CS 522

ADAVANCED DATA MINING

HOME WORK – 2

“Classification and Analysis using Random Forests, Feature Selection and Principal Component Analysis”

SUBMITTED BY :

SACHIN KRISHNA MURTHY

CWID : A20354077

1. Preparation

We will use, churn, Titanic and reuters data for this assignment. The data was provided in .csv format. The first dataset ‘churn’ has the phone data of various customers which includes the number of calls they made, call minutes for days, evenings and nights separately, services that were enabled on their connection and a churn attribute which corresponds to ‘churn ratio’. Churn ratio is the number of customers that choose to leave the services over a specific period of time. Thus the churn attribute tells us whether or not that customer stopped using the company services as a factor ‘Yes’ or ‘No’.

The second data set Titanic records the information of all the passengers on the ship i.e. their age, sex, class (first, second or third) and whether or not they survived the shipwreck or not. We will use this data to predict whether or not a given passenger survives the disaster or not using random forests.

In the preparation phase we installed the required packages in R and loaded them using the ‘install.packages’ command and the library function respectively, as follows:

```
install.packages("randomForest")

install.packages("car")

install.packages("rpart")

install.packages("plyr")

install.packages("lattice")

# Loading the libraries

Library(randomForest)

Library(car)

Library(rpart)

Library(plyr)

Library(lattice)
```

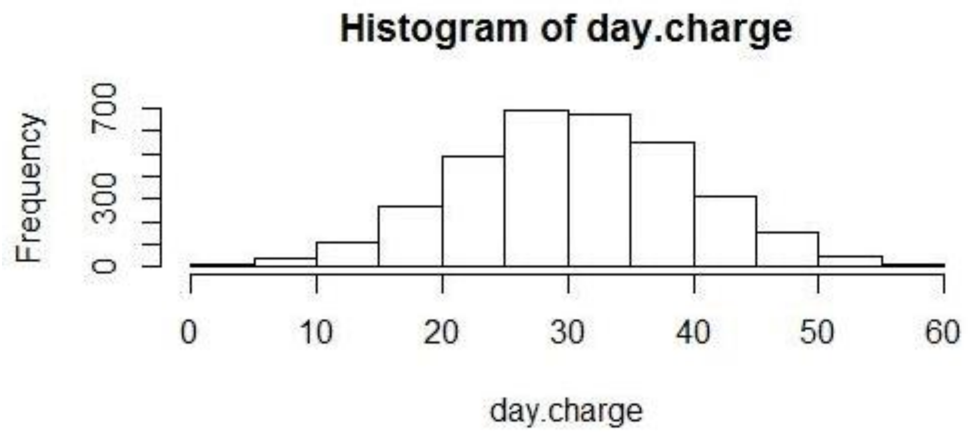
2. Data Analysis

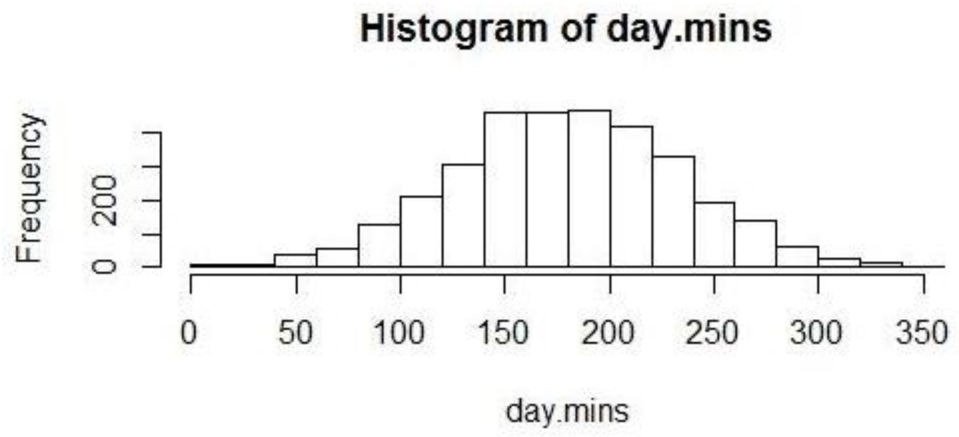
After reading all the files in R as data frames we recode the churn attribute. In using R it is very important to code this variable or at least understand how R treats this variable. We recode this attribute to a 0/1 variable where 1 indicates event of interest. In R terms these are referred to as positive events in some packages with 0 as negative events. We perform this recoding using the ‘recode’ function in the ‘car’ package in R as follows:

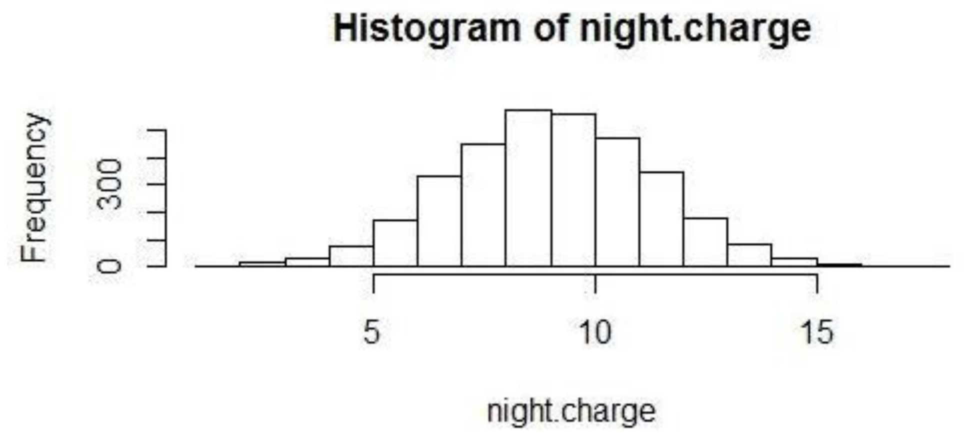
```
data$churn <- recode(data$churn, "'Yes' = 1;else = 0")
```

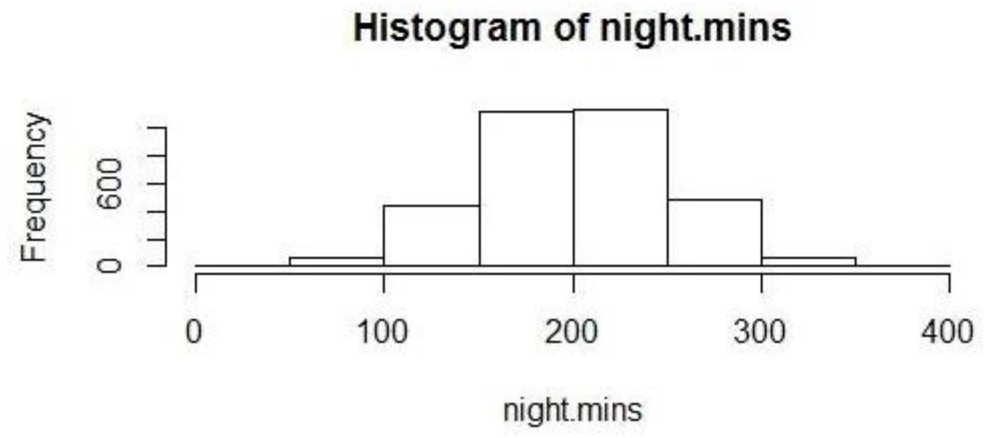
During this phase of the assignment we used several functions to perform initial analysis of our data. We represented attributes as histograms and bar-plots and tabled the class variables. Results and analysis of these applied functions are below.

Histograms offer a quick view of the attributes and their distribution. We made histograms for various attributes of the Churn data.









Looking at all these histograms we can see that these attributes are normally distributed. This means that values for these attributes are as likely to occur on one side of the average as the other.
Tabling the data gives the class distribution of the dataset i.e. it tells us how many data points belong to each class.

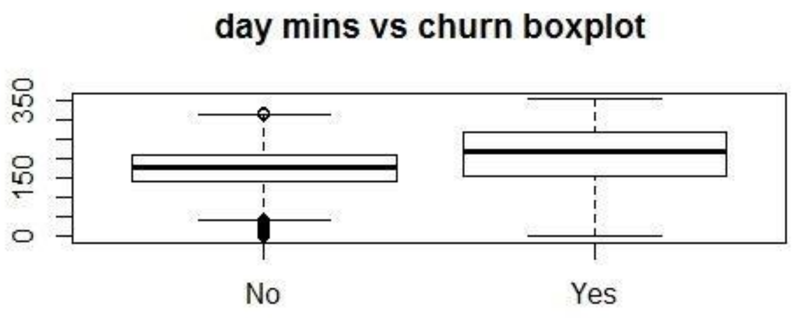
churn	
No	Yes
2850	483

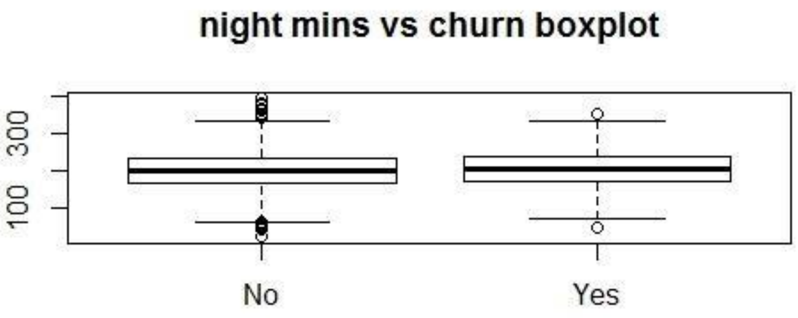
Survived	
No	Yes
1490	711

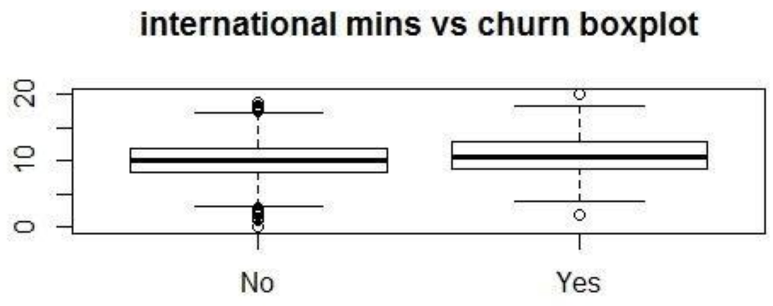
Looking at the values of this table we can say that there are total 3333 customers in the churn data, 483 churn while 2850 do not churn. Similarly in the Titanic data, out of 3201 total passengers, only 711 survived the disaster, while the rest 1490 did not.

