Human Resource Analytics

Project ID: 12069

B.Tech.Final Project Report

submitted for fulfillment of

the requirements for the

Degree of Bachelor of Technology

Under Biju Pattnaik University of Technology

Submitted By

Ashis Sahu	**ROLL NO. 201411005**
Kritika Shah	**ROLL NO. 201449401**

2017 - 2018

Under the guidance of

Mr. Shom Prasad Das

NATIONAL INSTITUTE OF SCIENCE &TECHNOLOGY

Palur Hills, Berhampur, Odisha – 761008, India

ABSTRACT

Human Resource Analytics is a business analytics project, where we need to design model(s), which shall predict when an experienced employee will leave the organization. Predictive data analytics is in its essence of technology that learns from existing data and it uses this to forecast individual behavior. This means that predictions are very specific. Predictive data analytics would be used to predict when an experienced employee will leave the organization. The data from the organization uses historic data of thousands of employees in the past to predict whether an experienced employee will leave the organization or not. Predictive analytics involve a set of various statistical (data mining) techniques used to predict uncertain outcomes.

Based on the data we could see why employees would leave the organization. Pay scale, promotion and better performance ratings where, negatively related to employees leaving the organization. For instance, when someone received a promotion but did not get a substantial raise, this person would still be much more likely to quit. So, there must be various reasons due to which an employee is likely to leave a company such as Employee satisfaction level, Last evaluation, Number of projects, Average monthly hours, Time spent at the company, whether they have had a work accident, whether they have had a promotion in the last 5 years, Sales, Salary. High turnover generally leads to high recruitment costs and lost revenue due to productivity loss and onboarding. Additionally, leaving employees take their knowledge and network with them, and sometimes even customer.

ACKNOWLEDGEMENT

We express our sincere thanks to my esteemed project faculty advisor, **Mr. Shom Prasad Das,** for his valuable guidance in carrying out this seminar with encouragement, enlightenment and cooperation with a deep sense of gratitude. I would also like to thank my project coordinator.

We give our sincere thanks to **Dr. Sandipan Mallik,** Project coordinator, National Institute of Science and Technology, for allowing me to present my project report.

We acknowledge with immense pleasure the sustained interest, encouraging attitude and constant inspiration provided by **Prof. Sangram Mudali** (Director) and **Prof. Geetika Mudali** (Placement Director) N.I.S.T. Their continuous drive for better quality in everything that happens at N.I.S.T. and selfless inspiration has always helped us to move ahead with renewed zeal at our lowest times.

Ashis Sahu

Kritika Shah

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

R is a powerful language used widely for data analysis and statistical computing. It was developed in early 90s. Since then, endless efforts have been made to improve R's user interface. The journey of R language from a rudimentary text editor to interactive R Studio and more recently Jupyter Notebooks has engaged many data science communities across the world.

This was possible only because of generous contributions by R users globally. Inclusion of powerful packages in R has made it more and more powerful with time. Packages such as dplyr, tidyr, readr, data.table, SparkR, ggplot2 have made data manipulation, visualization and computation much faster.

Data Manipulation:

R has a fantastic collection of packages for data manipulation. These packages allows you to do basic & advanced computations quickly. These packages are dplyr, plyr, tidyr, lubridate, stringr.

Modeling / Machine Learning:

For modeling, caret package in R is powerful enough to cater to every need for creating machine learning model. Packages algorithms wise such as randomForest, rpart, gbm,etc. There are also individual packages for each ML algorithm.

Exploratory Data Analysis in R

Data Exploration is a crucial stage of predictive model. This stage forms a concrete foundation for data manipulation.

Train Data: The predictive model is always built on train data set. An intuitive way to identify the train data is, that it always has the 'response variable' included.

Test Data: Once the model is built, it's accuracy is 'tested' on test data. This data always contains less number of observations than train data set. Also, it does not include 'response variable'.

```
#working directory
path<- "D:\Project"
#set working directory
setwd(path)
#Load Datasets
train<- read.csv("Hr_data")
#check dimesions( number of row & columns) in data set
>dim(train)
[1] 14999 10

>table(is.na(train))
FALSE
```

Machine Learning:

R is a software for statistical computing. Good thing, R has enough provisions to implement machine learning algorithms in a fast and simple manner.

