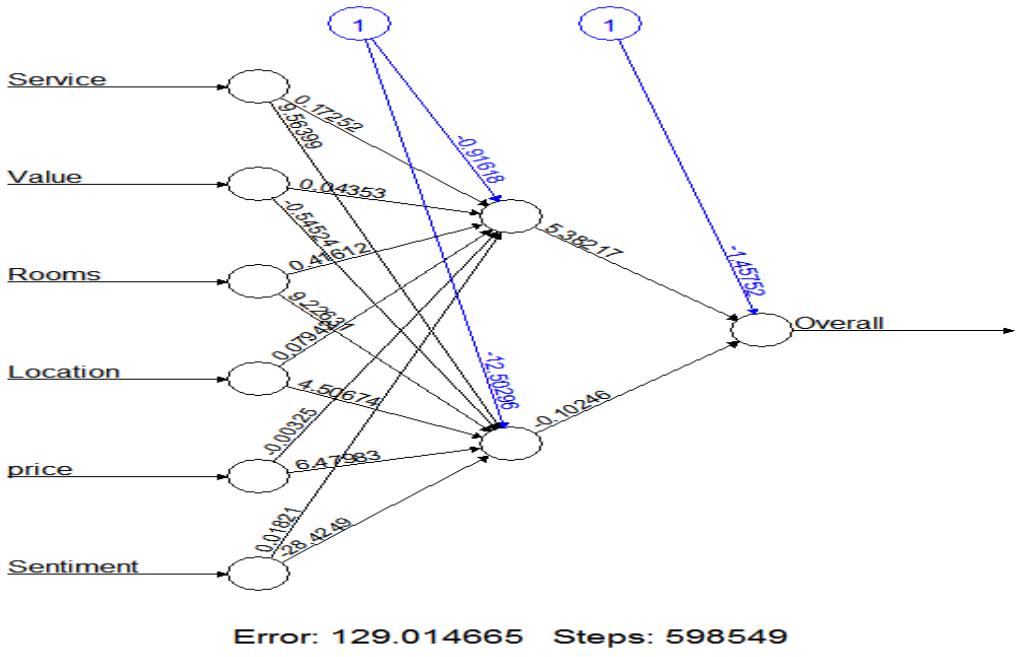


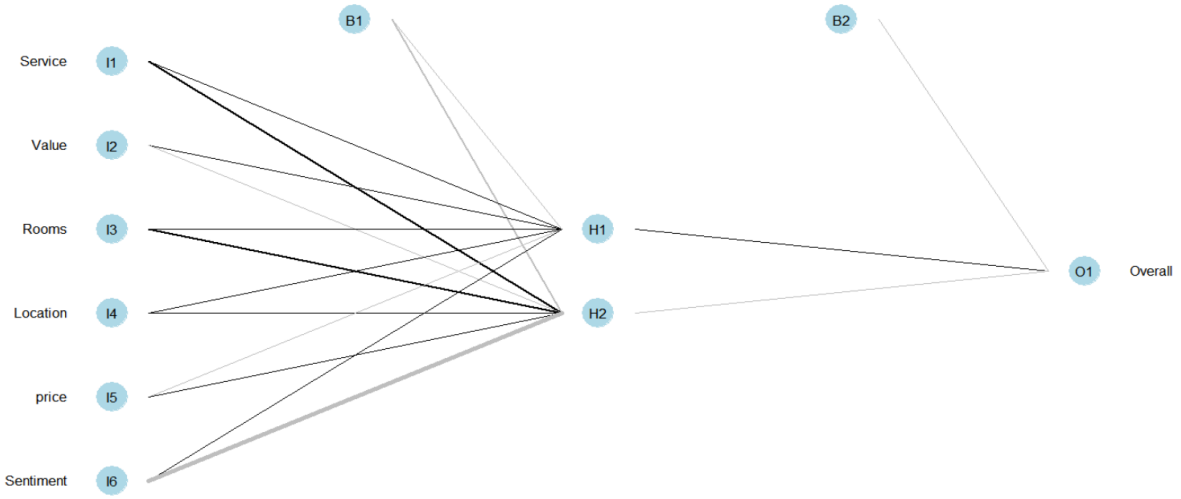
3.5)Model 6- neural net with 2 hidden nodes and logistic activation function

a) Building the model

```
Q1NNmodel15 <- neuralnet(formula = Overall~Service+Value+Rooms+Location+price+Sentiment,
data = train_trip, hidden = 2, act.fct = "logistic",stepmax = 1e6)
```

b) plotting the model



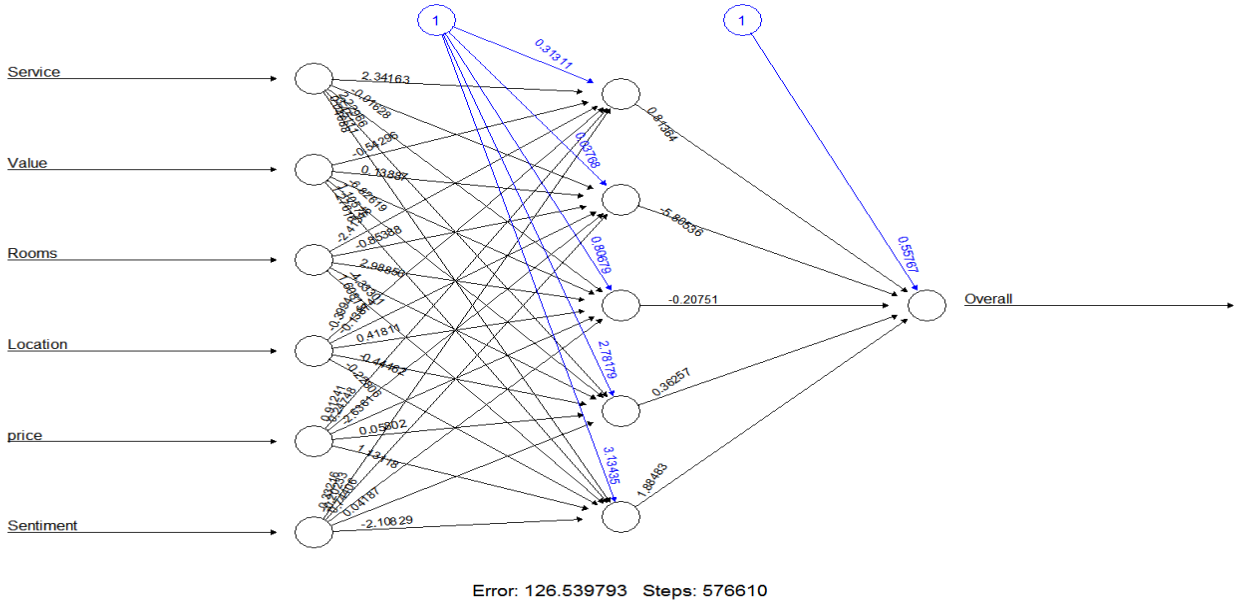


3.6) Model 6- neural net with 2 hidden layers (2 nodes in 1st and 3 nodes in 2nd) and tangent hyperbolicus activation function

a) Building the model:

```
Q1NNmodel16 <- neuralnet(formula = Overall~Service+Value+Rooms+Location+price+Sentiment, data = tra
in_trip, hidden = c(2,3), act.fct = "tanh",stepmax = 1e6)
```

b) Plotting the model:



Correlation table:

We have tabulated all the correlation values in a table:

Model	correlation
NN with 1 hidden node - default	0.818
NN with five hidden node	0.802
NN with 3 hidden layers	0.764
NN with 2 hidden nodes and tangent hyperbolicus activation function	0.836
NN with 2 hidden nodes and logistic activation function	0.735
NN with 2 hidden layers (2 nodes in 1st and 3 nodes in 2nd) and tangent hyperbolicus activation function	0.793

We can see that the best fit model is model 4 – neural net with 2 hidden nodes and tangent hyperbolicus activation function with a correlation of 0.836

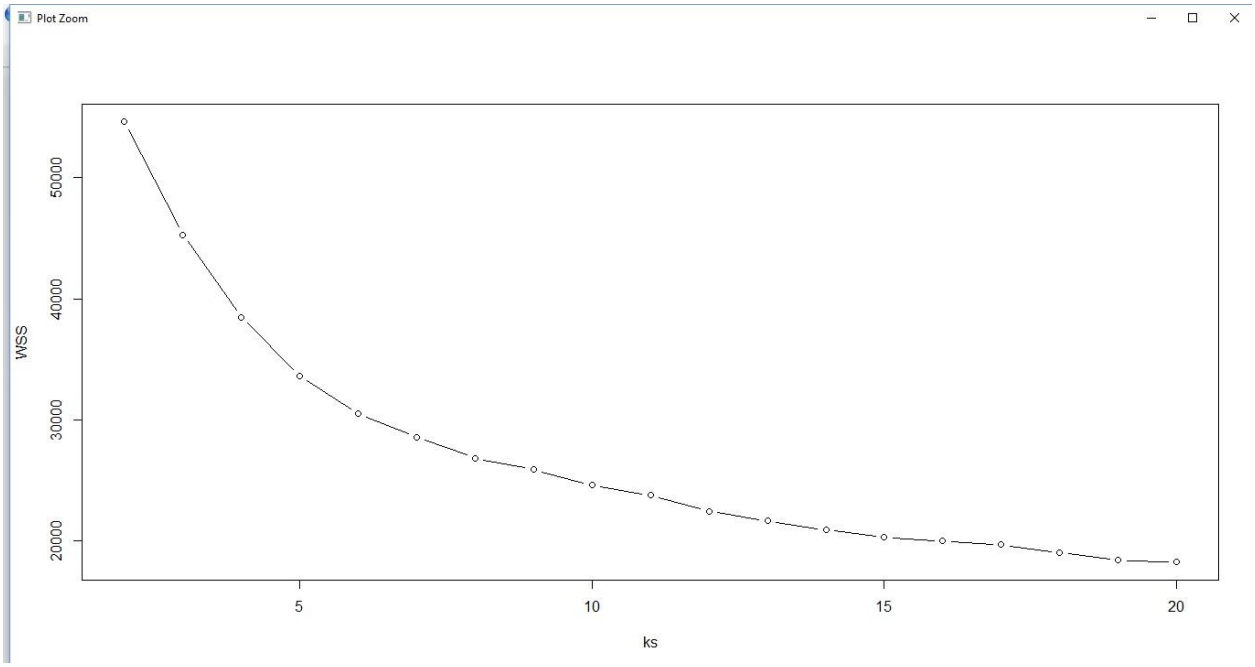
The worst fit model is model 5 – neural net with 2 hidden nodes and logistic activation function with a correlation of 0.735

