Milestone 3: Report

Predictive Analysis of Los Angeles Airbnb Data

INST 737: Introduction to Data Science

Team 1: Members

Samarjith Jawaharlal Sathyanarayan

Sheryl Mathias

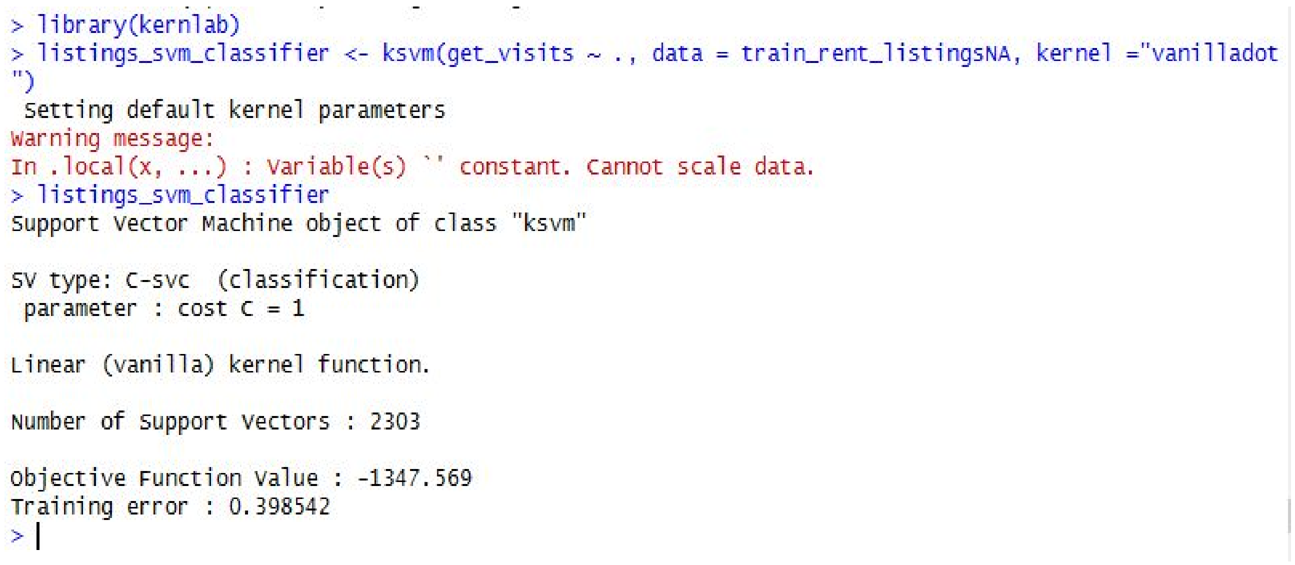
Ruchira Kapoor

Question 1: SVMs

Research Question: **Classify whether a listing gets a visit or not based on various features.**

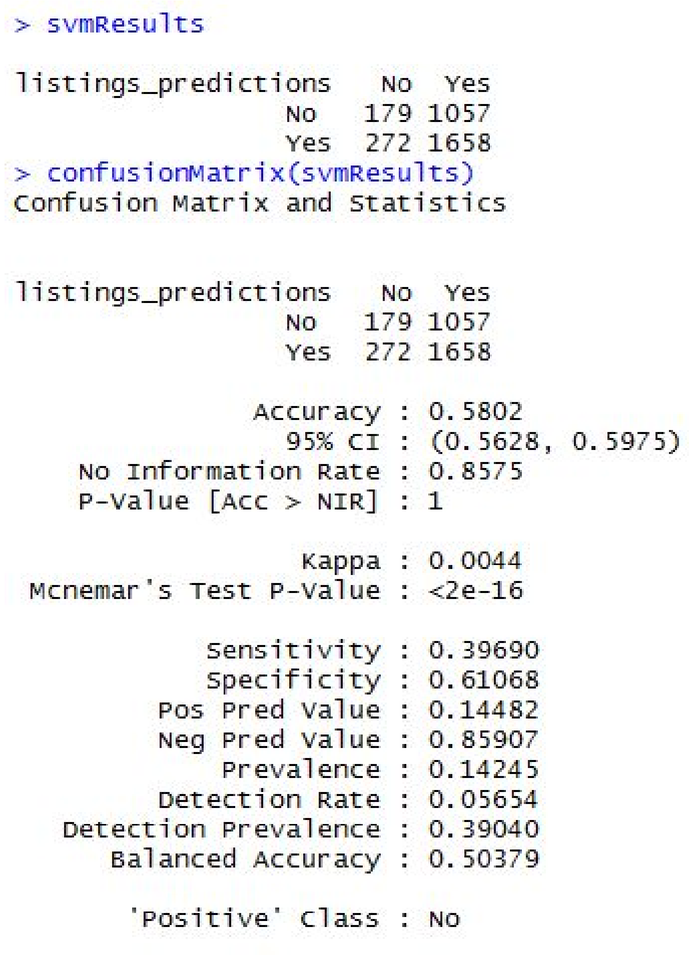
Linear kernel function

Results:



```
> library(kernlab)
> listings_svm_classifier <- ksvm(get_visits ~ ., data = train_rent_listingsNA, kernel ="vanilladot
")
 Setting default kernel parameters
Warning message:
In .local(x, ...) : Variable(s) `' constant. Cannot scale data.
> listings_svm_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 : 2303

Objective Function Value : -1347.569
Training error : 0.398542
> |
```

Confusion Matrix results using Linear Kernel Function



```
> svmResults

listings_predictions   No  Yes
                 No   179 1057
                 Yes  272 1658
> confusionMatrix(svmResults)
Confusion Matrix and Statistics


listings_predictions   No  Yes
                 No   179 1057
                 Yes  272 1658

               Accuracy : 0.5802
                 95% CI : (0.5628, 0.5975)
    No Information Rate : 0.8575
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0044
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.39690
            Specificity : 0.61068
         Pos Pred Value : 0.14482
         Neg Pred Value : 0.85907
             Prevalence : 0.14245
         Detection Rate : 0.05654
   Detection Prevalence : 0.39040
      Balanced Accuracy : 0.50379

       'Positive' Class : No
```

Precision: This gives us the percentage of positive predictions that the listings were correctly predicted that they will get a visit. For this we check the Yes column and Yes row from our confusion matrix.

True Positive (TP) = 1658

False Negative (FN) = 272

False Positive (FP) = 1057

True Negative (TN) = 179

Precision = TP / (TP + FP) = 1658 / (1658 + 1057) = 0.61

Recall: This gives us the percentage of listings that were labelled as getting a visit that were predicted as getting a visit.

Recall/Sensitivity: TP / (TP + FN) = 1658 / (1658 + 272) = 0.86

Specificity : This gives us the percentage of listings that were labelled as not getting a visit that were predicted as not getting a visit.

Specificity: TN / (TN + FP) = 179 / (179 + 1057) = 0.14

Accuracy: This gives us the percentage of listings that were correctly predicted.

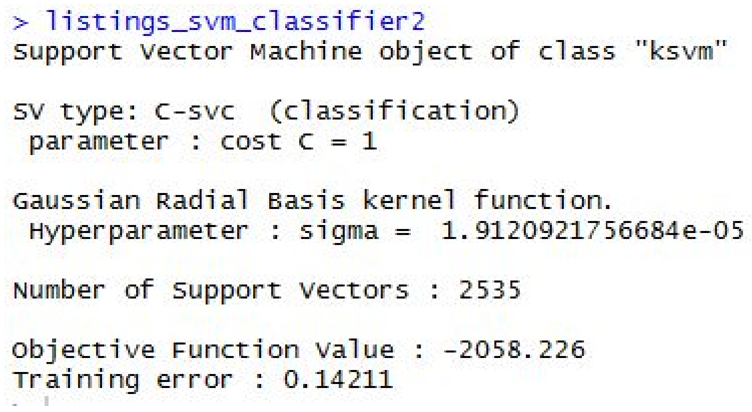
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (1658 + 179) / (1658 + 272 + 1057 + 179) = 1837/3166 = 0.58

F - measure = (2 * precision * recall) / (precision + recall) = (2*0.61*0.86)/ (0.61 + 0.86) = 1.0492 / 1.47 = 0.71

<u>Non-linear kernel function:</u>

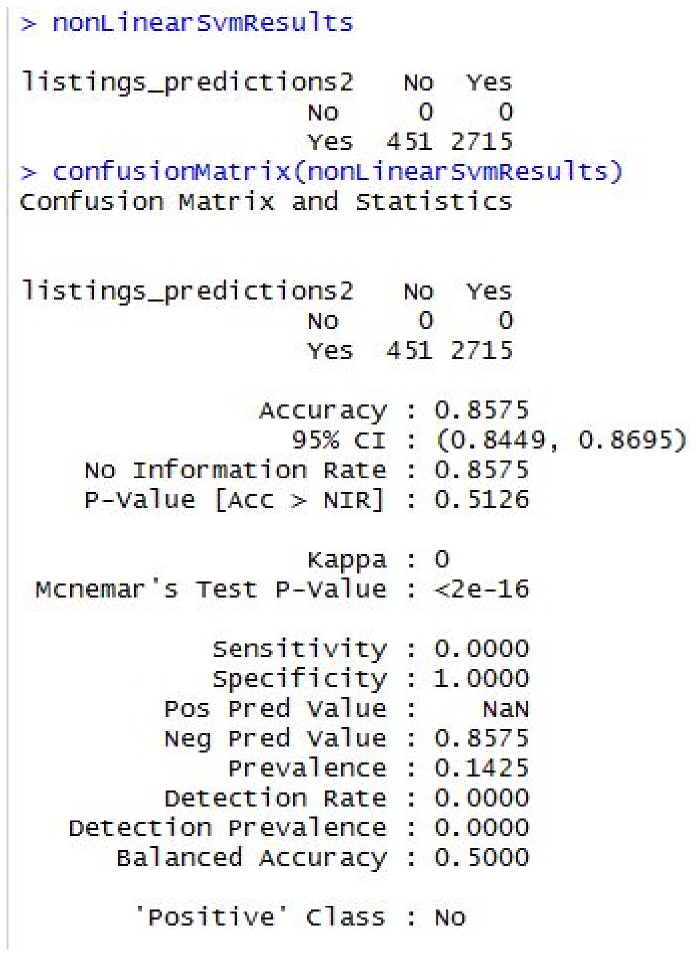
Using rbf kernel

Results



```
> listings_svm_classifier2
Support Vector Machine object of class "ksvm"

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

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  1.9120921756684e-05

Number of Support Vectors : 2535

Objective Function Value : -2058.226
Training error : 0.14211
```

Confusion Matrix results using Non-linear Kernel Function



```
> nonLinearSvmResults

listings_predictions2   No  Yes
                  No     0    0
                  Yes  451 2715
> confusionMatrix(nonLinearSvmResults)
Confusion Matrix and Statistics


listings_predictions2   No  Yes
                  No     0    0
                  Yes  451 2715

               Accuracy : 0.8575
                 95% CI : (0.8449, 0.8695)
    No Information Rate : 0.8575
    P-Value [Acc > NIR] : 0.5126

                  Kappa : 0
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.8575
             Prevalence : 0.1425
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : No
```

Precision: This gives us the percentage of positive predictions that the listings were correctly predicted that they will get a visit. For this we check the Yes column and Yes row from our confusion matrix.

True Positive (TP) = 2715

False Negative (FN) = 451

False Positive (FP) = 0

True Negative (TN) = 0

Precision = TP / (TP + FP) = 2715 / (2715 + 0) = 1

Recall: This gives us the percentage of listings that were labelled as getting a visit that were predicted as getting a visit.

Recall/Sensitivity: TP / (TP + FN) = 2715 / (2715 + 451) = 0.85

Specificity : This gives us the percentage of listings that were labelled as not getting a visit that were predicted as not getting a visit.

Specificity: TN / (TN + FP) = 0 / (0 + 0) = 0

Accuracy: This gives us the percentage of listings that were correctly predicted.

Accuracy = (TP + TN) / (TP + TN + FP + FN) = (2715 + 0) / (2715 + 0 + 0 + 451) = 1837/3166 = 0.85

F - measure = (2 * precision * recall) / (precision + recall) = (2*1*0.85)/ (1 + 0.85) = 1.7 / 1.85 = 0.91

Research Question: **Based on the data available predict the price of a listing.**

SVM can be used for both classification and regression (SVR). The dependent variable for this question is continuous in nature and thus SVR is carried out. For this we make use of the package "e1071". After the cleaning and the standardization of data is done, we are left with two sets of data namely Data1 and Data2. Data1 comprises of both quantitative and qualitative data, whereas Data1 consists of only quantitative data.