Here are some benefits I found after using R:

1. The style of coding is quite easy.
2. It’s open source. No need to pay any subscription charges.
3. Availability of instant access to over 7800 packages customized for various computation tasks.
4. The community support is overwhelming. There are numerous forums to help you out.
5. Get high performance computing experience (require packages)
6. One of highly sought skill by analytics and data science companies.

2. METHODOLOGY

Human Resource Analytics to design models, which shall predict when an experienced employee will leave the organization. The main objective of the project is to find the reasons and tentative timeframe when employees will leave the organization.

Human Resource Analytics to design model(s), which shall predict when an experienced employee will leave the organization. Predictive data analytics is in its essence of technology that learns from existing data and it uses this to forecast individual behavior. This means that predictions are very specific. Predictive data analytics would be used to predict the when an experienced employee will leave the organization. The data from the organization uses historic data of thousands of employees in the past to predict whether an experienced employee will leave the organization or not. Predictive analytics involve a set of various statistical (data mining) techniques used to predict uncertain outcomes.

Based on the data we could see why employees would leave the organization. Pay scale, promotions and better performance ratings where negatively related to employees leaving the organization. For instance, when someone received a promotion but did not get a substantial raise, this person would still be much more likely to quit. So, there must be various reasons due to which an employee is likely to leave a company such as Employee satisfaction level, Last evaluation, Number of projects, Average monthly hours, Time spent at the company, Whether they have had a work accident, Whether they have had a promotion in the last 5 years, Sales, Salary. High turnover generally leads to high recruitment costs and lost revenue due to productivity loss and onboarding. Additionally, leaving employees take their knowledge and network with them, and sometimes even customer.

1. The entire project will be done R Programming language.
2. Predictive models which includes Decision tree, linear regression model, SVM, etc.
3. SparkR will be used to work on HDFS, S3, Cassandra for big data analytics.

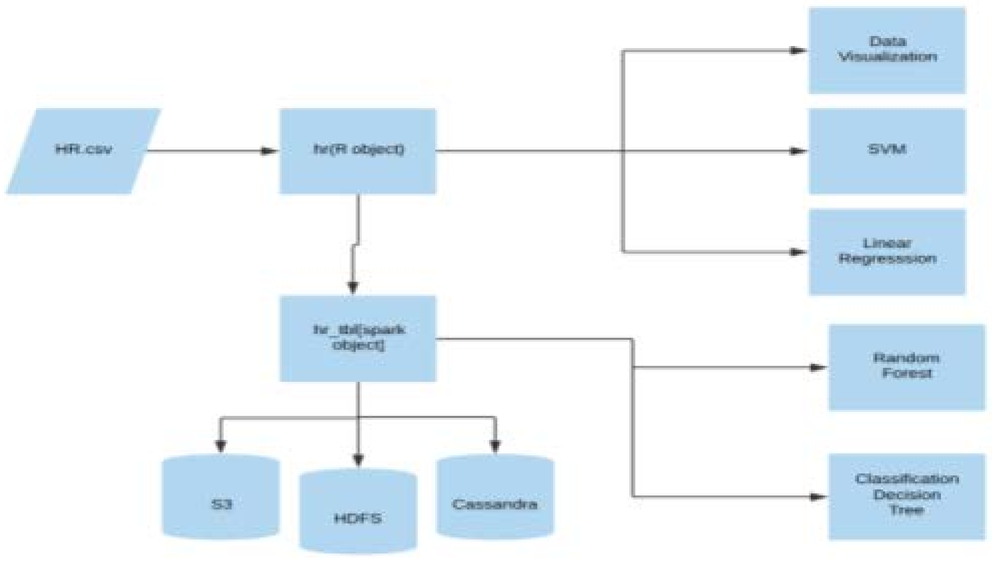


Figure 2.1 Module Representation

3. HOW TO INSTALL R / R STUDIO?

You could download and install the R version 3.4.3 from:
https://cran.r-project.org/bin/windows/base/

Then, install RStudio. It provides much better coding experience. For Windows users, R Studio is available for Windows Vista and above versions. Follow the steps below for installing R Studio:
Go to https://www.rstudio.com/products/rstudio/download/

In 'Installers for Supported Platforms' section, choose and click the R Studio installer based on your operating system. The download should begin as soon as you click.
Click Next..Next..Finish.
Download Complete.
To Start R Studio, click on its desktop icon or use 'search windows' to access the program.