Boxplots display difference between populations with regards to another variable and without making any assumptions about the underlying statistical distribution. The bottom and top of the box are always the first and third quartiles, the end of the whiskers represent lowest datum still within 1.5 interquartile range (IQR) of the lower quartile and the highest datum still within 1.5 IQR of the upper quartile. The band inside the box represents the median of the values of that attribute.

Therefore to examine the relationship between each variable and the churn variable looking for differences between the group who churned and those who did not churn we plotted the following barplots for some of the attributes against the class variable.







After experimenting with various attributes and the class variable we found that most of the attributes are equally distributed around the mean as seen in the histograms. For the day charge vs the class variable we see that the data distribution of the phone usage of people who churn is larger and more scattered than those who don't churn.

We also generate summary statistics for each group. To perform this in R we used the tapply function in which the first argument is the variable of interest, the second the grouping variable and the third the function to be applied to each group we also added the each function within tapply which allows us to concatenate a number of functions.
Below are the table results for summary statistics of all the variables in the churn dataset that show the mean and standard deviation of all the attributes

a)Day

	Yes		No	
Day.mins	mean	sd	mean	sd
	206.91408	68.99779	175.17575	50.18166
Day.calls	mean	sd	mean	sd
	101.33540	21.58231	100.28316	19.80116
Day.charge	mean	sd	mean	sd
	35.17592	11.72971	29.780421	8.530835

b)Evening

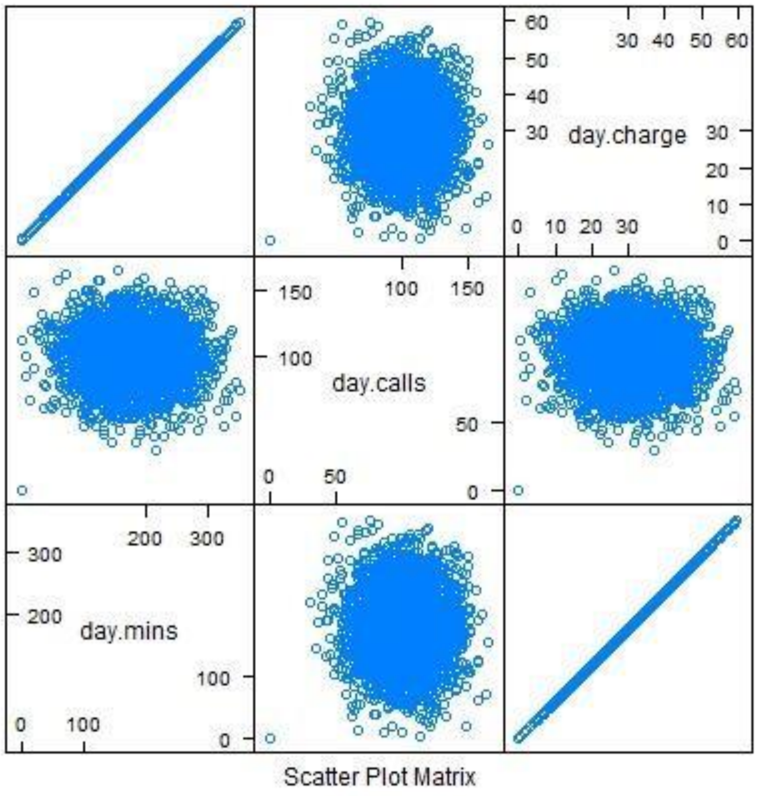
	Yes		No	
Eve.mins	mean	sd	mean	sd
	212.41014	51.72891	199.04330	50.29217
Eve.calls	mean	sd	mean	sd
	100.56108	19.72471	100.03860	19.95841
Eve.charge	mean	sd	mean	sd
	18.054969	4.396762	16.918909	4.274863

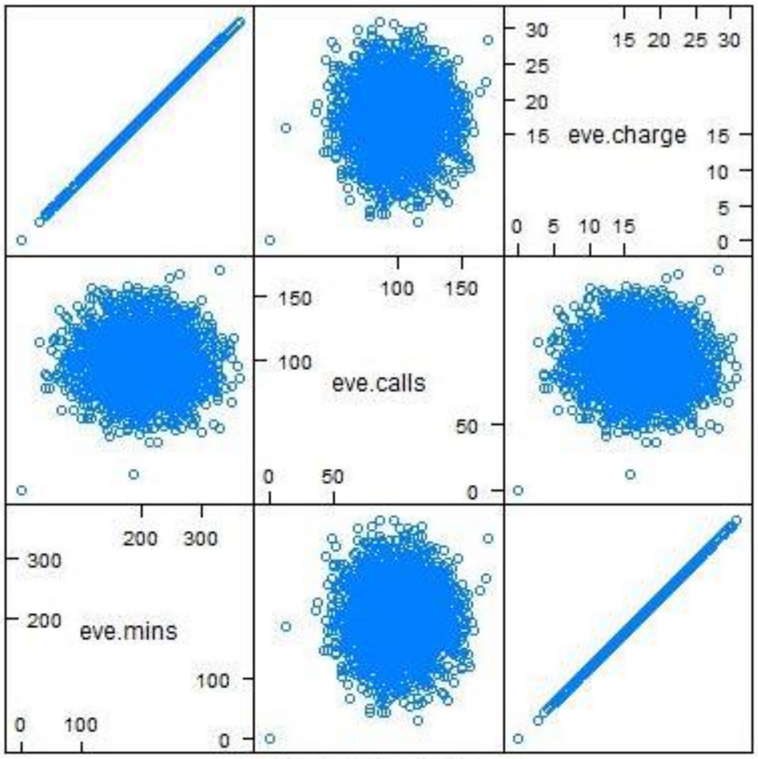
c)Night

	Yes		No	
Night.mins	mean	sd	mean	sd
	205.23168	47.13282	200.13319	51.10503
Night.calls	mean	sd	mean	sd
	100.39959	19.9506	100.05825	19.50625
Night.charge	mean	sd	mean	sd
	9.235528	2.121081	9.006074	2.299768

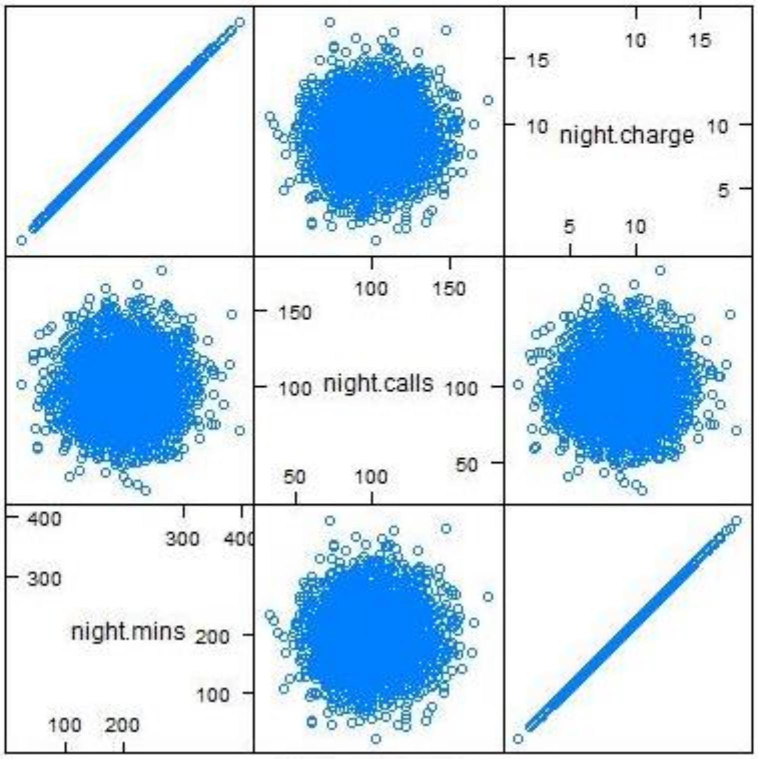
Other than using boxplots to examine the relationships between quantitative variables, we can also use scatter plots. Scatter plots display the data as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis. In R we can make scatter plots of more than two variables at the same time using the splom function available in the library 'lattice'.

A scatter plot can suggest various kinds of correlations between variables. Correlations may be positive (rising), negative (falling), or null (uncorrelated). If the pattern of dots slopes from lower left to upper right, it suggests a positive correlation between the variables being studied. If the pattern of dots slopes from upper left to lower right, it suggests a negative correlation. Below are scatter plots of a few combination of variables.

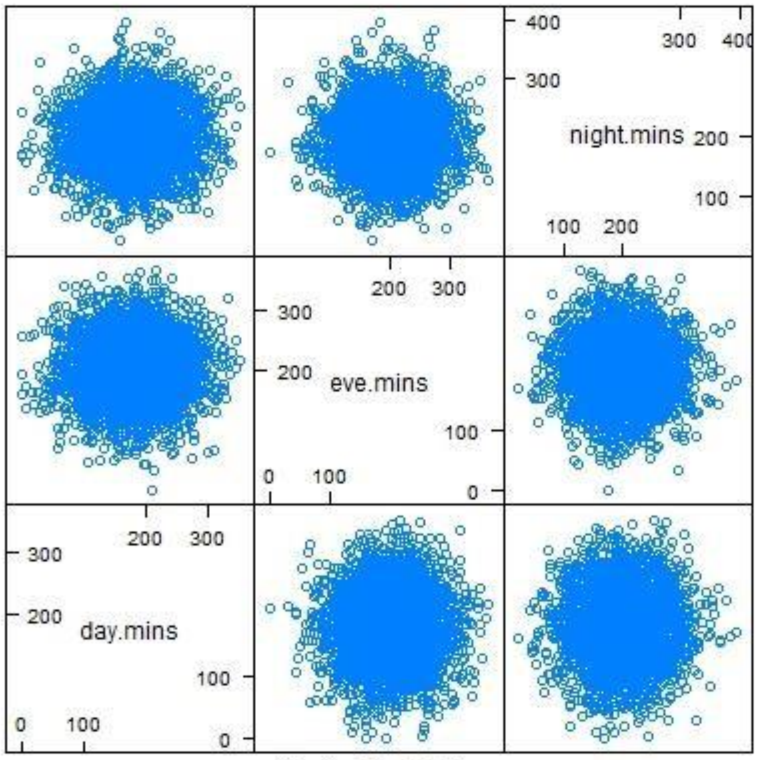




Scatter Plot Matrix



Scatter Plot Matrix



Scatter Plot Matrix

The first three scatterplots between day charges, calls and minutes, evening charges, calls and minutes and night charges, calls and minutes respectively tell us that there is a strong positive correlation between the minutes and charge attributes for day, evening and night. This makes sense because the minutes used on the phone will determine the charges made to the account.

On the other hand, we did not see any such or similar relation between other variables as suggested by the last scatter plot.