The Root Mean Square Error (RMSE) value for Data1 is 74.90137 while the RMSE value for Data2 is 75.08783. It can be observed that the error value for Data1 is lower that of Data2. Thus Data1 is more effective as it contains more independent variables.

Question 2: Neural Networks

Research Question: **Based on the data available predict the price of a listing.**

For this question we make use of the function "neuralnet" in R. The dependent variable considered is price which is a continuous variable. Neural networks take in inputs in the form of quantitative values only. For the categorical predictive features which had more than 3 levels, we used one-hot encoding technique to convert them into numeric. All binary predictive features are encoded into 1 and -1. Preferably all true values are coded as 1 and false values as -1.

Furthermore it is recommended to normalize the quantitative data as it helps in making a better neural network model. For this there are a number of ways to go about it, but the most common ones are min/max and z-score method. For the purpose of this question we use the min/max method to normalize the quantitative variables.

After the cleaning and the standardization of data is done, we are left with two sets of data namely Data1 and Data2. Data1 comprises of both quantitative and qualitative data, whereas Data1 consists of only quantitative data. We go about computing four variations of neural networks for each of the data-sets.

A neural network is made up of three layers - an input layer, a hidden layer and an output layer. The hidden layer is optional and can range from to none to as many based on the requirement.

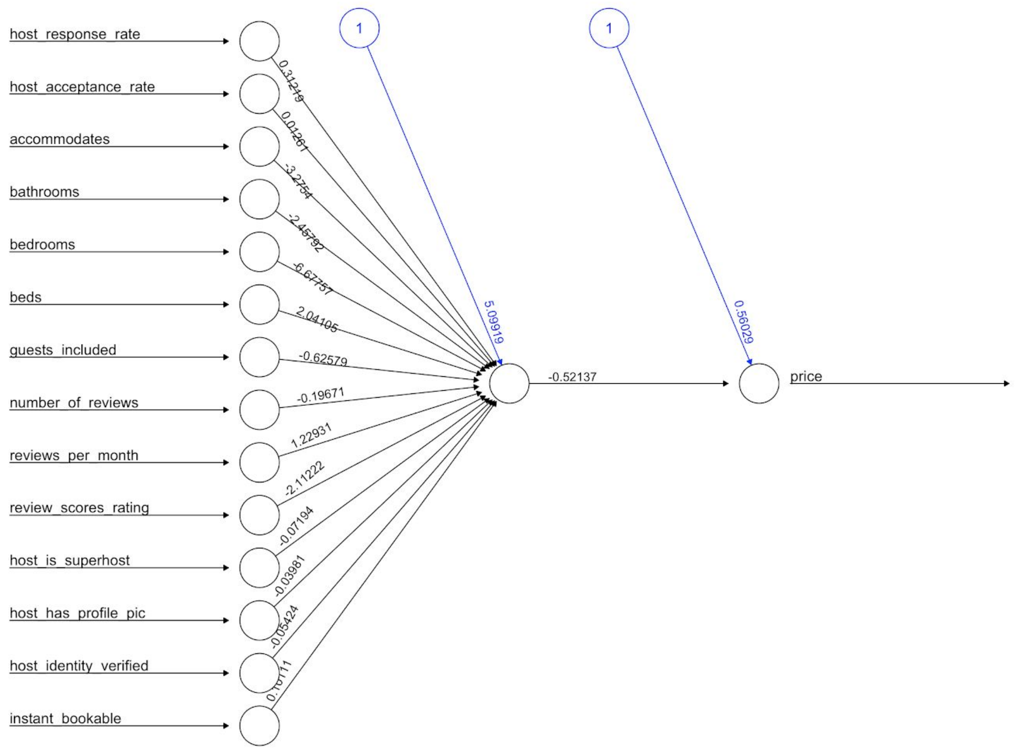
Model	Data-set	Error	Steps	Correlation
1	Data1	42.773923713220	5702	0.7578401324
2	Data1	38.08541160020658	27446	0.7820626123
3	Data1	37.897616604513	4761	0.7812115288
4	Data1	42.774036868011	35360	0.7578192702
5	Data2	43.053855892990	4748	0.7566905411
6	Data2	39.216567456235	44714	0.7812808871
7	Data2	38.30474682754	19594	0.7810708648
8	Data2	43.053350107459	16186	0.7567431985

The default activation function in the “neuralnet” function is logistic. But the fourth and eighth model make use of the tanh activation function.

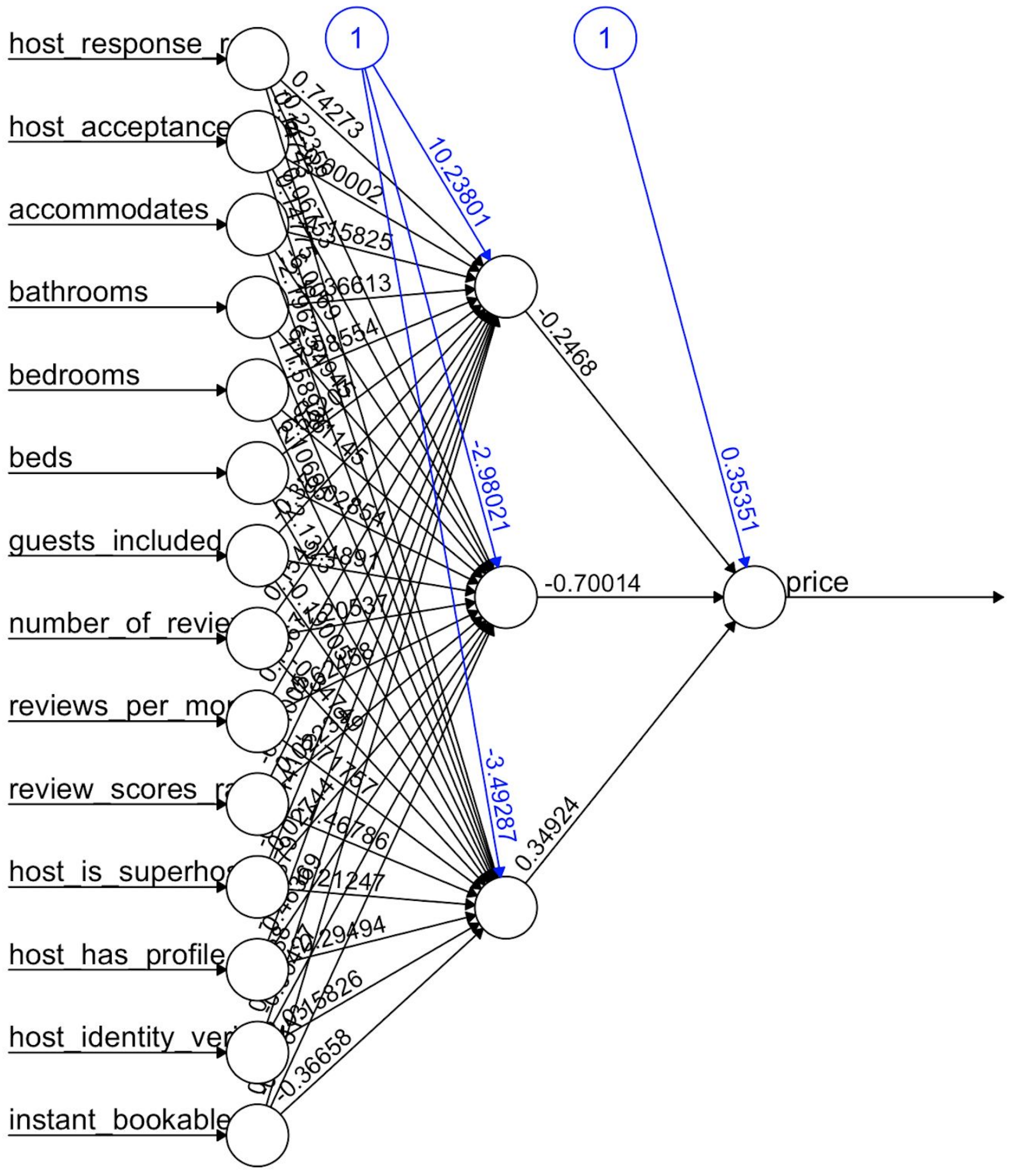
Given below are the plots of all the models. Of all the variation it is the variation that has the most number of hidden layers that is effective. However this is a case by case basis and can be zeroed in only by trial and error method.

Choosing too many hidden layers can result in overfitting the model while not having hidden layer can result in underfitting the model. Data2 seems to be more predictive than Data1 and this is due to the extra number of variables considered in Data2.