3.1 Interface of R Studio

R Console: This area shows the output of code you run. Also, you can directly write codes in console. Code entered directly in R console cannot be traced later. This is where R script comes to use.

R Script: As the name suggest, here you get space to write codes. To run those codes, simply select the line(s) of code and press Ctrl + Enter. Alternatively, you can click on little 'Run' button location at top right corner of R Script.

R environment: This space displays the set of external elements added. This includes data set, variables, vectors, functions etc. To check if data has been loaded properly in R, always look at this area.

Graphical Output: This space display the graphs created during exploratory data analysis. Not just graphs, you could select packages, seek help with embedded R's official documentation.

3.2 How to install R Packages?

The sheer power of R lies in its incredible packages. In R, most data handling tasks can be performed in 2 ways: Using R packages and R base functions. In this tutorial, I'll also introduce you with the most handy and powerful R packages. To install a package, simply type:

```
install.packages("package name")
```

As a first time user, a pop might appear to select your CRAN mirror (country server), choose accordingly and press OK.

Everything you see or create in R is an object. A vector, matrix, data frame, even a variable is an object. R treats it that way. So, R has 5 basic classes of objects. This includes:

- Character
- Numeric
- Integer
- Complex
- Logical

Since these classes are self-explanatory by names, I wouldn't elaborate on that. These classes have attributes. Think of attributes as their 'identifier', a name or number which aptly identifies them. An object can have following attributes:

- names
- dimensions
- class
- length

4. DATA TYPES IN R

R has various type of ‘data types’ which includes vector (numeric, integer etc), matrices, data frames and list. Let’s understand them one by one.

Vector: As mentioned above, a vector contains object of same class. But, you can mix objects of different classes too. When objects of different classes are mixed in a list, coercion occurs. This effect causes the objects of different types to ‘convert’ into one class. For ex

```
qt<- c("Time", 24, "October", TRUE, 3.33)  #character
ab<- c(TRUE, 24) #numeric
 cd <- c(2.5, "May") #character
```

List: A list is a special type of vector which contain elements of different data types. For example:

```
my_list<- list(22, "ab", TRUE, 1 + 2i)
my_list[[3]]
 [1] TRUE
```

5. DATA VISUALIZATION

R has in built plotting commands as well. They are good to create simple graphs. But, becomes complex when it comes to creating advanced graphics. Hence, should install ggplot2.

R Programming offers a satisfactory set of inbuilt function and libraries (such as ggplot2, leaflet, lattice) to build visualizations and present data.

5.1 Data Visualization in R

Basic Visualization

1. Histogram
2. Bar / Line Chart
3. Box plot
4. Scatter plot

6. INTRODUCTION TO TREE

Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation. Unlike linear models, they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand (classification or regression).

Methods like decision trees, random forest, gradient boosting are being popularly used in all kinds of data science problems. Hence, for every analyst (fresher also), it’s important to learn these algorithms and use them for modeling.

Tree-Based Models:-

Recursive partitioning is a fundamental tool in data mining. It helps us explore the stucture of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome. This section briefly describes CART modeling, conditional inference trees, and random forests.

CART Modeling via rpart

Classification and regression trees (as described by Brieman, Freidman, Olshen, and Stone) can be generated through the rpart package. Detailed information on rpart is available in An Introduction to Recursive Partitioning Using the RPART Routines. The general steps are provided below followed by an examples.

C**ode:**

```
fit<- rpart(left ~ satisfaction_level +last_evaluation + number_project +
average_montly_hours + promotion_last_5years +  salary , method="class", data = hr)
plot(fit, uniform=TRUE)
text(fit, use.n=TRUE, all=TRUE)
```

7. UNDERSTANDING SparkR

SparkR is an easy way for transition from R to Spark. Dataframe API provides a very natural way to code. SparkR is an R package that provides a light-weight frontend to use Apache Spark from R. In Spark 2.2.0, SparkR provides a distributed data frame implementation that supports operations like selection, filtering, aggregation etc. (similar to R data frames, dplyr) but on large datasets. SparkR also supports distributed machine learning using MLlib.