To see if there is a significant difference between the averages of day minutes in the population in the two groups (churn vs no churn) we perform a Welch Two Sample t-test where the null hypothesis is that there is no difference between the means of the two groups with respect to their daily usage of their phone.

```
Welch Two Sample t-test

data:  day.mins by churn
t = -9.6846, df = 571.513, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -38.17516 -25.30148
sample estimates:
 mean in group No mean in group Yes
         175.1758          206.9141
```

In this example, the resulting ‘t’ value is -9.6846, the number of degrees of freedom is 571.513 and the p-value is very low, less than 2.2e-16. This low value of p allows us to strongly reject the null-hypothesis because we can see there is an increase in the mean of day minutes between the two class groups.

The 95 percent confidence interval is (-38.17516, -25.30148). This interval means that if the experiment were repeated a large number of times, each time generating about 95% probability confidence interval from the sample, thus the 95% of the generated intervals will contain the true percentage of people which speak during t minutes during the day.

This confidence interval shows that the difference of the means of the amount of time spent on phone between people who churn vs people who don’t churn is between 25 and 38. Therefore we can conclude that there in fact, is a significant difference between the averages of day minutes in the population in the two groups.

(vii) Now we take a look at the categorical attribute of Vmail in the data which describes if a customer had a voice mail service associated with the account. The motive it to find if the churn variable is dependent on the vmail attribute or not. We plot a barplot for churn vs vmail which gives us separate bars for churn and not churn for each ‘yes’ and ‘no’ value for vmail.
Below is the plot that we obtained.

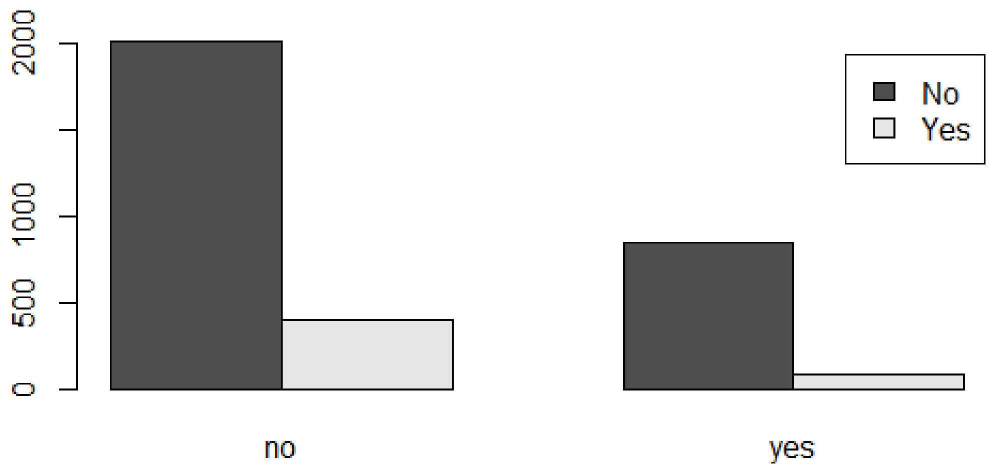


Fig. Barplot for churn and not churn for each ‘yes’ and ‘no’ value for vmail

This barplot only tells us about how many users who did not have vmail service churn and do not churn and for the users who had a vmail service. Thus to find out if the two attributes are dependent or not we calculate a chisquare test type.

```
Pearson's Chi-squared test with Yates' continuity
        correction

data:  table(churn, vmail)
X-squared = 34.1317, df = 1, p-value = 5.151e-09
```

From the above results we can see that the p-value is less than the significance value and thus we can say that there is correlation between vmail and churn attributes.

3. Basic Tree Classifier

In this section we will look at the Titanic data set, which provides information on the fate of passengers on the maiden voyage of the ocean liner 'Titanic'. There are 4 variables Class Sex, Age and Survived. We are interested in predicting who survived.

3.1 Explore data

We start by exploring the data e.g. frequencies of each variable separately and some tables. We use the table function to put together a full set of summary tables that uses different variable combinations. Below are the table results.

```
> table(data)
, , Age = Adult, Survived = No

    Sex
Class  Female Male
 1st      4  118
 2nd     13  154
 3rd     89  387
 Crew     3  670

, , Age = Child, Survived = No

    Sex
Class  Female Male
 1st      0    0
 2nd      0    0
 3rd     17   35
 Crew     0    0

, , Age = Adult, Survived = Yes

    Sex
Class  Female Male
 1st    140   57
 2nd      80   14
 3rd     76   75
 Crew    20  192

, , Age = Child, Survived = Yes

     Sex
Class  Female Male
 1st      1    5
 2nd      13   11
 3rd     14   13
 Crew     0    0
```

We see four tables, each headed by a combination of age (adult/child) and survived (yes/no). The tables are further subdivided into counts by class (1st/2nd/3rd/crew) and sex (male/female). This gives us the basic idea of the values and number of values for each class in our dataset.

Q: Which variables are related to Survived?

3.3 Graphical Output

A graphical output is useful for conveying information quickly and naturally it is often easier to quickly pick out anything unusual than from the numbers.
After exploring our data we found that all the variables in the Titanic dataset are categorical thus we cannot produce box plots for this dataset. Therefore, just as we plotted bar graphs for the categorical attributes in our churn dataset we plot bar graphs for our Titanic dataset. We plot three such bar graphs. The first one describes the number of survived and not survived passengers in each class. The second one describes the number of survived and not survived passengers with respect to their age. The third one describes the number of survived and not survived passengers with respect to Sex.

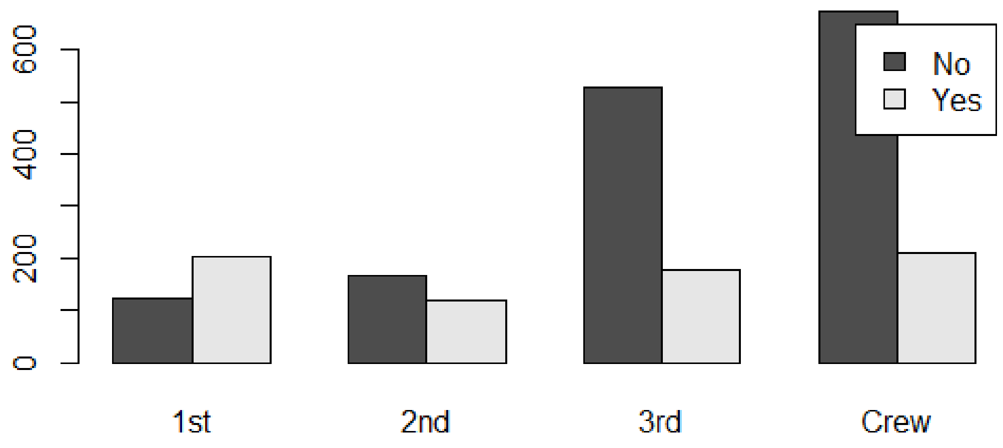


Fig. Barplot survived vs not survived for each class

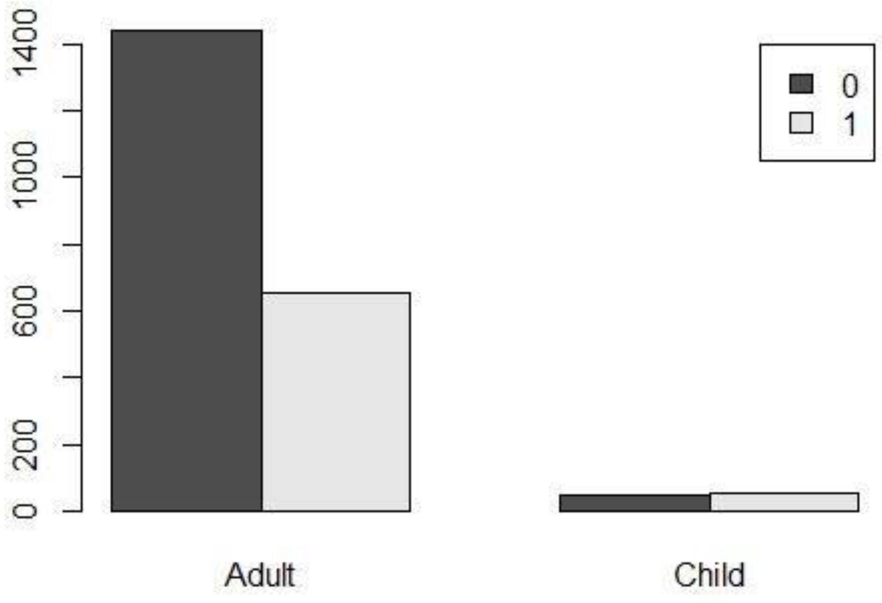


Fig. Barplot survived vs not survived with respect to Age

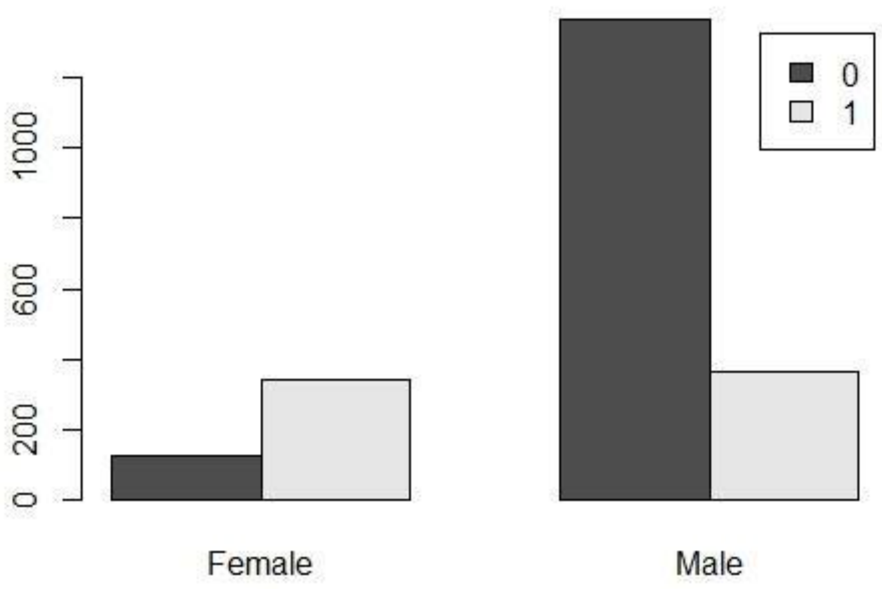
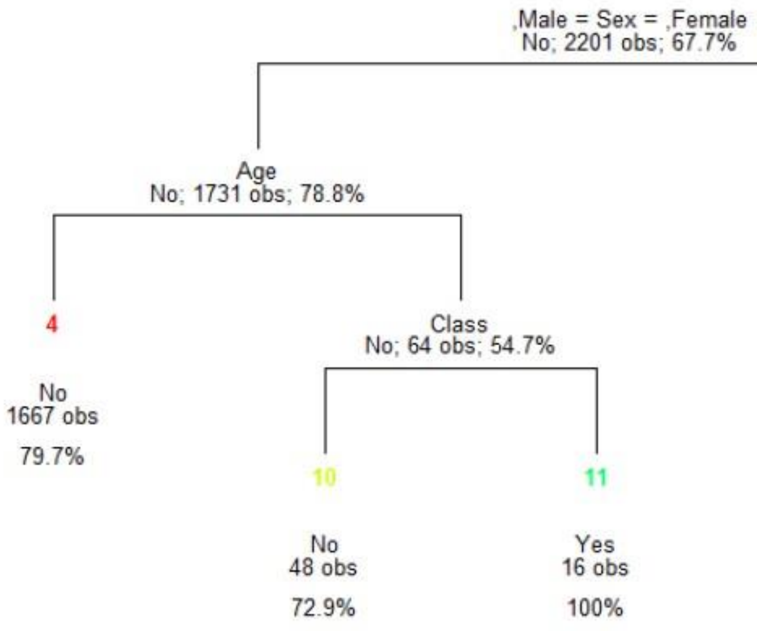


Fig. Barplot survived vs not survived with respect to Age

From the above graphs we can make the observations that survival rate of 1st class members and females is higher than in all the other categories. The number of males and females that survived are nearly equal. The number of children that survived vs that did not survive is also almost equal. The lowest survival was in the 2nd class but, this could just be because of overall low number of 2nd class passengers.

