1. INTRODUCTION

1.1 INTRODUCTION

Machine learning (ML):

- **Machine Learning** is the study of algorithms and statistical models that computer systems use to progressively improve their performance on a specific task.
- Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.
- Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

  Types of learning algorithms:

  a) Supervised Learning b) Unsupervised Learning

a) Supervised learning:

- Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.
- The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and a desired output.
- Supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.
- An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data.

- An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.
- We use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

- Supervised learning problems can be further grouped into regression and classification problems.

- **Classification**: A classification problem is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease".
  - A classification problem requires that examples be classified into one of two or more classes.
  - A classification can have real-valued or discrete input variables.
  - A problem with two classes is often called a two-class or binary classification problem.
  - A problem with more than two classes is often called a multi-class classification problem.
  - A problem where an example is assigned multiple classes is called a multi-label classification problem.
  - An algorithm that is capable of learning a classification predictive model is called a classification algorithm.

- **Regression**: A regression problem is when the output variable is a real value, such as "dollars" or "weight".

- A continuous variable is a real-value, such as an integer or floating-point value. These are often quantities, such as amounts and sizes.
- For example, a house may be predicted to sell for a specific dollar value, perhaps in the range of $100,000 to $200,000.
- A regression can have real-valued or discrete input variables.
- A problem with multiple input variables is often called a multivariate regression problem.
- An algorithm that is capable of learning a regression predictive model is called a regression algorithm.

b) **Unsupervised Learning:**

- Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points.
- Unsupervised learning is where you only have input data (X) and no corresponding output variables.
- The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.
- These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.
- Unsupervised learning problems can be further grouped into clustering and association problems.

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data.

1.2 . DECISION TREE:

- A decision tree is a tree-shaped diagram that people use to determine a course of action.
- In a decision tree, each branch represents a possible decision, occurrence or reaction.
- Simply, Decision tree is a graph to represent choices and their results in form of a tree.
- It is mostly used in Machine Learning and Data Mining applications using R.
- Examples of use of decision trees is − predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these.
- Generally, a model is created with observed data also called training data. Then a set of validation data is used to verify and improve the model.
- R has packages which are used to create and visualize decision trees.
- For new set of predictor variable, we use this model to arrive at a decision on the category (yes/No, spam/not spam) of the data.
- They involve creating a set of binary splits on the predictor variables in order to create a tree that can be used to classify new observations into one of two groups.
- we have two types of decision trees: classical trees and conditional inference trees.
- A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning**,** covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

- In Tree Construction, Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable.
- Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.
- Information gain is used to decide which feature to split on at each step in building the tree. A commonly used measure of purity is called information.
- For each node of the tree, the information value "represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no".
- Information gain is based on the concept of entropy.
- **Entropy**, as it relates to machine learning, is a measure of the randomness in the information being processed.

1.3 RANDOM FOREST

- Ensemble Learning is a type of Supervised Learning Technique in which the basic idea is to generate multiple Models on a training dataset and then simply combining(average) their Output Rules.
- The idea is that instead of producing a single complicated and complex Model which might have a high variance which will lead to Overfitting or might be too simple and have a high bias which leads to Underfitting, we will generate lots of Models by training on Training Set and at the end combine them.
- Such a technique is Random Forest which is a popular technique is used to improve the predictive performance of Decision Trees by reducing the variance in the Trees by averaging them.
- Averaging the Trees helps us to reduce the variance and also improve the Performance of Decision Trees on Test Set and eventually avoid Overfitting.
- An error estimate is made for the cases which were not used while building the tree. That is called an **OOB (Out-of-bag)** error estimate which is mentioned as a percentage.
- The R package **"randomForest"** is used to create random forests.
- The forest error rate depends on two things:
    - The *correlation* between any two trees in the forest. Increasing the correlation increases the forest error rate.
    - The *strength* of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

- Random forests are grown using the randomForest () function in the randomForest package.
- The default number of trees is 500, the default number of variables sampled at each node is sqrt(M), and the minimum node size is 1.

2. DECISION TREE

2.1 FEATURES OF DECISION TREE:

- **Explanatory Power:** The output of decision trees is interpretable. It can be understood by people without analytical or mathematical backgrounds. It does not require any statistical knowledge to interpret them.
- **Exploratory data analysis:** Decision trees can enable analysts to identify significant variables and important relations between two or more variables, helping to surface the signal contained by many input variables.
- **Minimal data cleaning:** Because decision trees are resilient to outliers and missing values, they require less data cleaning than some other algorithms.
- **Any data type:** Decision trees can make classifications based on both numerical and categorical variables.
- **Non-parametric:** A decision tree is a non-parametric algorithm, as opposed to neural networks, which process input data transformed into a tensor, via tensor multiplication using large number of coefficients, known as parameters.

2.2 ALGORITHM

- We have two types of decision trees: classical trees and conditional inference trees.

Classical Trees:

The process of building a classical decision tree starts with a binary outcome variable and a set of predictor variables. The algorithm is as follows:

- Choose the predictor variable that best splits the data into two groups such that the purity (homogeneity) of the outcome in the two groups is maximized

  If the predictor is continuous, choose a cut-point that maximizes purity for the two groups created. If the predictor variable is categorical, combine the categories to obtain two groups with maximum purity.
- Separate the data into these two groups, and continue the process for each subgroup.
- Repeat steps 1 and 2 until a subgroup contains fewer than a minimum number of observations or no splits decrease the impurity beyond a specified threshold. The subgroups in the final set are called *terminal nodes*. Each terminal node is classified as one category of the outcome or the other based on the most frequent value of the outcome for the sample in that node.
- To classify a case, run it down the tree to a terminal node, and assign it the modal outcome value assigned in step 3.

Unfortunately, this process tends to produce a tree that is too large and suffers from Overfitting.

Classical trees are provided by the rpart () function in the rpart package.

Conditional inference trees

Conditional inference trees are similar to traditional trees, but variables and splits are selected based on significance tests rather than purity/homogeneity measures.

In this case, the algorithm is as follows:

- Calculate p-values for the relationship between each predictor and the outcome variable.
- Select the predictor with the lowest p-value.
- Explore all possible binary splits on the chosen predictor and dependent variable (using permutation tests), and pick the most significant split.
- Separate the data into these two groups, and continue the process for each subgroup.
- Continue until splits are no longer significant or the minimum node size is reached.

Conditional inference trees are provided by the ctree () function in the party package.

2.3 SAMPLE EXAMPLE

We have considered a dataset ‘readingSkills’. It is a data frame with 200 observations on the following 4 variables. They are:

nativeSpeaker

a factor with levels no and yes, where yes indicates that the child is a native speaker of the language of the reading test.

age

age of the child in years.

shoeSize

shoe size of the child in cm.

score

raw score on the reading test.

In this artificial data set, that was generated by means of a linear model, age and nativeSpeakers are actual predictors of the score, while the spurious correlation between score and shoeSize is merely caused by the fact that both depend on age.

The basic syntax for creating a decision tree in R is

ctree (formula, data)

Following is the description of the parameters used −

- **formula** is a formula describing the predictor and response variables.
- **data** is the name of the data set used.

Code Implementation:

```
# Load the party package. It will automatically load other

library(party)

# Print some records from data set readingSkills.

print(head(readingSkills))

library(party)

data("readingSkills")

# Create the input data frame.

input.dat <- readingSkills[c(1:105),]

View(input.dat)

# Create the tree.

output.tree <- ctree(

  nativeSpeaker ~ age + shoeSize + score,

  data = input.dat)

# Plot the tree.

plot(output.tree)
```

OUTPUT:

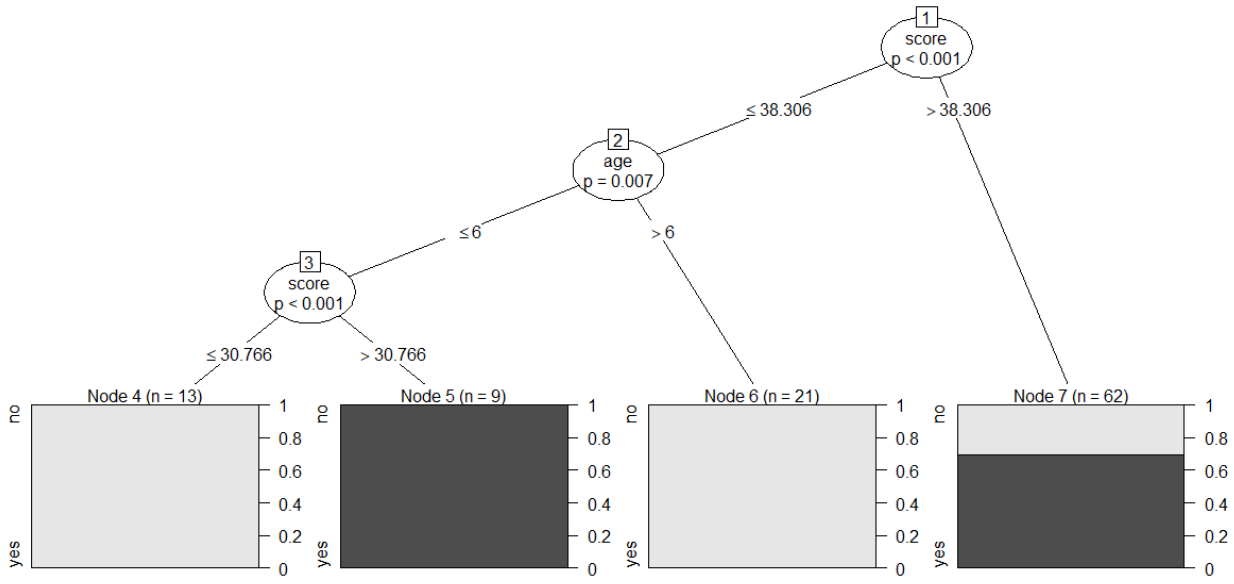


FIG 2.4

From the decision tree, we can conclude that anyone whose readingSkills score is less than 38.3 and age is more than 6 is not a native Speaker.

2.4 ADVANTAGES AND DISADVANTAGES OF DECISION TREE:

ADVANTAGES:

Decision trees have a number of advantages as a practical, useful managerial tool.

Comprehensive

A significant advantage of a decision tree is that it forces the consideration of all possible outcomes of a decision and traces each path to a conclusion. It creates a comprehensive analysis of the consequences along each branch and identifies decision nodes that need further analysis.

Specific

Decision trees assign specific values to each problem, decision path and outcome. Using monetary values makes costs and benefits explicit. This approach identifies the relevant decision paths, reduces uncertainty, clears up ambiguity and clarifies the financial consequences of various courses of action.

When factual information is not available, decision trees use probabilities for conditions to keep choices in perspective with each other for easy comparisons.

Easy to Use

Decision trees are easy to use and explain with simple math, no complex formulas. They present visually all of the decision alternatives for quick comparisons in a format that is easy to understand with only brief explanations.

They are intuitive and follow the same pattern of thinking that humans use when making decisions.

Versatile

A multitude of business problems can be analyzed and solved by decision trees. They are useful tools for business managers, technicians, engineers, medical staff and anyone else who has to make decisions under uncertain conditions.

Simple decision trees can be manually constructed or used with computer programs for more complicated diagrams.

Decision trees is a technique to find the best solutions to problems with uncertainty.

DISADVANTAGES:

Overfitting

Over fitting is a common flaw of decision trees. Setting constraints on model parameters (depth limitation) and making the model simpler through pruning are two ways to regularize a decision tree and improve its ability to generalize onto the test set.

Predicting continuous variables

While decision trees can ingest continuous numerical input, they are not a practical way to predict such values, since decision-tree predictions must be separated into discrete categories, which results in a loss of information when applying the model to continuous values.

Heavy feature engineering

The flip side of a decision tree's explanatory power is that it requires heavy feature engineering. When dealing with unstructured data or data with latent factors, this makes decision trees sub-optimal. Neural networks are clearly superior in this regard.

3. RANDOM FOREST

3.1 FEATURES OF RANDOM FOREST

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

3.2 ALGORITHM

Assume that *N* is the number of cases in the training sample and *M* is the number of variables. Then the algorithm is as follows:

- Grow a large number of decision trees by sampling *N* cases with replacement from the training set.
- Sample $m < M$ variables at each node. These variables are considered candidates for splitting in that node. The value *m* is the same for each node.
- Grow each tree fully without pruning (the minimum node size is set to 1).
- Terminal nodes are assigned to a class based on the mode of cases in that node.
- Classify new cases by sending them down all the trees and taking a vote majority rules.

Variable importance

- In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class. Now randomly permute the values of variable m in the oob cases and put these cases down the tree. Subtract the number of votes for the correct class in the variable-m-permuted oob data from the number of votes for the correct class in the untouched oob data. The average of this number over all trees in the forest is the raw importance score for variable m.
- If the values of this score from tree to tree are independent, then the standard error can be computed by a standard computation. The correlations of these scores between trees have been computed for a number of data sets and proved to be quite low, therefore we compute standard errors in the classical way, divide the raw score

by its standard error to get a z-score, ands assign a significance level to the z-score assuming normality.

- If the number of variables is very large, forests can be run once with all the variables, then run again using only the most important variables from the first run.

Missing value replacement for the training set

- If the mth variable is not categorical, the method computes the median of all values of this variable in class j, then it uses this value to replace all missing values of the mth variable in class j. If the mth variable is categorical, the replacement is the most frequent non-missing value in class j.

3.3 SAMPLE EXAMPLE

We have considered a dataset ‘readingSkills’. It is a data frame with 200 observations on the following 4 variables. They are:

nativeSpeaker

a factor with levels no and yes, where yes indicates that the child is a native speaker of the language of the reading test.

age

age of the child in years.

shoeSize

shoe size of the child in cm.

score

raw score on the reading test.

In this artificial data set, that was generated by means of a linear model, age and nativeSpeakers are actual predictors of the score, while the spurious correlation between score and shoeSize is merely caused by the fact that both depend on age.

The basic syntax for creating a random forest in R is

randomForest (formula, data)

Following is the description of the parameters used −

- **formula** is a formula describing the predictor and response variables.
- **data** is the name of the data set used.

Code Implementation:

```
library(randomForest)

# Create the forest.

output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score,

                              data = readingSkills)

# View the forest results.

print(output.forest)

# Importance of each predictor.

print(importance(output.forest,type = 2))
```

OUTPUT:

```
 randomForest(formula = nativeSpeaker ~ age + shoeSize + score,     data = readingSkills)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1


        OOB estimate of  error rate: 1%
Confusion matrix:
     no yes class.error
no  99   1        0.01
```

yes 1 99 0.01

	MeanDecreaseGini
age	13.26635
shoeSize	19.18101
score	58.22390

From the random forest shown above we can conclude that the shoesize and score are the important factors deciding if someone is a native speaker or not. Also the model has only 1% error which means we can predict with 99% accuracy.