CLUSTERING

1) K-means

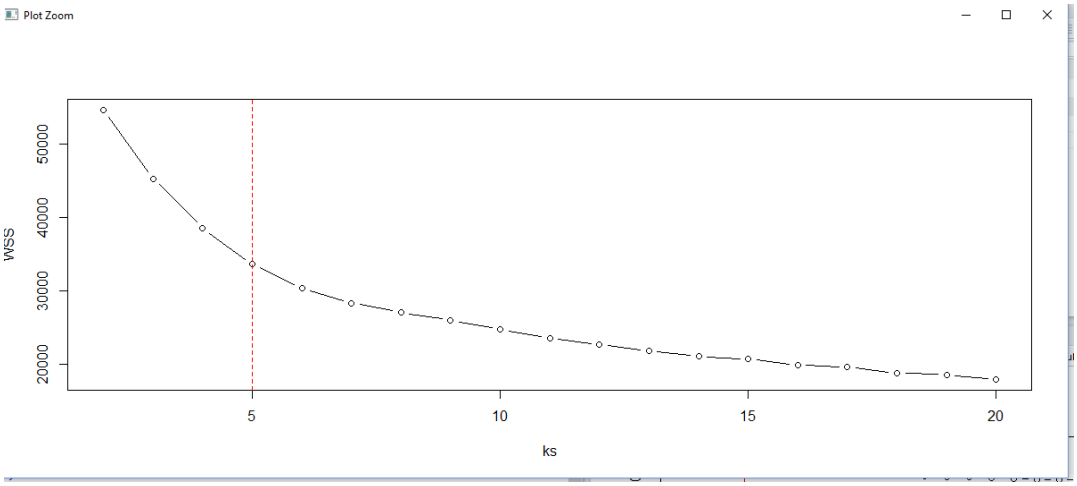
We aim to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. We used the sapply function to return a list a list of the same length as k and used the elbow function to determine our k. We performed k means on our balanced data of 10864 instances. As we can see that the drop is at different values varying from 5 to 7 but the first drop is at 5. We have used three different versions of k in our analysis

such as 5, 6 and 7. For each k value, we iterated our clustering algorithm till the number of data points in the clusters were nearly the same.



Above is the plot w received to check for the k values by using the plot function.
ks <- 2:20
WSS <- sapply(ks, FUN=function(k) {kmeans(new, centers=k, nstart=5)$tot.withinss})
plot(ks, WSS, type="b")

- **k = 5**



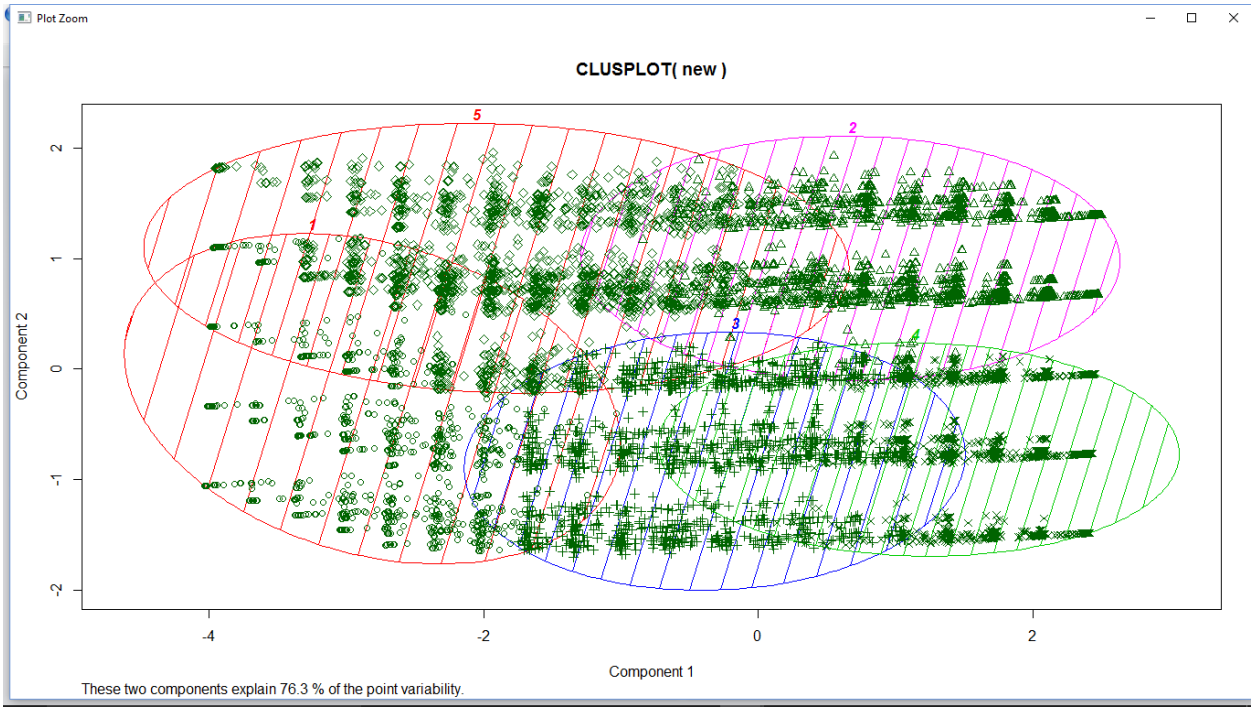
Here we have used 5 as the number of clusters. We repeated our algorithm for approximately 8 iterations to find the following data points in clusters to be nearly same.

The below table shows the distribution of data points in different clusters. We can read this table as Class 1 contributes maximum data points to cluster 1, 2 to cluster 5, 3 to cluster 3, 4 to cluster 2 and 5 to cluster 4.

```
> table(tripnew$overall,kc5$cluster)

      1    2    3    4    5
1   653   17  141    3  420
2   552   78  438   33  675
3   225  471  683  145  853
4    48  823  623  786  193
5     7 1142  137 1670   48
```

The below image represents the clusters and the data points in them. Since we have categorical independent variables we get such results.

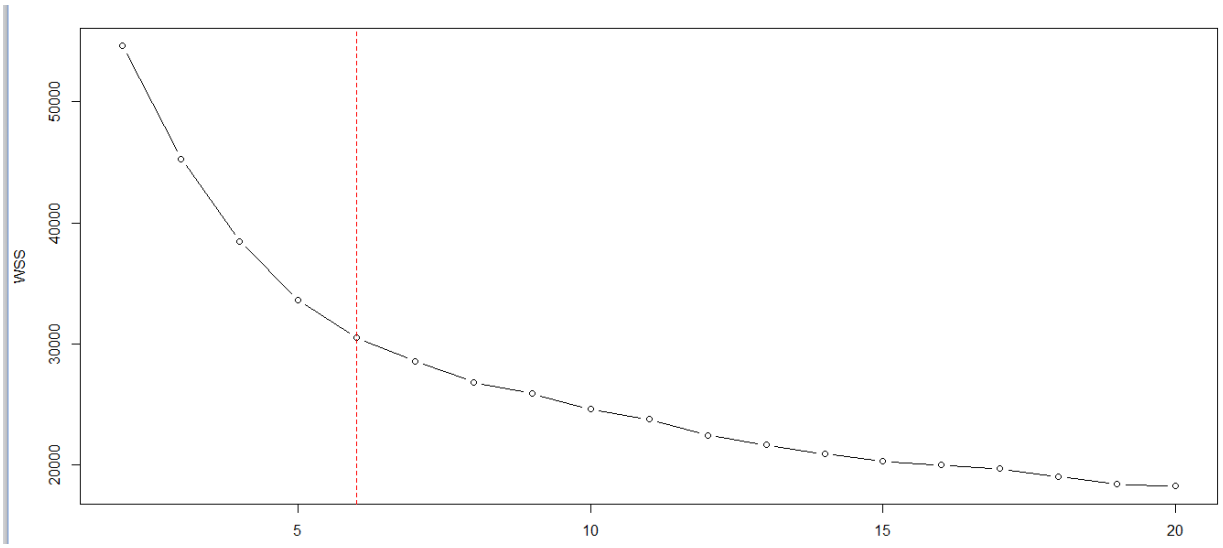
```
Within cluster sum of squares by cluster:
[1] 4099.408 7036.840 7290.243 9840.062 5736.921
 (between_SS / total_SS =  65.3 %)
```



Total iterations = 8. The size of the clusters varied widely. Cluster 4 is the largest cluster. The variance explained by the model is 65.3% All the independent variables contribute significantly to the cluster.