4 Building Classification Trees

In this section we will see how well we are able to fit a classification tree in order to predict survival based on passenger class, age and sex. We use the rpart function provided by the rpart package to fit a tree where Survived is to be predicted from Class, Age and Sex and then store that tree for later reference in the variable fit. Below is the resulting classification tree we obtained using the drawTreeNodes function in the rattle package.



Now at this point we have trained a classification tree and would like to make predictions based on it, using a subset of the data. We determine the predictions for 1st class/child/female, 2nd class/adult/male and crew/adult/male all at once by setting up a data frame with multiple observations and passing it to predict() along with our fitted tree model. We get the following results.

Class	Age	Sex	Survived probability
2nd	Child	Male	1
1st	Child	Female	.93
2nd	Adult	Male	.20
Crew	Adult	Male	.20

This table surely gives the survival probability but does not say much about the classification accuracy. To calculate accuracy of our classification tree we subset our data to pick out only those observations who did not survive and store them in newdata variable. Then, we use the predict() function to predict only those "No"'s – so we should get all "No" predictions if it is completely accurate.

Then out of the prediction results we pick out only the probabilities of answer "No" and test if the prediction is over 0.5. The motivation to do this is that no is more strongly predicted than yes. This produces a vector of true/false values where, true indicates that the prediction for that observation number was correct since the probability of "No" was over 0.5 and false indicates it was incorrectly predicted. We repeat the above prediction method to predict only on the records with survived as 'yes'. After performing these experiments we get the following results.

True Label	Predicted Not-survived	Predicted Survived
Not-survived	1470	20
Survived	441	270

We see that in subset of all not-survived 1470 were correctly classified and 20 were incorrect, giving accuracy as 98%. However, the accuracy for subset of survived passengers only 270 were classified correctly and 441 were classified incorrectly giving a very low accuracy of 37.97%. The overall accuracy being 79.05%.

4.7 Comparison with logistic regression

We also performed logistic regression using the lm function for both the survived and not-survived subsets of the data and got the following result of number of correct and incorrect predictions.

True Label	Predicted Not-survived	Predicted Survived
Not Survived	1364	362
Survived	362	349

We see that logistic regression did not perform a whole lot better than the classification tree. The overall accuracy 77.82% is even lower than before. The accuracy for not survived decreased to 91.54%. However accuracy for survived increased slightly to a 49.08%.

5 Random Forests

In this section we will be studying, using and analyzing random forests on the Churn dataset. We start with fitting a model and then find variables that are relatively more important that the others for classification. After learning how to use the randomForest function to fit in R we will later experiment with the number of trees and the subset of the data that is used to make decisions at a split. Below is the resulting random forest obtained in the first part.

```
Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 3

        OOB estimate of  error rate: 7.26%
Confusion matrix:
      No Yes class.error
No  2830  20 0.007017544
Yes  222 261 0.459627329
```

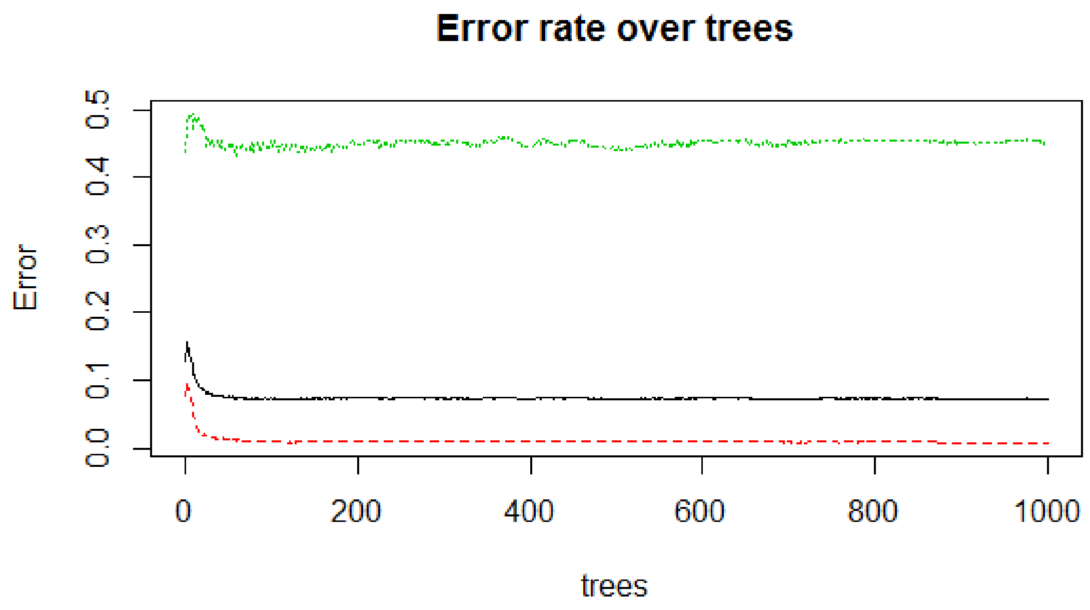
This results gives us important information about the random forest that we built. The type is classification in this case. The other type of random forest that can be made in R using this function is of type regression. The number of variables tried at every split is the number of attributes that are used at each split to make the split decision while working on the local optimum for high information gain using the gini index.

In the end we get a confusion matrix that tell us the number of cases that were correctly and incorrectly classified. Using this information we found that the accuracy of this random forest comes out to be 92.27%. We calculate this accuracy to evaluate the performance of the classifier. However we can also use the error rate (7.26%) to do the same job. The error rate is the percentage error made by the classifier.

We also find out the various attributes associated with the random forest object using the names() function and get the following result.

```
"call"             "type"             "predicted"
 [4] "err.rate"        "confusion"        "votes"
 [7] "oob.times"       "classes"          "importance"
[10] "importanceSD"    "localImportance" "proximity"
[13] "ntree"           "mtry"             "forest"
[16] "y"               "test"             "inbag"
[19] "terms"
```

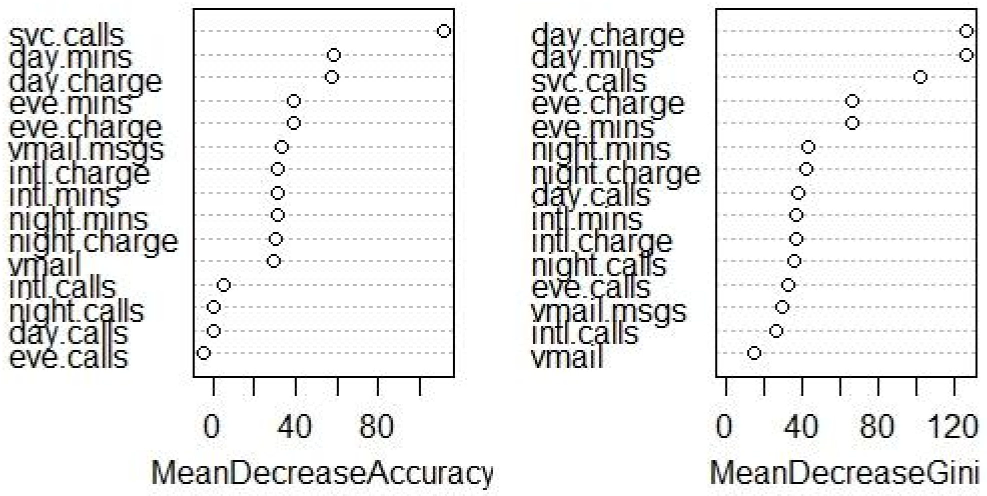
We will talk about all these attributes in our further analysis.
The error rate over trees can be shown in the following graph.