3.4 ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

- The overfitting problem will never come when we use the random forest algorithm in any classification problem.
- The same random forest algorithm can be used for both classification and regression task.
- The random forest algorithm can be used for feature engineering.
    - Which means identifying the most important features out of the available features from the training dataset.

DISADVANTAGES:

- Not easily interpretable.
- It may not work if the dependent variables considered in the model are linearly related.

4. DESIGN

4.1 SOFTWARE REQUIREMENTS:

- **RStudio** is a free and open-source integrated development environment (IDE) for R, a programming language for statistical computing and graphics. RStudio was founded by JJ Allaire, creator of the programming language ColdFusion. Hadley Wickham is the Chief Scientist at RStudio.
- RStudio is available in two editions: RStudio Desktop, where the program is run locally as a regular desktop application; and RStudio Server, which allows accessing RStudio using a web browser while it is running on a remote Linux server. Prepackaged distributions of RStudio Desktop are available for Windows, macOS, and Linux.
- RStudio is available in open source and commercial editions and runs on the desktop (Windows, macOS, and Linux) or in a browser connected to RStudio Server or RStudio Server Pro (Debian, Ubuntu, Red Hat Linux, CentOS, openSUSE and SLES).
- RStudio is partly written in the C++ programming language and uses the Qt framework for its graphical user interface. The bigger percentage of the code is written in Java. JavaScript is also amongst the languages used.
- Work on RStudio started around December 2010, and the first public beta version (v0.92) was officially announced in February 2011.Version 1.0 was released on 1 November 2016. Version 1.1 was released on 9 October 2017.

- In April 2018 it was announced RStudio will be providing operational and infrastructure support for Ursa Labs. Ursa Labs will focus on building a new data science runtime powered by Apache Arrow.

Introduction to R:

- R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

- R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

- One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

- R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and

runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

The R environment:

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

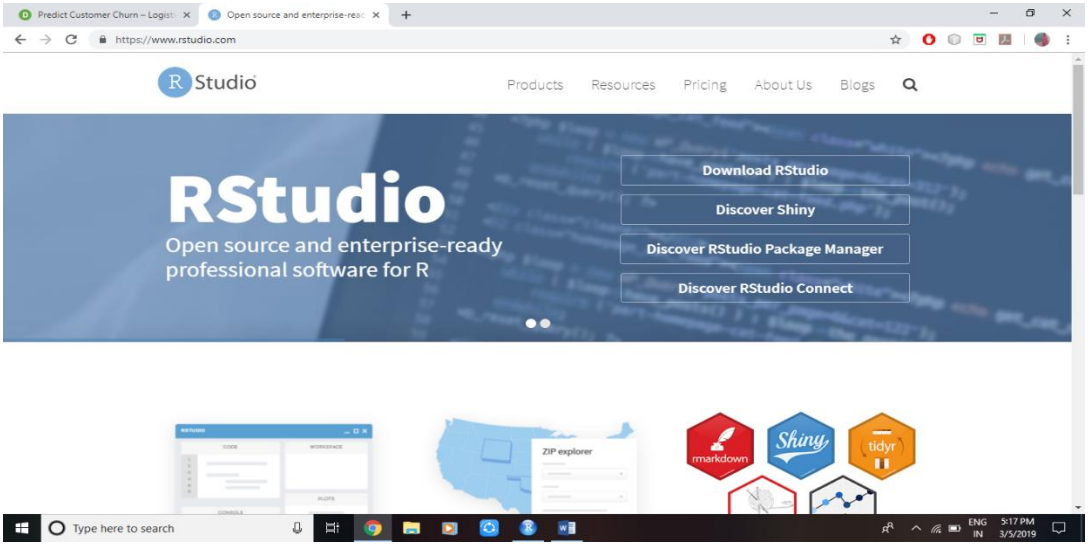
R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it of an environment within which statistical techniques are implemented. R can be extended (easily) via *packages*. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

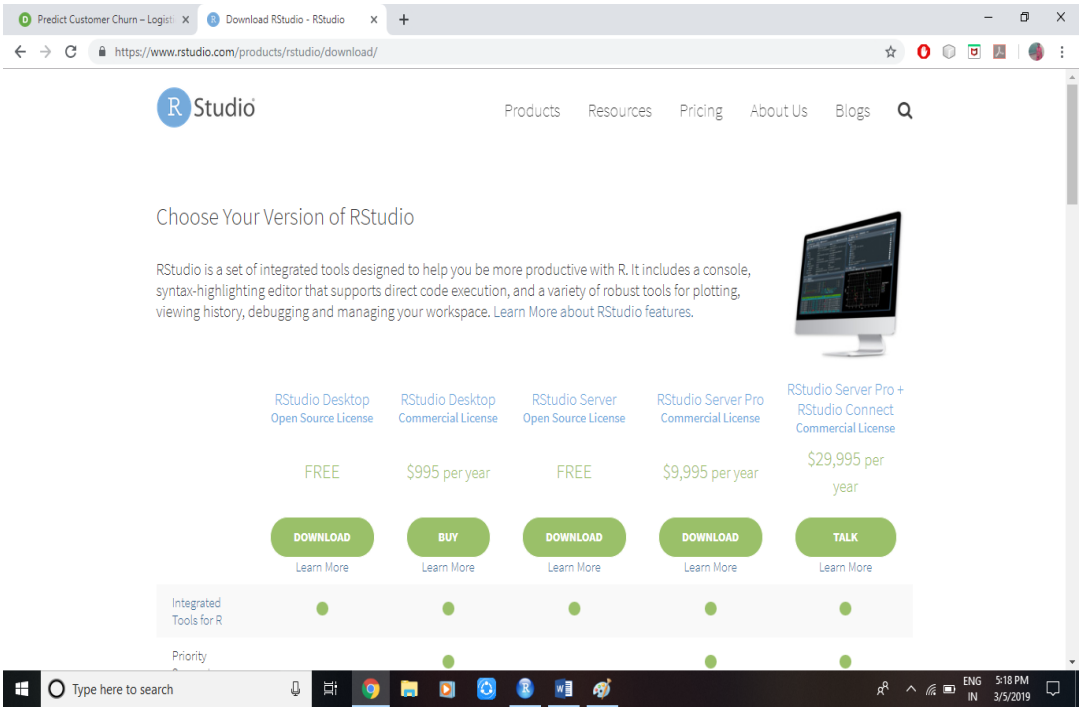
INSTALLING RSTUDIO:

1)Go to the website 'https://www.rstudio.com'.

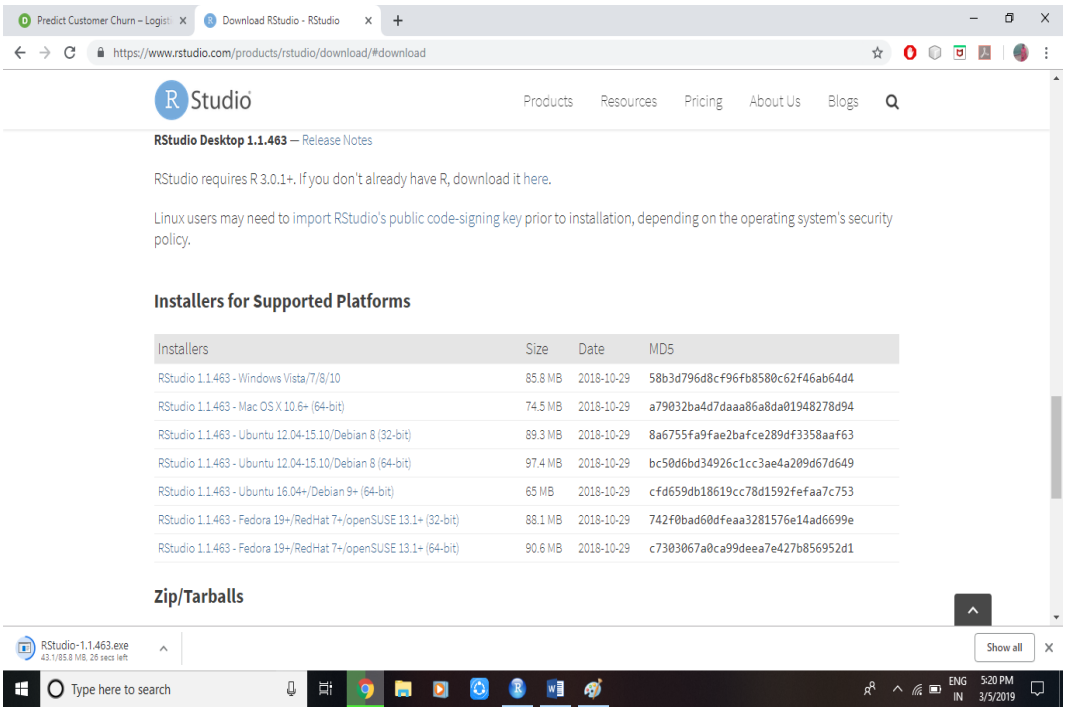
Download RStudio.



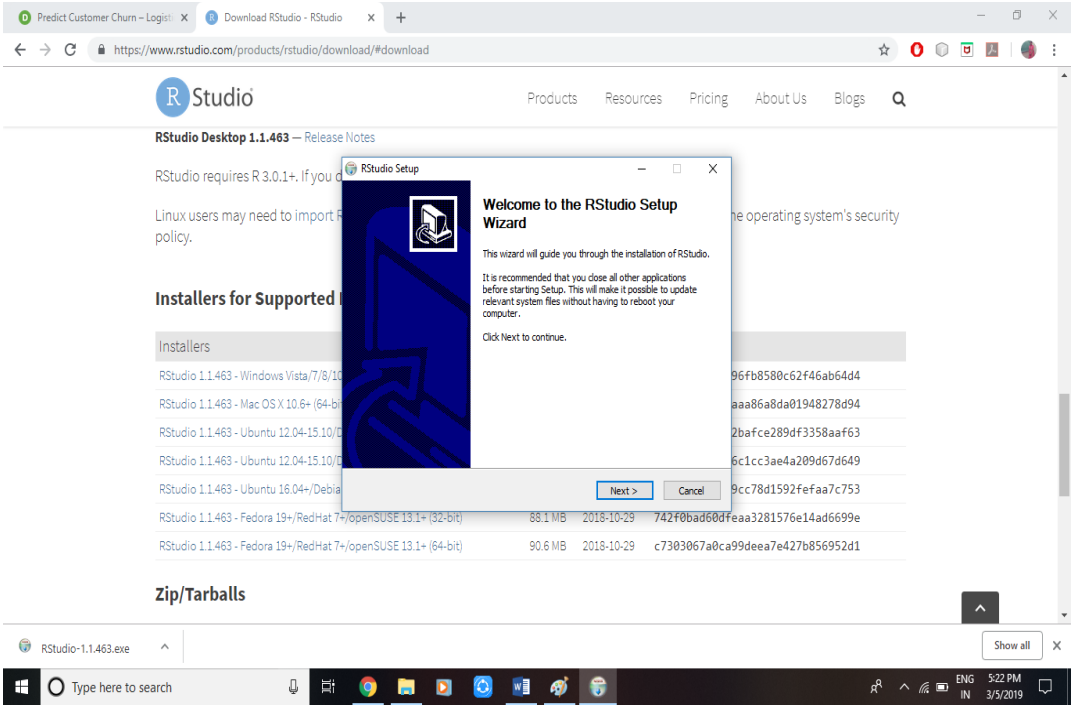
2) Select Download Button – RStudio Desktop Open Source License



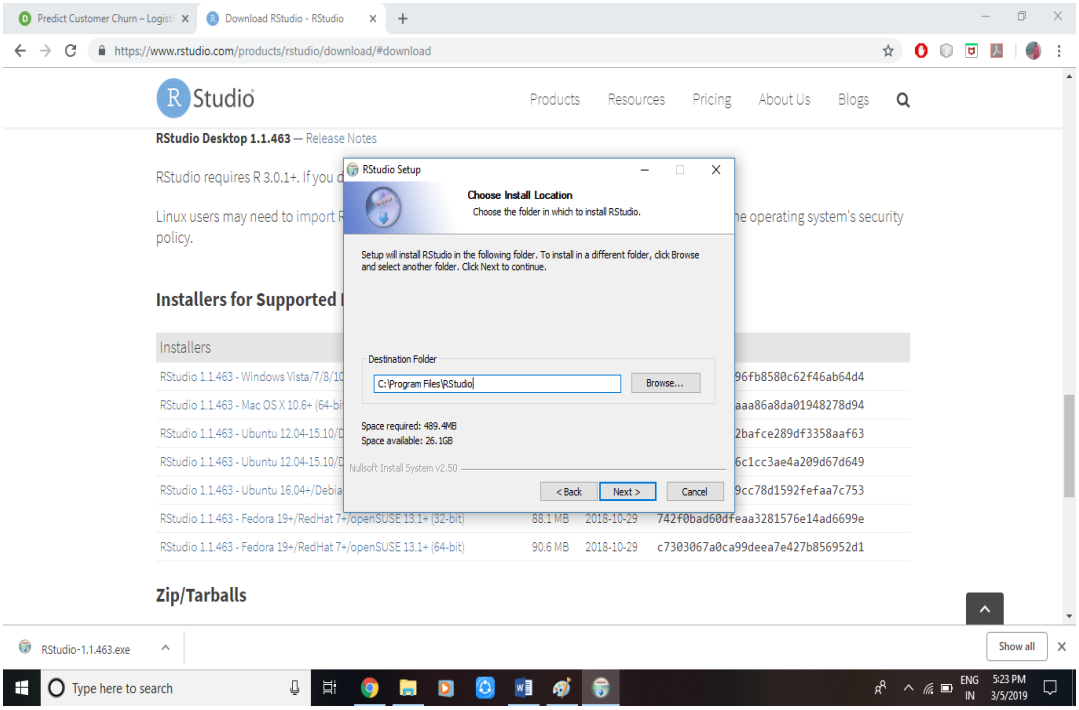
3)Download RStudio 1.1.383 Windows Vista 7.8.10