- **k = 6**

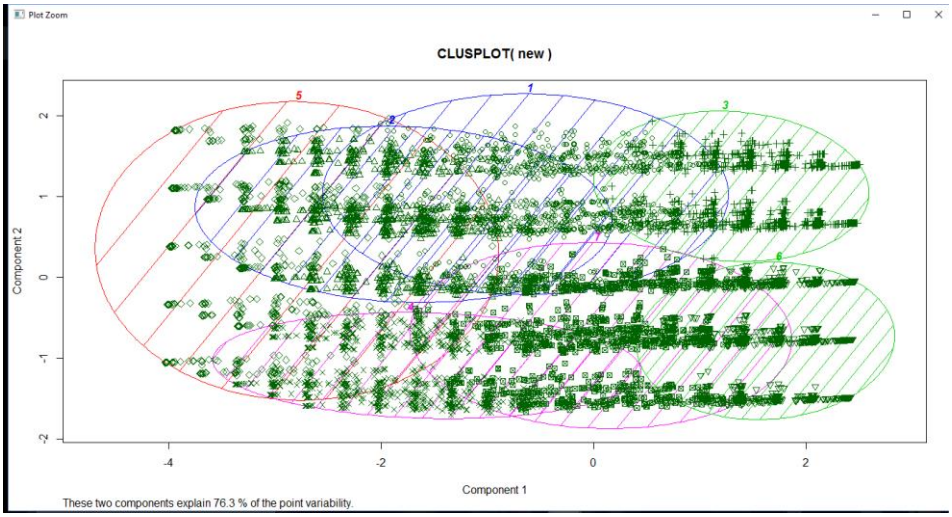
Now we tried using 6 clusters for our data set. Below you can see few iterations when the number of data points in each clusters nearly repeats itself. And finally we get the clusters for those data points.



The below table shows the distribution of data points in different clusters. In the last two iteration we can see that class 5 has significantly contributed to cluster 1. Also in these final iterations we see that the data points are starting to remain constant in clusters

```
> table(tripnew$Overall,kc6$cluster)

      1    2    3    4    5    6
1   491   85    4    2  522  130
2   575  298   33   15  456  399
3   450  778  157  162  194  636
4    66  447  830  552   47  531
5    20  112 1673 1072    7  120
> kc6 <- kmeans(new, 6)
> table(tripnew$Overall,kc6$cluster)

      1    2    3    4    5    6
1   150  507    3    3   85  486
2   449  451   28   20  282  546
3   691  196  114  209  767  400
4   640   46  620  759  354   54
5   141    7 1150 1604   84   18
> kc6 <- kmeans(new, 6)
> table(tripnew$Overall,kc6$cluster)

      1    2    3    4    5    6
1     4  129  500    4  479  118
2    40  412  518   27  436  343
3   178  899  359  242  194  505
4   940  417   42  647   46  381
5  1694  119   11 1098    7   75
> kc6 <- kmeans(new, 6)
> table(tripnew$Overall,kc6$cluster)

      1    2    3    4    5    6
1     4   95  488  507  138    2
2    37  309  556  449  409   16
3   174  785  424  194  631  169
4   900  428   58   46  482  559
5  1689  102   19    7  109 1078
>
```

```
Within cluster sum of squares by cluster:
[1] 6264.048 5153.376 5492.780 4851.173 4041.996 4584.383
 (between_SS / total_SS =  69.0 %)
```



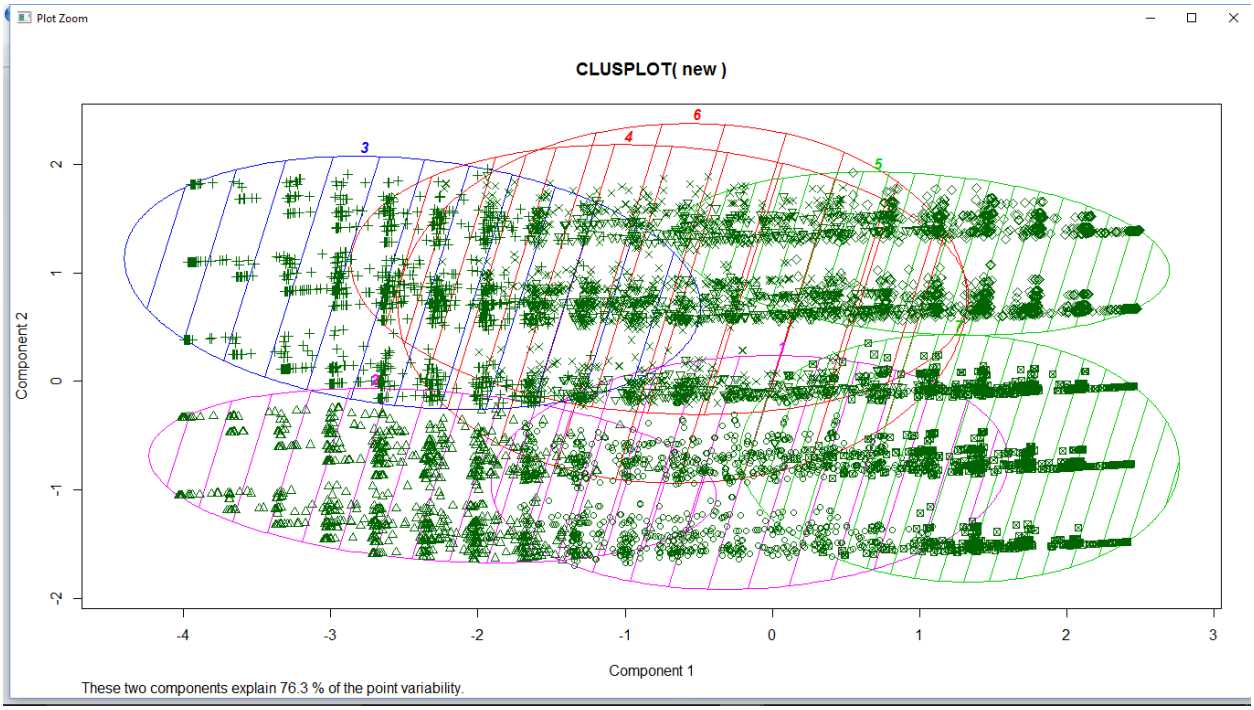
Total iterations = 10. The sizes of the clusters vary widely. Cluster 1 was the largest cluster. The variance explained by the model is 69%. The clusters overlap with each other. All the independent variables contribute significantly to the cluster.

- **k = 7**

The below table shows the distribution of data points in different clusters. We can read this table as Class 5 contributes maximum data points to cluster 1 and then to cluster 4 in the last run. Similarly we can see class how they are divided into the clusters.

```
> table(tripnew$Overall,kc7$cluster)

      1    2    3    4    5    6    7
  1   79  515  496   44    3   93    4
  2  281  510  515  189   18  233   30
  3  528  222  345  519  189  430  144
  4  520   56   44  224  579  246  804
  5  108    8   13   79 1080   50 1666
> kc7 <- kmeans(new, 7)
> table(tripnew$Overall,kc7$cluster)

      1    2    3    4    5    6    7
  1   70    4    2  350  350   56  402
  2  288   29   15  467  263  237  477
  3  612  125  157  410  102  715  256
  4  763  667  541   72   15  357   58
  5  148 1655 1065   27    0   95   14
> kc7 <- kmeans(new, 7)
> table(tripnew$Overall,kc7$cluster)

      1    2    3    4    5    6    7
  1   26  479  184    3  124  417    1
  2  117  436  427   28  378  381    9
  3  560  194  621  112  585  224   81
  4  668   46  171  619  497   20  452
  5  300    7   46 1149   89    3 1410
> kc7 <- kmeans(new, 7)
> table(tripnew$Overall,kc7$cluster)

      1    2    3    4    5    6    7
  1    3   83  375    3  287  406   77
  2   20  282  406   27  308  439  294
  3  197  780  161  104  242  300  593
  4  747  364   24  594   68   38  638
  5 1602   87    7 1144   26    4  134
>
```

```
within cluster sum of squares by cluster:
[1] 4835.620 1920.615 2900.493 5821.575 4240.959 3583.810 5970.103
 (between_SS / total_SS =  70.1 %)
```



Total iterations = 10. The sizes of the clusters vary widely. The variance explained by the model is 70.1%. The clusters overlap with each other. All the independent variables contribute significantly to the cluster.

2) Hierarchical Clustering

After K- Means clustering we ran Hierarchical Clustering. 3 random samples of 50 were chosen from a total of 10864 records of the balanced data. As our data was skewed, we normalized the data to get better results. 3 different hierarchical clustering techniques for the Euclidean Algorithm were applied: Complete, Average and Ward.d2. We also applied the clustering algorithm on our entire data set but the dendogram formation was not visible. Finally we applied the clustering techniques on our samples to show the dendrogram formation that was not visible otherwise.

Whole Data:

- Ward.d2

The table below can be read as: class 1 contributes maximum to cluster 6, class 2 contributes maximum to cluster 6, class 3 contributes maximum to cluster 2, class 4 to cluster 4 and class 5 to cluster 4. We can read the rest of the data point distributions.

```
> group.7.trip_wardd2 <- cutree(fit_whole,k=7)
> table(group.7.trip_wardd2,tripnew$Overall)

group.7.trip_wardd2    1    2    3    4    5
                  1  291  228  133   11    1
                  2   94  277  768  405  156
                  3  105  279  353   84   34
                  4    6   25  202  678 1441
                  5   52  193  422  454   66
                  6  680  731  347   67   22
                  7    6   43  152  774 1284
> group.7.trip_com <- cutree(fit_whole_C,k=7)
```

- Complete

```
> group.7.trip_com <- cutree(fit_whole_C,k=7)
> table(group.7.trip_com,tripnew$Overall)

group.7.trip_com    1    2    3    4    5
               1  549  730  767  172   54
               2    6    3    9    2    2
               3   49  222  888 1927 2863
               4    8   11    8    5    2
               5  618  806  698  357   76
               6    1    0    1    5    2
               7    3    4    6    5    5
```