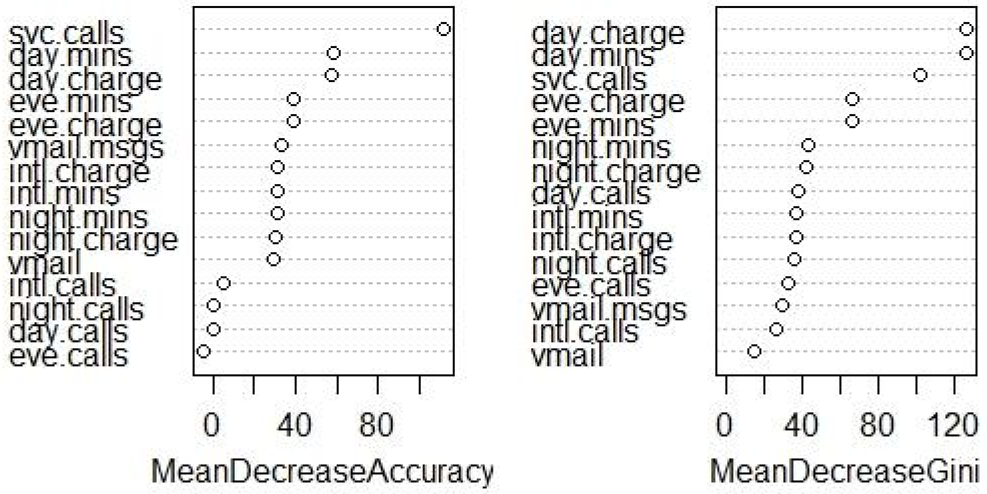
5.2 Variable Importance

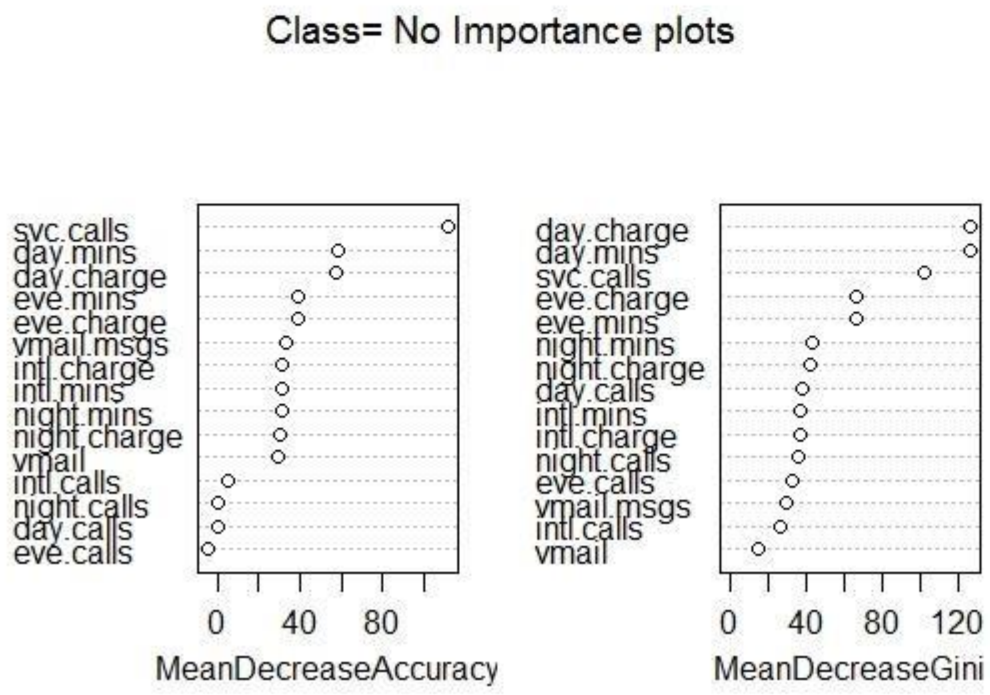
Variable importance is a measure of relevance of an attribute to make better decisions. It tells us which variables are better than the others when used to make a split decision. We can use this information for feature selection and select only the most important features. I R we can plot the variables with respect to their importance using the varImpPlot.We make these plots for average importance of all variables in the data and separately for records that belong to the two different classes. Below are the results.

fit



Class= Yes Importance plots

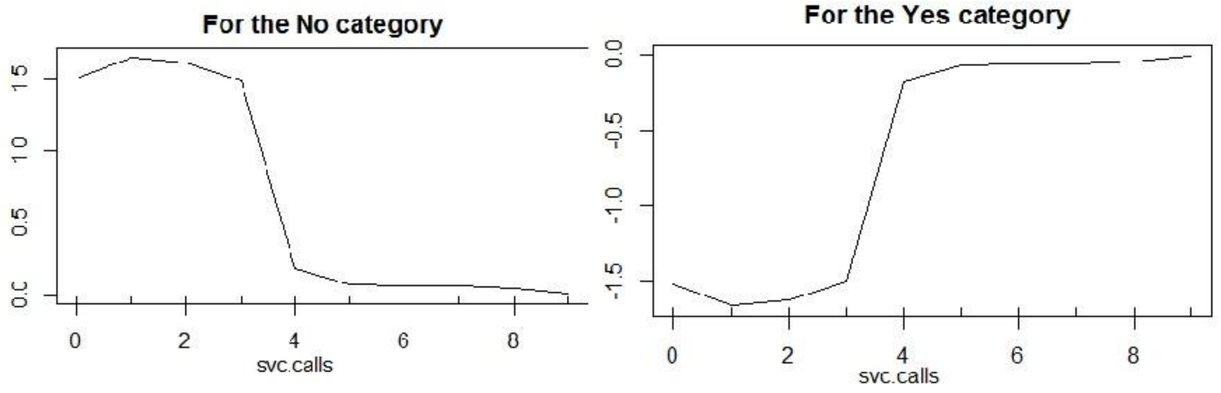


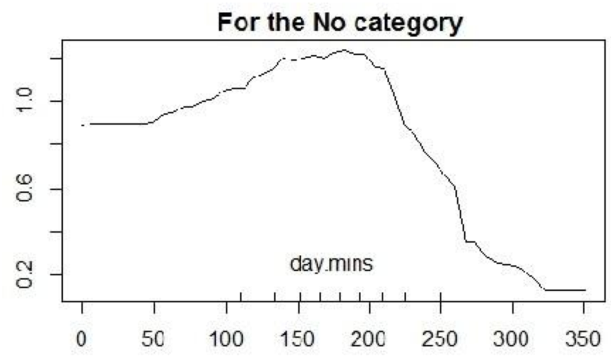


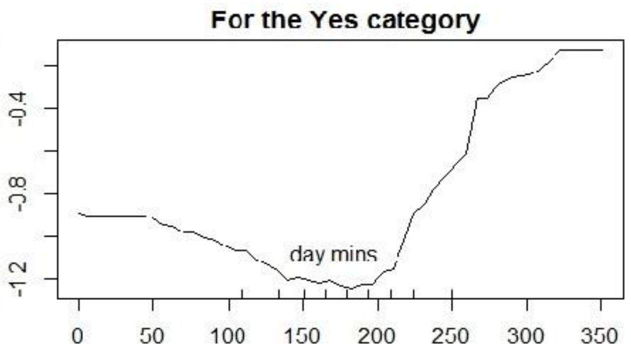
We see that all these three graphs are identical to each other. This means that the variables that are important for decision making are important throughout the different class labels. We can also see that some of the variables ranked very high for e.g. svd.calls, day.mins and day.charge are among the top most important variables.

5.3 Partial Dependence

Other than just finding which variables are important, we can also find out the effect that a variable has on the class prediction. In order to do this we look at the partial dependence plots generated by the partialPlot function of R for different variables and classes. Below the plots we generated for few of the important variables.



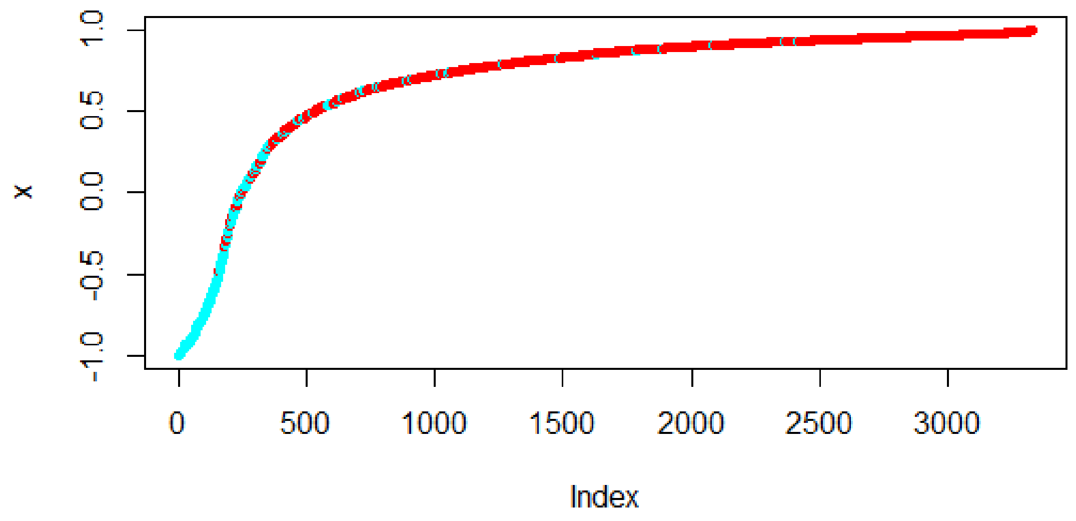




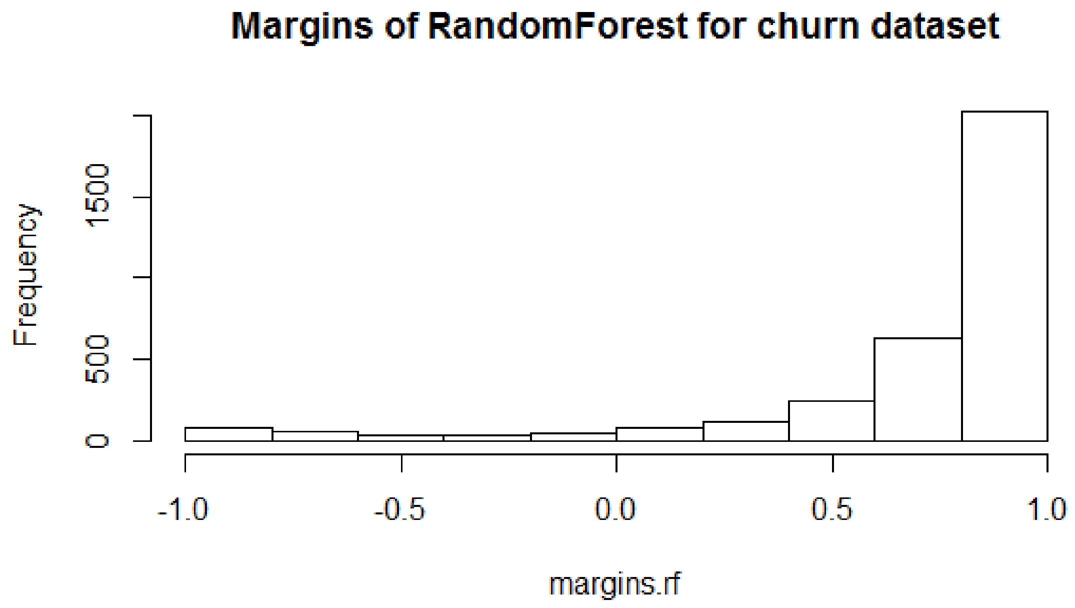
Form the above graphs we can see the behavior of all the values of a variable towards that particular class.