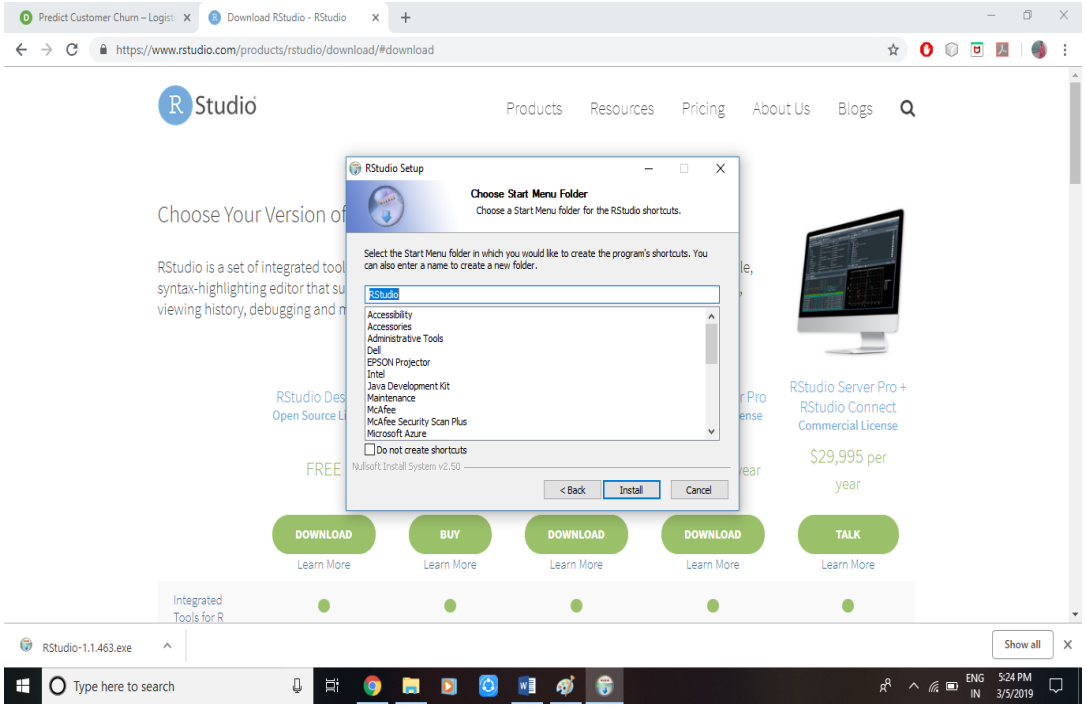
4)Click ‘Next’.



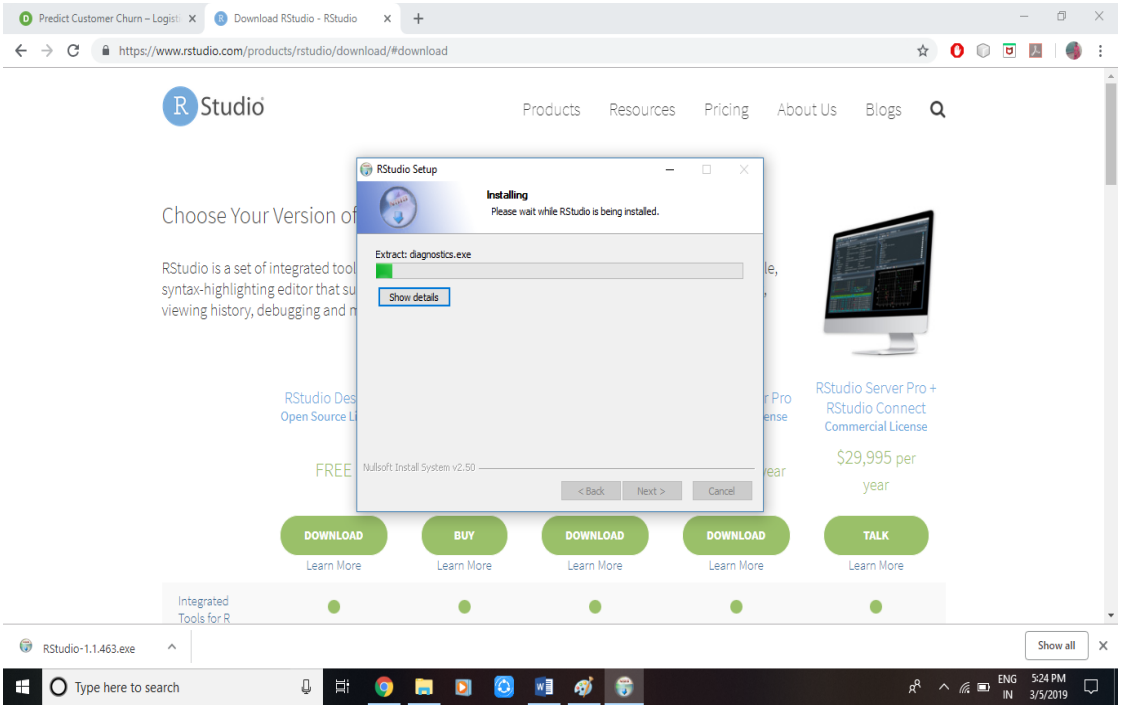
5)Select Destination Folder.



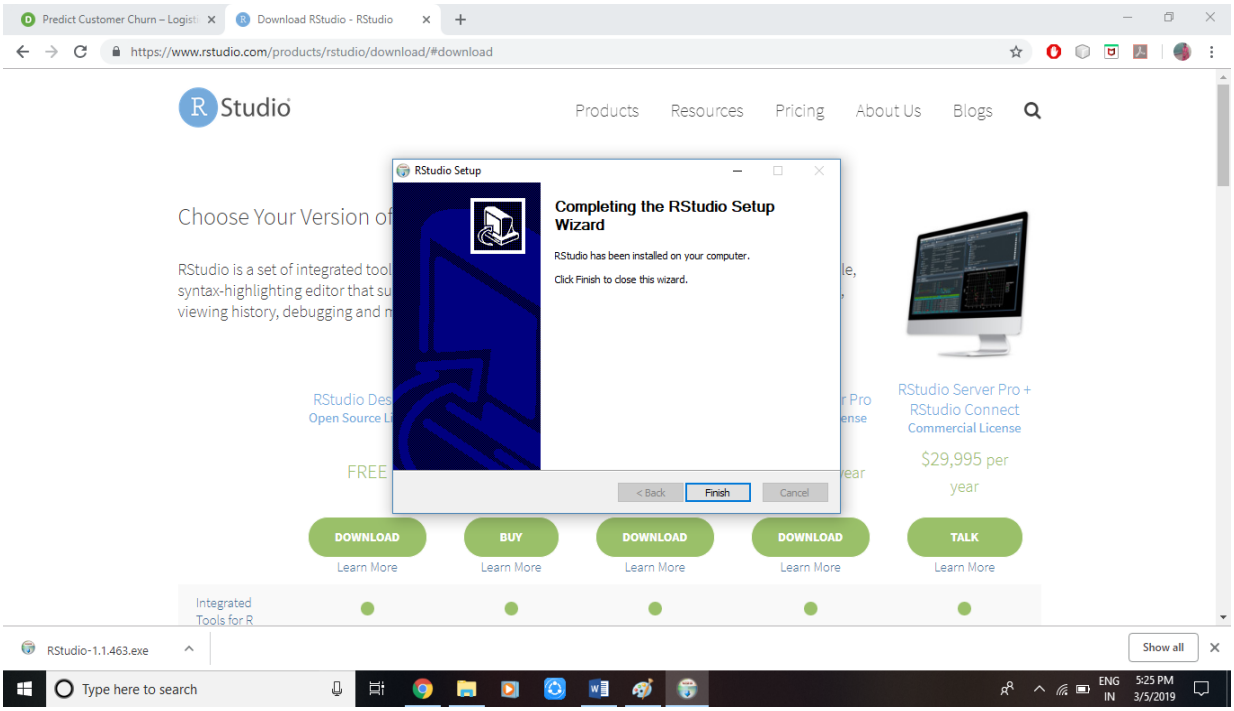
6)Choose Menu Folder

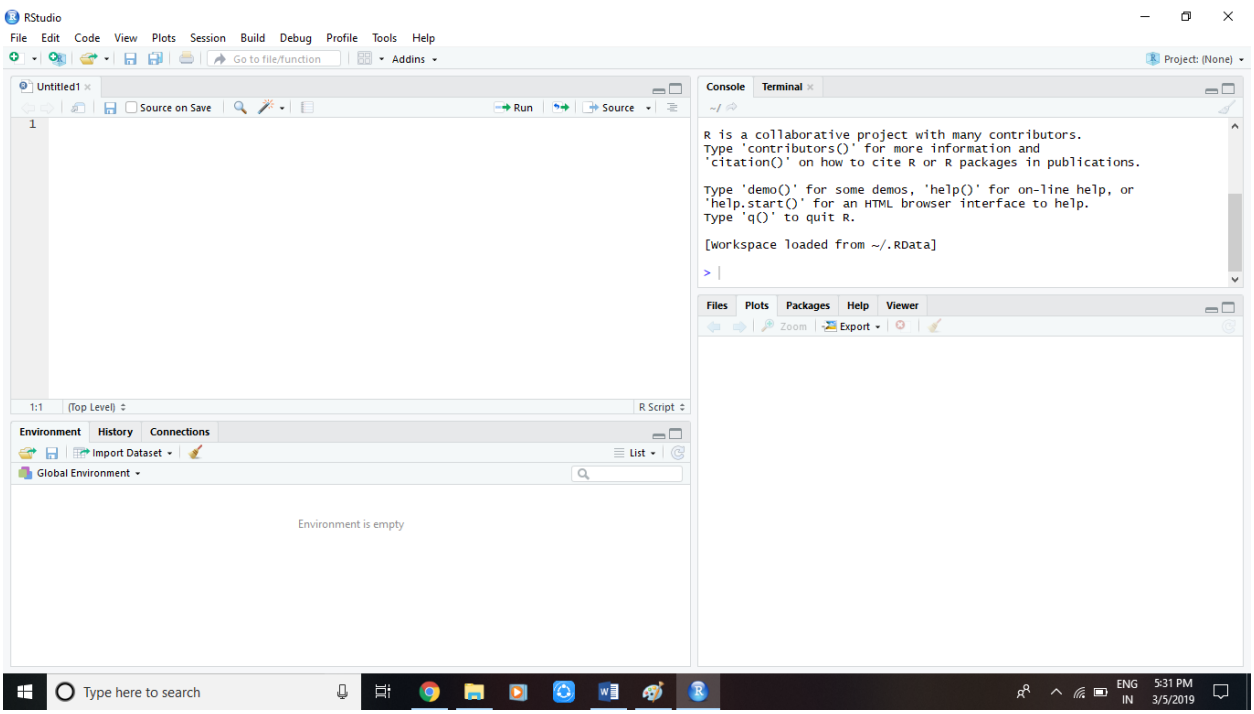


7)Installing process



8)Click‘Finish’.

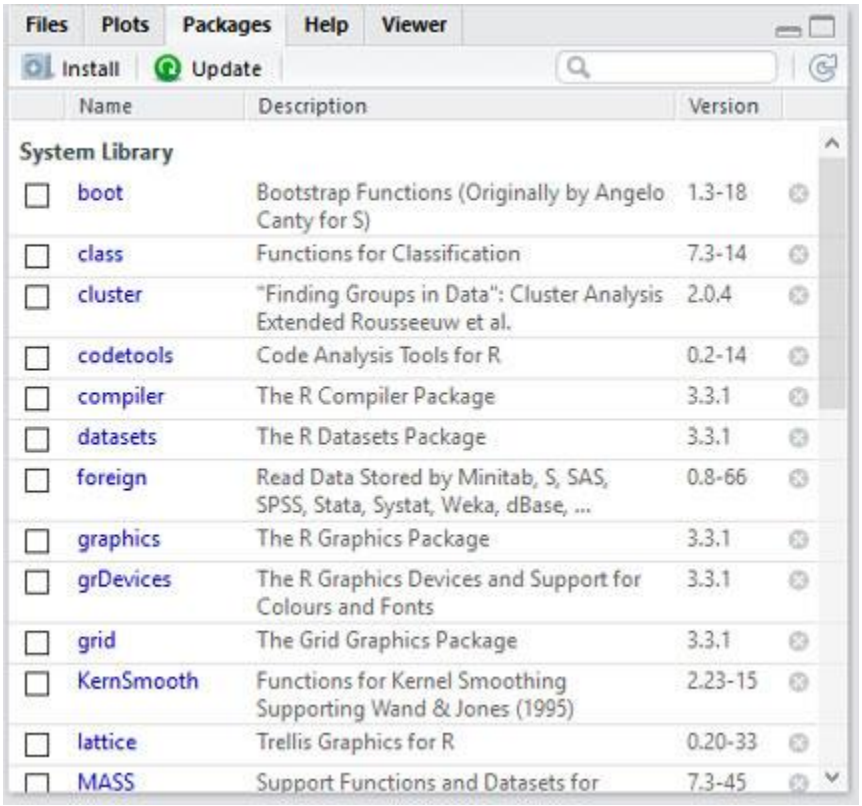




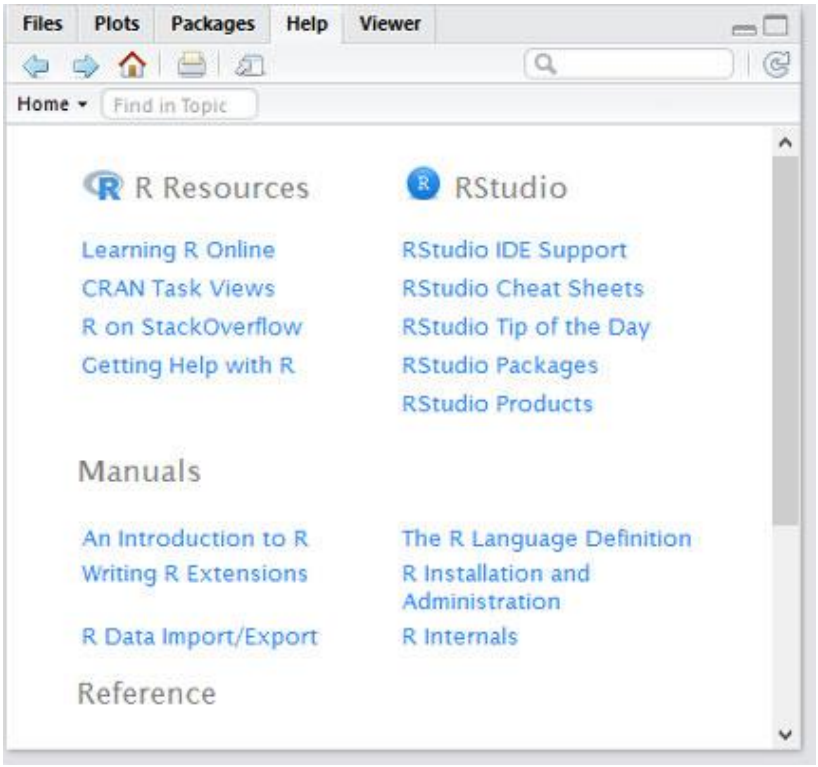
The large Console pane on the left runs R code. One way to run R code is to type it directly into the Console pane.