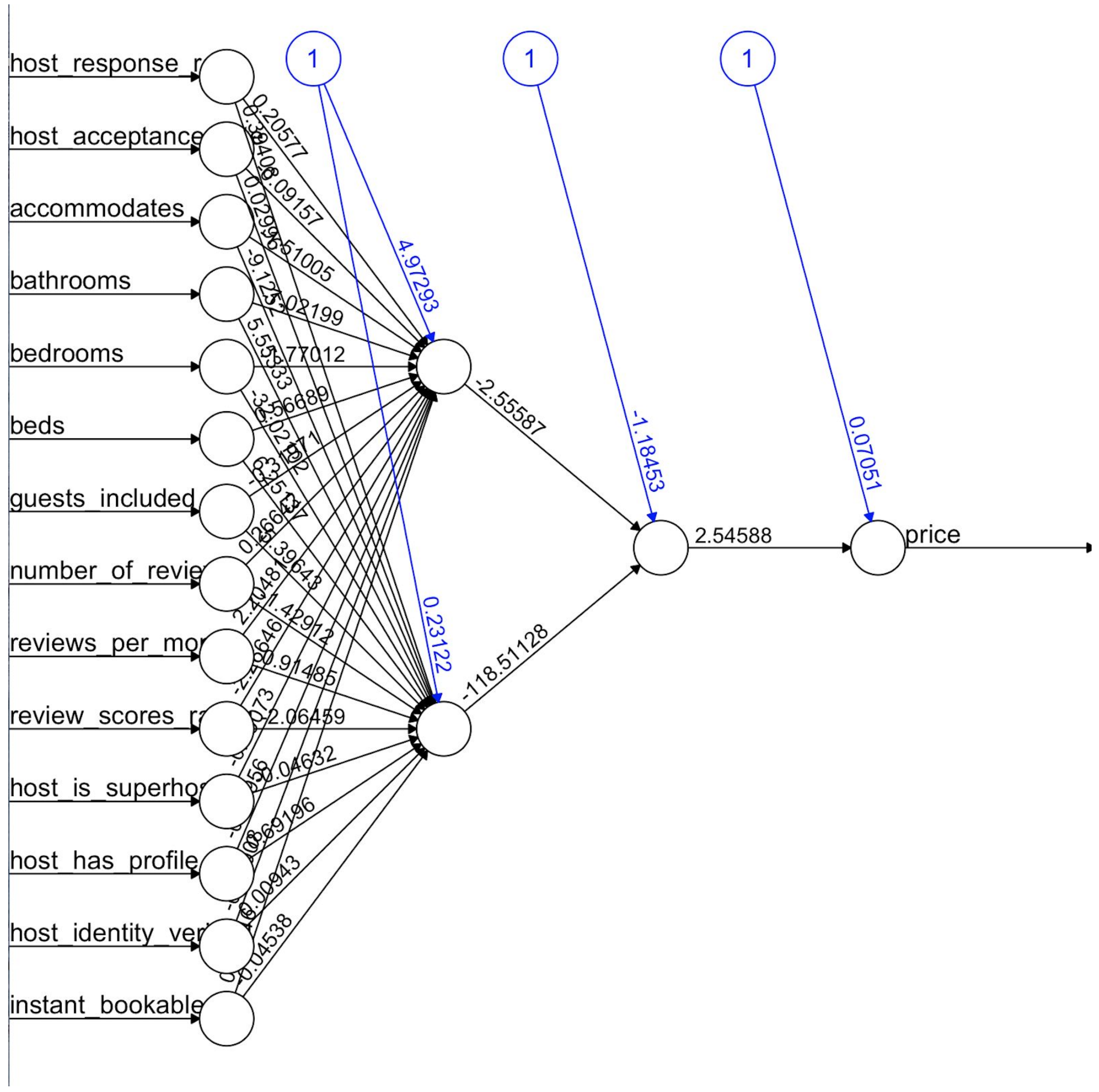
Model 1 - No Hidden Layer



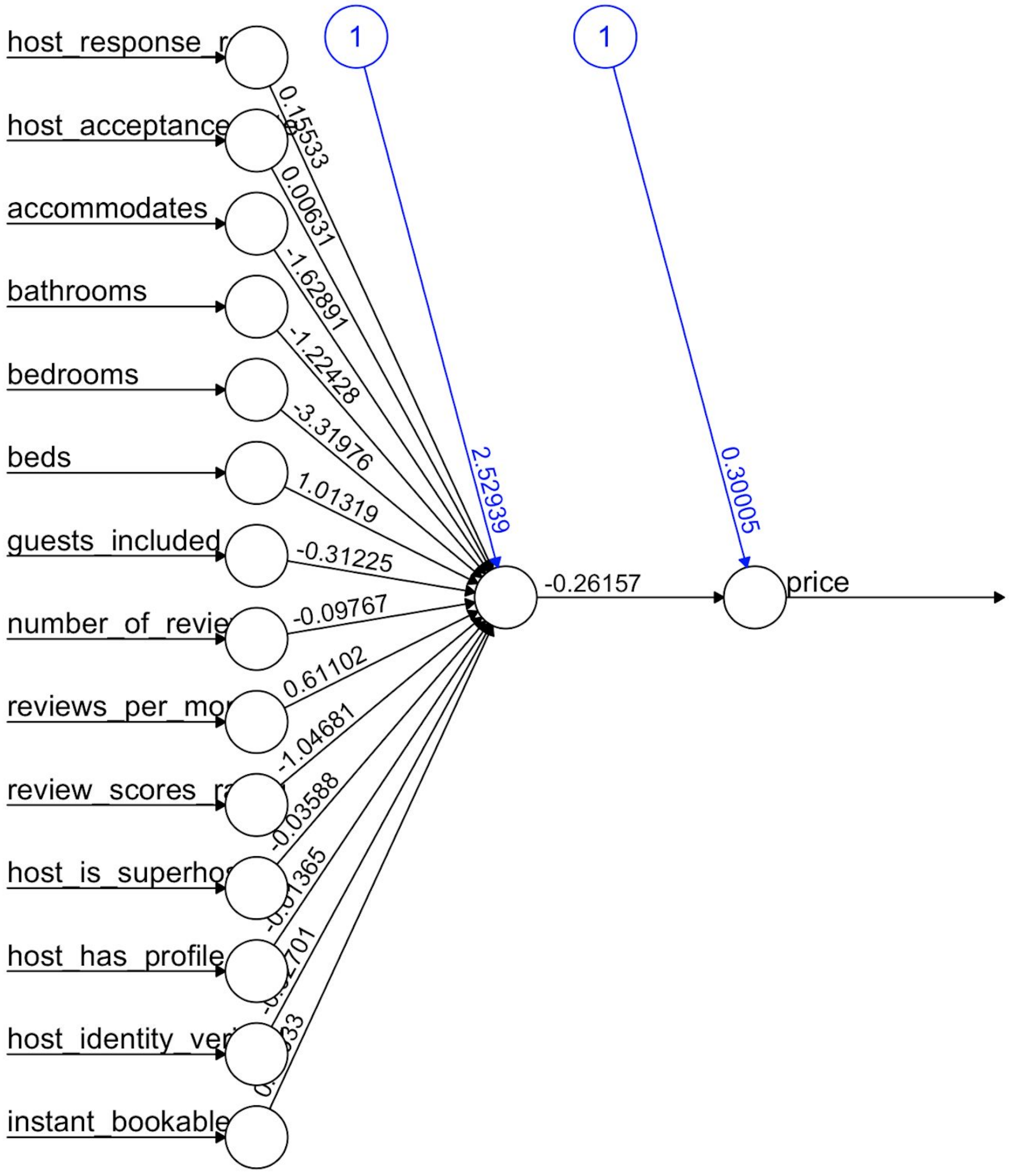
Model 2 - One Hidden Layer (3)



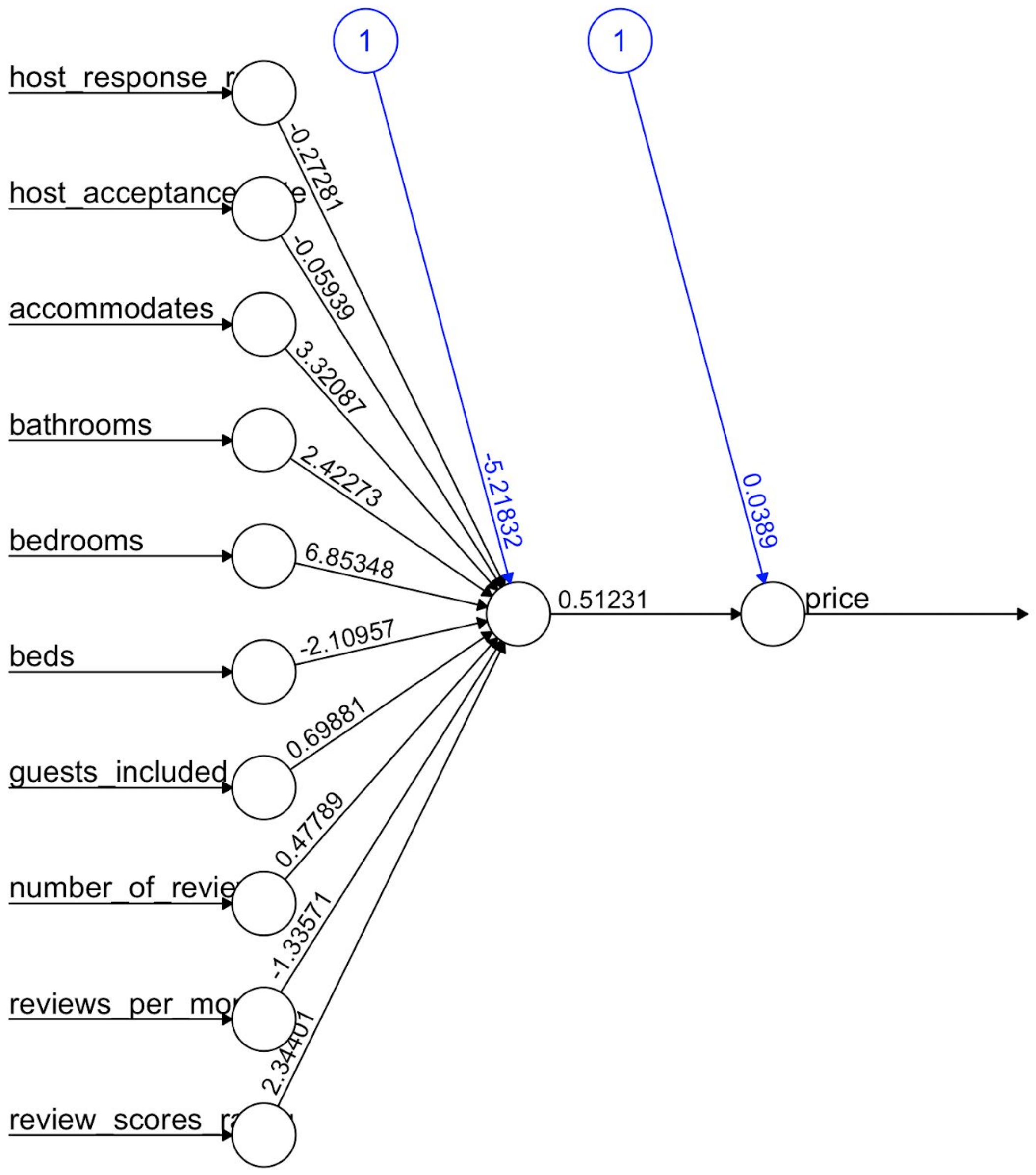
Model 3 - Two Hidden Layers (2,1)



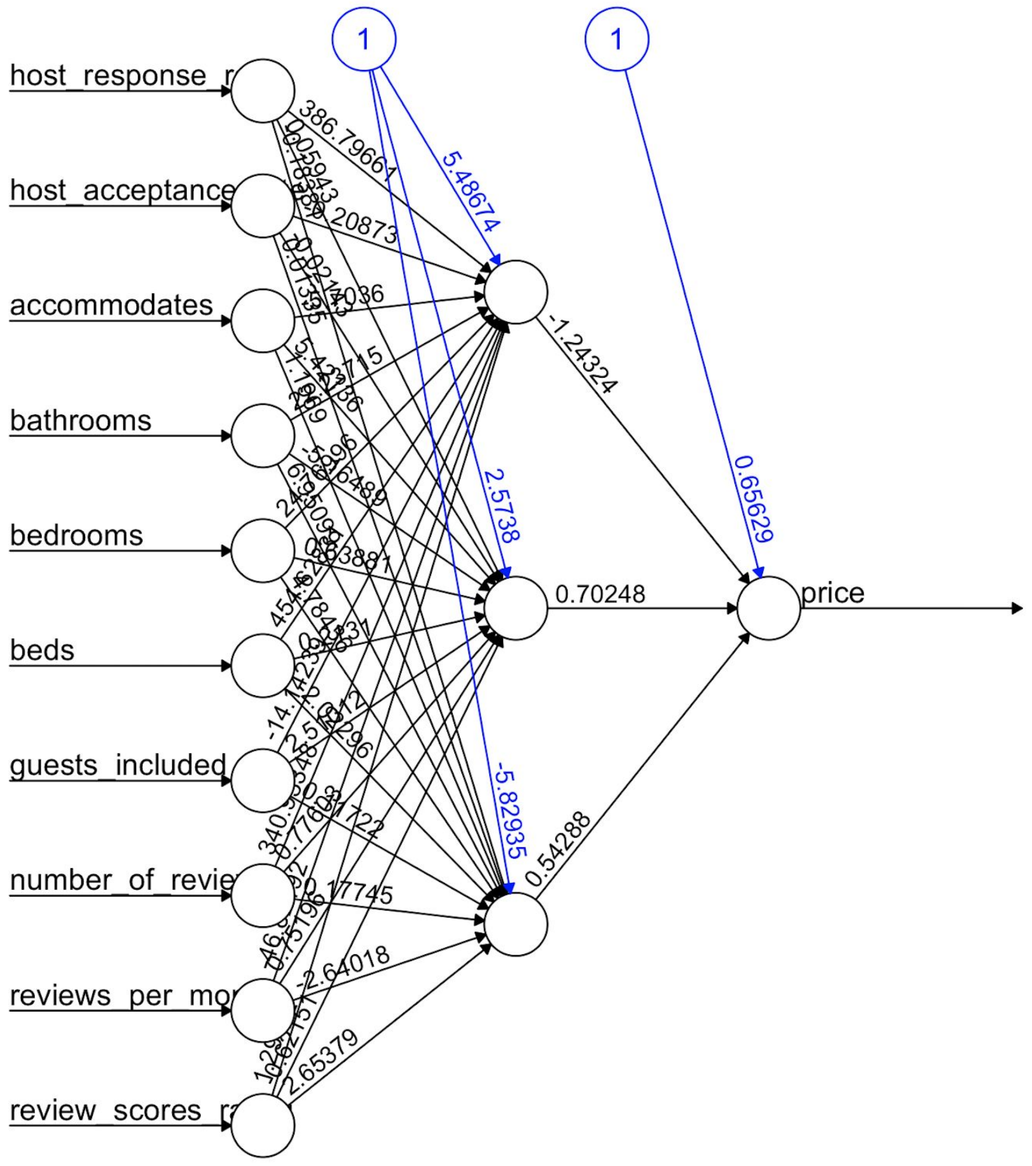
Model 4 - No Hidden Layer, Activation Function - tanh