Spark is a library of code that can be used to process data in parallel on a cluster. The basic idea of Spark is parallelism, meaning Spark breaks the data into pieces, sends the pieces to differnt computers for processing, then sends the results back and process the combination to get the final result. More specifically, the basic computing paradigm is: distribute a large data set on multiple nodes, map functions row by row, group data by a key, and then perform aggregate operations.

7.1 Basic concepts of Spark include: RDDs, transformation, action

1. **RDD**

RDD stands for resilient distributed datasets. It is the basic data structure defined by Spark so the data can be distributed among cluster nodes. RDD contains large information, we can apply actions to RDD to return values, and transformations to return new RDD.

2. RDD transformation

Commonly used RDD transformation include map, filter, reduce, and reduceByKey. rdd.map: map is like the map function in R, it applies a function to each element of RDD.

8. INSTALLING SPARK

8.1 INSTALLING SPARK IN UBUNTU

Step 1: Installing Java

Check to see if Java is already installed by typing:

java -version

If you see something like "The program 'java' can be found in the following packages…"

It probably means you actually need to install Java.

The easiest option for installing Java is using the version packaged with Ubuntu. Specifically, this will install OpenJDK 8, the latest and recommended version.

First, update the package index by in your terminal typing:

sudo apt-get update

After entering your password it will update some stuff.

Now you can install the JDK with the following command:

sudo apt-get install default-jdk

Then hit Enter and continue with "y".

Step 2: Install Scala

sudo apt-get install scala

Type scala into your terminal:

scala

You should see the scala REPL running. Test it with:

println("Hello World")

You can then quit the Scala REPL with

:q

Step 3: Install Spark

In terminal type:

sudo apt-get install git

Next, go to https://spark.apache.org/downloads.html and download a pre-built for Hadoop 2.7 version of Spark (preferably Spark 2.0 or later). Then download the .tgz file and remember where you save it on your computer.

Then in your terminal change directory to where you saved that .tgz file (or just move the file to your home folder), then use

```
tarxvf spark-2.0.2-bin-hadoop2.7.tgz
Then once its done extracting the Spark folder, use:
cd spark-2.0.2-bin-hadoop2.7.tgz
then use
cd bin
and then type
./spark-shell
```

9. Spark

9.1 Multi-platform Support

Apache Spark provides extended interoperability regarding its running platform or supported data structure. Spark supports applications running in –