The other two panes provide helpful information as you work with R. The Environment and History pane is in the upper right. The Environment tab keeps track of the things you create (which R calls objects) as you work with R. The History tab tracks R code that you enter.

The Files, Plots, Packages, and Help tabs are in the pane in the lower right. The Files tab shows files you create. The Plots tab holds graphs you create from your data. The Packages tab shows add-ons (called packages) you downloaded as part of the R installation. Bear in mind that "downloaded" doesn't mean "ready to use." To use a package's capabilities, one more step is necessary, and you'll want to use packages.



The Help tab, shown here, provides links to a wealth of information about R and RStudio.



4.2 HARDWARE REQUIREMENTS

- Operating System : Windows 7 or 8 or 10
- Processor Speed : 1.5 GHz or above
- RAM Size : 8 GB
- Hard Disk Size : 256 GB

5.SOURCE CODE IMPLEMENTATION

Customer churn occurs when customers or subscribers stop doing business with a company or service, also known as customer attrition. It is also referred as loss of clients or customers. One industry in which churn rates are particularly useful is the telecommunications industry, because most customers have multiple options from which to choose within a geographic location.

Data Preprocessing

Each row represents a customer, each column contains that customer's attributes:

```
library(plyr)

library(corrplot)

library(ggplot2)

library(gridExtra)

library(ggthemes)

library(caret)

library(MASS)

library(randomForest)

library(party)

churn <- read.csv('Telco-Customer-Churn.csv')

str(churn)
```

The raw data contains 7043 rows (customers) and 21 columns (features). The “Churn” column is our target.

We use sapply to check the number if missing values in each columns. We found that there are 11 missing values in “TotalCharges” columns. So, let’s remove all rows with missing values.

```
sapply(churn, function(x) sum(is.na(x)))

churn <- churn[complete.cases(churn),]
```

Look at the variables, we can see that we have some wranglings to do.

1. We will change “No internet service” to “No” for six columns, they are: “OnlineSecurity”, “OnlineBackup”, “DeviceProtection”, “TechSupport”, “streamingTV”, “streamingMovies”.

```
cols_recode1 <- c(10:15)

for(i in 1:ncol(churn[,cols_recode1])) {

    churn[,cols_recode1][,i] <- as.factor(mapvalues

    (churn[,cols_recode1][,i], from =c("No internet service"),to=c("No")))

}
```

2. We will change “No phone service” to “No” for column “MultipleLines”

```
churn$MultipleLines <- as.factor(mapvalues(churn$MultipleLines,

                                from=c("No phone service"),
```

```
                                to=c("No")))
```

3. Since the minimum tenure is 1 month and maximum tenure is 72 months, we can group them into five tenure groups: “0–12 Month”, “12–24 Month”, “24–48 Months”, “48–60 Month”, “> 60 Month”

```
min(churn$tenure)

max(churn$tenure)

group_tenure <- function(tenure){
 if (tenure >= 0 & tenure <= 12){
   return('0-12 Month')
 }else if(tenure > 12 & tenure <= 24){
   return('12-24 Month')
 }else if (tenure > 24 & tenure <= 48){
   return('24-48 Month')
 }else if (tenure > 48 & tenure <=60){
   return('48-60 Month')
 }else if (tenure > 60){
   return('> 60 Month')
 }
}
```

```
churn$tenure_group <- sapply(churn$tenure,group_tenure)

churn$tenure_group <- as.factor(churn$tenure_group)
```

4. Change the values in column “SeniorCitizen” from 0 or 1 to “No” or “Yes”.

```
churn$SeniorCitizen <- as.factor(mapvalues(churn$SeniorCitizen,

                                    from=c("0","1"),

                                    to=c("No", "Yes")))
```

5. Remove the columns we do not need for the analysis.

```
churn$customerID <- NULL

churn$tenure <- NULL
```

Exploratory data analysis and feature selection

Correlation between numeric variables

```
numeric.var <- sapply(churn, is.numeric)

corr.matrix <- cor(churn[,numeric.var])

corrplot(corr.matrix, main="\n\nCorrelation Plot for Numerical Variables", method="number")
```

The Monthly Charges and Total Charges are correlated. So one of them will be removed from the model. We remove Total Charges.

```
churn$TotalCharges <- NULL
```

Bar plots of categorical variables

```
p1 <- ggplot(churn, aes(x=gender)) + ggtitle("Gender") + xlab("Gender") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p2 <- ggplot(churn, aes(x=SeniorCitizen)) + ggtitle("Senior Citizen") + xlab("Senior
Citizen") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p3 <- ggplot(churn, aes(x=Partner)) + ggtitle("Partner") + xlab("Partner") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p4 <- ggplot(churn, aes(x=Dependents)) + ggtitle("Dependents") + xlab("Dependents") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

grid.arrange(p1, p2, p3, p4, ncol=2)


p5 <- ggplot(churn, aes(x=PhoneService)) + ggtitle("Phone Service") + xlab("Phone
Service") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()
```

```
p6 <- ggplot(churn, aes(x=MultipleLines)) + ggtitle("Multiple Lines") + xlab("Multiple Lines") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()

p7 <- ggplot(churn, aes(x=InternetService)) + ggtitle("Internet Service") + xlab("Internet Service") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()

p8 <- ggplot(churn, aes(x=OnlineSecurity)) + ggtitle("Online Security") + xlab("Online Security") +
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()

grid.arrange(p5, p6, p7, p8, ncol=2)


p9 <- ggplot(churn, aes(x=OnlineBackup)) + ggtitle("Online Backup") + xlab("Online Backup") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()

p10 <- ggplot(churn, aes(x=DeviceProtection)) + ggtitle("Device Protection") + xlab("Device Protection") +
```

```
 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p11 <- ggplot(churn, aes(x=TechSupport)) + ggtitle("Tech Support") + xlab("Tech
Support") +

 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p12 <- ggplot(churn, aes(x=StreamingTV)) + ggtitle("Streaming TV") + xlab("Streaming
TV") +

 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

grid.arrange(p9, p10, p11, p12, ncol=2)

p13 <- ggplot(churn, aes(x=StreamingMovies)) + ggtitle("Streaming Movies") +
xlab("Streaming Movies") +

geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p14 <- ggplot(churn, aes(x=Contract)) + ggtitle("Contract") + xlab("Contract") +

 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p15 <- ggplot(churn, aes(x=PaperlessBilling)) + ggtitle("Paperless Billing") +
xlab("Paperless Billing") +
```

```
 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p16 <- ggplot(churn, aes(x=PaymentMethod)) + ggtitle("Payment Method") +
xlab("Payment Method") +

 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

p17 <- ggplot(churn, aes(x=tenure_group)) + ggtitle("Tenure Group") + xlab("Tenure
Group") +

 geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") +
coord_flip() + theme_minimal()

grid.arrange(p13, p14, p15, p16, p17, ncol=2)
```

All of the categorical variables seem to have a reasonably broad distribution, therefore, all of them will be kept for the further analysis.

First, we split the data into training and testing sets

```
intrain<- createDataPartition(churn$Churn,p=0.7,list=FALSE)

set.seed(2017)

training<- churn[intrain,]

testing<- churn[-intrain,]
```

```
dim(training)

dim(testing)
```

Decision Tree

For illustration purpose, we are going to use only three variables for plotting Decision Trees, they are “Contract”, “tenure_group” and “PaperlessBilling”.

```
tree <- ctree(Churn~Contract+tenure_group+PaperlessBilling, training)

plot(tree, type='simple')
```

1. Out of three variables we use, Contract is the most important variable to predict customer churn or not churn.
2. If a customer in a one-year or two-year contract, no matter he (she) has Paperless Billing or not, he (she) is less likely to churn.
3. On the other hand, if a customer is in a month-to-month contract, and in the tenure group of 0–12 month, and using Paperless Billing, then this customer is more likely to churn.

Decision Tree Confusion Matrix:

We are using all the variables to product confusion matrix table and make predictions.

```
pred_tree <- predict(tree, testing)
```

```
print("Confusion Matrix for Decision Tree")

table(Predicted = pred_tree, Actual = testing$Churn
```

Decision Tree Accuracy

```
p1 <- predict(tree, training)

tab1 <- table(Predicted = p1, Actual = training$Churn)

tab2 <- table(Predicted = pred_tree, Actual = testing$Churn)

print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))
```

The accuracy for Decision Tree has hardly improved. Let's see if we can do better using Random Forest.

Random Forest

Random Forest Initial Model

```
rfModel <- randomForest(Churn ~., data = training)

print(rfModel)
```

The error rate is relatively low when predicting "No", and the error rate is much higher when predicting "Yes".

Random Forest Prediction and Confusion Matrix

```
pred_rf <- predict(rfModel, testing)

caret::confusionMatrix(pred_rf, testing$Churn)
```

Random Forest Error Rate

```
plot(rfModel)
```

We use this plot to help us determine the number of trees. As the number of trees increases, the OOB error rate decreases, and then becomes almost constant. We are not able to decrease the OOB error rate after about 100 to 200 trees.

Tune Random Forest Model

```
t <- tuneRF(training[, -18], training[, 18], stepFactor = 0.5, plot = TRUE, ntreeTry = 200, trace = TRUE, improve = 0.05)
```

We use this plot to give us some ideas on the number of mtry to choose. OOB error rate is at the lowest when mtry is 2. Therefore, we choose mtry=2.

Fit the Random Forest Model After Tuning

```
rfModel_new <- randomForest(Churn ~., data = training, ntree = 200, mtry = 2, importance = TRUE, proximity = TRUE)

print(rfModel_new)
```

OOB error rate decreased to 19.7% from 20.65% earlier.

Random Forest Predictions and Confusion Matrix After Tuning

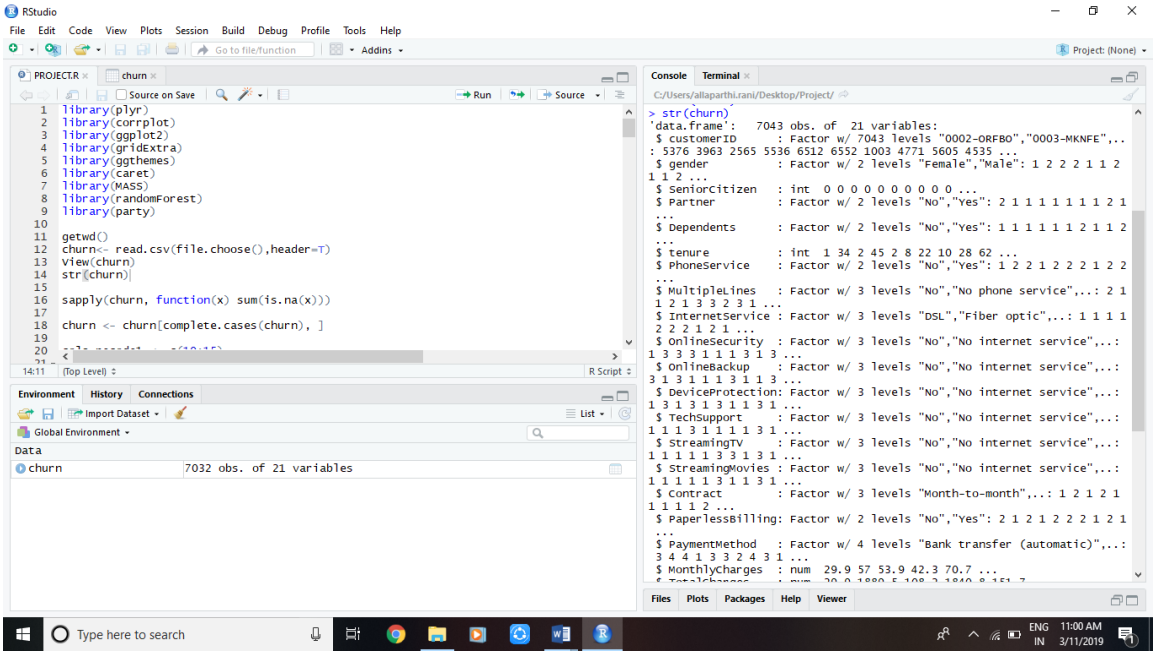
```
pred_rf_new <- predict(rfModel_new, testing)

caret::confusionMatrix(pred_rf_new, testing$Churn)
```

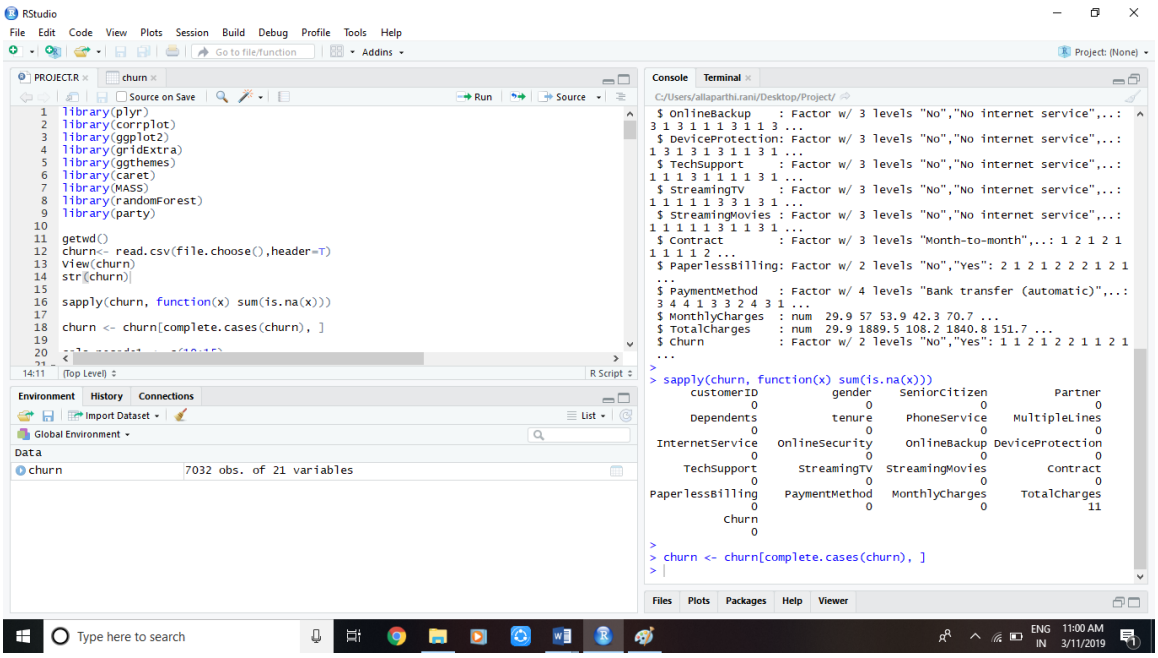
The accuracy did not increase but the sensitivity improved, compare with the initial Random Forest model.