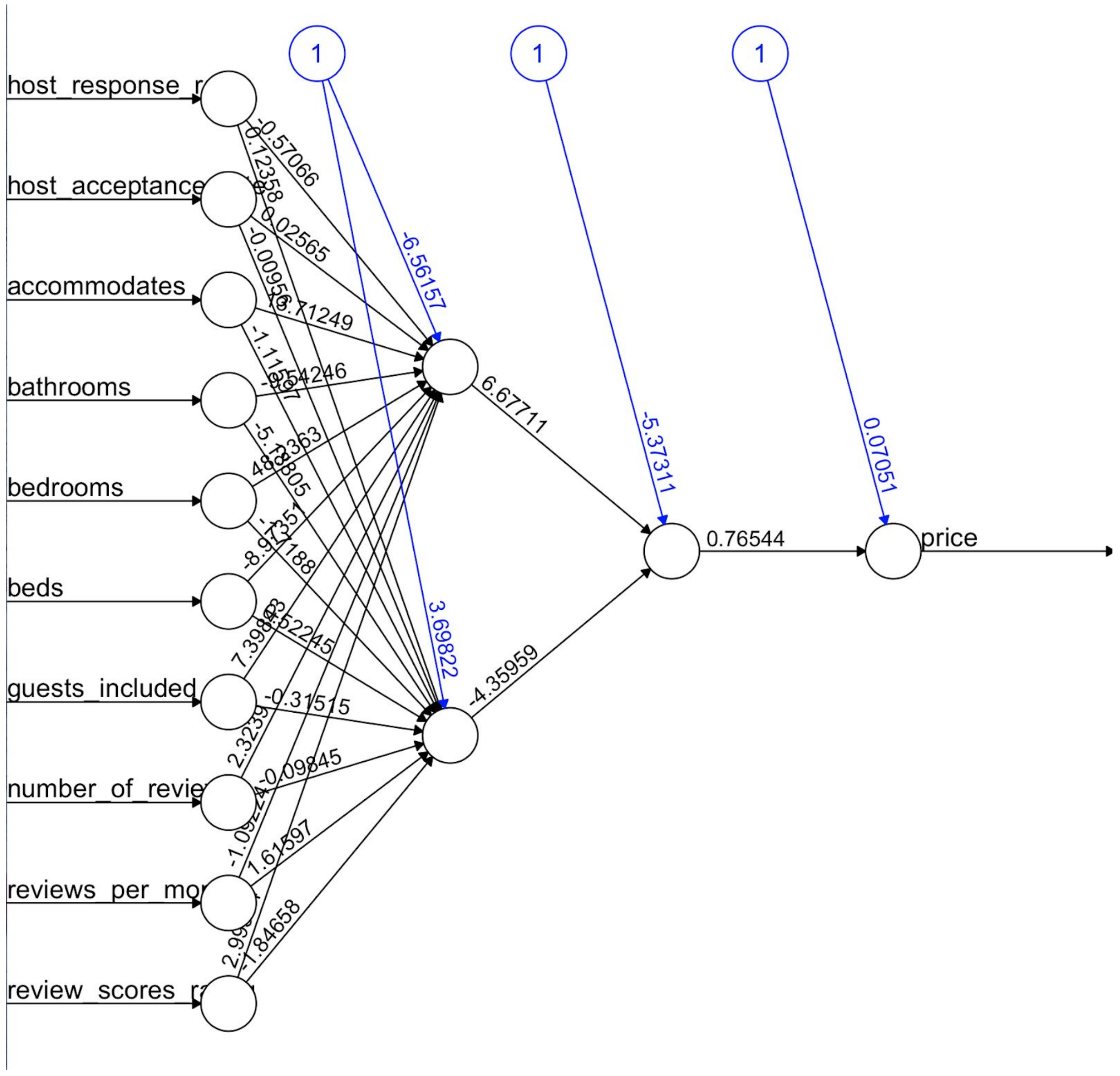
Model 5 - No Hidden Layer



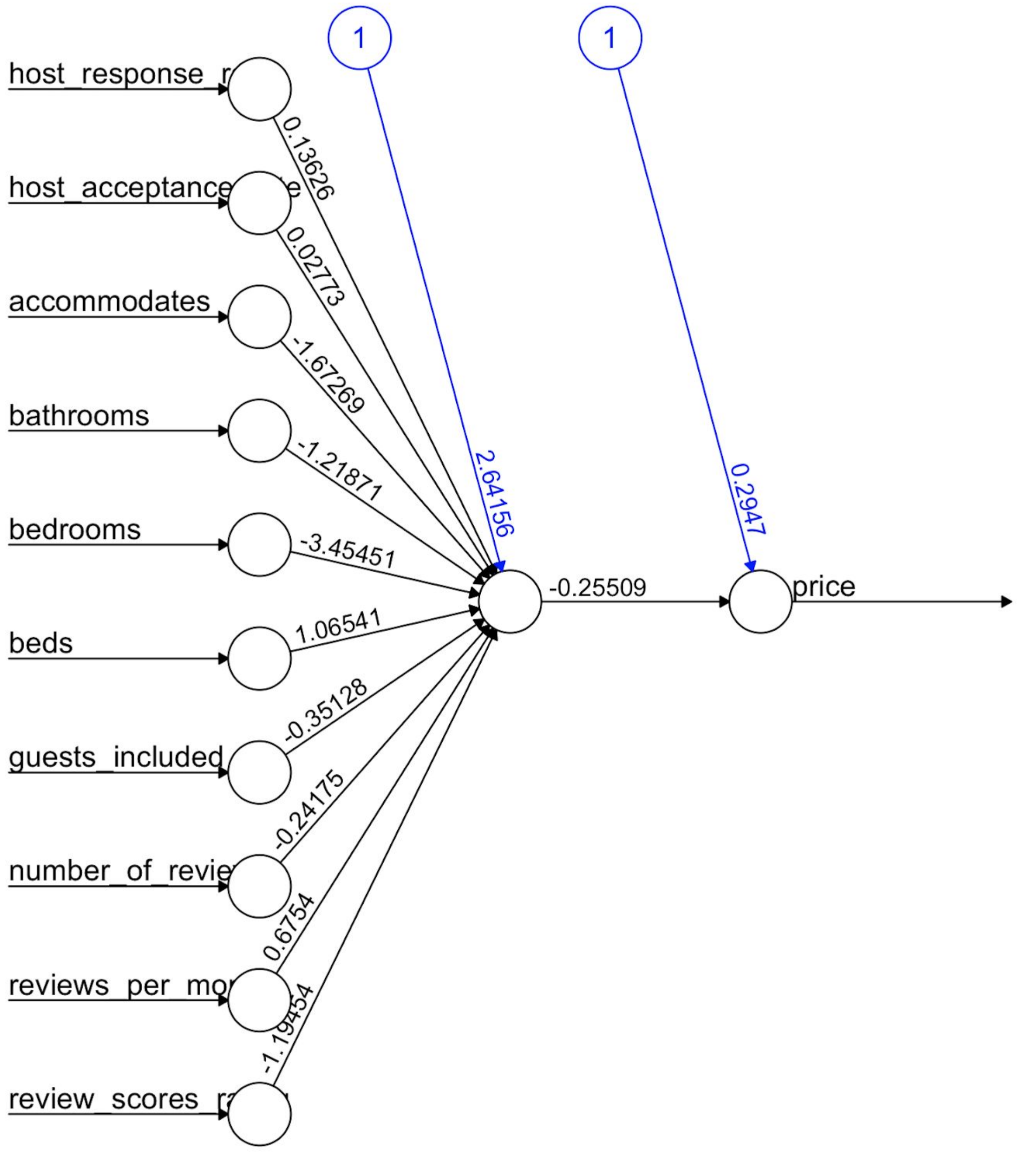
Model 6 - One Hidden Layer (3)



Model 7 - Two Hidden Layers (2,1)



Model 8 - No Hidden Layer. Activation Function - tanh



Research Question: **Classify whether a listing gets a visit or not based on various features.**

In this question our outcome feature 'get_visits' is binary in nature. Thus we are using Neural Network to classify the listings into 2 classes.

Based on the number of reviews per listing, we classify listings whether they get a visit or not. Our assumption is that if a listing has more than 1 review, it will be assigned a label ‘YES’. Those listings that get 0 reviews, will be assigned the label ‘NO’.

Normalization:

These are then encoded as 1s (for YES) and 0s (for NO). Our outcome column ‘get_visits’ is thus encoded in 1s and 0s for computing the Neural Network model.

For the categorical predictive features like room_type, cancellation_policy and host_response_time which had more than 3 levels, we used one-hot encoding technique to convert them into numeric as Neural Networks work with only quantitative data like SVM classifiers.

R code:

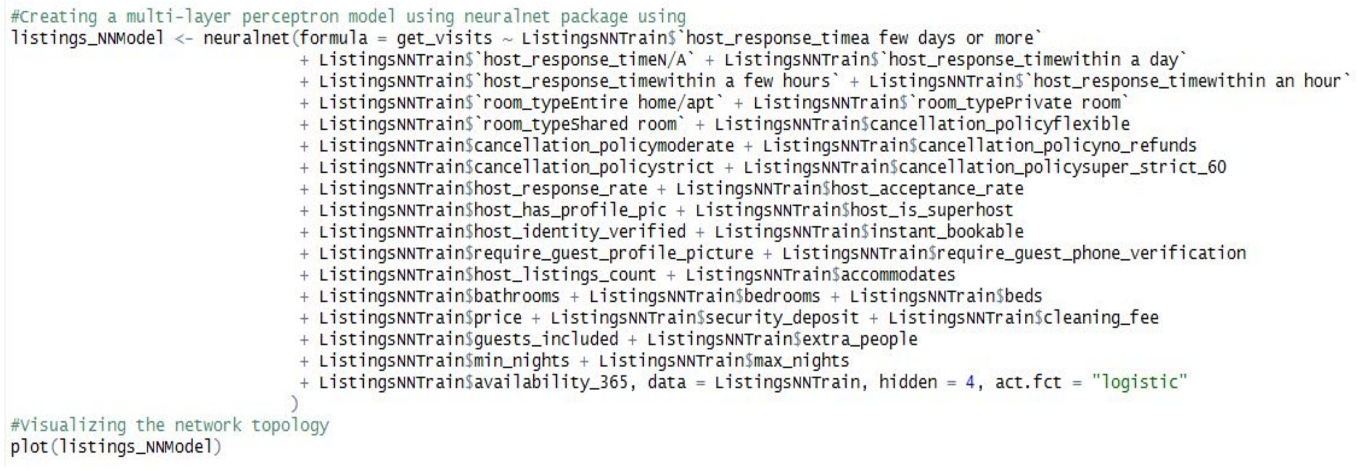
```
FinalListings <- cbind(with(ListingsDF, model.matrix(~ host_response_time + 0)),
                  with(ListingsDF, model.matrix(~ room_type + 0)),
                   with(ListingsDF, model.matrix(~ cancellation_policy + 0))
```

All binary predictive features are encoded into 1 and -1. Preferably all true values are coded as 1 and false values as -1.

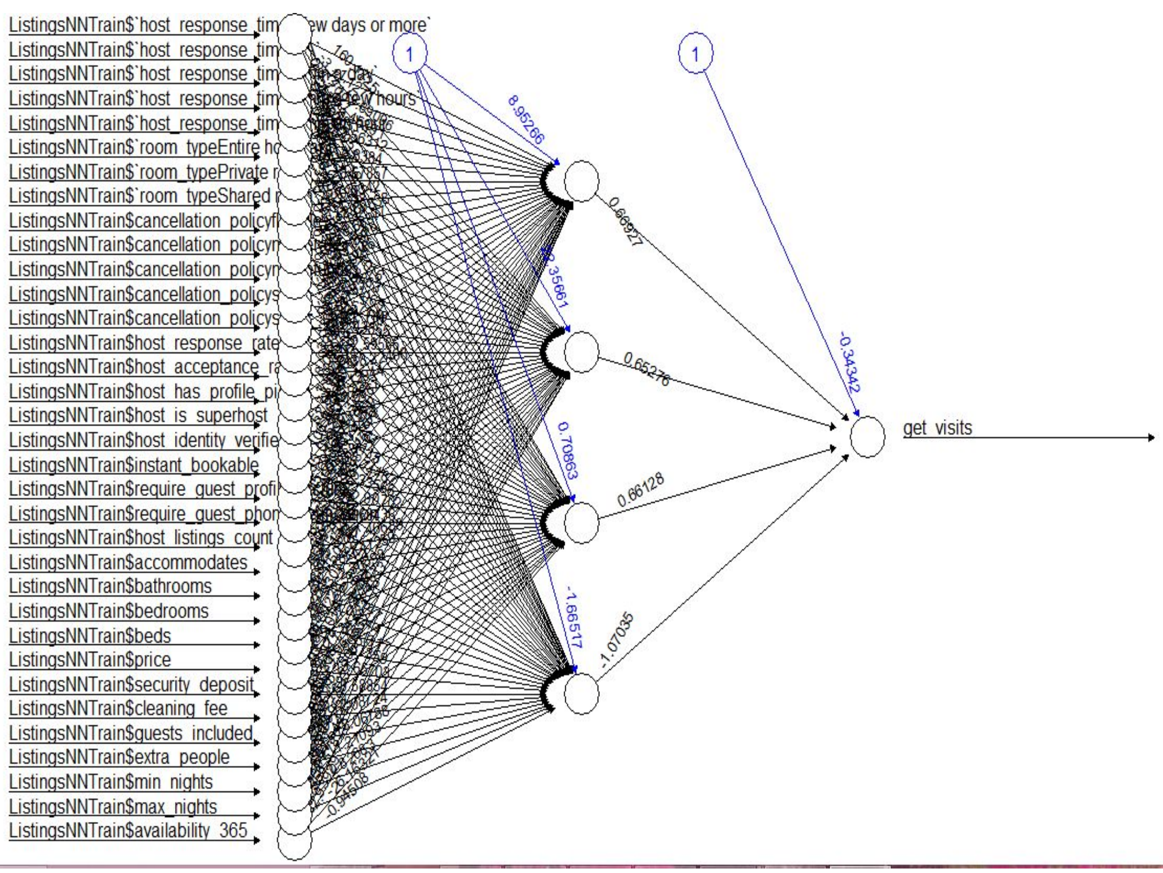
Finally all continuous variables are normalized to values ranging between 0 and 1 using min-max normalization function.