5.4 Other plots

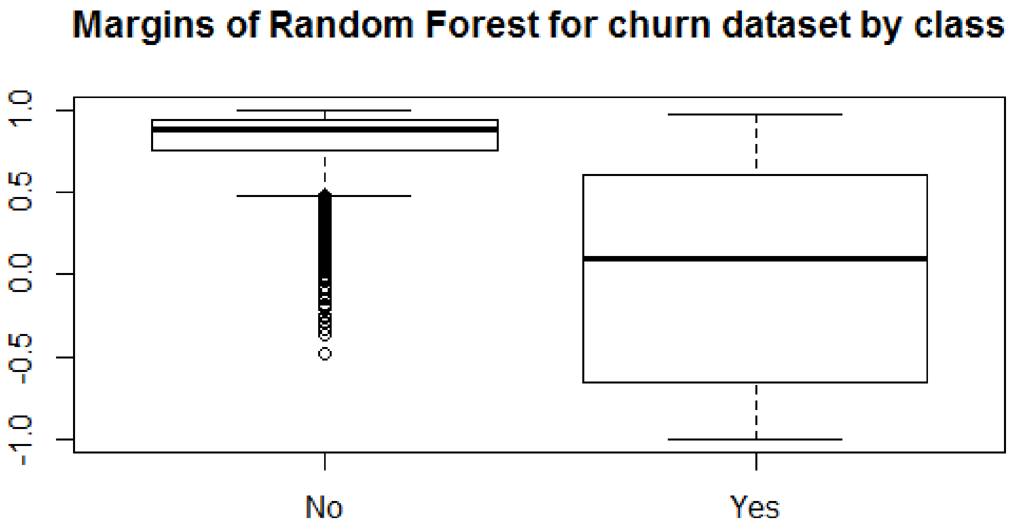
The margin of a data point is defined as the proportion of votes for the correct class minus maximum proportion of votes for the other classes. Thus under majority votes, positive margin means correct classification, and vice versa. The results can be seen below:



The same information can be showed in histograms of the margins



And also a boxplot for the margins of the random forest.



6. Influence of Parameterization

In this section we experiment with different number of trees and different subset ratio when building the random forest to see how these parameters influence the decision tree learning and accuracy.

There are many parameters that are used for the training of the random forests. Below is the list of all those parametrs and their documentation.

data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which randomForest is called from.
subset	an index vector indicating which rows should be used. (NOTE: If given, this argument must be named.)
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)
x, formula	a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the print method, an randomForest object).
y	A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, randomForest will run in unsupervised mode.
xtest	a data frame or matrix (like x) containing predictors for the test set.
ytest	response for the test set.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one. Ignored for regression.
cutoff	(Classification only) A vector of length equal to number of classes. The ‘winning’ class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
strata	A (factor) variable that is used for stratified sampling.
sampsize	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
nodesize	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
maxnodes	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by nodesize). If set larger than maximum possible, a warning is issued.
importance	Should importance of predictors be assessed?

localImp	Should casewise importance measure be computed? (Setting this to TRUE will override importance.)
nPerm	Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.
proximity	Should proximity measure among the rows be calculated?
oob.prox	Should proximity be calculated only on “out-of-bag” data?
norm.votes	If TRUE (default), the final result of votes are expressed as fractions. If FALSE, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
do.trace	If set to TRUE, give a more verbose output as randomForest is run. If set to some integer, then running output is printed for every do.trace trees.
keep.forest	If set to FALSE, the forest will not be retained in the output object. If xtest is given, defaults to FALSE.
corr.bias	perform bias correction for regression? Note: Experimental. Use at your own risk.
keep.inbag	Should an n by ntree matrix be returned that keeps track of which samples are “in-bag” in which trees (but not how many times, if sampling with replacement)
...	optional parameters to be passed to the low level functionrandomForest.default.

Even though there are 26 total parameters, only the number of trees, i.e. how many random trees per forest, and the subset ratio, i.e. size of attribute subset, are really parameters of the random forest itself, all the others rather influence how and when nodes are split during decision tree learning.

6.2 Influence of number of trees on classification accuracy and training time.

The default number of trees in the random forest function is 500. We now experiment with different number of trees and monitor the change in accuracy. We chose values from 100 to 1000 with an increase of 100 trees at every step thus calculating accuracy when the number of trees is 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. In R it can be done using the following code.

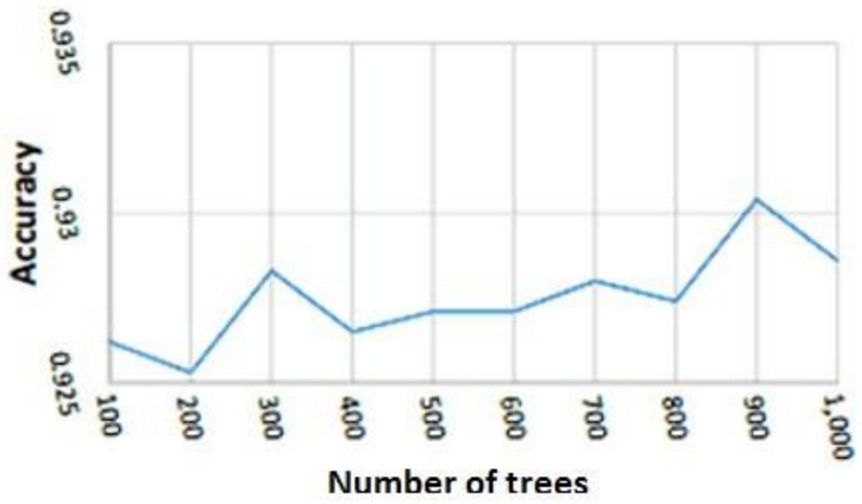
```
For(i in seq(100, 1000, 100){
      fit = randomForest(churn~., data=data[3:17], ntree = i,
      importance=TRUE, Proximity = TRUE)
      print(c(i, fit$err.rate[i,1]))
}
```

This gives us the following table:

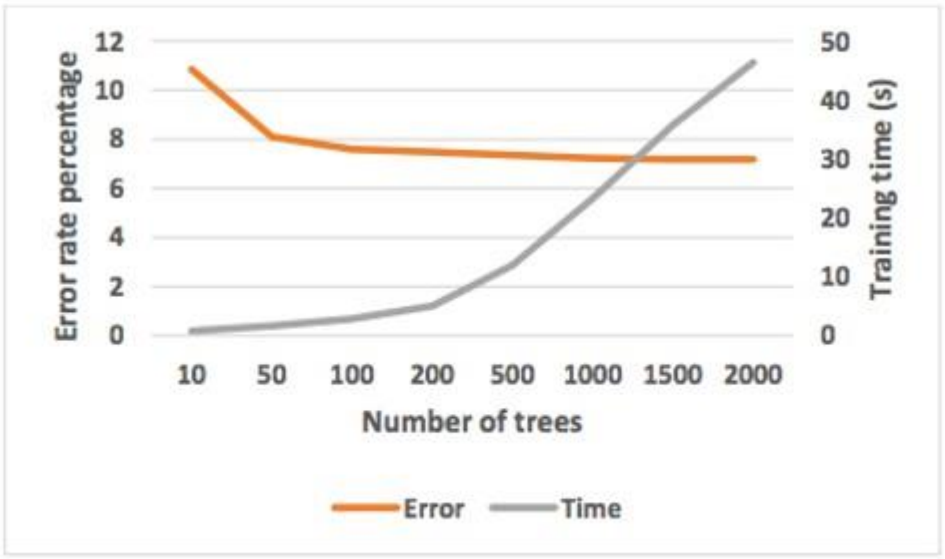
No. of trees	Error Rate(%)	Accuracy(%)
100	7.38	92.62

200	7.47	92.53
300	7.17	92.83
400	7.35	92.65
500	7.29	92.71
600	7.29	92.71
700	7.20	92.80
800	7.26	92.74
900	6.96	93.04
1000	7.14	92.86

When we plot these accuracies against the number of trees we get the following graph.

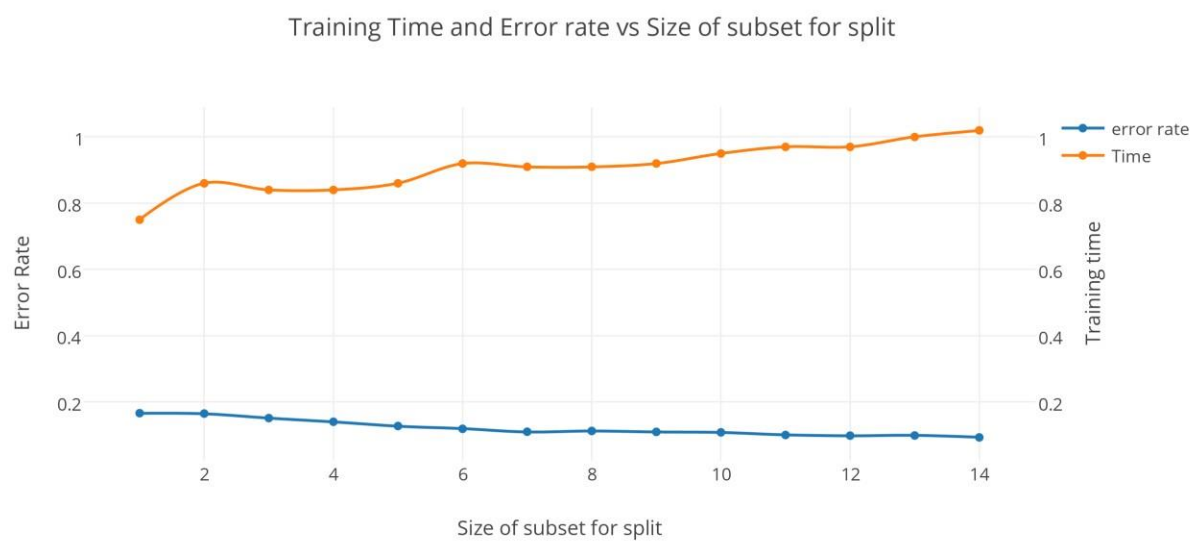


We can say by looking at this graph that for different number of trees we get different accuracies however in the long run we cannot really make a statement about how increasing or decreasing the number of trees influence the error rate or accuracy. To combat this problem we increase the range of number of trees from 100 to 100 to 10 to 2000. At this point we also calculate the time taken by the function to train the random forests using the system.time() function in R. We then plot the training time and error rate vs the number of trees and get the following result.



We can see from the above graph that the error rate decreases when the number of trees increases from 10 to 2000. However the curve becomes flat when the number trees goes beyond 100. Due to this flattening it is reasonable that beyond a certain number of trees, this parameter will not have any significant effect on error rate as the model becomes stable after a certain number of trees and thus won't improve the classification model.
On the other hand the training time increases with increase in the number of trees and this makes sense because it will take longer to train a higher number of trees than a smaller number of trees. From the graph we also see that the training time significantly increases after 200 number of trees.
Therefore there is a tradeoff between the number of trees and accuracy and the training time. This compels us to choose a balanced number of trees so as to not compromise with training time or accuracy. In our example an optimum number of trees can be from anywhere between, 100 and 200.

Now we experiment with number of attributes taken from the dataset at every split decision. In the randomForest function in R this measure is given by 'mtry'. In this section we will keep the number of trees fixed at 200 and change the mtry value. In our dataset Churn we have a total of 17 attributes out of which one is class label therefore for this part of the experiment we will keep mtry from 16 to 5 number of attributes. After performing the experiment we get the following plot of accuracy and time vs the of columns chosen at split decisions.