- Average

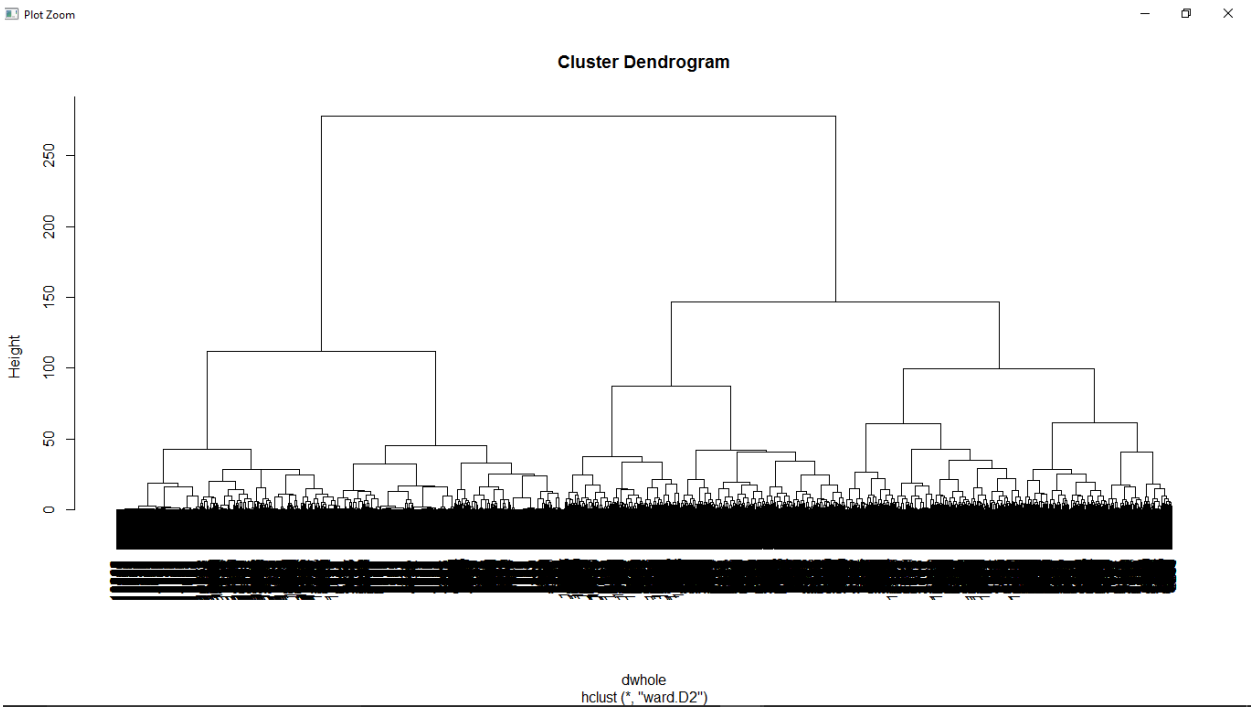
```
> group.7.trip_avg <- cutree(fit_whole_A,k=7)
> table(group.7.trip_avg,tripnew$Overall)

group.7.trip_avg    1    2    3    4    5
               1  167  151  130   17    3
               2  346  558  608  218  111
               3   38  187  736 1163 1625
               4   16   42   38   13    2
               5  368  550  571  324   96
               6  283  220   88   25    1
               7   16   68  206  713 1166
```

As the range of all the IV's varies widely in all the three methods, we can say that all the IV's contributes significantly towards cluster formation.

```
> aggregate(trip,list(group.7.trip_avg),mean)
  Group.1  Service    Value    Rooms  Overall Location    price Sentiment
1       1 2.014957 1.572650 1.702991 2.012821 2.019231 1.649573 0.1135009
2       2 2.288973 2.033134 2.336773 2.560022 4.205323 2.165671 0.2180904
3       3 4.272073 4.204588 4.026140 4.106962 4.497999 1.956255 0.8287054
4       4 1.765766 2.225225 3.369369 2.486486 1.891892 3.513514 0.1321175
5       5 2.505500 2.375589 2.468308 2.596647 4.388685 4.365636 0.2996785
6       6 1.847650 1.440843 1.482982 1.769854 2.124797 4.401945 0.1172591
7       7 4.342554 4.255417 4.517289 4.357769 4.643154 4.277086 0.8626089
> aggregate(trip,list(group.7.trip_com),mean)
  Group.1  Service    Value    Rooms  Overall Location    price  Sentiment
1       1 1.992958 1.907570 2.071303 2.318662 3.528169 1.966549 0.15486900
2       2 4.409091 3.545455 1.772727 2.590909 2.863636 1.454545 0.31118794
3       3 4.399731 4.203564 4.210456 4.232644 4.685157 2.880988 0.86122595
4       4 1.588235 1.647059 4.294118 2.470588 2.000000 3.558824 0.08303799
5       5 2.264971 2.160861 2.310763 2.400000 3.674755 4.284540 0.23187567
6       6 3.555556 4.777778 4.333333 3.777778 1.555556 1.555556 0.46912553
7       7 1.695652 4.521739 4.782609 3.217391 4.260870 3.869565 0.43082223
> aggregate(trip,list(group.7.trip_wardd2),mean)
  Group.1  Service    Value    Rooms  Overall Location    price  Sentiment
1       1 1.522590 1.331325 1.350904 1.799699 2.658133 1.671687 0.08633136
2       2 3.468824 2.966471 2.945294 3.148235 4.168824 1.787059 0.50526844
3       3 1.581287 2.163743 2.692398 2.605848 4.173099 2.196491 0.16241518
4       4 4.560374 4.542517 4.531037 4.497874 4.684099 1.872449 0.92314988
5       5 3.333614 3.085088 3.203033 3.243471 4.228307 4.100253 0.50457434
6       6 1.809962 1.704385 1.723877 1.927991 3.382783 4.351381 0.11016080
7       7 4.614874 4.370075 4.490040 4.455069 4.787959 4.194776 0.91312428
>
```

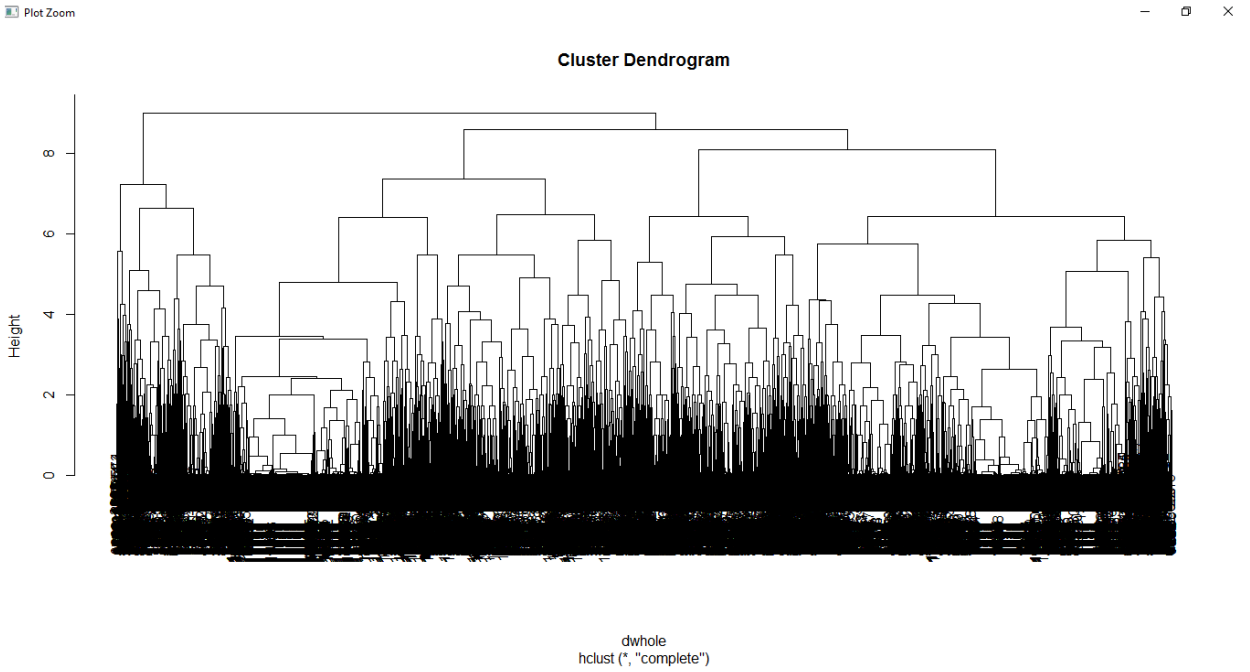
Dendogram

- Ward.d2



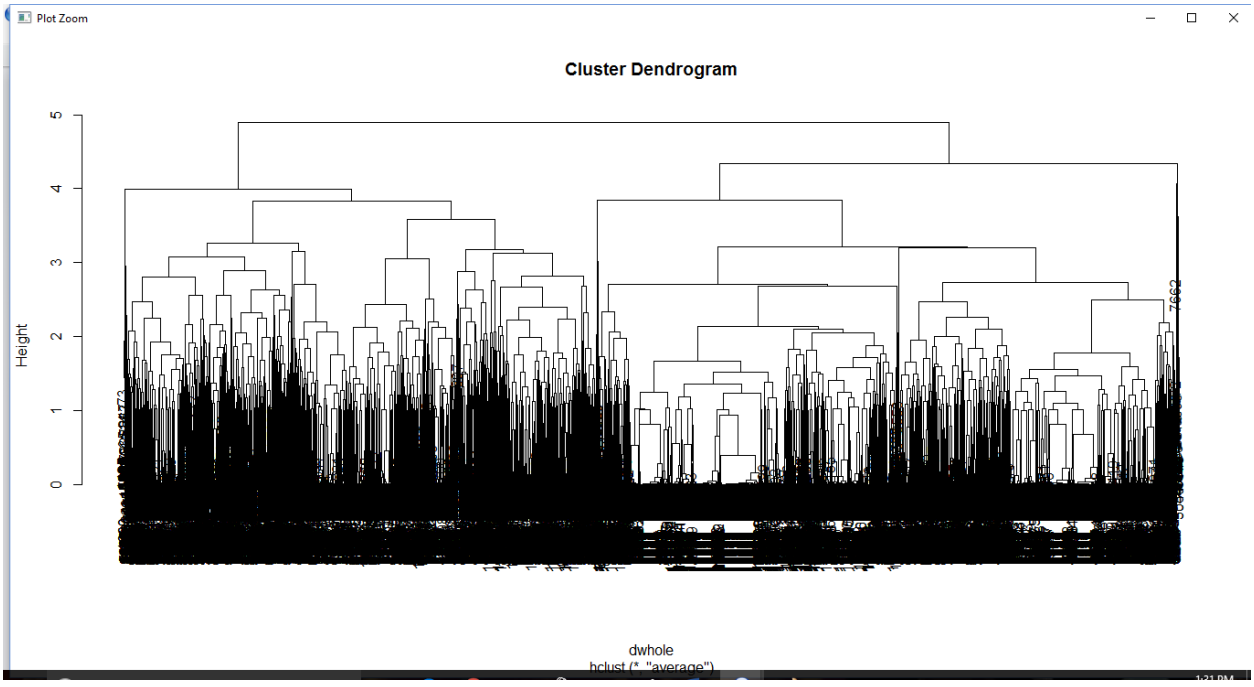
As the data was huge, the dendrogram is unreadable as the screen was not big enough for all the data to be displayed.