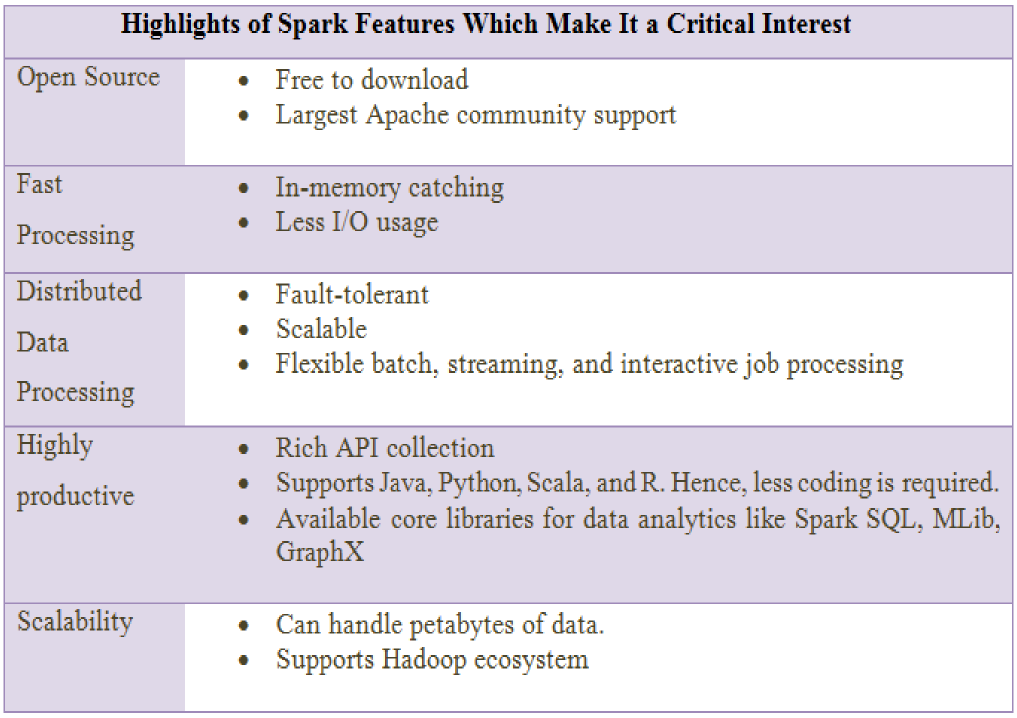
- cloud
- standalone cluster mode

Besides, that Spark can access varied data structures

- HBase
- Tachyon
- HDFS
- Cassandra

Spark can be deployed on

- A distributed framework such as YARN or Mesos
- Standalone server



Highlights of Spark Features Which Make It a Critical Interest

Open Source	• Free to download • Largest Apache community support
Fast Processing	• In-memory catching • Less I/O usage
Distributed Data Processing	• Fault-tolerant • Scalable • Flexible batch, streaming, and interactive job processing
Highly productive	• Rich API collection • Supports Java, Python, Scala, and R. Hence, less coding is required. • Available core libraries for data analytics like Spark SQL, MLib, GraphX
Scalability	• Can handle petabytes of data. • Supports Hadoop ecosystem

9.2 Important Features That Make Apache Spark a Better Choice

Apache Spark Data Streaming is Superior to Traditional Systems

Given below is a figure displaying why Spark streaming is superior to traditional systems:

Traditionally data streaming follows static task scheduling. On the other hand in Spark data streaming it is dynamic scheduling of tasks which make the overall processing faster.

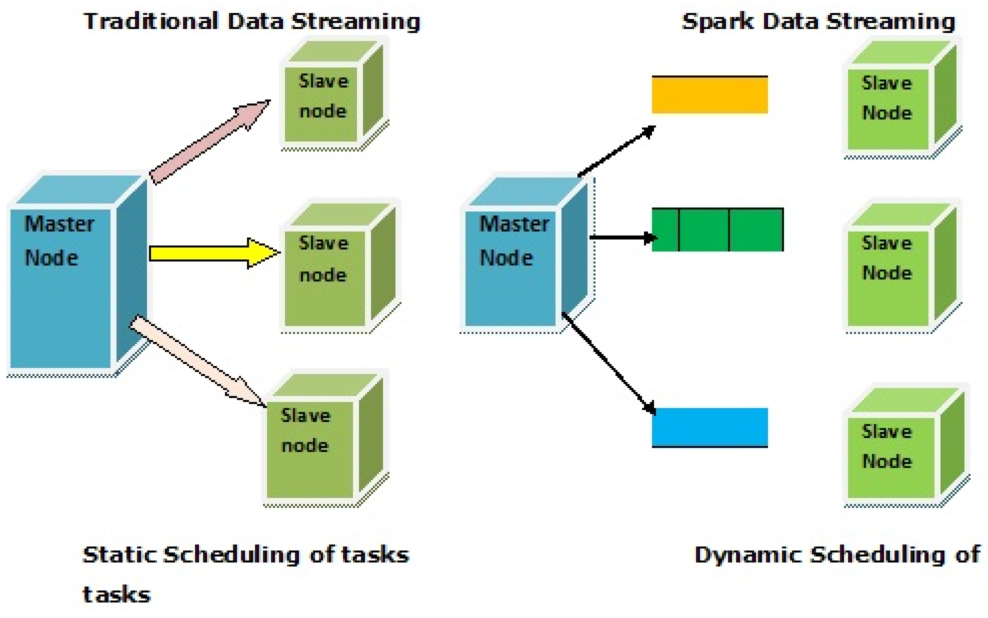


Figure 9.2.1 Traditional Data Streaming vs Spark Data Streaming

Apache Spark Structured Streaming for Infinite Data Streaming

Structured Streaming is the part of Spark 2.x which is a higher-level API. It helps in creating a more natural abstraction for writing applications. Using Structure Streaming, developers can create infinite streaming data frames as well datasets. With Structured Streaming, a user can efficiently handle message delivery.

Structured streaming facilitates users with the Catalyst query optimizer. Moreover, it can run in an interactive manner. As a result, it allows users to perform SQL queries for live streaming data.