Model 1

Activation function used is logistic which is the default function used while using the neuralnet package in R to compute the Neural Network model for the training data set. This model uses 1 hidden layer with 4 neurons as can be seen in the plot below.



```
#Creating a multi-layer perceptron model using neuralnet package using
listings_NNModel <- neuralnet(formula = get_visits ~ ListingsNNTrain$`host_response_timea few days or more`
                             + ListingsNNTrain$`host_response_timeN/A` + ListingsNNTrain$`host_response_timewithin a day`
                             + ListingsNNTrain$`host_response_timewithin a few hours` + ListingsNNTrain$`host_response_timewithin an hour`
                             + ListingsNNTrain$`room_typeEntire home/apt` + ListingsNNTrain$`room_typePrivate room`
                             + ListingsNNTrain$`room_typeShared room` + ListingsNNTrain$cancellation_policyflexible
                             + ListingsNNTrain$cancellation_policymoderate + ListingsNNTrain$cancellation_policyno_refunds
                             + ListingsNNTrain$cancellation_policystrict + ListingsNNTrain$cancellation_policysuper_strict_60
                             + ListingsNNTrain$host_response_rate + ListingsNNTrain$host_acceptance_rate
                             + ListingsNNTrain$host_has_profile_pic + ListingsNNTrain$host_is_superhost
                             + ListingsNNTrain$host_identity_verified + ListingsNNTrain$instant_bookable
                             + ListingsNNTrain$require_guest_profile_picture + ListingsNNTrain$require_guest_phone_verification
                             + ListingsNNTrain$host_listings_count + ListingsNNTrain$accommodates
                             + ListingsNNTrain$bathrooms + ListingsNNTrain$bedrooms + ListingsNNTrain$beds
                             + ListingsNNTrain$price + ListingsNNTrain$security_deposit + ListingsNNTrain$cleaning_fee
                             + ListingsNNTrain$guests_included + ListingsNNTrain$extra_people
                             + ListingsNNTrain$min_nights + ListingsNNTrain$max_nights
                             + ListingsNNTrain$availability_365, data = ListingsNNTrain, hidden = 4, act.fct = "logistic"
                             )
#Visualizing the network topology
plot(listings_NNModel)
```



ListingsNNTrain$`host_response_tim
ew days or more`
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`room_typeEntire h
ListingsNNTrain$`room_typePrivate
ListingsNNTrain$`room_typeShared
ListingsNNTrain$cancellation_policyf
ListingsNNTrain$cancellation_policym
ListingsNNTrain$cancellation_policyn
ListingsNNTrain$cancellation_policys
ListingsNNTrain$cancellation_policys
ListingsNNTrain$host_response_rate
ListingsNNTrain$host_acceptance_ra
ListingsNNTrain$host_has_profile_pi
ListingsNNTrain$host_is_superhost
ListingsNNTrain$host_identity_verifie
ListingsNNTrain$instant_bookable
ListingsNNTrain$require_guest_profi
ListingsNNTrain$require_guest_phon
ListingsNNTrain$host_listings_count
ListingsNNTrain$accommodates
ListingsNNTrain$bathrooms
ListingsNNTrain$bedrooms
ListingsNNTrain$beds
ListingsNNTrain$price
ListingsNNTrain$security_deposit
ListingsNNTrain$cleaning_fee
ListingsNNTrain$guests_included
ListingsNNTrain$extra_people
ListingsNNTrain$min_nights
ListingsNNTrain$max_nights
ListingsNNTrain$availability_365
1
1
8.95266
0.66927
-0.35661
0.65276
-0.34342
0.70863
0.66128
-1.66517
-1.07035
get_visits

Accuracy:

Confusion Matrix for the Neural Network binary classifier based on NN Model 1 is:

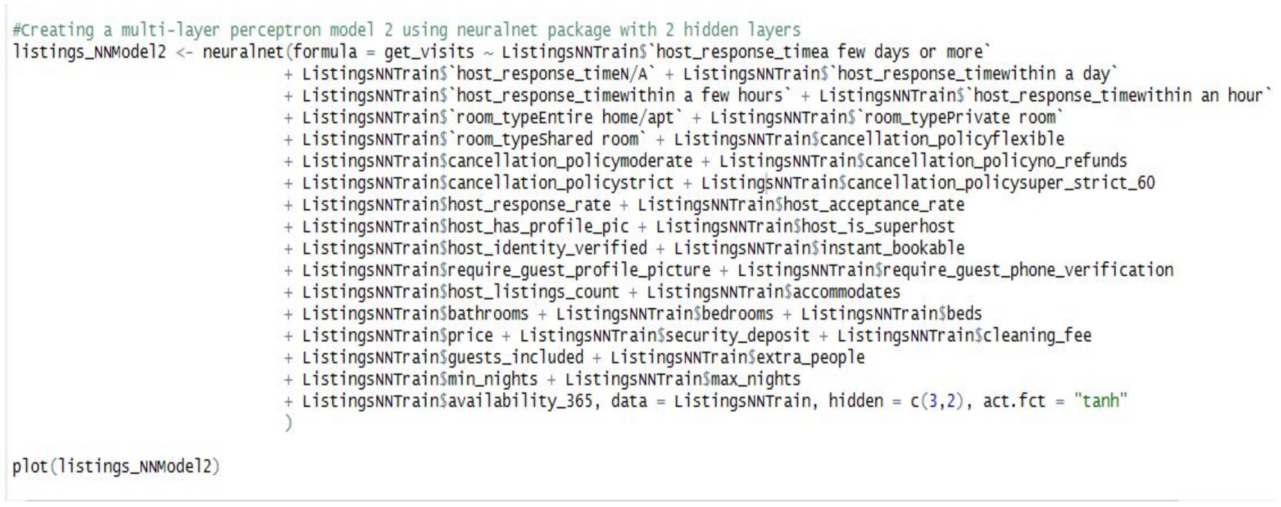
```
norm_predicted_get_visits    0    1

                  0            58  103

                  1           426 2544
```

58 listings were correctly classified as listings that won't get any visit while 2544 listings were correctly classified as listings that will get a visit.

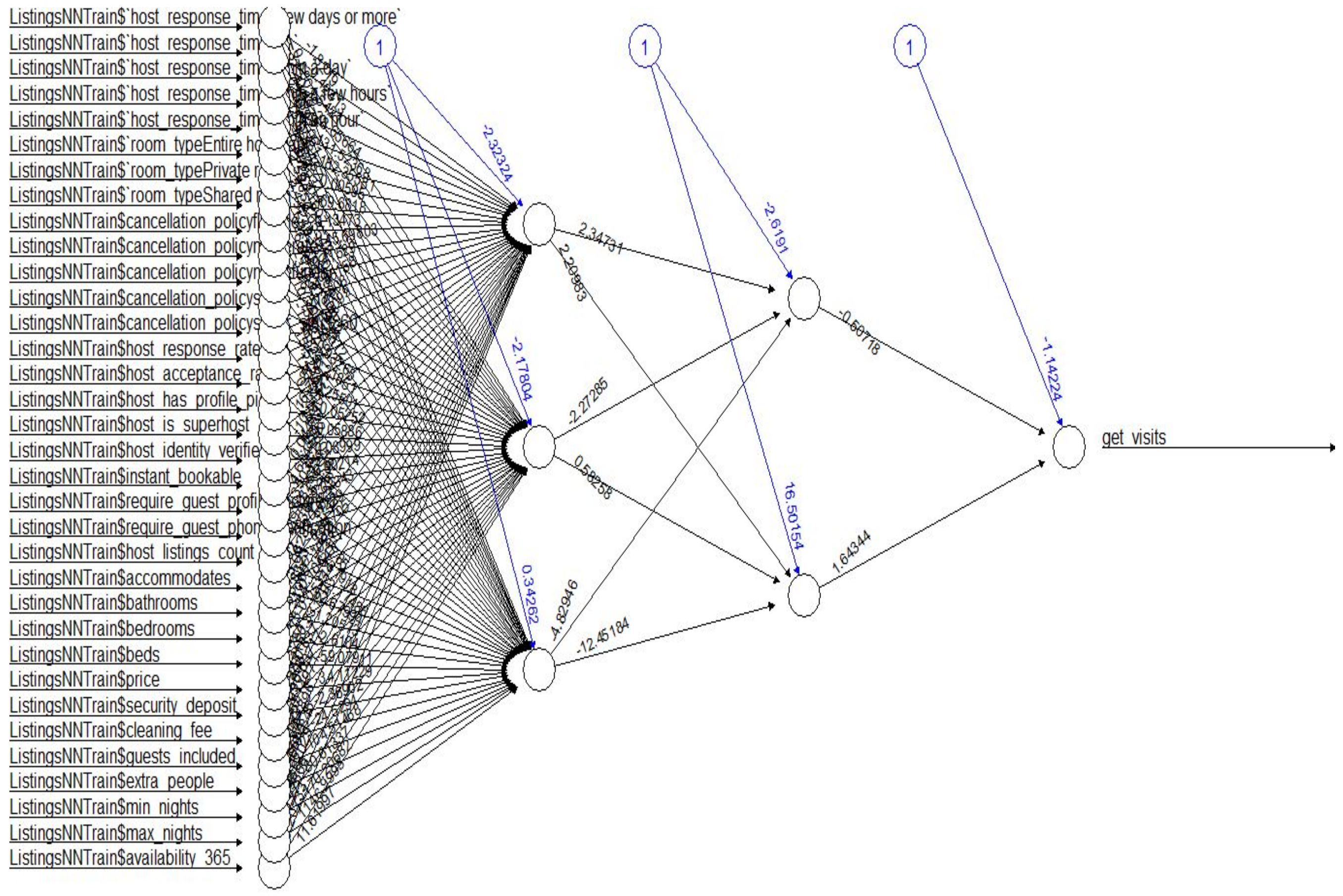
Model 2

Activation function used is tanh which is the default function used while using the neuralnet package in R to compute the Neural Network model for the training data set. This model uses 2 hidden layers with 3 neurons in the first layer and 2 neurons in the second layer as can be seen in the plot below.