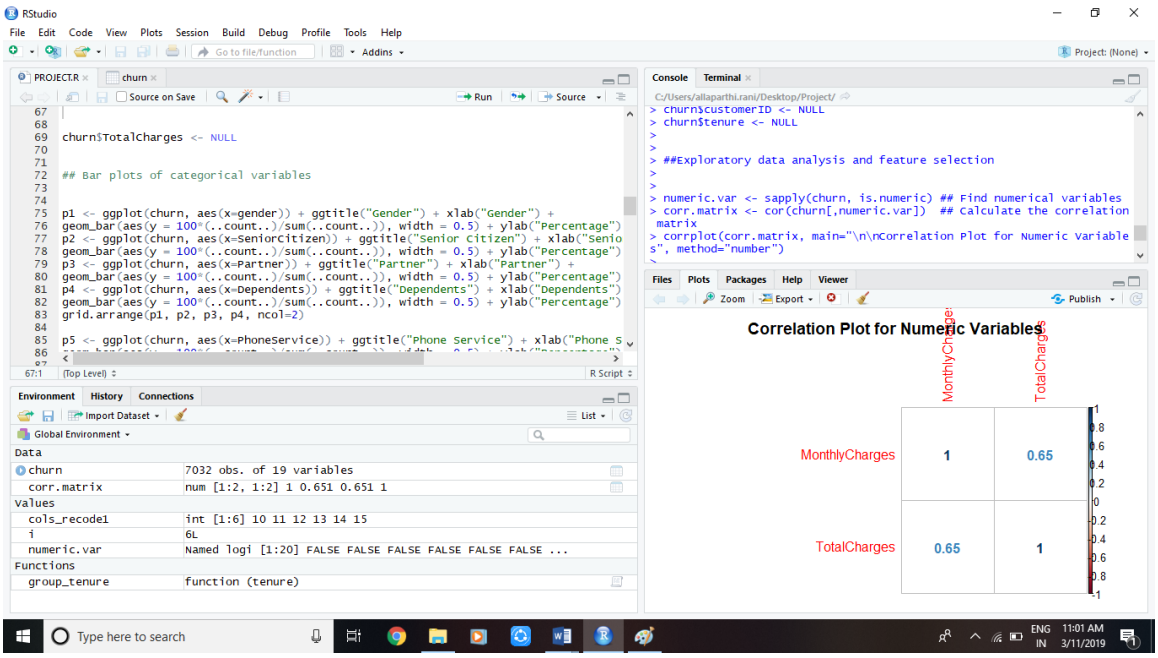
Random Forest Feature Importance

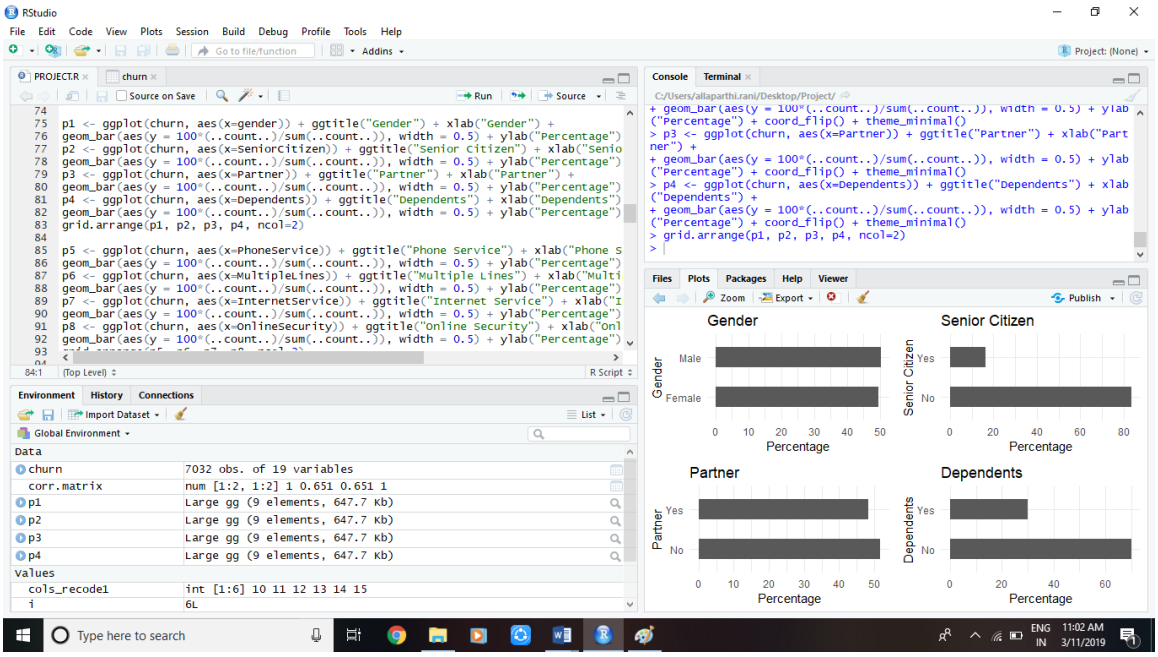
```
varImpPlot(rfModel_new, sort=T, n.var = 10, main = 'Top 10 Feature Importance')
```

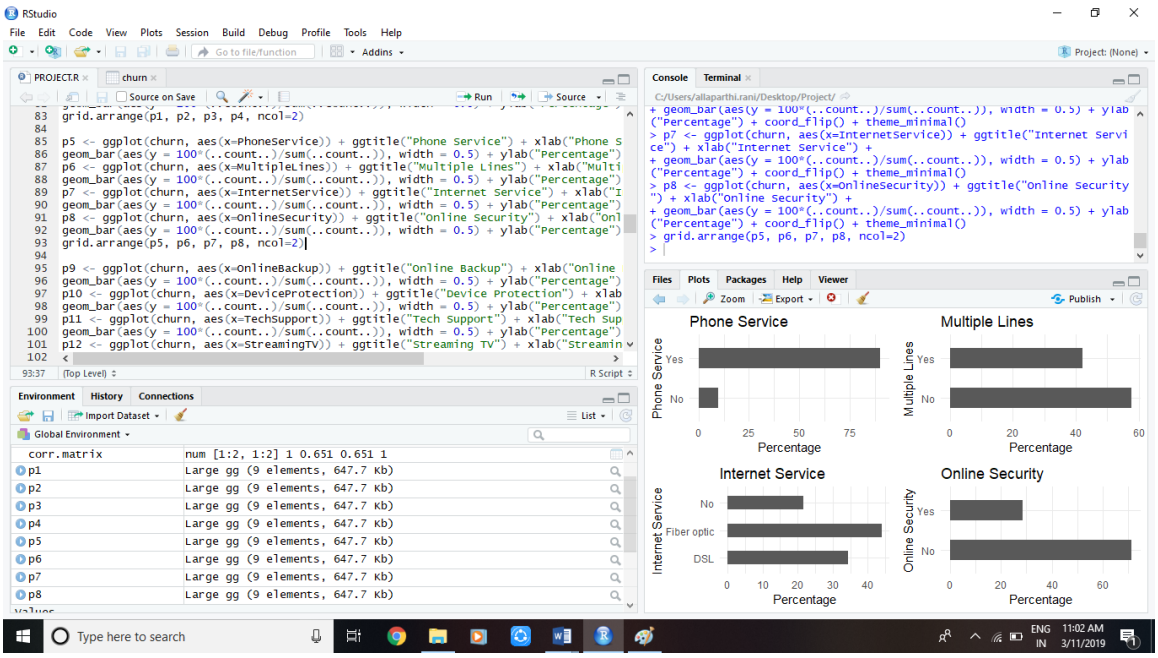
OUTPUTS:

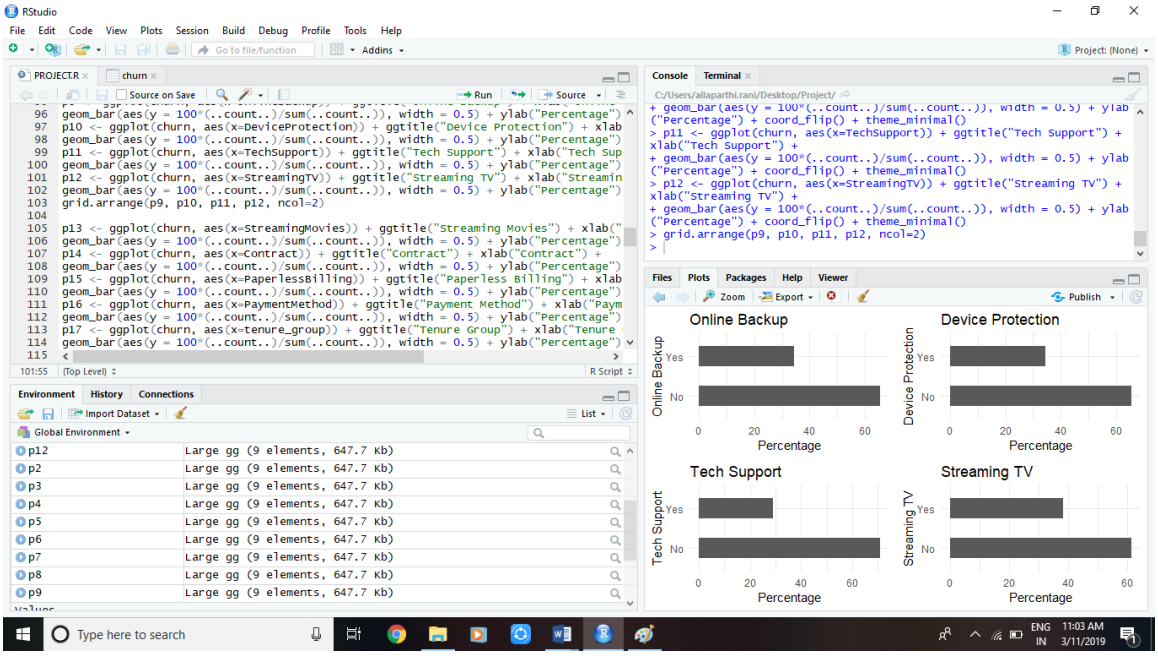

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
Source on Save
Run
Source
1 library(plyr)
2 library(corrplot)
3 library(ggplot2)
4 library(gridExtra)
5 library(ggthemes)
6 library(caret)
7 library(MASS)
8 library(randomForest)
9 library(party)
10
11 getwd()
12 churn<- read.csv(file.choose(),header=T)
13 View(churn)
14 str(churn)
15
16 sapply(churn, function(x) sum(is.na(x)))
17
18 churn <- churn[complete.cases(churn),]
19
20
14:11 (Top Level) R Script
Environment History Connections
Import Dataset List
Global Environment
Data
churn 7032 obs. of 21 variables
Console Terminal
C:/Users/allaparthi.rani/Desktop/Project/
> str(churn)
'data.frame': 7043 obs. of 21 variables:
$ customerID : Factor w/ 7043 levels "0002-ORFBO","0003-MKNFE",..
: 5376 3963 2565 5536 6512 6552 1003 4771 5605 4535 ...
$ gender : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 1 2
1 1 2 ...
$ SeniorCitizen : int 0 0 0 0 0 0 0 0 0 0 ...
$ Partner : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 2 1
...
$ Dependents : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 2
...
$ tenure : int 1 34 2 45 2 8 22 10 28 62 ...
$ PhoneService : Factor w/ 2 levels "No","Yes": 1 2 2 1 2 2 2 1 2 2
...
$ MultipleLines : Factor w/ 3 levels "No","No phone service",..: 2 1
1 2 1 3 3 2 3 1 ...
$ InternetService : Factor w/ 3 levels "DSL","Fiber optic",..: 1 1 1 1
2 2 2 1 2 1 ...
$ OnlineSecurity : Factor w/ 3 levels "No","No internet service",..:
1 3 3 3 1 1 1 3 1 3 ...
$ OnlineBackup : Factor w/ 3 levels "No","No internet service",..:
3 1 3 1 1 1 3 1 1 3 ...
$ DeviceProtection: Factor w/ 3 levels "No","No internet service",..:
1 3 1 3 1 1 3 1 1 3 ...
$ TechSupport : Factor w/ 3 levels "No","No internet service",..:
1 1 1 3 1 1 1 3 1 3 ...
$ StreamingTV : Factor w/ 3 levels "No","No internet service",..:
1 1 1 1 1 3 3 1 3 1 ...
$ StreamingMovies : Factor w/ 3 levels "No","No internet service",..:
1 1 1 1 1 3 1 1 3 1 ...
$ Contract : Factor w/ 3 levels "Month-to-month",..: 1 2 1 2 1
1 1 1 2 ...
$ PaperlessBilling: Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 2 1
...
$ PaymentMethod : Factor w/ 4 levels "Bank transfer (automatic)",..:
3 4 4 1 3 3 2 4 3 1 ...
$ MonthlyCharges : num 29.9 57 53.9 42.3 70.7 ...
Files Plots Packages Help Viewer
Type here to search
ENG IN 11:00 AM 3/11/2019