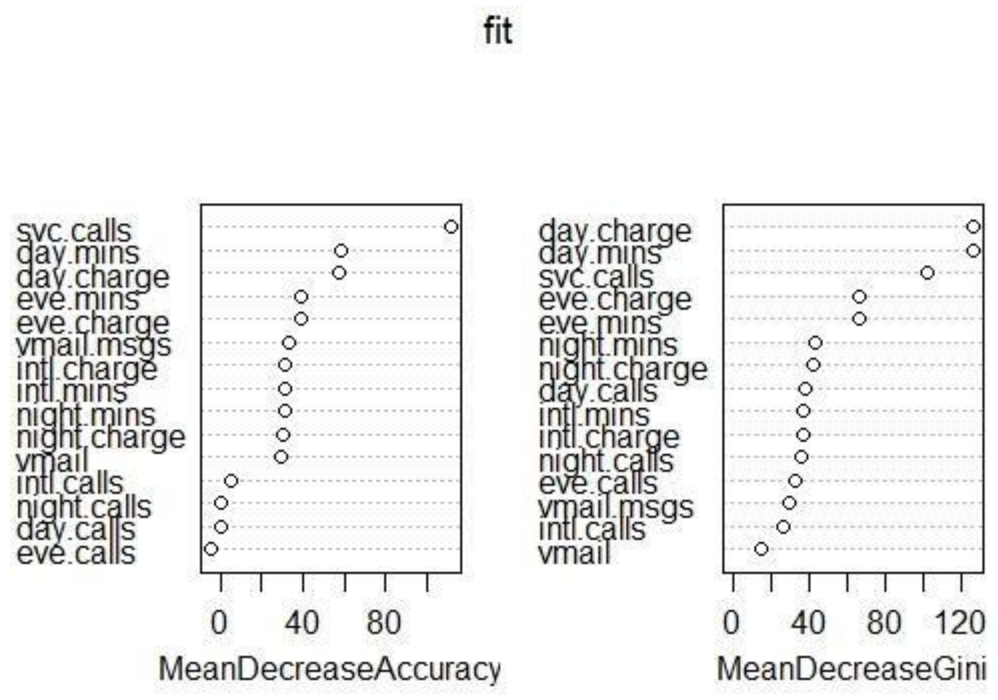


The time taken Increased with the increase in the size of the subset chosen for splitting at every split. However the error rate decreased with increase in the size of the subset. This makes sense because at every split the algorithm will have more attributes to choose from therefore the higher time and lower error rate.

7. Feature Selection

In this section we try to find the important features and check if the performance improves using only the important variables. In order to do this we make 6 different random forests with only 1 tree and record the error rate for the trees. Now from this point we can go into two different directions:
1. Take the top most important variables and train the random forest using only those variables.
2. Take the least important variables and remove them from the data and train using the ones left.

Below is plot of important variables in the data.



Let's start with the first approach. After recording the top 5 most important variables in these tree we get a list of 8 most important variables. We find the important variables using the MeanDecreaseAccuracy(MDA) plot of the trees and the variables. The key is to identify the variables above the absolute value of MDA for variable with the minimum value of this measure. The idea behind this is that the MDA measure of the non-important variables lie around the absolute value of the variable with the least MDA.

We then create a subset of those variables against the class attribute and train the randomforest using only those attributes.

```
d = subset(data, select=c("churn",
"day.charge","svc.calls","day.mins","eve.mins","eve.charge","nig
ht.charge","night.mins",
"intl.charge"))
```

These are the details of the new random forest with only the important variables.

```
Type of random forest: classification
Number of trees: 1000
No. of variables tried at each split: 2
OOB estimate of error rate: 8.79%
Confusion matrix:
No Yes class.error
No 2785 65 0.02280702
Yes 228 255 0.47204969
```

And these are the details of the original random forest

```
Type of random forest: classification
Number of trees: 1000
No. of variables tried at each split: 4
OOB estimate of error rate: 7.17%
Confusion matrix:
No Yes class.error
No 2823 27 0.009473684
Yes 212 271 0.438923395
```

We see that the original random forest is still better at accurately classifying the data but using only the important variables comes very close to accurately classifying considering that we only used 8 attributes out of 17 in total. The important variables are indeed important for the classification of the data. We also see a different number of variables that are tried at each split.

Now, let's try the second approach. We find a list of least important variables for all these trees and remove them from the original random forest of 1000 trees. After experimentation we find the least important variables to be Eve.calls intl.calls day.calls night.calls vmail area.
After removing these variables and selecting only the remaining important variables we get the following result.

```
Type of random forest: classification
Number of trees: 1000
No. of variables tried at each split: 3
OOB estimate of error rate: 6.93%
Confusion matrix:
No Yes class.error
No 2822 28 0.009824561
Yes 203 280 0.420289855
```

We see the error rate has significantly reduced and it is even lesser than that of the original random forest. If we look at the non-important variables we can say that these are the number of types of calls mostly e.g. eve.calls, intl.calls which determine the eve.charge and intl.charge on the basis of price of per call. However the *.charge variables already cover the aspect of number of calls therefore the variables of number of calls is not important. Now after the experiment we can say it was rather an obvious choice of important variables.

8. Compare to Text Data

In this part of the assignment we test the performance of random forests on text data. We take documents from three different topics, Crude, Grain and trade with a total of 1191 documents.

After preprocessing the text by removing stopwrods, numbers, punctuations, white spaces, stemming and turning everything to lower case we get a 100% sparsity document term matrix. We further remove sparse terms to get 96% sparsity and a total of 1465 terms and 1191 documents. The weight used for the document term matrix is TF-IDF.
Using formula, for training random forest does not work well with large number of columns. Therefore we use the document term matrix as x and a label vector 'label' which has the topic information for the documents. The following code was used to fit the random forest.

```
fit=randomForest(x = data.matrix(dtmidf),y=as.factor(label)
,ntree = 1000, importance=TRUE, proximity=TRUE,
type=classification)
fit
```

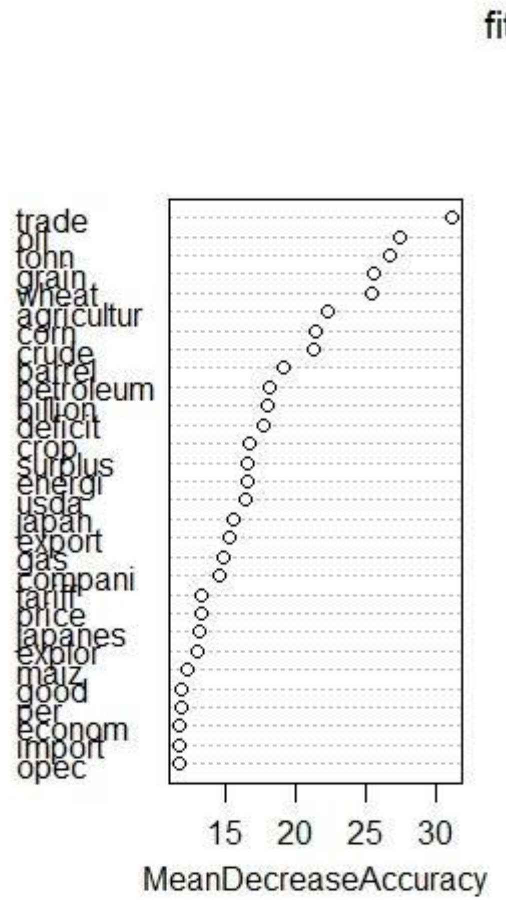
With this we get the following results:

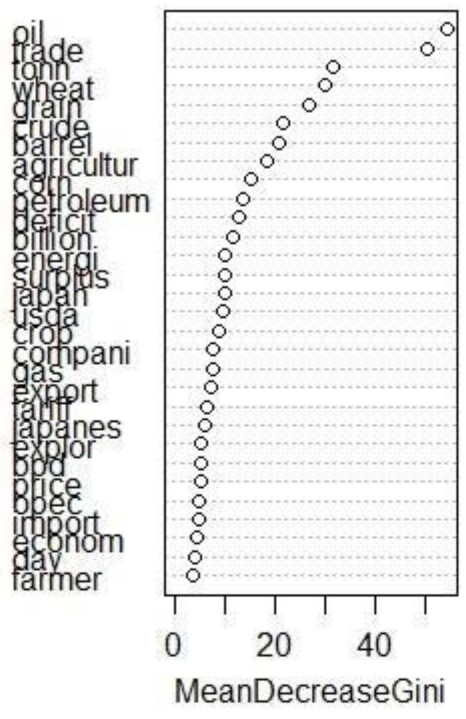
```
Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 38

        OOB estimate of  error rate: 4.11%
Confusion matrix:
    0   1   2 class.error
0  378   2   9  0.02827763
1    1 417  15  0.03695150
2   10  12 347  0.05962060
```

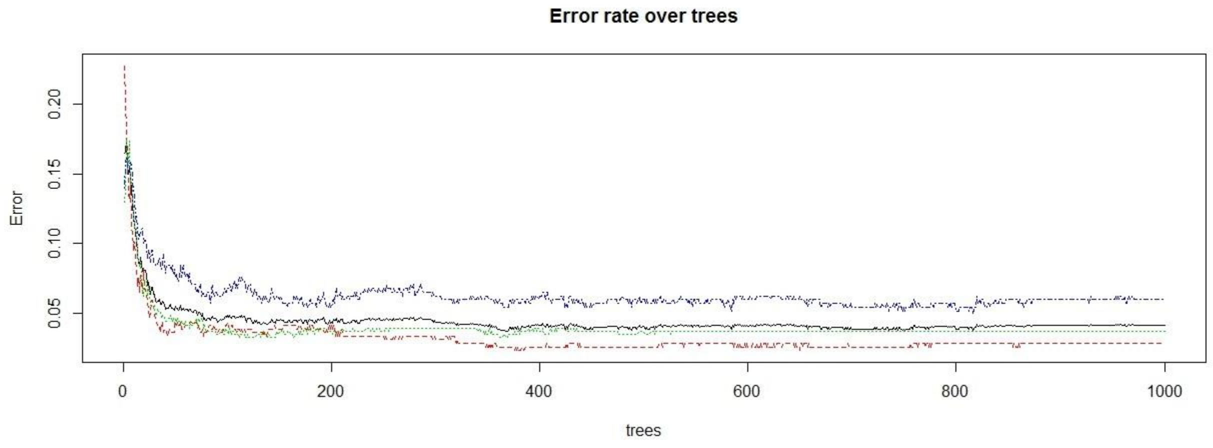
Comparing the churn data with reuters' there is a huge difference in the dimensions of the data. The number of rows for reuters is 1191 (no. of documents) whereas for churn we have 3333 rows. The number of column, hence the number of features for reuters is 1465 (no. of terms) whereas for churn we have only 16 feature columns. However counting the number of elements for churn it is more than 56,000 whereas for reuters we have more that 1,744000 elements thus the reuters dataset is significantly larger than Churn.