- Complete



As the data was huge, the dendrogram is unreadable as the screen was not big enough for all the data to be displayed.

- Average



As the data was huge, the dendrogram is unreadable as the screen was not big enough for all the data to be displayed.

Sample

- Ward.d2

As the range of all the IV's varies widely; we can say that all the IV's contributes significantly towards cluster formation.

```
> plot(sample1_average, hang=-1)
> rect.hclust(sample1_average, k=7, border="red")
> group.7.multi_sample1_wardd2 <- cutree(sample1_wardd2,k=7)
> table(group.7.multi_sample1_wardd2,Trip1$Overall)

group.7.multi_sample1_wardd2  1  2  3  4  5
                           1  0  1  1  3  3
                           2  3  1  2  0  0
                           3  0  0  1  4 10
                           4  0  0  3  0  0
                           5  0  2  2  2  0
                           6  0  3  3  2  0
                           7  1  2  0  0  1
> aggregate(Trip1,list(group.7.multi_sample1_wardd2),mean)
  Group.1  Service    Value    Rooms  Overall Location    price  Sentiment
1       1 4.125000 4.125000 4.000000 4.000000 3.750000 1.625000 0.85529997
2       2 1.166667 1.000000 1.666667 1.833333 3.000000 2.833333 0.29315991
3       3 4.400000 4.333333 4.533333 4.600000 4.666667 4.133333 0.98089443
4       4 4.666667 4.000000 1.666667 3.000000 4.666667 3.666667 0.74093712
5       5 3.000000 2.500000 3.333333 3.000000 4.500000 4.000000 0.46010850
6       6 3.000000 2.750000 2.375000 2.875000 4.500000 1.875000 0.22248024
7       7 3.000000 2.250000 2.000000 2.500000 2.500000 4.750000 0.02363241
> plot(sample1_wardd2, hang=-1)
> rect.hclust(sample1_wardd2, k=7, border="red")
>
```

- Complete

As the range of all the IV's varies widely; we can say that all the IV's contributes significantly towards cluster formation.

```
> group.7.multi_sample1_complete <- cutree(sample1_complete,k=7)
> table(group.7.multi_sample1_complete,Trip1$Overall)

group.7.multi_sample1_complete  1  2  3  4  5
                             1  0  0  1  2  3
                             2  2  2  2  1  0
                             3  0  0  1  4 10
                             4  0  3  3  2  0
                             5  0  0  3  0  0
                             6  1  3  2  2  1
                             7  1  1  0  0  0
> aggregate(Trip1,list(group.7.multi_sample1_complete),mean)
  Group.1  Service    Value    Rooms  Overall Location    price Sentiment
1       1 4.333333 4.333333 4.333333 4.333333 4.166667 1.500000 0.9684498
2       2 1.714286 1.142857 1.714286 2.285714 3.714286 2.714286 0.1161679
3       3 4.400000 4.333333 4.533333 4.600000 4.666667 4.133333 0.9808944
4       4 3.125000 3.250000 2.750000 2.875000 4.000000 1.750000 0.3510303
5       5 4.666667 4.000000 1.666667 3.000000 4.666667 3.666667 0.7409371
6       6 2.888889 2.555556 3.000000 2.888889 3.888889 4.333333 0.3168417
7       7 2.500000 1.000000 1.000000 1.500000 1.500000 3.500000 0.4763450
```

- Average

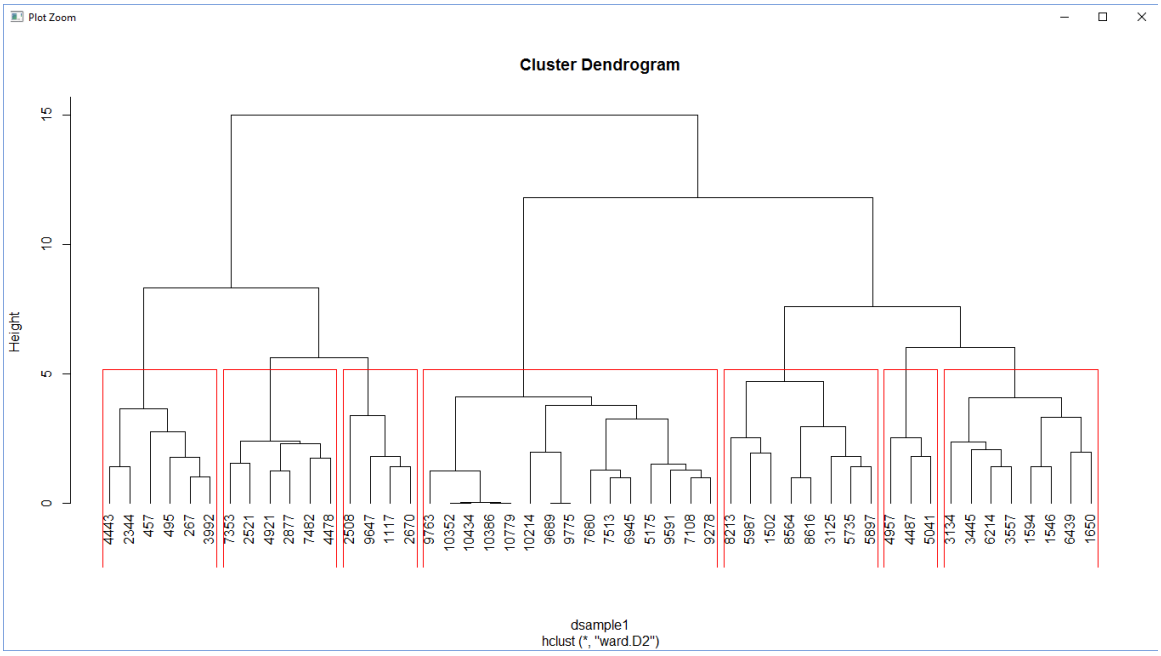
As the range of all the IV's varies widely; we can say that all the IV's contributes significantly towards cluster formation.

```
> rect.hclust(sample1_complete, k=7, border="red")
> group.7.multi_sample1_average <- cutree(sample1_average,k=7)
> table(group.7.multi_sample1_average,Trip1$Overall)

group.7.multi_sample1_average  1  2  3  4  5
                            1  0  0  4  4  3
                            2  2  5  2  1  0
                            3  0  0  1  4 10
                            4  0  0  3  0  0
                            5  1  3  2  2  1
                            6  1  0  0  0  0
                            7  0  1  0  0  0
> aggregate(Trip1,list(group.7.multi_sample1_average),mean)
  Group.1  Service    Value    Rooms  Overall Location    price   Sentiment
1       1 4.000000 3.909091 3.636364 3.909091 4.363636 1.545455 0.752008893
2       2 1.900000 1.700000 2.000000 2.200000 3.500000 2.500000 0.116001871
3       3 4.400000 4.333333 4.533333 4.600000 4.666667 4.133333 0.980894434
4       4 4.666667 4.000000 1.666667 3.000000 4.666667 3.666667 0.740937124
5       5 2.888889 2.555556 3.000000 2.888889 3.888889 4.333333 0.316841696
6       6 1.000000 1.000000 1.000000 1.000000 1.000000 3.000000 0.949084640
7       7 4.000000 1.000000 1.000000 2.000000 2.000000 4.000000 0.003605372
```