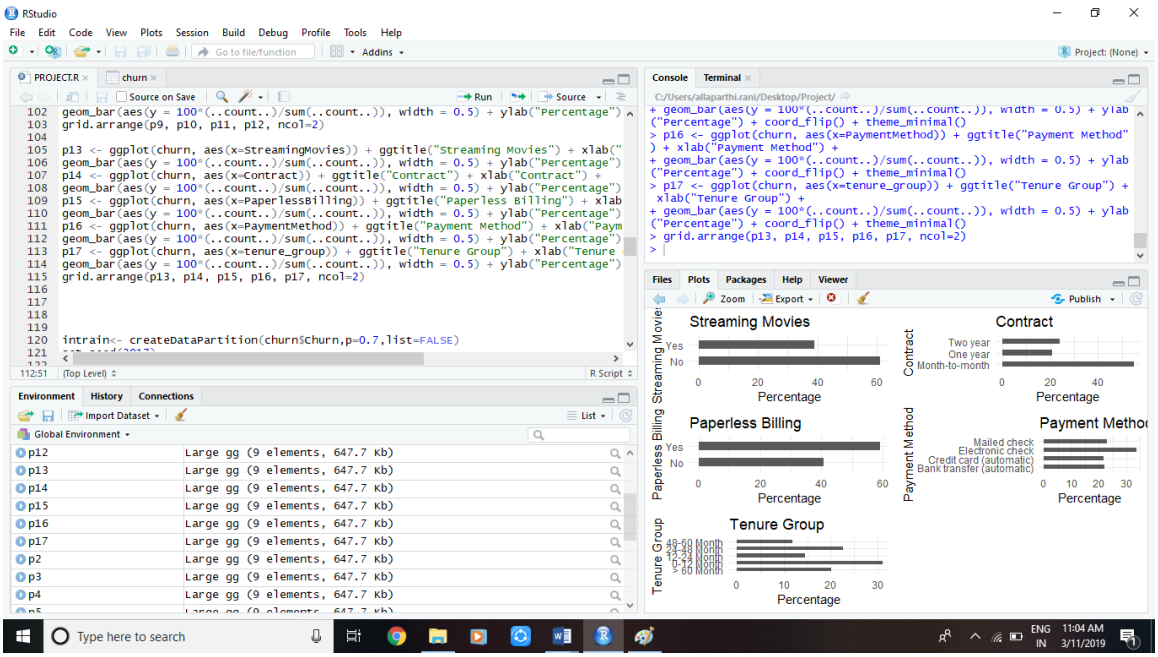

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
Source on Save
Run
Source
1 library(plyr)
2 library(corrplot)
3 library(ggplot2)
4 library(gridExtra)
5 library(ggthemes)
6 library(caret)
7 library(MASS)
8 library(randomForest)
9 library(party)
10
11 getwd()
12 churn<- read.csv(file.choose(),header=T)
13 View(churn)
14 str(churn)
15
16 sapply(churn, function(x) sum(is.na(x)))
17
18 churn <- churn[complete.cases(churn),]
19
20
14:11 (Top Level) R Script
Environment History Connections
Import Dataset List
Global Environment
Data
churn 7032 obs. of 21 variables
Console Terminal
C:/Users/allaparthi.rani/Desktop/Project/
$ OnlineBackup : Factor w/ 3 levels "No","No internet service",..:
3 1 3 1 1 1 3 1 1 3 ...
$ DeviceProtection: Factor w/ 3 levels "No","No internet service",..:
1 3 1 3 1 1 3 1 1 3 ...
$ TechSupport : Factor w/ 3 levels "No","No internet service",..:
1 1 1 3 1 1 1 3 1 3 ...
$ StreamingTV : Factor w/ 3 levels "No","No internet service",..:
1 1 1 1 1 3 3 1 3 1 ...
$ StreamingMovies : Factor w/ 3 levels "No","No internet service",..:
1 1 1 1 1 3 1 1 3 1 ...
$ Contract : Factor w/ 3 levels "Month-to-month",..: 1 2 1 2 1
1 1 1 2 ...
$ PaperlessBilling: Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 2 1
...
$ PaymentMethod : Factor w/ 4 levels "Bank transfer (automatic)",..:
3 4 4 1 3 3 2 4 3 1 ...
$ MonthlyCharges : num 29.9 57 53.9 42.3 70.7 ...
$ TotalCharges : num 29.9 1889.5 108.2 1840.8 151.7 ...
$ Churn : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 2 1 1 2 1
...
>
> sapply(churn, function(x) sum(is.na(x)))
customerID gender SeniorCitizen Partner
0 0 0 0
Dependents tenure PhoneService MultipleLines
0 0 0 0
InternetService OnlineSecurity OnlineBackup DeviceProtection
0 0 0 0
TechSupport StreamingTV StreamingMovies Contract
0 0 0 0
PaperlessBilling PaymentMethod MonthlyCharges TotalCharges
0 0 0 11
Churn
0
>
> churn <- churn[complete.cases(churn),]
>
Files Plots Packages Help Viewer
Type here to search
ENG IN 11:00 AM 3/11/2019

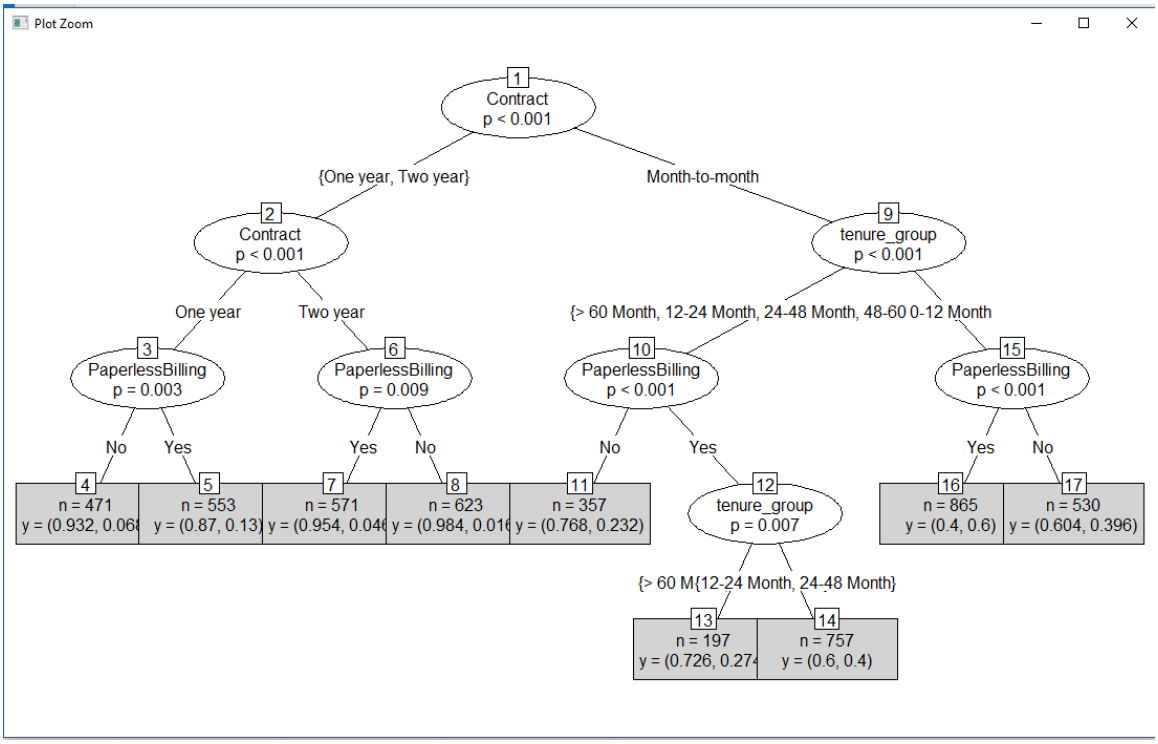

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
churn$TotalCharges <- NULL
Bar plots of categorical variables
p1 <- ggplot(churn, aes(x=gender)) + ggtitle("Gender") + xlab("Gender") +
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p2 <- ggplot(churn, aes(x=SeniorCitizen)) + ggtitle("Senior Citizen") + xlab("Senio
p3 <- ggplot(churn, aes(x=Partner)) + ggtitle("Partner") + xlab("Partner") +
p4 <- ggplot(churn, aes(x=Dependents)) + ggtitle("Dependents") + xlab("Dependents")
grid.arrange(p1, p2, p3, p4, ncol=2)
p5 <- ggplot(churn, aes(x=PhoneService)) + ggtitle("Phone Service") + xlab("Phone S
> churn$customerID <- NULL
> churn$tenure <- NULL
> ##Exploratory data analysis and feature selection
> numeric.var <- sapply(churn, is.numeric) ## Find numerical variables
> corr.matrix <- cor(churn[,numeric.var]) ## Calculate the correlation matrix
> corrplot(corr.matrix, main="\n\nCorrelation Plot for Numeric Variables", method="number")
Correlation Plot for Numeric Variables
MonthlyCharges
TotalCharges
1
0.65
0.65
1
Data
churn 7032 obs. of 19 variables
corr.matrix num [1:2, 1:2] 1 0.651 0.651 1
Values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L
numeric.var Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
Functions
group_tenure function (tenure)

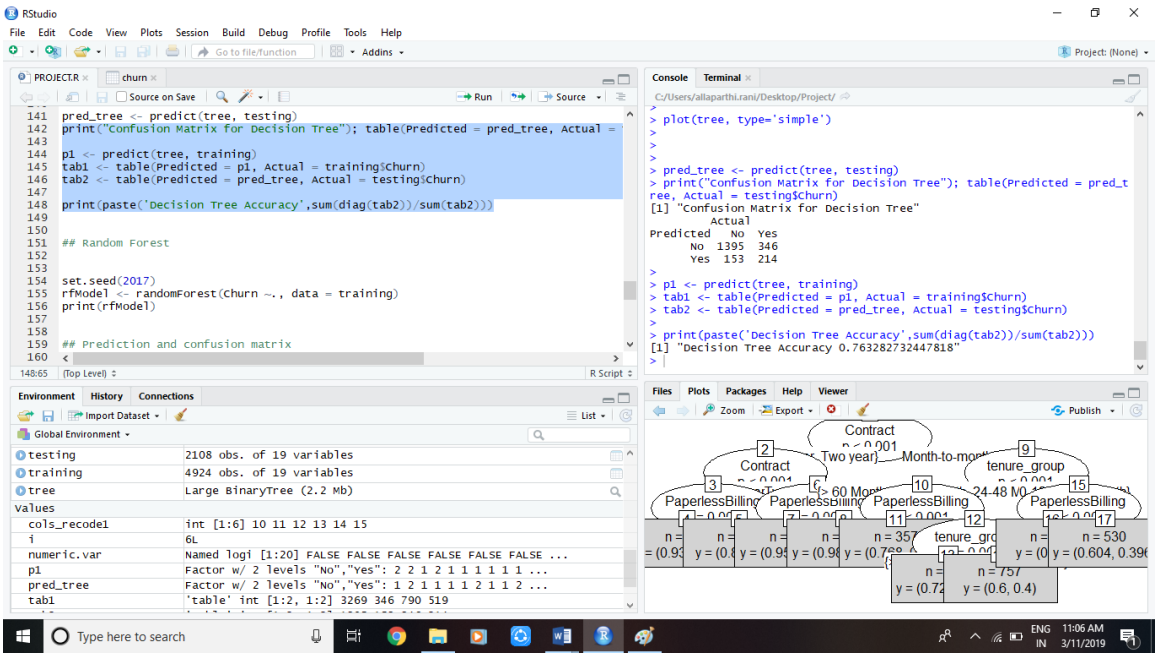

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
p1 <- ggplot(churn, aes(x=gender)) + ggtitle("Gender") + xlab("Gender") +
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p2 <- ggplot(churn, aes(x=SeniorCitizen)) + ggtitle("Senior Citizen") + xlab("Senio
p3 <- ggplot(churn, aes(x=Partner)) + ggtitle("Partner") + xlab("Partner") +
p4 <- ggplot(churn, aes(x=Dependents)) + ggtitle("Dependents") + xlab("Dependents")
grid.arrange(p1, p2, p3, p4, ncol=2)
p5 <- ggplot(churn, aes(x=PhoneService)) + ggtitle("Phone Service") + xlab("Phone S
p6 <- ggplot(churn, aes(x=MultipleLines)) + ggtitle("Multiple Lines") + xlab("Multi
p7 <- ggplot(churn, aes(x=InternetService)) + ggtitle("Internet Service") + xlab("I
p8 <- ggplot(churn, aes(x=OnlineSecurity)) + ggtitle("Online Security") + xlab("Onl
+ geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
> p3 <- ggplot(churn, aes(x=Partner)) + ggtitle("Partner") + xlab("Partner") +
> p4 <- ggplot(churn, aes(x=Dependents)) + ggtitle("Dependents") + xlab("Dependents") +
> grid.arrange(p1, p2, p3, p4, ncol=2)
Gender
Male
Female
Percentage
Senior Citizen
Yes
No
Partner
Dependents
Data
churn 7032 obs. of 19 variables
corr.matrix num [1:2, 1:2] 1 0.651 0.651 1
p1 Large gg (9 elements, 647.7 Kb)
p2 Large gg (9 elements, 647.7 Kb)
p3 Large gg (9 elements, 647.7 Kb)
p4 Large gg (9 elements, 647.7 Kb)
Values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L