<u>R code:</u>



```r
#Creating a multi-layer perceptron model 2 using neuralnet package with 2 hidden layers
listings_NNModel2 <- neuralnet(formula = get_visits ~ ListingsNNTrain$`host_response_timea few days or more`
                               + ListingsNNTrain$`host_response_timeN/A` + ListingsNNTrain$`host_response_timewithin a day`
                               + ListingsNNTrain$`host_response_timewithin a few hours` + ListingsNNTrain$`host_response_timewithin an hour`
                               + ListingsNNTrain$`room_typeEntire home/apt` + ListingsNNTrain$`room_typePrivate room`
                               + ListingsNNTrain$`room_typeShared room` + ListingsNNTrain$cancellation_policyflexible
                               + ListingsNNTrain$cancellation_policymoderate + ListingsNNTrain$cancellation_policyno_refunds
                               + ListingsNNTrain$cancellation_policystrict + ListingsNNTrain$cancellation_policysuper_strict_60
                               + ListingsNNTrain$host_response_rate + ListingsNNTrain$host_acceptance_rate
                               + ListingsNNTrain$host_has_profile_pic + ListingsNNTrain$host_is_superhost
                               + ListingsNNTrain$host_identity_verified + ListingsNNTrain$instant_bookable
                               + ListingsNNTrain$require_guest_profile_picture + ListingsNNTrain$require_guest_phone_verification
                               + ListingsNNTrain$host_listings_count + ListingsNNTrain$accommodates
                               + ListingsNNTrain$bathrooms + ListingsNNTrain$bedrooms + ListingsNNTrain$beds
                               + ListingsNNTrain$price + ListingsNNTrain$security_deposit + ListingsNNTrain$cleaning_fee
                               + ListingsNNTrain$guests_included + ListingsNNTrain$extra_people
                               + ListingsNNTrain$min_nights + ListingsNNTrain$max_nights
                               + ListingsNNTrain$availability_365, data = ListingsNNTrain, hidden = c(3,2), act.fct = "tanh"
                               )

plot(listings_NNModel2)
```


ListingsNNTrain$`host_response_time`a few days or more`
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`host_response_tim
ListingsNNTrain$`room_typeEntire ho
ListingsNNTrain$`room_typePrivate r
ListingsNNTrain$`room_typeShared r
ListingsNNTrain$cancellation_policyfl
ListingsNNTrain$cancellation_policym
ListingsNNTrain$cancellation_policym
ListingsNNTrain$cancellation_policys
ListingsNNTrain$cancellation_policys
ListingsNNTrain$host_response_rate
ListingsNNTrain$host_acceptance_ra
ListingsNNTrain$host_has_profile_pi
ListingsNNTrain$host_is_superhost
ListingsNNTrain$host_identity_verifie
ListingsNNTrain$instant_bookable
ListingsNNTrain$require_guest_profi
ListingsNNTrain$require_guest_phon
ListingsNNTrain$host_listings_count
ListingsNNTrain$accommodates
ListingsNNTrain$bathrooms
ListingsNNTrain$bedrooms
ListingsNNTrain$beds
ListingsNNTrain$price
ListingsNNTrain$security_deposit
ListingsNNTrain$cleaning_fee
ListingsNNTrain$guests_included
ListingsNNTrain$extra_people
ListingsNNTrain$min_nights
ListingsNNTrain$max_nights
ListingsNNTrain$availability_365
1
1
1
-2.32324
-2.17804
0.34262
2.34731
2.20983
-2.27285
0.58258
4.82946
-12.45184
-2.6191
16.50154
-0.60718
1.64344
-1.14224
get_visits

Accuracy:

Confusion Matrix for the Neural Network binary classifier based on NN Model 2 is:

```
norm_predicted_get_visits2    0    1

                    0            4    4

                    1          480 2643
```

4 listings were correctly classified as listings that won't get any visit while 2643 listings were correctly classified as listings that will get a visit.

Model 1 has a better accuracy.

Question 3: Clustering

Clustering groups a set of data points into groups in such a way that the points in the same group are similar to each other as compared to other points in another group. Clustering is efficient when we want to group quantitative or numerical data.

Method I: K-Means

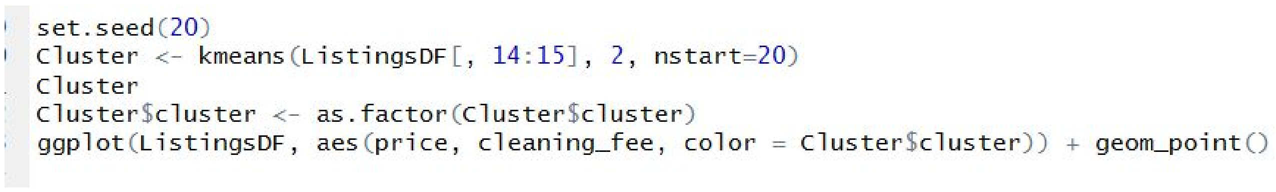
Research Question: **Classify whether a listing gets a visit or not based on various features.**

K-means clustering can be done accurately on numerical data. Since our data is predominantly categorical, we have performed clustering on columns price and cleaning fee and we have clustered it based on our predictor variable, gets_visit.

Get_visit is a binary variable that states if a listing got a visit or not. After performing clustering and giving the size of clusters as 2, we get the following result.

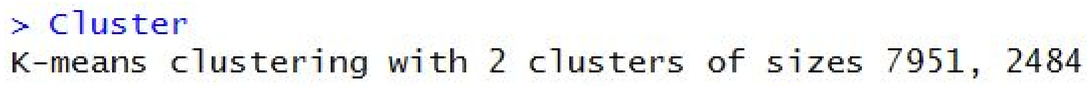
R code:

Code for making the clusters using k means algorithm.



```
set.seed(20)
Cluster <- kmeans(ListingsDF[, 14:15], 2, nstart=20)
Cluster
Cluster$cluster <- as.factor(Cluster$cluster)
ggplot(ListingsDF, aes(price, cleaning_fee, color = Cluster$cluster)) + geom_point()
```

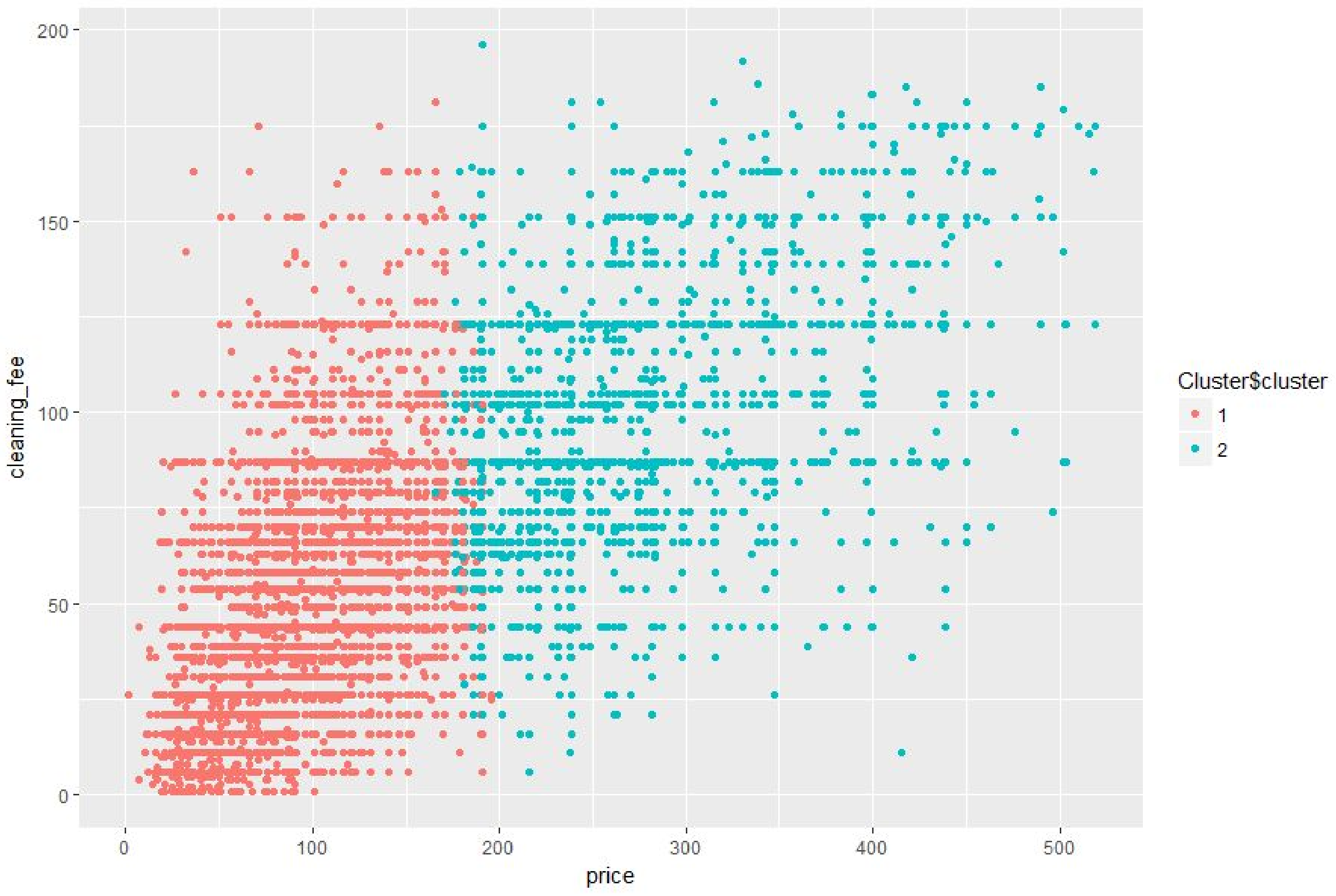
Output for the cluster being formed. Below we can see that 2 clusters are formed of sizes 7951 and 2484 using k means clustering algorithm.



```
> Cluster
K-means clustering with 2 clusters of sizes 7951, 2484
```

Now, we plot the points on a graph using ggplot. Below the graph shows how the two clusters are formed. One cluster is colored red and the other is colored green.

There is not a clear separation between the two clusters thus showing that the clustering is not very efficient.



Method II: Density Based Clustering

Density based clustering groups points together that are closely packed to each other, in a given space. Outliers are marked as points that are alone in low-density areas.

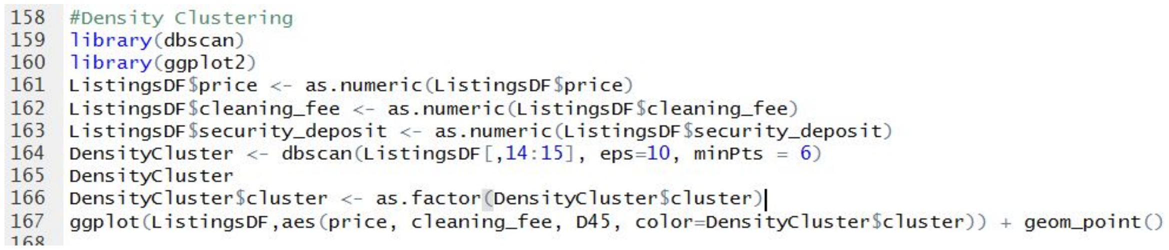
DBSCAN (density-based spatial clustering of applications with noise) is a supervised learning clustering method. We do not need to mention the number of clusters, as was the case with k-means clustering.

Research Question: **Classify whether a listing gets a visit or not based on various features.**

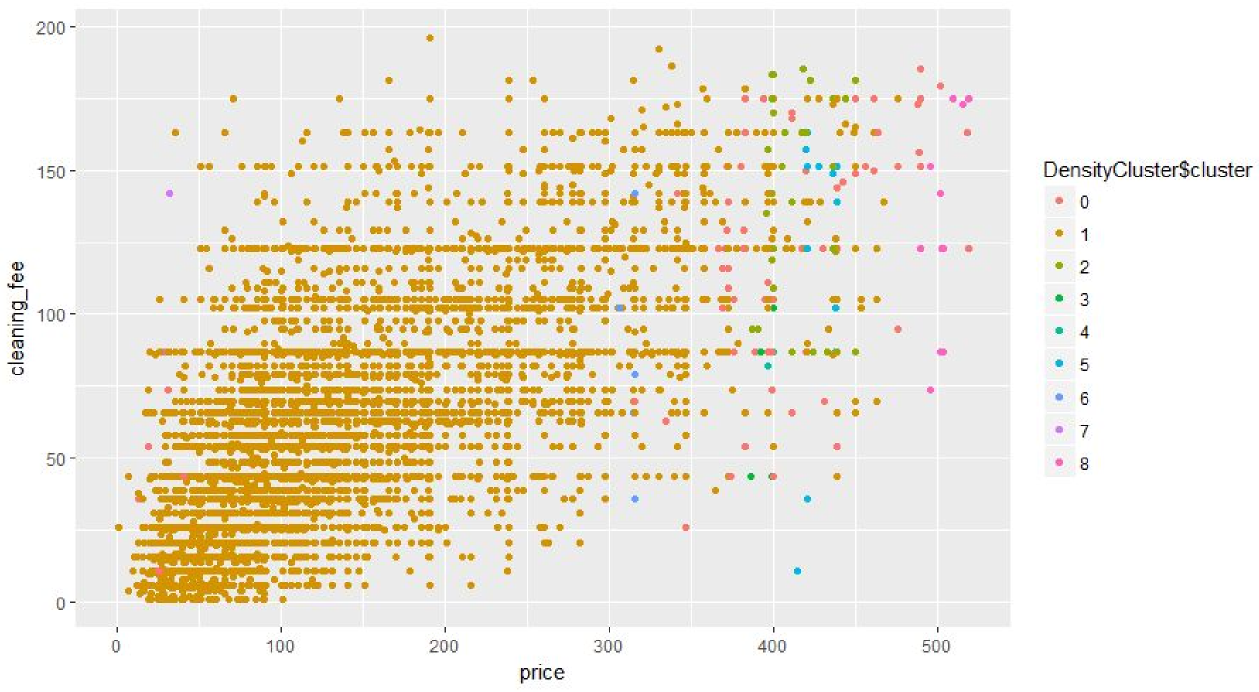
Again, density based clustering required numeric data so we take into consideration price and cleaning fee. After performing dbscan function on the data we get the following results.