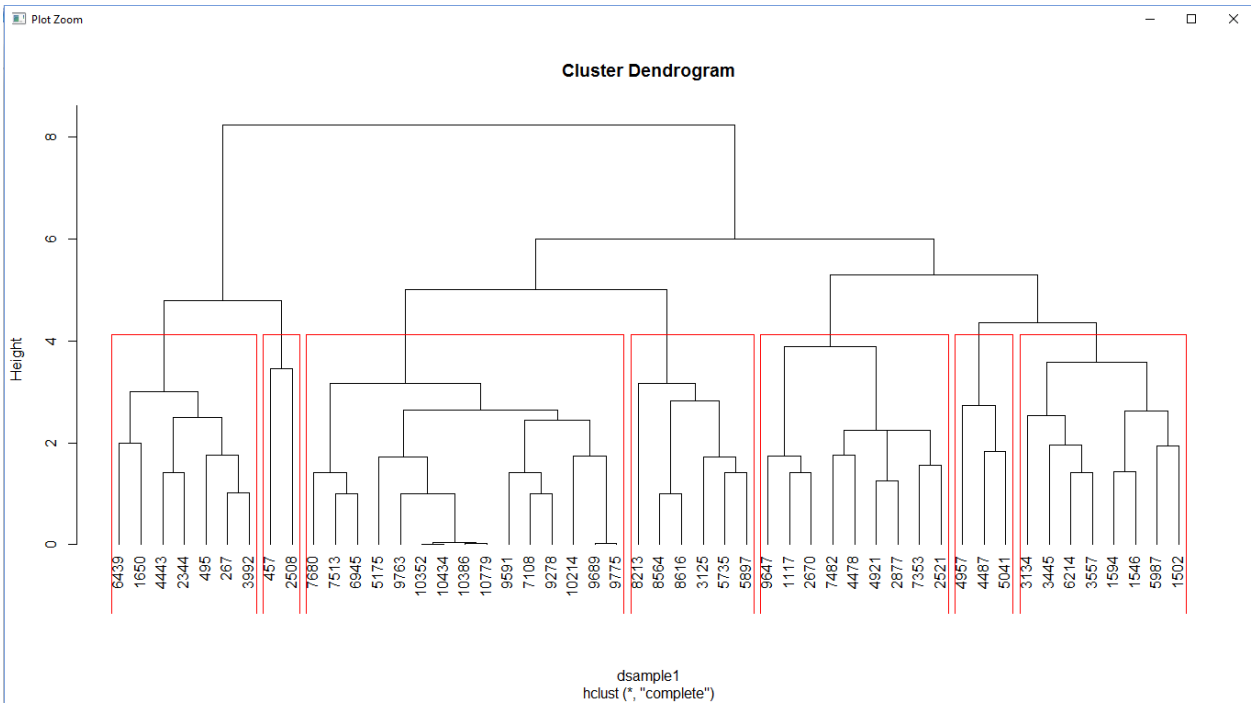
Dendogram

The graphs below is a sample of the full model formed above. This is just to show how the actual dendrogram is. We have used the cutree function to obtain different clusters and cut our dendogram at length 7.

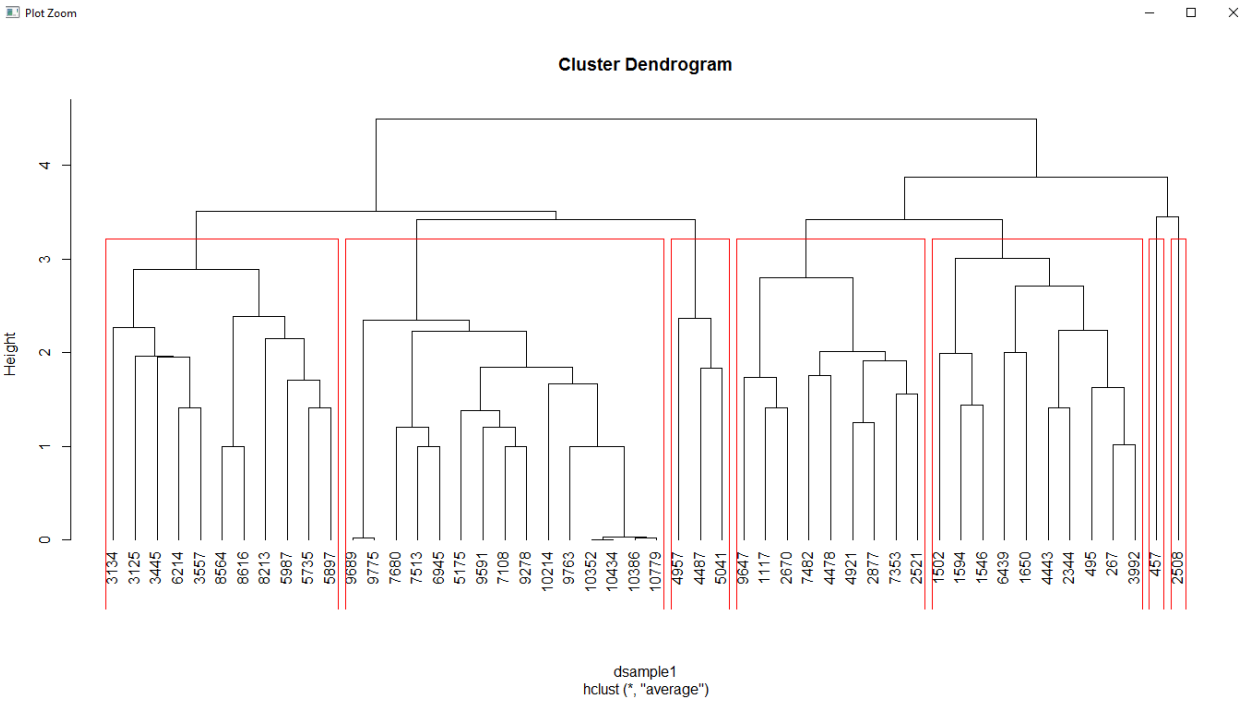
- Ward.d2



- Complete



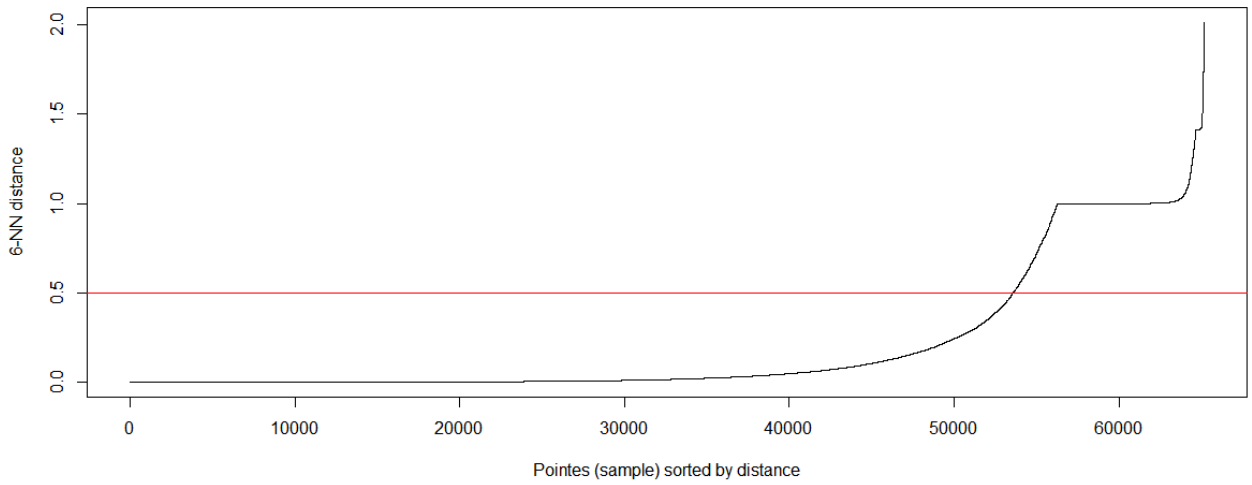
- Average



3) DBSCAN

We have used the density-based clustering algorithm. Given a set of data points, it groups together points that are closely packed together, marking as outliers points that lie alone in low-density regions. To get the desired number of clusters, number of iterations were done with the eps and minimum point values.

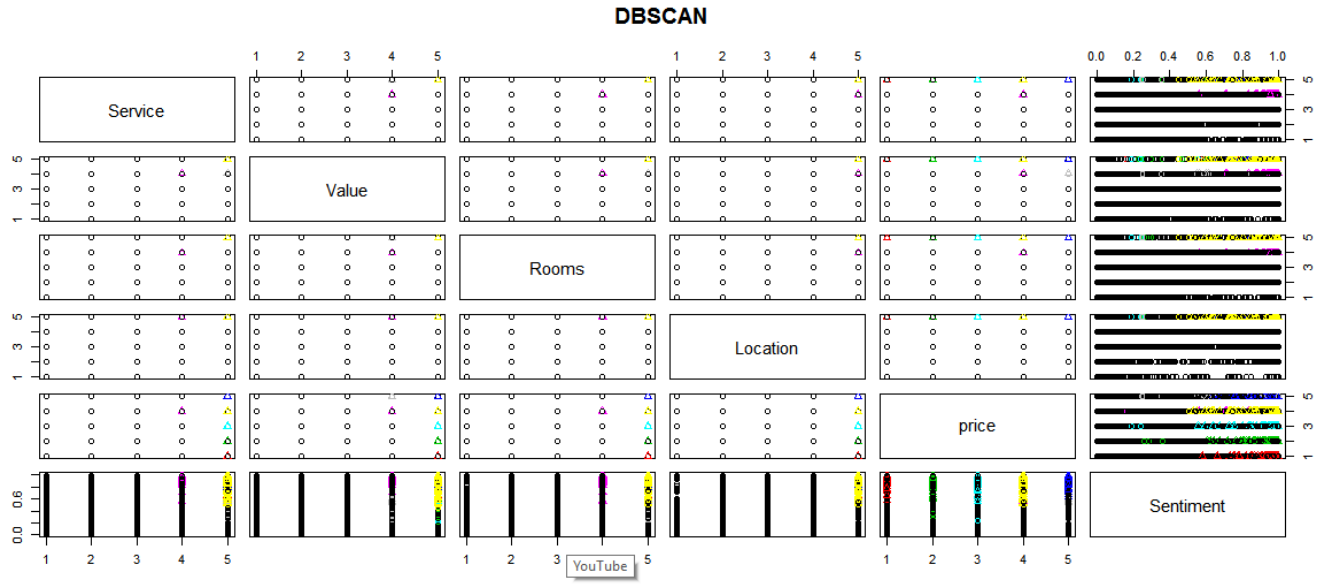
Here we use 6 as the value of k.
kNNdistplot(new, k = 6)
abline(h=.5, col="red")