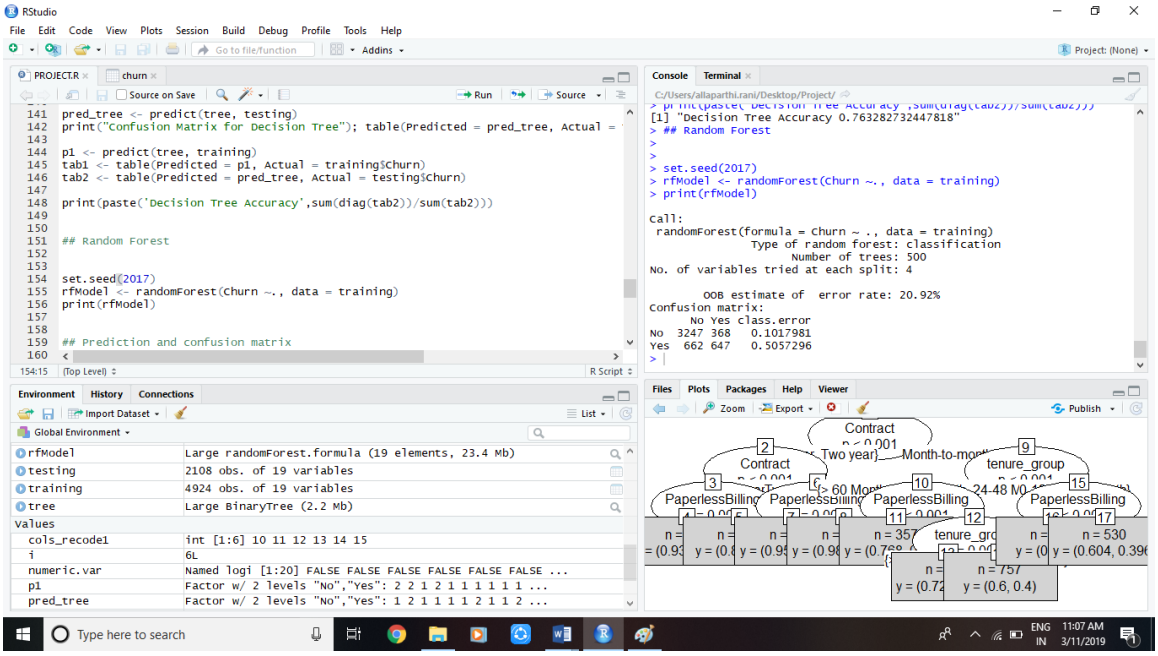

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
grid.arrange(p1, p2, p3, p4, ncol=2)
p5 <- ggplot(churn, aes(x=PhoneService)) + ggtitle("Phone Service") + xlab("Phone S
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p6 <- ggplot(churn, aes(x=MultipleLines)) + ggtitle("Multiple Lines") + xlab("Multi
p7 <- ggplot(churn, aes(x=InternetService)) + ggtitle("Internet Service") + xlab("I
p8 <- ggplot(churn, aes(x=OnlineSecurity)) + ggtitle("Online Security") + xlab("Onl
grid.arrange(p5, p6, p7, p8, ncol=2)
p9 <- ggplot(churn, aes(x=OnlineBackup)) + ggtitle("Online Backup") + xlab("Online
p10 <- ggplot(churn, aes(x=DeviceProtection)) + ggtitle("Device Protection") + xlab
p11 <- ggplot(churn, aes(x=TechSupport)) + ggtitle("Tech Support") + xlab("Tech Sup
p12 <- ggplot(churn, aes(x=StreamingTV)) + ggtitle("Streaming TV") + xlab("Streamin
Console Terminal
> p7 <- ggplot(churn, aes(x=InternetService)) + ggtitle("Internet Service") + xlab("Internet Service") +
> p8 <- ggplot(churn, aes(x=OnlineSecurity)) + ggtitle("Online Security") + xlab("Online Security") +
> grid.arrange(p5, p6, p7, p8, ncol=2)
Phone Service
Multiple Lines
Internet Service
Online Security
Percentage
corr.matrix num [1:2, 1:2] 1 0.651 0.651 1
p1 Large gg (9 elements, 647.7 Kb)
p2 Large gg (9 elements, 647.7 Kb)
p3 Large gg (9 elements, 647.7 Kb)
p4 Large gg (9 elements, 647.7 Kb)
p5 Large gg (9 elements, 647.7 Kb)
p6 Large gg (9 elements, 647.7 Kb)
p7 Large gg (9 elements, 647.7 Kb)
p8 Large gg (9 elements, 647.7 Kb)
Type here to search
ENG IN 11:02 AM 3/11/2019

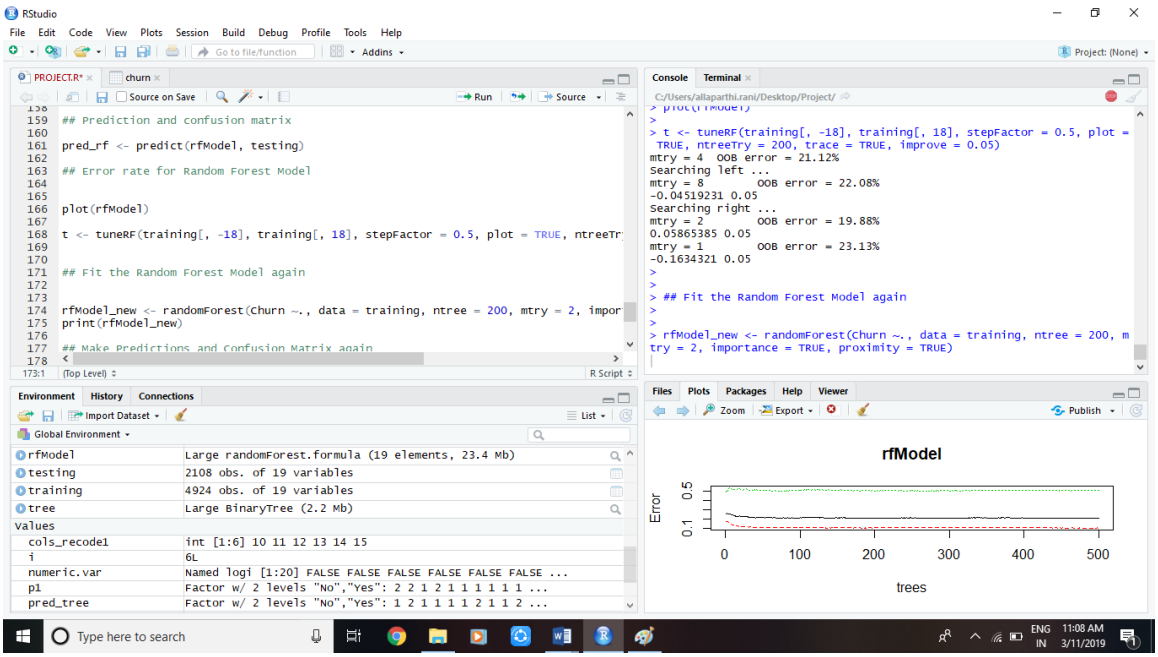

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p10 <- ggplot(churn, aes(x=DeviceProtection)) + ggtitle("Device Protection") + xlab
p11 <- ggplot(churn, aes(x=TechSupport)) + ggtitle("Tech Support") + xlab("Tech Sup
p12 <- ggplot(churn, aes(x=StreamingTV)) + ggtitle("Streaming TV") + xlab("Streamin
grid.arrange(p9, p10, p11, p12, ncol=2)
p13 <- ggplot(churn, aes(x=StreamingMovies)) + ggtitle("Streaming Movies") + xlab("
p14 <- ggplot(churn, aes(x=Contract)) + ggtitle("Contract") + xlab("Contract") +
p15 <- ggplot(churn, aes(x=PaperlessBilling)) + ggtitle("Paperless Billing") + xlab
p16 <- ggplot(churn, aes(x=PaymentMethod)) + ggtitle("Payment Method") + xlab("Paym
p17 <- ggplot(churn, aes(x=tenure_group)) + ggtitle("Tenure Group") + xlab("Tenure
Console Terminal
> p11 <- ggplot(churn, aes(x=TechSupport)) + ggtitle("Tech Support") + xlab("Tech Support") +
> p12 <- ggplot(churn, aes(x=StreamingTV)) + ggtitle("Streaming TV") + xlab("Streaming TV") +
> grid.arrange(p9, p10, p11, p12, ncol=2)
Online Backup
Device Protection
Tech Support
Streaming TV
Percentage
p12 Large gg (9 elements, 647.7 Kb)
p2 Large gg (9 elements, 647.7 Kb)
p3 Large gg (9 elements, 647.7 Kb)
p4 Large gg (9 elements, 647.7 Kb)
p5 Large gg (9 elements, 647.7 Kb)
p6 Large gg (9 elements, 647.7 Kb)
p7 Large gg (9 elements, 647.7 Kb)
p8 Large gg (9 elements, 647.7 Kb)
p9 Large gg (9 elements, 647.7 Kb)
Type here to search
ENG IN 11:03 AM 3/11/2019

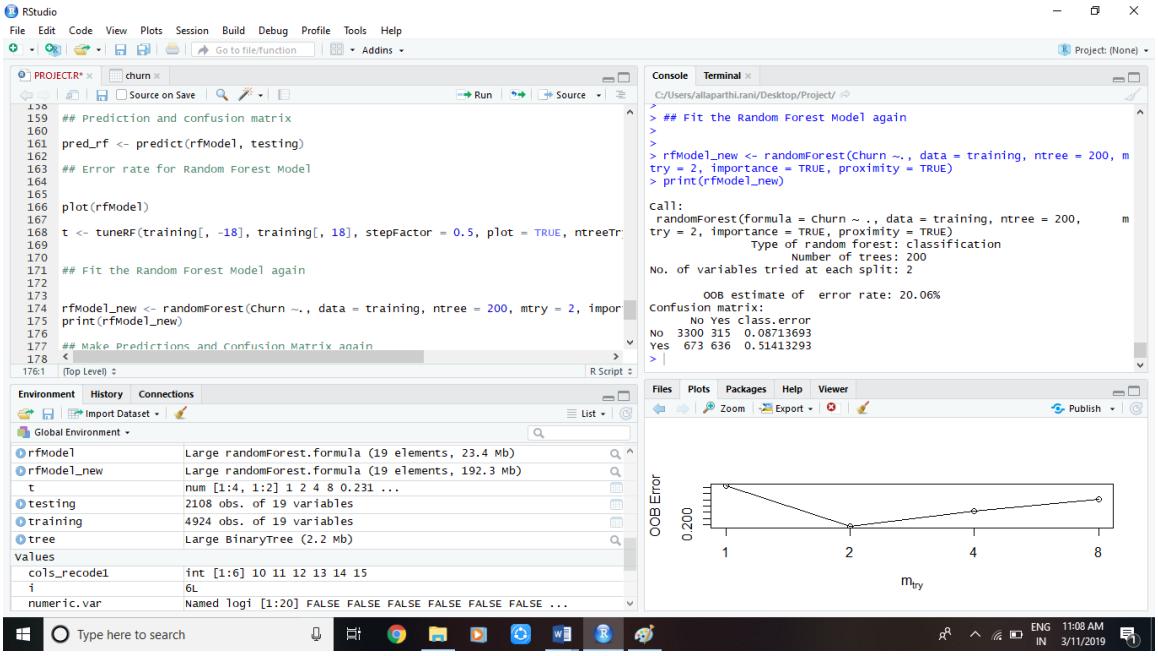

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
Source on Save
Run
Source
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
grid.arrange(p9, p10, p11, p12, ncol=2)
p13 <- ggplot(churn, aes(x=StreamingMovies)) + ggtitle("Streaming Movies") + xlab("
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p14 <- ggplot(churn, aes(x=Contract)) + ggtitle("Contract") + xlab("Contract") +
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p15 <- ggplot(churn, aes(x=PaperlessBilling)) + ggtitle("Paperless Billing") + xlab
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p16 <- ggplot(churn, aes(x=PaymentMethod)) + ggtitle("Payment Method") + xlab("Paym
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
p17 <- ggplot(churn, aes(x=tenure_group)) + ggtitle("Tenure Group") + xlab("Tenure
geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage")
grid.arrange(p13, p14, p15, p16, p17, ncol=2)
intrain<- createDataPartition(churn$Churn,p=0.7,list=FALSE)
(Top Level)
R Script
Console
Terminal
C:/Users/allaparthi.rani/Desktop/Project/
> p16 <- ggplot(churn, aes(x=PaymentMethod)) + ggtitle("Payment Method") + xlab("Payment Method") +
> p17 <- ggplot(churn, aes(x=tenure_group)) + ggtitle("Tenure Group") + xlab("Tenure Group") +
> grid.arrange(p13, p14, p15, p16, p17, ncol=2)
Environment
History
Connections
Import Dataset
List
Global Environment
p12 Large gg (9 elements, 647.7 Kb)
p13 Large gg (9 elements, 647.7 Kb)
p14 Large gg (9 elements, 647.7 Kb)
p15 Large gg (9 elements, 647.7 Kb)
p16 Large gg (9 elements, 647.7 Kb)
p17 Large gg (9 elements, 647.7 Kb)
p2 Large gg (9 elements, 647.7 Kb)
p3 Large gg (9 elements, 647.7 Kb)
p4 Large gg (9 elements, 647.7 Kb)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Streaming Movies
Yes
No
Percentage
Contract
Two year
One year
Month-to-month
Paperless Billing
Payment Method
Mailed check
Electronic check
Credit card (automatic)
Bank transfer (automatic)
Tenure Group
> 60 Month
Type here to search
ENG
IN
11:04 AM
3/11/2019

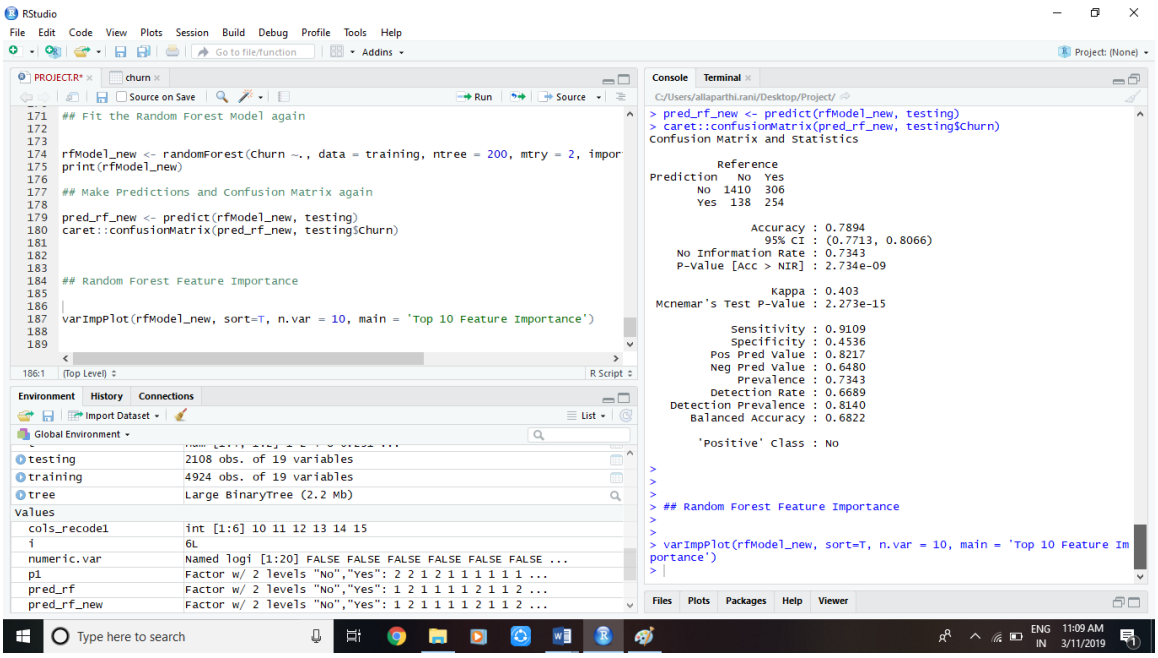

Plot Zoom
1
Contract
p < 0.001
{One year, Two year}
Month-to-month
2
Contract
p < 0.001
9
tenure_group
p < 0.001
One year
Two year
{> 60 Month, 12-24 Month, 24-48 Month, 48-60 0-12 Month
3
PaperlessBilling
p = 0.003
6
PaperlessBilling
p = 0.009
10
PaperlessBilling
p < 0.001
15
PaperlessBilling
p < 0.001
No
Yes
4
n = 471
y = (0.932, 0.068
5
n = 553
y = (0.87, 0.13)
7
n = 571
y = (0.954, 0.046
8
n = 623
y = (0.984, 0.016
11
n = 357
y = (0.768, 0.232)
12
tenure_group
p = 0.007
16
n = 865
y = (0.4, 0.6)
17
n = 530
y = (0.604, 0.396)
{> 60 M{12-24 Month, 24-48 Month}
13
n = 197
y = (0.726, 0.274
14
n = 757
y = (0.6, 0.4)