R Code:

Below is the code for clustering the data using density-based spatial clustering algorithm.



```
#Density Clustering
library(dbscan)
library(ggplot2)
ListingsDF$price <- as.numeric(ListingsDF$price)
ListingsDF$cleaning_fee <- as.numeric(ListingsDF$cleaning_fee)
ListingsDF$security_deposit <- as.numeric(ListingsDF$security_deposit)
DensityCluster <- dbscan(ListingsDF[,14:15], eps=10, minPts = 6)
DensityCluster
DensityCluster$cluster <- as.factor(DensityCluster$cluster)
ggplot(ListingsDF,aes(price, cleaning_fee, D45, color=DensityCluster$cluster)) + geom_point()
```



The above graph shows that the data was divided into 8 clusters. One cluster seems to be very large and prominent as compared to the other clusters.

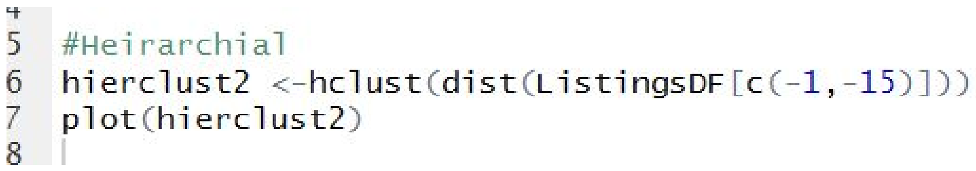
Method III: Hierarchical Clustering

Hierarchical clustering is an alternative approach which builds a hierarchy from the bottom-up, and doesn't require us to specify the number of clusters beforehand, as opposed to k-means.

Research Question: **Classify whether a listing gets a visit or not based on various features.**

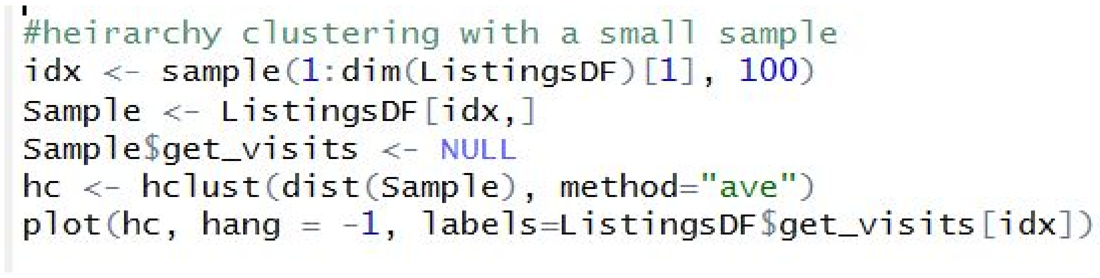
When hierarchical clustering was performed on the entire data set, the branches and nodes were not visible towards the bottom of the tree due to the density of data.

R code:



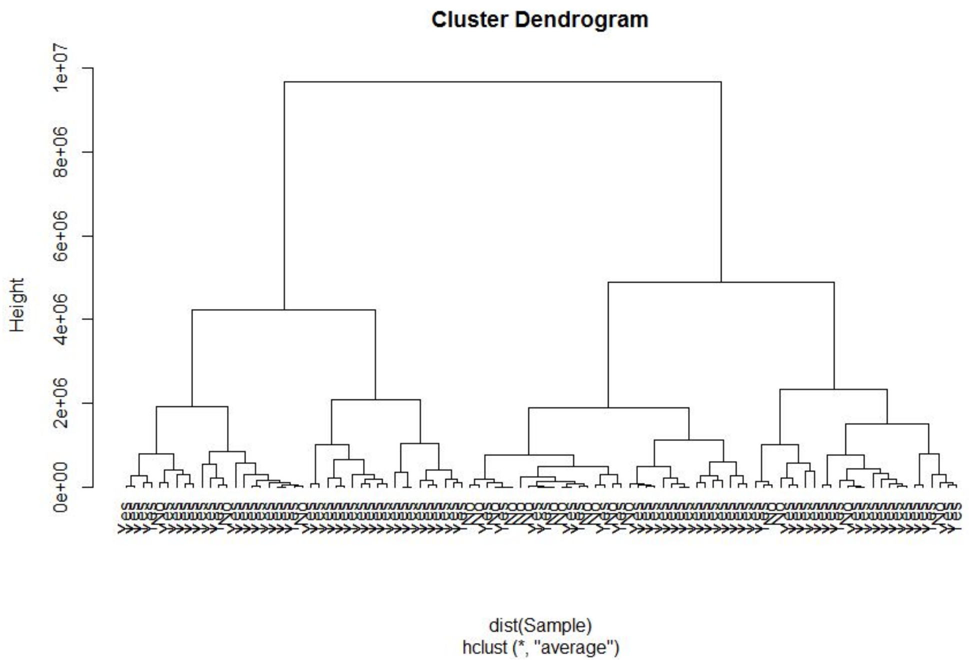
```
#Heirarchial
hierclust2 <-hclust(dist(ListingsDF[c(-1,-15)]))
plot(hierclust2)
```

This code creates a cluster dendrogram of the entire data. As it cannot be comprehended, we took a sample of the data and performed clustering.



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

The result for the above sample made is as follows.



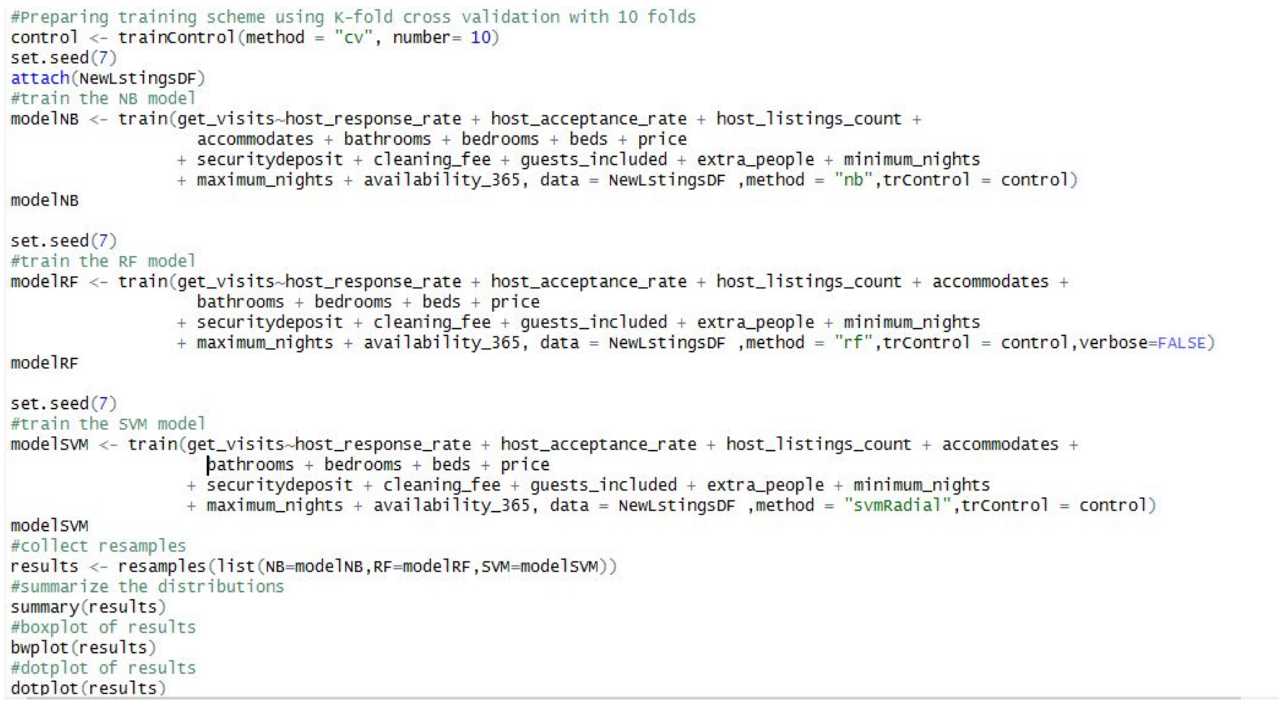
Question 4: Comparative Analysis

We have performed a comparative analysis of 3 classifiers SVM, RF and NB using Caret package in R.

CV Technique 1

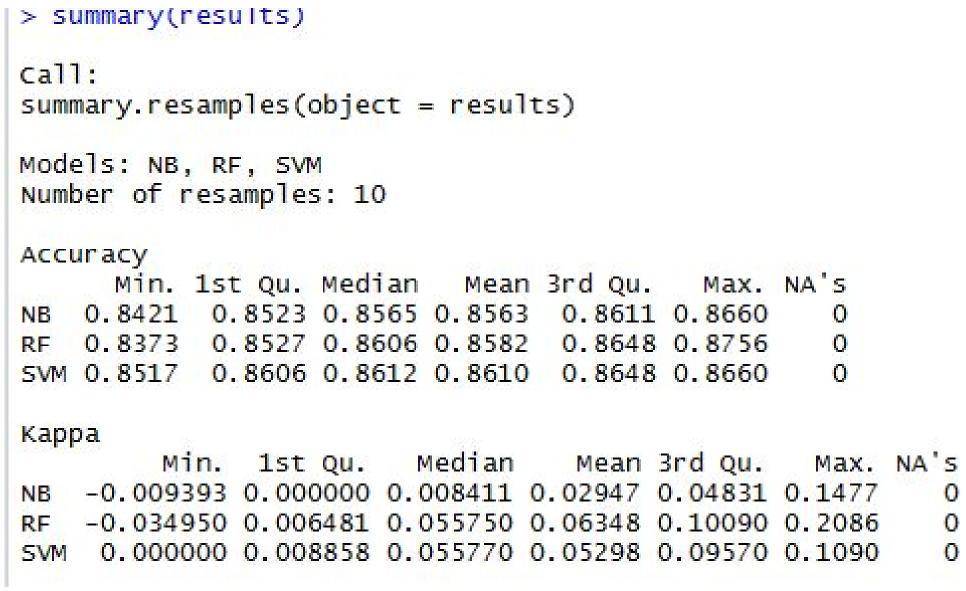
The cross-validation technique used in this case is K fold cross validation using 10 folds.

R code:



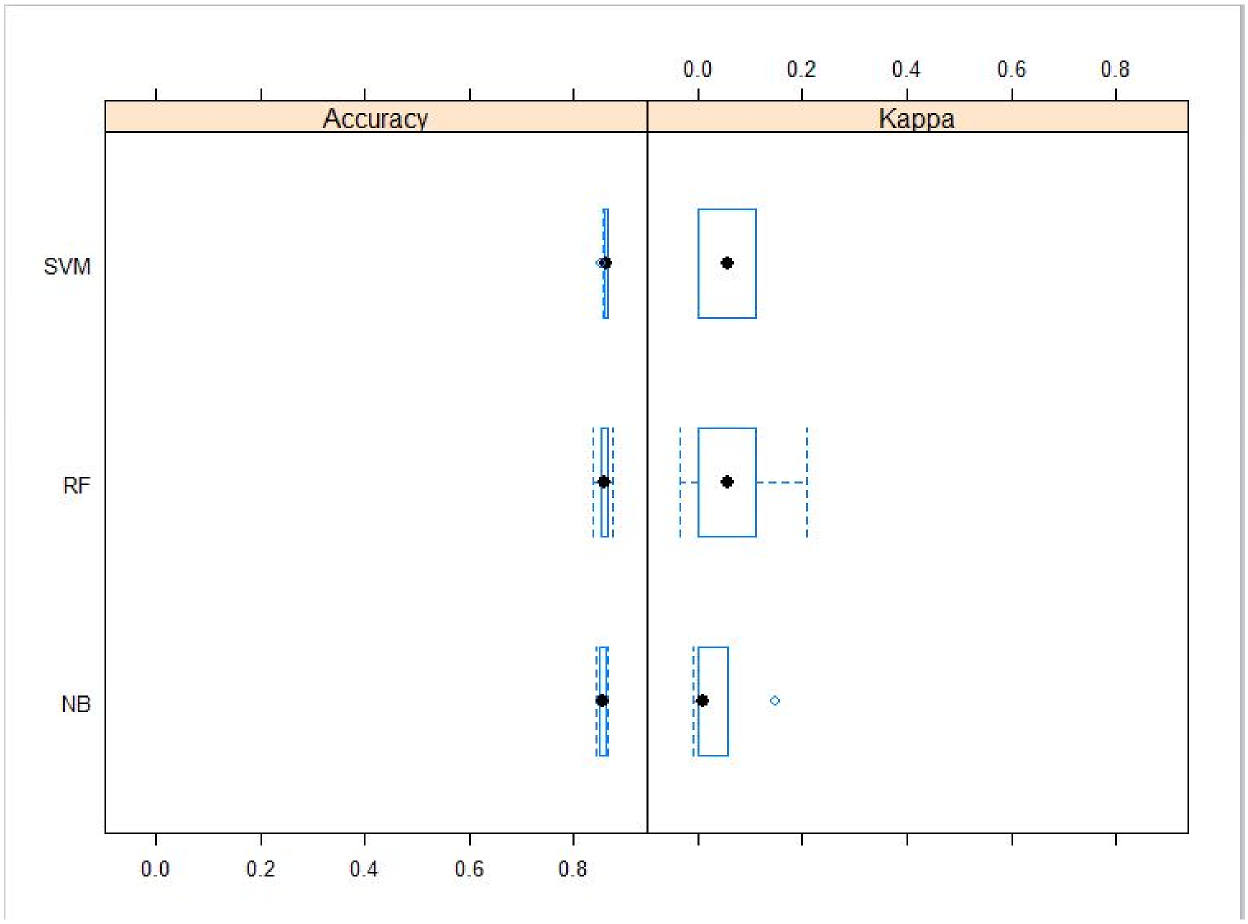
```
#Preparing training scheme using K-fold cross validation with 10 folds
control <- trainControl(method = "cv", number= 10)
set.seed(7)
attach(NewLstingsDF)
#train the NB model
modelNB <- train(get_visits~host_response_rate + host_acceptance_rate + host_listings_count +
                   accommodates + bathrooms + bedrooms + beds + price
                 + securitydeposit + cleaning_fee + guests_included + extra_people + minimum_nights
                 + maximum_nights + availability_365, data = NewLstingsDF ,method = "nb",trControl = control)
modelNB

set.seed(7)
#train the RF model
modelRF <- train(get_visits~host_response_rate + host_acceptance_rate + host_listings_count + accommodates +
                   bathrooms + bedrooms + beds + price
                 + securitydeposit + cleaning_fee + guests_included + extra_people + minimum_nights
                 + maximum_nights + availability_365, data = NewLstingsDF ,method = "rf",trControl = control,verbose=FALSE)
modelRF

set.seed(7)
#train the SVM model
modelSVM <- train(get_visits~host_response_rate + host_acceptance_rate + host_listings_count + accommodates +
                    bathrooms + bedrooms + beds + price
                  + securitydeposit + cleaning_fee + guests_included + extra_people + minimum_nights
                  + maximum_nights + availability_365, data = NewLstingsDF ,method = "svmRadial",trControl = control)
modelSVM
#collect resamples
results <- resamples(list(NB=modelNB,RF=modelRF,SVM=modelSVM))
#summarize the distributions
summary(results)
#boxplot of results
bwplot(results)
#dotplot of results
dotplot(results)
```

Summary of our results which shows that RF and SVM have performed almost equally well with a median value of 0.86



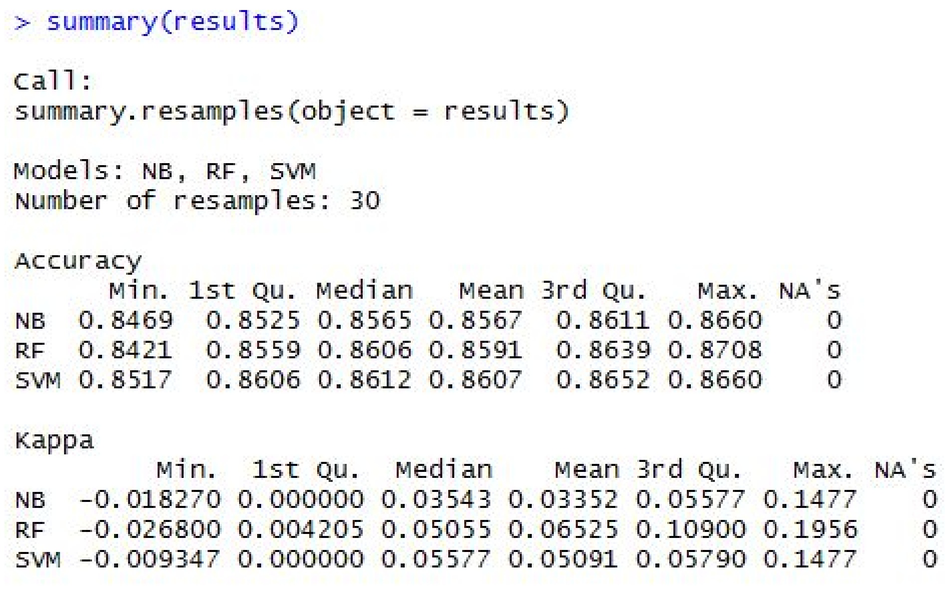
```
> summary(results)

Call:
summary.resamples(object = results)

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

Accuracy
      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NB  0.8421  0.8523 0.8565 0.8563  0.8611 0.8660    0
RF  0.8373  0.8527 0.8606 0.8582  0.8648 0.8756    0
SVM 0.8517  0.8606 0.8612 0.8610  0.8648 0.8660    0

Kappa
         Min.  1st Qu.   Median    Mean 3rd Qu.   Max. NA's
NB  -0.009393 0.000000 0.008411 0.02947 0.04831 0.1477    0
RF  -0.034950 0.006481 0.055750 0.06348 0.10090 0.2086    0
SVM  0.000000 0.008858 0.055770 0.05298 0.09570 0.1090    0
```

Boxplot of our results



CV Technique 2

The cross-validation technique used in this case is repeated K fold cross validation using 10 folds with sampling thrice.

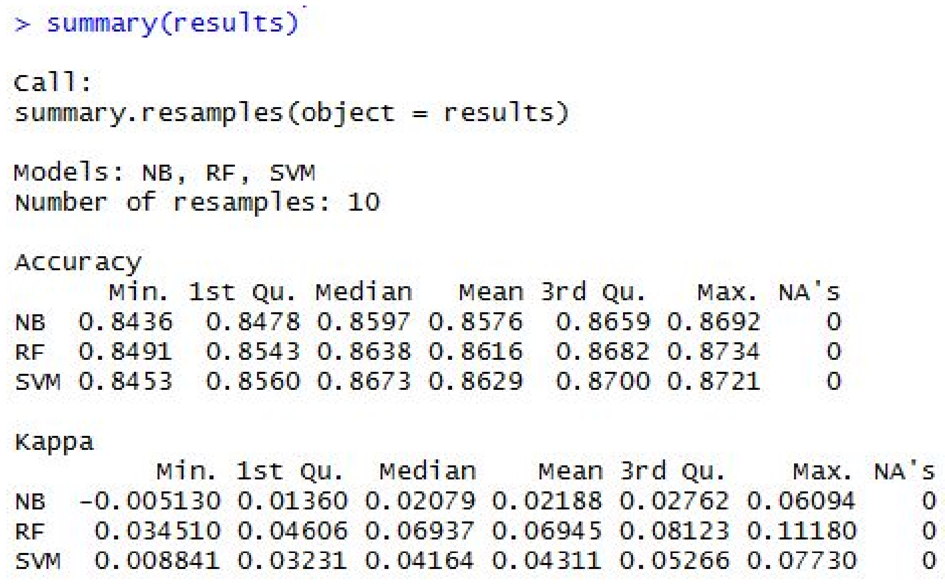


```
> summary(results)

Call:
summary.resamples(object = results)

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

Accuracy
      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NB  0.8469  0.8525 0.8565 0.8567  0.8611 0.8660    0
RF  0.8421  0.8559 0.8606 0.8591  0.8639 0.8708    0
SVM 0.8517  0.8606 0.8612 0.8607  0.8652 0.8660    0

Kappa
         Min.  1st Qu.  Median    Mean 3rd Qu.   Max. NA's
NB  -0.018270 0.000000 0.03543 0.03352 0.05577 0.1477    0
RF  -0.026800 0.004205 0.05055 0.06525 0.10900 0.1956    0
SVM -0.009347 0.000000 0.05577 0.05091 0.05790 0.1477    0
```

Summary of our results which shows that SVM has performed almost equally well with a median value of 0.86

CV Technique 3

The cross-validation technique used in this case is repeated bootstrap sampling using 10 folds.



```
> summary(results)

Call:
summary.resamples(object = results)

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

Accuracy
      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
NB  0.8436  0.8478 0.8597 0.8576  0.8659 0.8692    0
RF  0.8491  0.8543 0.8638 0.8616  0.8682 0.8734    0
SVM 0.8453  0.8560 0.8673 0.8629  0.8700 0.8721    0

Kappa
         Min. 1st Qu.  Median    Mean 3rd Qu.    Max. NA's
NB  -0.005130 0.01360 0.02079 0.02188 0.02762 0.06094    0
RF   0.034510 0.04606 0.06937 0.06945 0.08123 0.11180    0
SVM  0.008841 0.03231 0.04164 0.04311 0.05266 0.07730    0
```

Summary of our results which shows that SVM and RF have performed almost equally well with a median value of 0.86

Question 5: Feedback

Team 2: Automatic Rumor Prediction on Twitter

This team did a good job. We found that their questions were explained well and the analyses done on it. In between the audio was a little distracting so we couldn't hear it properly. Apart from that we liked that they went over the ROCR curves. Overall, the motivation behind the project is well appreciated.

Team 3: Detecting Cyberbullying on Twitter

The presentation is well structured and includes all the techniques applied in all three milestones. There were certain terms like Silhouette Score analysis in the clustering technique for which the term Silhouette coefficient term wasn't explained and why this technique was used. All the visualizations for different clustering techniques were included which looked very interesting. Besides, I liked the way, the comparison between techniques were visualized in the last few slides. The presentation was concluded very well with all limitations and techniques that were not performed.

Team 4: Consumer Complaints

A very concise and well made presentation. The presentation summarized all their analytical efforts well and was easy to understand. However, they seemed to have paced through it which made it difficult for us to keep up with the video. We liked the part where they summarized milestone 2 results which made it easier for us to comprehend. Overall, it was a well made presentation with good analyses.

Team 5: Super Bowl Babies

I liked that only techniques part of milestone 3 were covered in this presentation as there was sufficient time to explain their analysis. The analysis and results of various techniques that were applied is explained very well. Although, it would have been great if the research question was written for the first few slides that were explained. I would like to know that can exhaustive and non-exhaustive CV technique approaches be applied to different research questions as they deal with the same data set. It's good that data cleaning was explained. However, the structure for the presentation could have been organized in a better manner. Overall, this data set looked very interesting. A lot of effort was put in creating neural network models.

Team 6: Predicting the Performance of Crowdfunding Campaigns
We liked the topic chosen by this group. There is tremendous scope in predicting what crowdfunding campaigns may result in. For this particular milestone, we found that they had

scraped all the data from the Kickstarter website which was commendable. The results for the clustering part was not very clear but that could have been because of the nature of their dataset. Nice effort and good dataset.

Team 7: Pokémon Go - Predict'em All

This was probably the most interesting datasets chosen as compared to all the teams. The video was made well by this team and they explained the process of data cleaning and preparation very well. Since, the video did not have a brief summary of the previously done analyses in Milestone 2 it was a little difficult to follow up. However, overall the visualizations were good and easy to understand.

Team 8: Airbnb Host Beneficiary System

This was one of the most detailed presentations. The team really came together and formed a very comprehensive and well explained presentation. The flow was uninterrupted as they covered the previous milestones very well to prevent any ambiguity. The information provided was clear and concise. Good use of visualizations. The part that covered the neural networks especially stood out, due to the various variations considered. The visualizations for the clustering were not that clear and hence some extra description could have been added. Overall, great work done by Team 8!

Team 9: College Scorecard Data

The team did a good job in breaking down the videos as it was easier to go through them. They could have made a more informative recap of what had been done before. Some of the slides seemed a bit hurried. But the information presented was clear and easily understandable. Plenty of visualizations were presented. Some of the neural networks lacked variation in terms of the activation function used. In all, apart from a few parts where the explanation seemed rushed or seemed could have been more, the team did a good job in answering the questions and and drawing suitable conclusions.