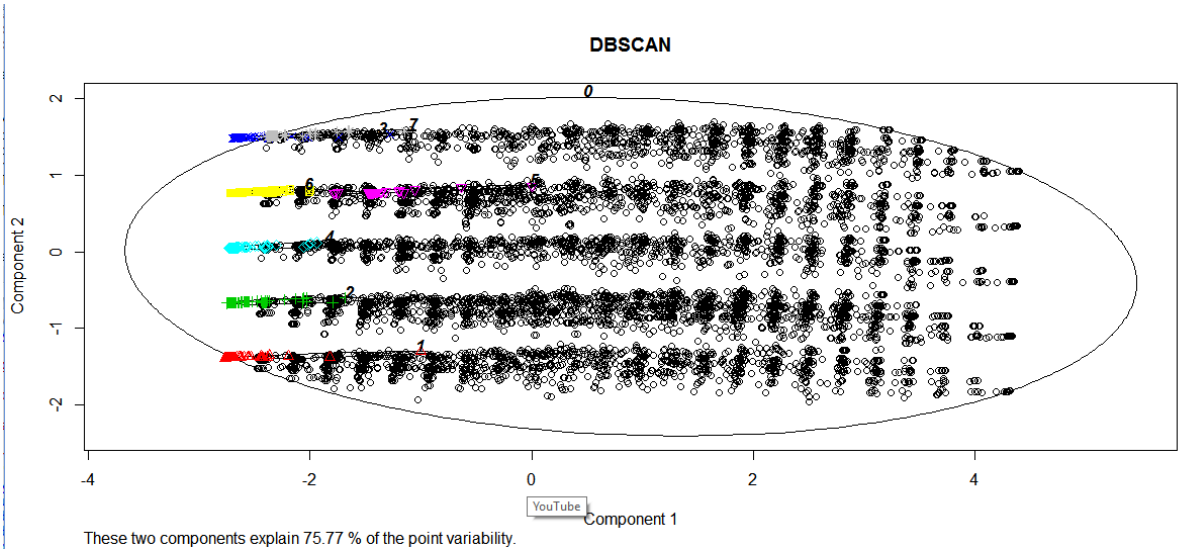
We tried a number of times with different eps value and minimum points to get the ideal value and after approximately 8 trials we got eps as 0.45 and minimum points as 100.

```
> db3 <- fpc::dbscan(new, eps=0.45, MinPts=100)
> print(db3)
dbscan Pts=10864 MinPts=100 eps=0.45
          0   1   2   3   4   5   6   7
border 9300   1   6   2   2   3   2  97
seed      0 209 243 243 238 101 413   4
total  9300 210 249 245 240 104 415 101
```

The above table shows the clusters and the border and seed points in those clusters respectively. We can see that for cluster 0, 9300 points are on the border and 0 in seed. Similarly we can check for remaining clusters as well. We can see that seed points are diversely spread throughout the clusters.



In the picture above we can see the clusters analysis between every independent variables and the patterns they form. The diagram below explains 75.77% of point variability.



Since we have categorical values in our independent variables, we cannot see trends between them. We can check them against the sentiment analysis variable which is continuous to see how the form clusters. We can say that the data is skewed hence the formation of the clusters is not evenly distributed.

Comparison Using Caret Package

We have compared different all the models below using three comparison methods. We have used caret package to carry out this comparison. The three cross validation techniques which we used to compare models are:

1. Bootstrapping (method = boot)
2. 5 fold cross validation (method = “cv” number =5)
3. 10 fold cross validation with 2 repeats. (method =”repeatedcv”, number =10, repeat=2)

(Note: We also tried running models using method LOOCV which is leave one out cross validation. This method required very high processing power and took lot of time to run linear svm. The computer was continuously running for more than 24 hours. That is why we stopped running it.)

Models built for cross validation

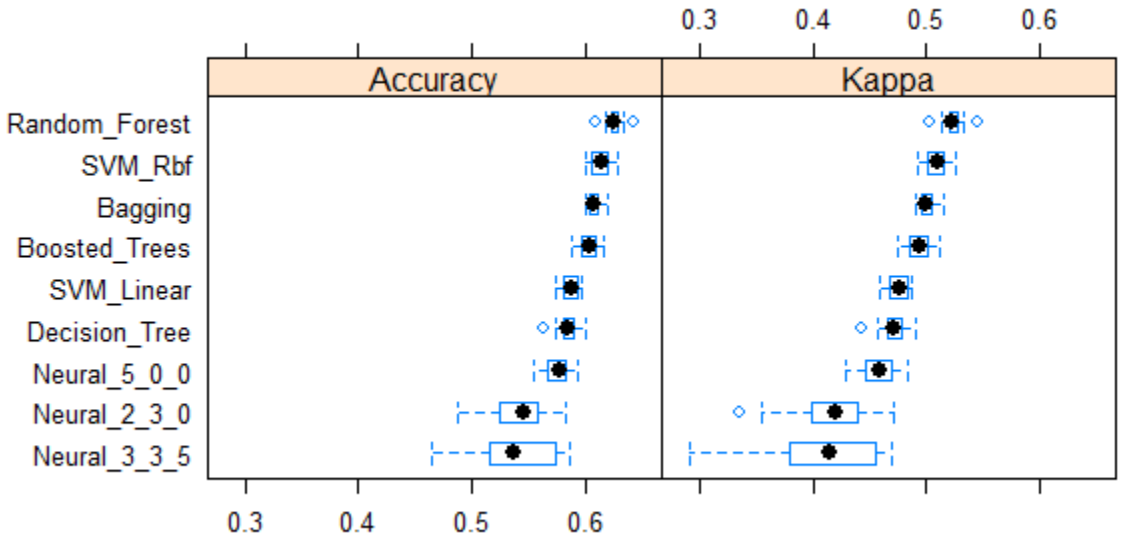
1. Linear svm (method: svmLinear)
2. Radial svm (method: svmRadial)
3. Decision tree with 1 trail (method: tree)
4. Random Forest with 3 trials (method: rf)
5. Bagging with 5 trials (method: rf)
6. Boosting with 10 and 20 trails (method: tree)
7. Neural network with 3 hidden layers. Neurons in each hidden layer are 1:5, 2:0,3:0 (method: mlpML)
8. Neural network with 3 hidden layers. Neurons in each hidden layer are 1:3, 2:3,3:5 (method: mlpML)
9. Neural network with 3 hidden layers. Neurons in each hidden layer are 1:2, 2:3,3:0 (method: mlpML)

Bootstrap:

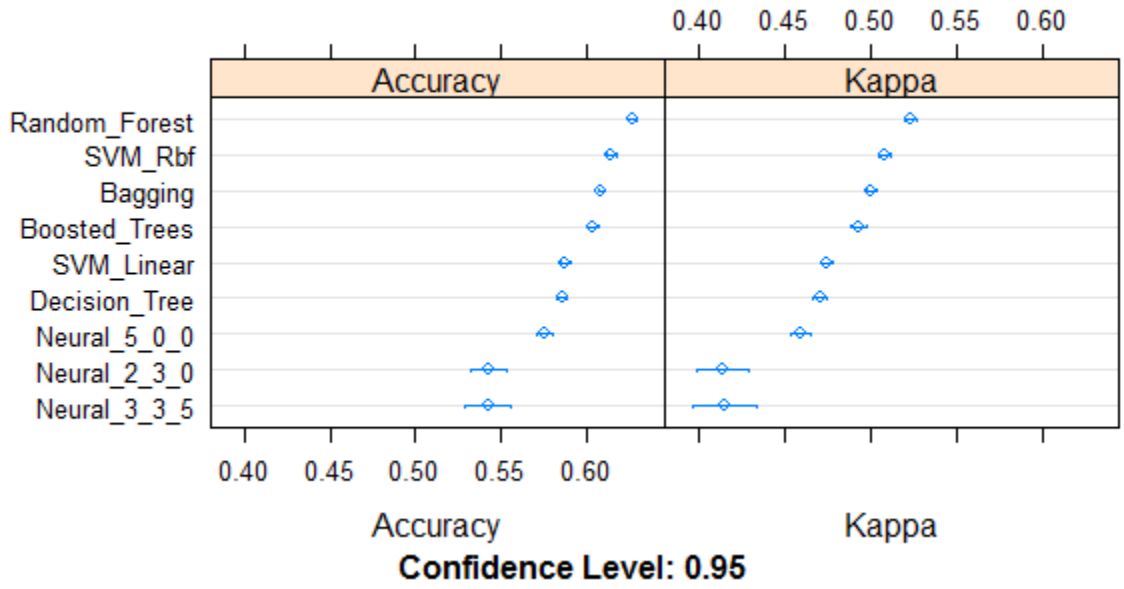
```
Call:
summary.resamples(object = result_boot)

Models: Decision_Tree, Boosted_Trees, Bagging, Random_Forest, SVM_Linear, SVM
_Rbf, Neural_5_0_0, Neural_3_3_5, Neural_2_3_0
Number of resamples: 25

Accuracy
                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
Decision_Tree 0.5639  0.5817 0.5853 0.5853  0.5909 0.6014    0
Boosted_Trees 0.5885  0.5963 0.6034 0.6030  0.6088 0.6173    0
Bagging       0.6002  0.6044 0.6080 0.6083  0.6119 0.6192    0
Random_Forest 0.6103  0.6238 0.6253 0.6264  0.6295 0.6431    0
SVM_Linear    0.5734  0.5815 0.5890 0.5872  0.5930 0.5973    0
```

```
SVM_Rbf       0.6009  0.6068 0.6141 0.6140  0.6193 0.6289    0
Neural_5_0_0  0.5551  0.5665 0.5771 0.5752  0.5835 0.5930    0
Neural_3_3_5  0.4640  0.5150 0.5377 0.5421  0.5748 0.5861    0
Neural_2_3_0  0.4872  0.5246 0.5454 0.5424  0.5590 0.5832    0

Kappa
                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
Decision_Tree 0.4435  0.4667 0.4711 0.4709  0.4775 0.4901    0
Boosted_Trees 0.4751  0.4855 0.4938 0.4934  0.5010 0.5113    0
Bagging       0.4903  0.4955 0.4998 0.5005  0.5051 0.5147    0
Random_Forest 0.5032  0.5200 0.5224 0.5235  0.5277 0.5447    0
SVM_Linear    0.4588  0.4675 0.4756 0.4748  0.4828 0.4876    0
SVM_Rbf       0.4924  0.5004 0.5095 0.5085  0.5146 0.5258    0
Neural_5_0_0  0.4282  0.4462 0.4591 0.4590  0.4694 0.4834    0
Neural_3_3_5  0.2910  0.3794 0.4140 0.4150  0.4556 0.4693    0
Neural_2_3_0  0.3347  0.3993 0.4205 0.4139  0.4401 0.4707    0
```