Though structured streaming is still a new venture in Apache Spark, it is the future of data streaming.

Enterprise can Use Apache Spark on the Top of Existing Hadoop Structure

Apache Spark can be considered as an enhancement on the existing Hadoop infrastructure of a company for a speedy Big Data processing. One can easily deploy Apache Spark applications. It can run on existing Hadoop v1 and v2 cluster using an existing Hadoop Distributed File System(HDFS).

Though HDFS works as the primary data storage by Spark, it can work with other data sources compatible with Hadoop like HBase, Cassandra, etc.

Apache Spark: A New Dimension in Big data Industry for Data Scientists

Apache Spark shows an arena for the data scientists where they can build sophisticated data analysis models. The volume and type of data they can use for such analysis were beyond imagination before Spark.

Visualization is an integral part while dealing with data analysis for business purposes. This is more important for Big data analysis. Spark Core helps data scientists to create such reports and dashboards using Java, Python, R scripts, etc.

Spark's Machine Learning Capability may Help in Data Lake Flow

Recent organization trends towards data lake which is millions of pieces of data need predictive and automatic rules on accessibility. It not only enhances the business agility but also escapes manual interventions.

Apache Spark with its inbuilt machine learning algorithms can help in this data lake processing.

Spark Edges Over Other Open Source Projects in Enterprise Adoption

Among all the Apache open source projects, Apache Spark has become the most in-demand technology in Big data industry across multiple verticals. In the current market scenario, there is an increasing demand to support BI related workloads with Spark SQL and Hadoop.

Moreover, there is a strong open source community support for Spark which makes increasing adoption rate of Spark by the enterprises.

10. DOWNLOAD AND INSTALLATION SparkR

Spark can be downloaded directly from the link: http://spark.apache.org/downloads.html

After downloading, save the zipped file to a directory of choice, and then unzip the file. We can then set up Spark in R environment.

1. Set system environment by pointing R session to the installed SparkR.
2. Set library path and load Sparklyr library
3. Initialize Spark context and SQL context
4. Grouping and Aggregation

   Calculating the mean of sepal length of each species, and also the number of observation of each species.Sort the output data frame by the mean sepal length
5. SQL Queries

   We can register Spark Dataframe as sql table, which enables us to run SQL queries and return the output as a data frame.

   Register Spark Dataframe as a table.

Spark DataFrame

A Spark Data Frame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R, but with richer optimizations under the hood. Spark Data Frames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing local R data frames.

Starting Up: SparkSession

The entry point into SparkR is the SparkSession which connects your R program to a Spark cluster. You can create a SparkSession usingspark_connect(master="local")and master as local, which would start the master node of spark cluster.

```
sc<- spark_connect(master="local")
```

From Data Sources

SparkR supports operating on a variety of data sources through the Spark DataFrame interface. This section describes the general methods for loading and saving data using Data Sources. You can check the Spark SQL programming guide for more specific options that are available for the built-in data sources.

```
hr_spark<- copyto(sc, hr)
```

Spark session: disconnect

The following code is used:-

```
spark_disconnect(sc)
```

11. Predictive Models

11 .1 Support Vector Machine

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

The illustration below shows the basic idea behind Support Vector Machines. Here we see the original objects (left side of the schematic) mapped, i.e., rearranged, using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as mapping (transformation).

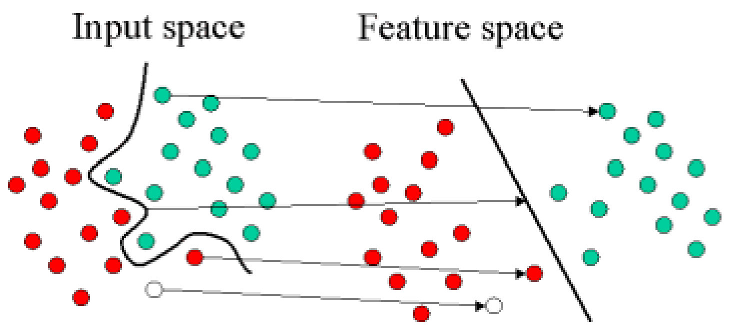


Figure 11.1 Support Vector Machine mechanism

Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1. To construct an optimal hyperplane, SVM employs an iterative training algorithm, which is used to minimize an error function. According to the form of the error function, SVM models can be classified into four distinct groups:

- Classification SVM Type 1 (also known as C-SVM classification)
- Classification SVM Type 2 (also known as nu-SVM classification)
- Regression SVM Type 1 (also known as epsilon-SVM regression)
- Regression SVM Type 2 (also known as nu-SVM regression)

Code:-

```
 library(e1071)
library(caret)
svm_model <- svm(left ~
average_montly_hours+last_evaluation+number_project+time_spend_company+satisfactio
n_level, data=train,type="C-classification")
summary(svm_model)
pred <- predict(svm_model,test)
hr_model_svm<-cbind(test,pred)
confusionMatrix_svm<- confusionMatrix(hr_model_svm$pred,hr_model_svm$left)
confusionMatrix_svm
```

11.2 Decision Tree

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

11.3 Linear Regression

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.[1] (This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.)

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Linear regression is similar to correlation in that the purpose is to measure to what extent there is a linear relationship between two variables. The major difference between the two is that correlation makes no distinction between independent and dependent variables while linear regression does. In particular, the purpose of linear regression is to "predict" the value of the dependent variable based upon the values of one or more independent variables.

The general mathematical equation for a linear regression is −

$$y = ax + b$$

Following is the description of the parameters used −

- **y** is the response variable.
- **x** is the predictor variable.
- **a** and **b** are constants which are called the coefficients.

Code:-

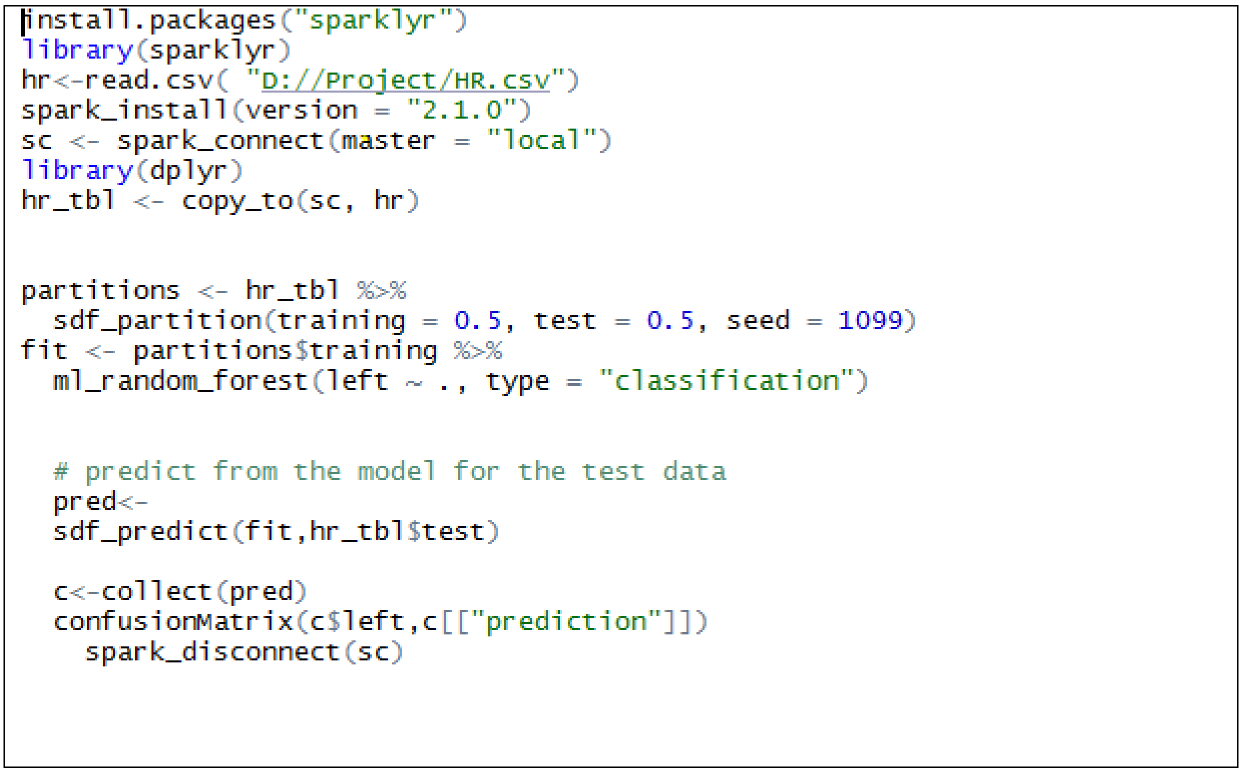
```
log_reg_model <- glm(left
~average_montly_hours+last_evaluation+number_project+time_spend_company+satisfaction_level,f
amily=binomial(link='logit'),data=train)
predictions<- predict(log_reg_model,test)
hr_model_logreg<- cbind(test,predictions)


i<-1
for(i in 1:3750)
{
 if(hr_model_logreg$predictions[i] > 0)
  {
   hr_model_logreg$predictions[i] <- 1
  }
  else
  {
    hr_model_logreg$predictions[i] <- 0
  }
}


# summarize results
confusionMatrix_logreg<- confusionMatrix(hr_model_logreg$predictions,hr_model_logreg$left)
confusionMatrix_logreg
```