The number of attributes tried for each data set churn and reuters is 4 and 38 when the number of trees were 1000 and all the other values were set to default.
Below is a graph plot of important variables for reuters data set.





We can see that as compared to the plot of important variables for churn there is a large number of variables for this plot for reuters. Below is a graph for errors in trees for the reuters data set.



We can see that the curve starts flattening around 20 number of trees which is significantly lower than the 200 for the churn dataset.

Conclusion

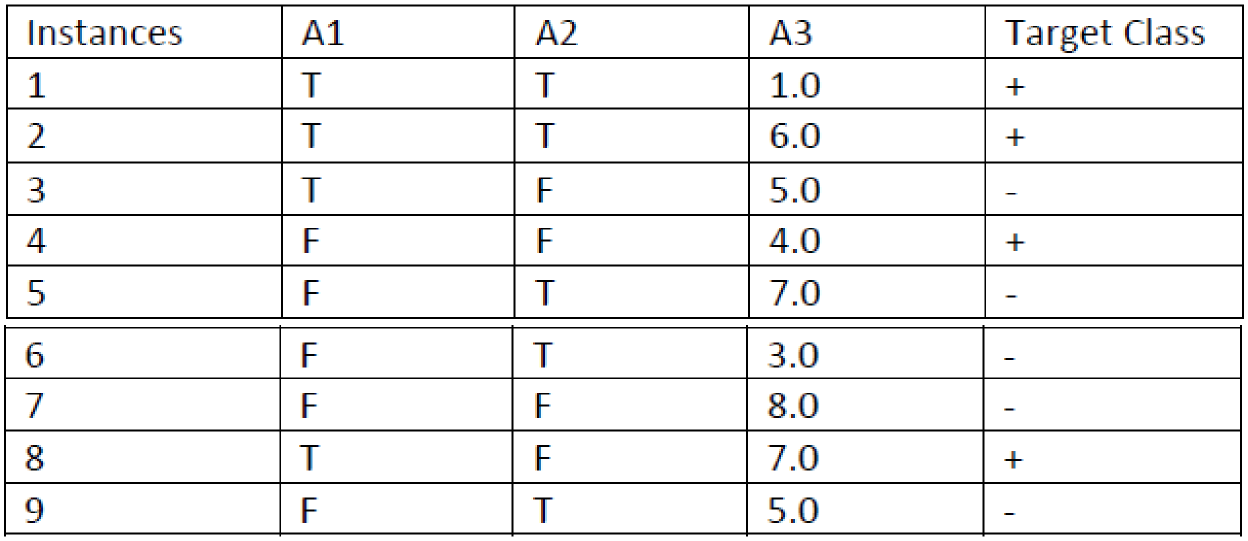
In conclusion regarding the classification accuracy the forest worked very well over the text data as compared to the churn data. An improved error rate from 7.17% to a 6.93% one reason for this to happen was the size of the data. The more data was there, the better the forest worked.

Because it works on the randomly permuted values over the whole subset of trees and being capable of differentiating the most importance attributes and enabling correlations between variables the accuracy indeed has to increase due to the increase in size. The ways in which random forests can be used on a variety of scale and over a lot of different statures were a fruitful conclusion to this experiment.

On the other hand when we compare churn with titanic dataset the one big difference we see is the type of variables in both the data sets. All of the variables in the titanic data have an overlap of values and thus we cannot attain a 100% accuracy in any way. This explains the lower accuracy of 79.05 percent for titanic against 92.74% for churn when the number of trees were set to be 1000.

Part 2

9.1



Instances	A1	A2	A3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	-
4	F	F	4.0	+
5	F	T	7.0	-
6	F	T	3.0	-
7	F	F	8.0	-
8	T	F	7.0	+
9	F	T	5.0	-

Count of + = 4, Count of - = 5 and Total (t)=9

Entropy $= \sum_{i=0}^{c-1} [P(i/t)] \log_2 P(i/t)$

$= - (4/9) \log_2 (4/9) - (5/9) \log_2 (5/9)$

$= 0.5199 + 0.4711 = 0.991$

9.2 What are the information gains of a1 and a2 relative to these training examples?

Ans:

Confusion matrix for a1 is:

a1	+	-
T	3	1
F	1	4

1. $GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$ (Calculating entropy as above)

I. Information gain of a1 →

When a1 = T $\Rightarrow$ C = 3 positives , 1 negatives

When a1 = F $\Rightarrow$ C = 1 positives , 4 negatives

Entropy for a1 = 4/9 * (-3/4 $\log_2$ 3/4 -1/4 $\log_2$ 1/4)

+5/9 * (-1/5 $\log_2$ 1/5 -4/5 $\log_2$ 4/5) = 0.7616

$GAIN_{splita1}$ = 0.991 - 0.7616 = 0.2294

The confusion matrix for a2 is

a2	+	-
T	2	3
F	2	2

II. Information gain of a2 →

When a2 = T $\Rightarrow$ C = 2 positives , 3 negatives

When a2 = F $\Rightarrow$ C = 2 positives , 2 negatives

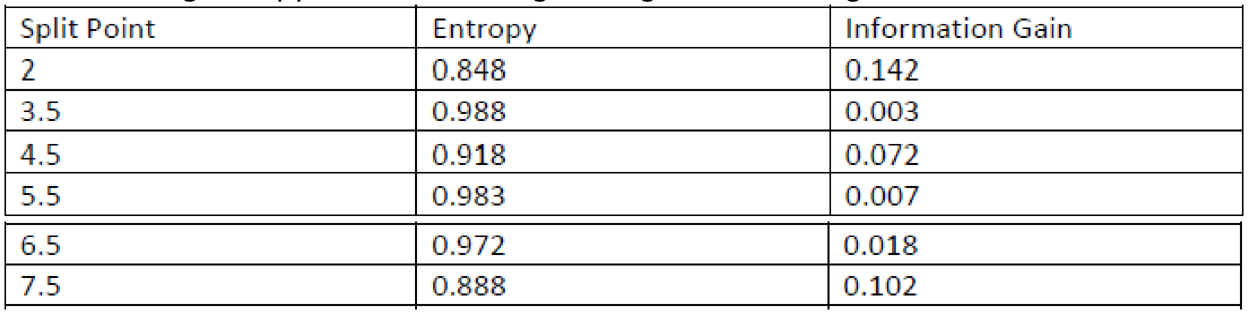
Entropy for a2 = 5/9 * (-2/5 $\log_2$ 2/5 -3/5 $\log_2$ 3/5)

+4/9 * (-2/4 $\log_2$ 2/4 -2/4 $\log_2$ 2/4) = 0.9839

$GAIN_{splita2}$ = 0.991 - 0.9839 = 0.0072

9.3 For a3, which is a continuous attribute, compute the information gain for every possible split.

Ans:
The values of the column a3 range from 1.0, 3.0, 4.0, 5.0, 6.0, 7.0 to 8.0 with an interval of 1. So the possible splits can be done on values between those intervals i.e. 2, 3.5, 4.5, 5.5, 6.5 and 7.5. Calculating entropy and information gain we get the following values.



Split Point	Entropy	Information Gain
2	0.848	0.142
3.5	0.988	0.003
4.5	0.918	0.072
5.5	0.983	0.007
6.5	0.972	0.018
7.5	0.888	0.102

The best split will be at split point 2 for a3

9.4 What is the best split (among a1, a2, and a3) according to the information gain?
Ans: The best split should be at a1 because it gives us the highest information gain of 0.23.

Part 3: Principal Component Analysis on GreenGold Dataset

Question 1: Interpret the summary that appears below. How many components are needed to explain a significant portion of the variance?
Ans: It is possible to notice by analyzing Proportion of Variance and the Cumulative Proportion that the first *five* components hold around 60% of the variance. The other components represent smaller and smaller portions of the variance data so it is possible to ignore them.

We can say this confidently by also looking at the graph for question 2. In the graph we can see a flattening in the histogram after the 5th component. There seems to be a flattening between second and third component but it cannot be accounted for because the values still decrease after the 3rd component and only after the fifth component the graph flattens out.

Question 2: Where does the scree plot flatten out in the plot that follows?
Ans: The scree plot starts to flatten out after the first component. However, there are two times that the scree plot “flatten out”, one is just after the first component and the second one is after the third component. So, it is possible to considerer the components 2 and 3 as useful. Then, the first three components can be considered useful.

Question 3: Interpret the loadings that you see in the R results (see the results in the table that follows). For example, for the first component, there is a (relatively) high correlation with average annual pay, solid waste recycling, environmental policy, and low emissions per job possible indicator of both wealth and "environmental consciousness." Which states are likely to have high scores for this component?

Ans: To interpret those results, the loadings matrices were spotted by its components and each component was sorted. As a result from the first questions, only the first three components were analyzed.

For the component 1:

	Variable 1	Variable 2	Variable 3
Highest rank	Environment policy record	Air quality	Average miles per gallon gas
Lowest rank	Total Energy consumption	Emission Jobs ratio	Workers in high disease industries

For the component 2:

	Variable 1	Variable 2	Variable 3
Highest rank	Education attainment: High School	State budged spent on enviorment	Fertilizer use per capta
Lowest rank	Households with income below poverty	Gaps in income distribution	Unemployment rate

For the component 3:

	Variable 1	Variable 2	Variable 3
Highest rank	Fertilizer use per capta	Employment growth between 1985 and 1993	Average miles per gallon gas
Lowest rank	State spendiment on the environment	Total Energy consumption on million BTU	Average annual pay

It is possible to analyze by those tables that considering Component 1, States with high environmental policy have correlations with the air quality and negative correlation with the total energy consumption, jobs ratio and workers in high disease industries. This suggests a component whose high scores indicate states in which wealth and environment protection are relevant. For the component 2, low employment rate and poverty line are negative correlated with the educational level measured by High School attendance. This component suggest states where educational indices are high and unemployment rate (which affects the poverty), low.

Furthermore, this component suggests that the educational levels are negatively correlated with unemployment. Finally, the component three suggests that states that high levels of “fertilizer use per capita” 1(environmental negative), have lower levels of “state spending on the environment” (environmental positive). Also, the employment growth is negatively correlated with average annual play, for example. States scoring high on this component probably have good employment rates (economic good), but probably don’t care so much with the environment. As a conclusion, it is perceptive how good PCA can perform to analyze the data. By using it, it is possible to find hidden relationships between variables, having the possibility to analyze the collection in a deeper level. Also, after analyze more specifically which states are top ranked for some variables, it easy to realize that the economic and environmental importance given by a state can differ by state to state among all the US.