BoxPlot



DotPlot

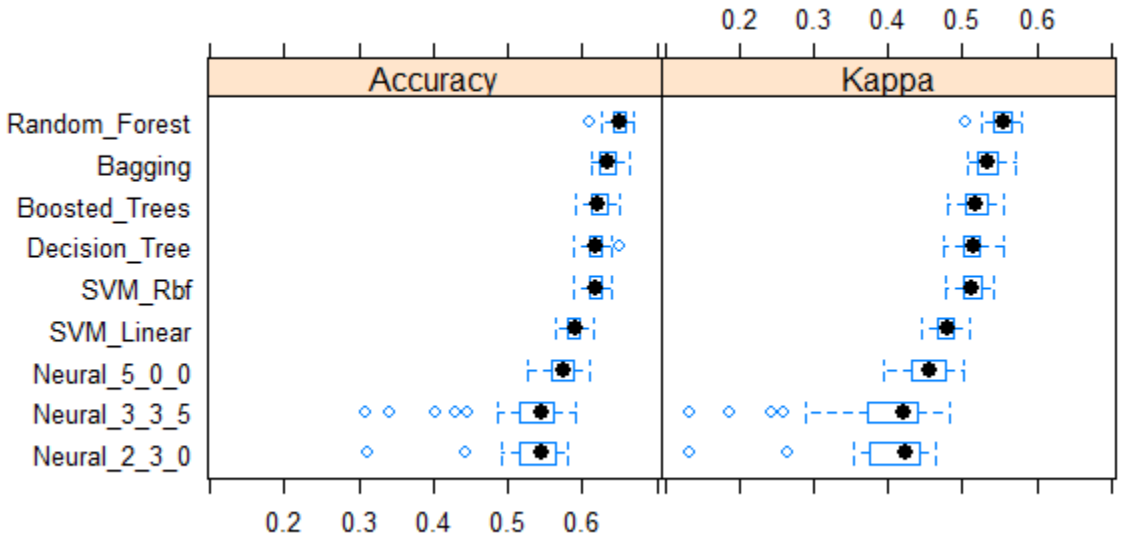


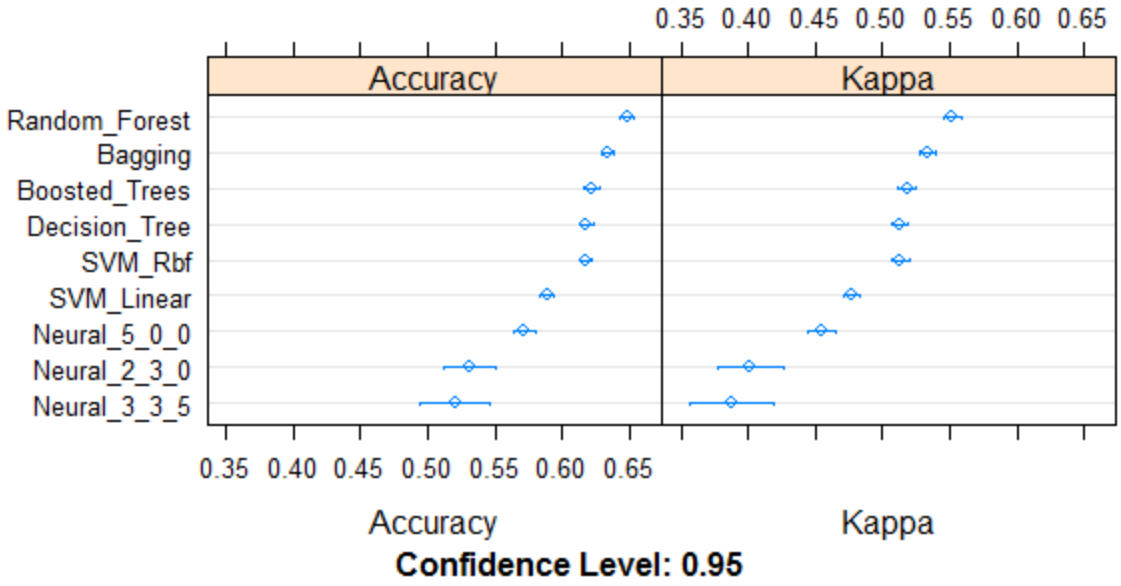
It can be seen from the results that the accuracy is highest for Random Forest method. It can be used as the model to give predictions. The value of accuracy is not very high around 0.63 which is not significant enough to give good predicted results. The highest observed value of kappa is also for random forest with value being around 0.525. The value of Kappa helps us gauge the expected accuracy against expected accuracy.

Repeated CV with 10 folds and 2 repeats.

```
Accuracy
                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
Decision_Tree 0.5897  0.6096 0.6183 0.6180  0.6253 0.6510    0
Boosted_Trees 0.5915  0.6125 0.6208 0.6223  0.6331 0.6507    0
Bagging       0.6127  0.6246 0.6341 0.6342  0.6436 0.6645    0
Random_Forest 0.6103  0.6418 0.6499 0.6484  0.6588 0.6691    0
SVM_Linear    0.5639  0.5805 0.5919 0.5888  0.5970 0.6149    0
SVM_Rbf       0.5893  0.6095 0.6178 0.6176  0.6263 0.6403    0
Neural_5_0_0  0.5253  0.5579 0.5752 0.5720  0.5878 0.6089    0
Neural_3_3_5  0.3100  0.5175 0.5455 0.5202  0.5596 0.5919    0
Neural_2_3_0  0.3119  0.5176 0.5444 0.5312  0.5633 0.5792    0

Kappa
                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
Decision_Tree 0.4753  0.5026 0.5137 0.5130  0.5229 0.5555    0
Boosted_Trees 0.4802  0.5057 0.5165 0.5181  0.5321 0.5546    0
Bagging       0.5057  0.5214 0.5333 0.5336  0.5461 0.5722    0
Random_Forest 0.5029  0.5434 0.5538 0.5520  0.5654 0.5793    0
SVM_Linear    0.4444  0.4656 0.4808 0.4768  0.4873 0.5103    0
```

```
SVM_Rbf       0.4779  0.5029 0.5134 0.5132  0.5247 0.5421    0
Neural_5_0_0  0.3934  0.4325 0.4568 0.4542  0.4772 0.5024    0
Neural_3_3_5  0.1331  0.3738 0.4208 0.3880  0.4395 0.4823    0
Neural_2_3_0  0.1339  0.3767 0.4228 0.4013  0.4425 0.4629    0
```





Confidence Level: 0.95

The value of accuracy increased for this model compared to other two models. It can be seen from the results that the accuracy is highest for Random Forest method. It can be used as the model to give predictions. The value of accuracy is not very high but it increased to 0.65 from 0.63 but is not significant enough to give good predicted results. The highest observed value of kappa is also for random forest with value being around 0.554. The second is followed by bagging which is very near to value of random forest. The value of Kappa helps us gauge the expected accuracy against expected accuracy.

Repeated CV with 5 folds

Accuracy

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Decision_Tree	0.5797	0.6093	0.6273	0.6370	0.6113	0.6180	0
Boosted_Trees	0.5235	0.6015	0.6368	0.6473	0.6151	0.6258	0
Bagging	0.6127	0.6224	0.6369	0.6347	0.6521	0.6554	0
Random_Forest	0.6204	0.6328	0.6350	0.6353	0.6477	0.6587	0
SVM_Linear	0.5639	0.5805	0.5919	0.5668	0.5550	0.6079	0
SVM_Rbf	0.5143	0.6075	0.6178	0.6176	0.6253	0.6403	0
Neural_5_0_0	0.5253	0.5889	0.5232	0.5720	0.5878	0.6089	0
Neural_3_3_5	0.3100	0.5175	0.5455	0.5202	0.5566	0.5919	0
Neural_2_3_0	0.3119	0.5188	0.5466	0.5389	0.5633	0.5792	0

Kappa

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Decision_Tree	0.4722	0.5047	0.5128	0.5136	0.5288	0.5510	0
Boosted_Trees	0.4702	0.5047	0.5165	0.5181	0.5321	0.5538	0
Bagging	0.5157	0.5214	0.5313	0.5136	0.5471	0.5714	0
Random_Forest	0.5599	0.5332	0.5258	0.5420	0.5744	0.5583	0
SVM_Linear	0.4444	0.4656	0.4808	0.4768	0.4873	0.5128	0
SVM_Rbf	0.4779	0.5029	0.5134	0.5132	0.5247	0.5415	0
Neural_5_0_0	0.3934	0.4325	0.4568	0.4542	0.4772	0.5011	0
Neural_3_3_5	0.1331	0.3738	0.4208	0.3880	0.4395	0.4765	0
Neural_2_3_0	0.1321	0.3778	0.4211	0.4048	0.4416	0.4629	0

The value of accuracy increased for this model compared to other first model. It can be seen from the results that the accuracy is highest for Random Forest method. It can be used as the model to give predictions. The value of accuracy is not very high but it increased to 0.63 from 0.635 but is not significant enough to give good predictive results. The highest observed value of kappa is also for random forest with value being around 0.54. The second is followed by bagging which is very near to value of random forest. The value of Kappa helps us gauge the expected accuracy against expected accuracy.

10 fold cross validation with 2 repeats is the best method to generate models from the observed models. It gives us better accuracy and kappa value.