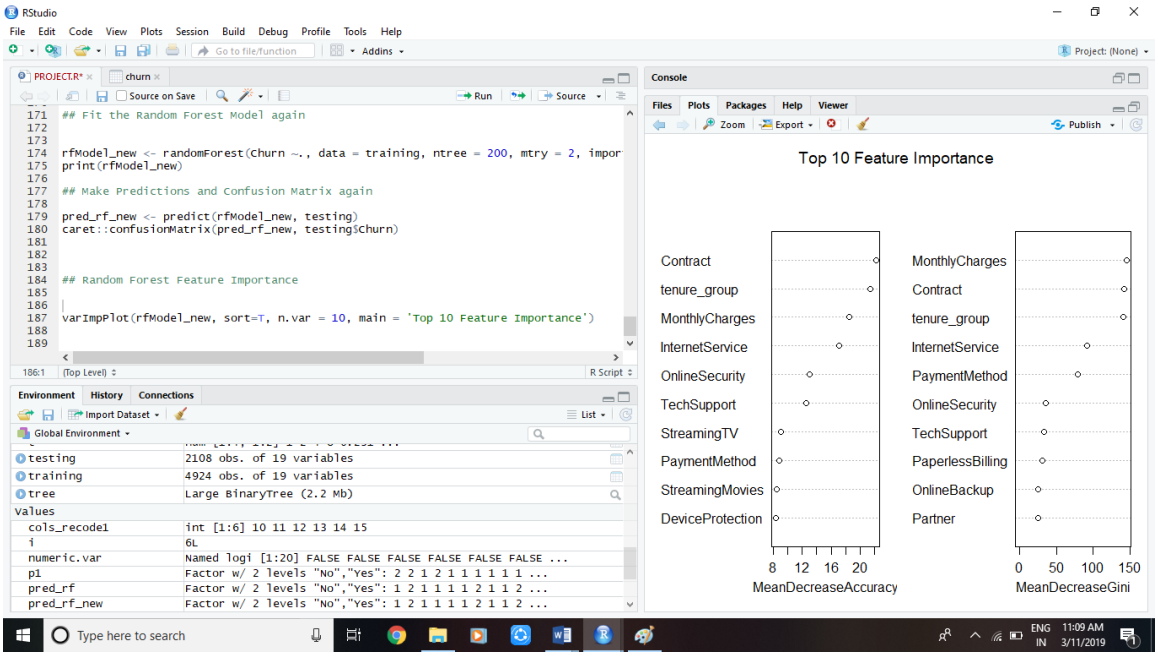

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
Source on Save
Run
Source
pred_tree <- predict(tree, testing)
print("Confusion Matrix for Decision Tree"); table(Predicted = pred_tree, Actual =
p1 <- predict(tree, training)
tab1 <- table(Predicted = p1, Actual = training$Churn)
tab2 <- table(Predicted = pred_tree, Actual = testing$Churn)
print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))
Random Forest
set.seed(2017)
rfModel <- randomForest(Churn ~., data = training)
print(rfModel)
Prediction and confusion matrix
(Top Level)
R Script
Console
Terminal
C:/Users/allaparthi.rani/Desktop/Project/
> plot(tree, type='simple')
> pred_tree <- predict(tree, testing)
> print("Confusion Matrix for Decision Tree"); table(Predicted = pred_tree, Actual = testing$Churn)
[1] "Confusion Matrix for Decision Tree"
Actual
Predicted No Yes
No 1395 346
Yes 153 214
> p1 <- predict(tree, training)
> tab1 <- table(Predicted = p1, Actual = training$Churn)
> tab2 <- table(Predicted = pred_tree, Actual = testing$Churn)
> print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))
[1] "Decision Tree Accuracy 0.763282732447818"
Environment
History
Connections
Import Dataset
Global Environment
testing 2108 obs. of 19 variables
training 4924 obs. of 19 variables
tree Large BinaryTree (2.2 Mb)
Values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L
numeric.var Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
p1 Factor w/ 2 levels "No","Yes": 2 2 1 2 1 1 1 1 1 1 ...
pred_tree Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 2 1 1 2 ...
tab1 'table' int [1:2, 1:2] 3269 346 790 519
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Contract
Two year
Month-to-mon
tenure_group
PaperlessBilling
n = 757
y = (0.6, 0.4)
n = 530
Type here to search
ENG
IN
11:06 AM
3/11/2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R
churn
Source on Save
Run
Source
pred_tree <- predict(tree, testing)
print("Confusion Matrix for Decision Tree"); table(Predicted = pred_tree, Actual =
p1 <- predict(tree, training)
tab1 <- table(Predicted = p1, Actual = training$Churn)
tab2 <- table(Predicted = pred_tree, Actual = testing$Churn)
print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))
Random Forest
set.seed(2017)
rfModel <- randomForest(Churn ~., data = training)
print(rfModel)
Prediction and confusion matrix
(Top Level)
R Script
Console
Terminal
C:/Users/allaparthi.rani/Desktop/Project/
[1] "Decision Tree Accuracy 0.763282732447818"
> ## Random Forest
> set.seed(2017)
> rfModel <- randomForest(Churn ~., data = training)
> print(rfModel)
Call:
randomForest(formula = Churn ~ ., data = training)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 4
OOB estimate of error rate: 20.92%
Confusion matrix:
No Yes class.error
No 3247 368 0.1017981
Yes 662 647 0.5057296
Environment
History
Connections
Import Dataset
Global Environment
rfModel Large randomForest.formula (19 elements, 23.4 Mb)
testing 2108 obs. of 19 variables
training 4924 obs. of 19 variables
tree Large BinaryTree (2.2 Mb)
Values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L
numeric.var Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
p1 Factor w/ 2 levels "No","Yes": 2 2 1 2 1 1 1 1 1 1 ...
pred_tree Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 2 1 1 2 ...
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Contract
Two year
Month-to-mon
tenure_group
PaperlessBilling
n = 757
y = (0.6, 0.4)
n = 530
Type here to search
ENG
IN
11:07 AM
3/11/2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R*
churn
Source on Save
Run
Source
Prediction and confusion matrix
pred_rf <- predict(rfModel, testing)
Error rate for Random Forest Model
plot(rfModel)
t <- tuneRF(training[, -18], training[, 18], stepFactor = 0.5, plot = TRUE, ntreeTr
Fit the Random Forest Model again
rfModel_new <- randomForest(Churn ~., data = training, ntree = 200, mtry = 2, impor
print(rfModel_new)
Make Predictions and Confusion Matrix again
(Top Level)
R Script
Console
Terminal
C:/Users/allaparthi.rani/Desktop/Project/
> t <- tuneRF(training[, -18], training[, 18], stepFactor = 0.5, plot =
TRUE, ntreeTry = 200, trace = TRUE, improve = 0.05)
mtry = 4 OOB error = 21.12%
Searching left ...
mtry = 8 OOB error = 22.08%
-0.04519231 0.05
Searching right ...
mtry = 2 OOB error = 19.88%
0.05865385 0.05
mtry = 1 OOB error = 23.13%
-0.1634321 0.05
> ## Fit the Random Forest Model again
> rfModel_new <- randomForest(Churn ~., data = training, ntree = 200, m
try = 2, importance = TRUE, proximity = TRUE)
Environment History Connections
Import Dataset
List
Global Environment
rfModel Large randomForest.formula (19 elements, 23.4 Mb)
testing 2108 obs. of 19 variables
training 4924 obs. of 19 variables
tree Large BinaryTree (2.2 Mb)
Values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L
numeric.var Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
p1 Factor w/ 2 levels "No","Yes": 2 2 1 2 1 1 1 1 1 1 ...
pred_tree Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 2 1 1 2 ...
Files Plots Packages Help Viewer
Zoom
Export
Publish
rfModel
Error
0.1
0.5
0
100
200
300
400
500
trees
Type here to search
ENG
IN
11:08 AM
3/11/2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R*
churn
Source on Save
Run
Source
Prediction and confusion matrix
pred_rf <- predict(rfModel, testing)
Error rate for Random Forest Model
plot(rfModel)
t <- tuneRF(training[, -18], training[, 18], stepFactor = 0.5, plot = TRUE, ntreeTr
Fit the Random Forest Model again
rfModel_new <- randomForest(Churn ~., data = training, ntree = 200, mtry = 2, impor
print(rfModel_new)
Make Predictions and Confusion Matrix again
(Top Level)
R Script
Console
Terminal
C:/Users/allaparthi.rani/Desktop/Project/
> ## Fit the Random Forest Model again
> rfModel_new <- randomForest(Churn ~., data = training, ntree = 200, m
try = 2, importance = TRUE, proximity = TRUE)
> print(rfModel_new)
Call:
randomForest(formula = Churn ~ ., data = training, ntree = 200, m
try = 2, importance = TRUE, proximity = TRUE)
Type of random forest: classification
Number of trees: 200
No. of variables tried at each split: 2
OOB estimate of error rate: 20.06%
Confusion matrix:
No Yes class.error
No 3300 315 0.08713693
Yes 673 636 0.51413293
Environment History Connections
Import Dataset
List
Global Environment
rfModel Large randomForest.formula (19 elements, 23.4 Mb)
rfModel_new Large randomForest.formula (19 elements, 192.3 Mb)
t num [1:4, 1:2] 1 2 4 8 0.231 ...
testing 2108 obs. of 19 variables
training 4924 obs. of 19 variables
tree Large BinaryTree (2.2 Mb)
Values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L
numeric.var Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
Files Plots Packages Help Viewer
Zoom
Export
Publish
OOB Error
0.200
1
2
4
8
mtry
Type here to search
ENG
IN
11:08 AM
3/11/2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
PROJECT.R*
churn
Source on Save
Run
Source
171 ## Fit the Random Forest Model again
172
173
174 rfModel_new <- randomForest(Churn ~., data = training, ntree = 200, mtry = 2, impor
175 print(rfModel_new)
176
177 ## Make Predictions and Confusion Matrix again
178
179 pred_rf_new <- predict(rfModel_new, testing)
180 caret::confusionMatrix(pred_rf_new, testing$Churn)
181
182
183
184 ## Random Forest Feature Importance
185
186
187 varImpPlot(rfModel_new, sort=T, n.var = 10, main = 'Top 10 Feature Importance')
188
189
186:1 (Top Level) R Script
Environment History Connections
Import Dataset List
Global Environment
testing 2108 obs. of 19 variables
training 4924 obs. of 19 variables
tree Large BinaryTree (2.2 Mb)
values
cols_recode1 int [1:6] 10 11 12 13 14 15
i 6L
numeric.var Named logi [1:20] FALSE FALSE FALSE FALSE FALSE FALSE ...
p1 Factor w/ 2 levels "No","Yes": 2 2 1 2 1 1 1 1 1 1 ...
pred_rf Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 2 1 1 2 ...
pred_rf_new Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 2 1 1 2 ...
Console Terminal
C:/Users/allaparthi.rani/Desktop/Project/
> pred_rf_new <- predict(rfModel_new, testing)
> caret::confusionMatrix(pred_rf_new, testing$Churn)
Confusion Matrix and Statistics
Reference
Prediction No Yes
No 1410 306
Yes 138 254
Accuracy : 0.7894
95% CI : (0.7713, 0.8066)
No Information Rate : 0.7343
P-Value [Acc > NIR] : 2.734e-09
Kappa : 0.403
Mcnemar's Test P-Value : 2.273e-15
Sensitivity : 0.9109
Specificity : 0.4536
Pos Pred Value : 0.8217
Neg Pred Value : 0.6480
Prevalence : 0.7343
Detection Rate : 0.6689
Detection Prevalence : 0.8140
Balanced Accuracy : 0.6822
'Positive' Class : No
> ## Random Forest Feature Importance
> varImpPlot(rfModel_new, sort=T, n.var = 10, main = 'Top 10 Feature Importance')
Files Plots Packages Help Viewer
Type here to search
ENG IN 11:09 AM 3/11/2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console
Files Plots Packages Help Viewer
Zoom Export Publish
Top 10 Feature Importance
Contract
tenure_group
MonthlyCharges
InternetService
OnlineSecurity
TechSupport
StreamingTV
PaymentMethod
StreamingMovies
DeviceProtection
8 12 16 20
MeanDecreaseAccuracy
MonthlyCharges
Contract
tenure_group
InternetService
PaymentMethod
OnlineSecurity
TechSupport
PaperlessBilling
OnlineBackup
Partner
0 50 100 150
MeanDecreaseGini

7.CONCLUSION

From the above example, we can see that Random Forest performed better than Decision Tree for customer churn analysis for this particular dataset.

Throughout the analysis, we have learned several important things:

1. Features such as tenure_group, Contract, PaperlessBilling, MonthlyCharges and InternetService appear to play a role in customer churn.
2. There does not seem to be a relationship between gender and churn.
3. Customers in a month-to-month contract, with PaperlessBilling and are within 12 months tenure, are more likely to churn; On the other hand, customers with one or two year contract, with longer than 12 months tenure, that are not using PaperlessBilling, are less likely to churn.

8.REFERENCES

[1] R in Action, Data Analysis and graphics with R,Robert L Kabacoff, Manning Publisher

[2]https://datascience.stackexchange.com/questions/6015/assumptions-limitations-of-random-forest-models

[3] D.J. Hand, Construction and Assessment of Classification Rules, John Wiley and Sons, New York, 1997.

[4] D. Hand, Discrimination and Classification, John Wiley, 1981.

[5] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.