11.4 Random forest tree

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Random forest tries to build multiple CART model with different sample and different initial variables. Random forest gives much more accurate predictions when compared to simple CART/CHAID or regression models in many scenarios. These cases generally have high number of predictive variables and huge sample size. This is because it captures the variance of several input variables at the same time and enables high number of observations to participate in the prediction. In some of the coming articles, we will talk more about the algorithm in more detail and talk about how to build a simple random forest on R.

Code:



```
install.packages("sparklyr")
library(sparklyr)
hr<-read.csv( "D://Project/HR.csv")
spark_install(version = "2.1.0")
sc <- spark_connect(master = "local")
library(dplyr)
hr_tbl <- copy_to(sc, hr)


partitions <- hr_tbl %>%
  sdf_partition(training = 0.5, test = 0.5, seed = 1099)
fit <- partitions$training %>%
  ml_random_forest(left ~ ., type = "classification")


  # predict from the model for the test data
  pred<-
  sdf_predict(fit,hr_tbl$test)

  c<-collect(pred)
  confusionMatrix(c$left,c[["prediction"]])
    spark_disconnect(sc)
```

12. OBSERVATIONS

Code:

```
g1 <- ggplot(hr,aes(sales,fill=left))+geom_bar()

plot(g1)
```

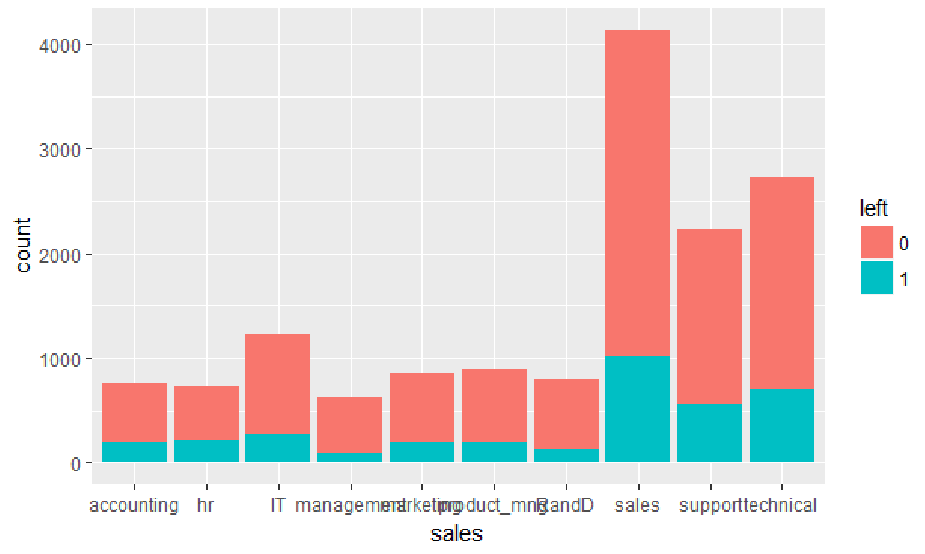


Figure 12.1 Plot for Sales

Code:

```
g2 <- ggplot(hr,aes(average_montly_hours,fill=left))+geom_bar()
plot(g2)
```

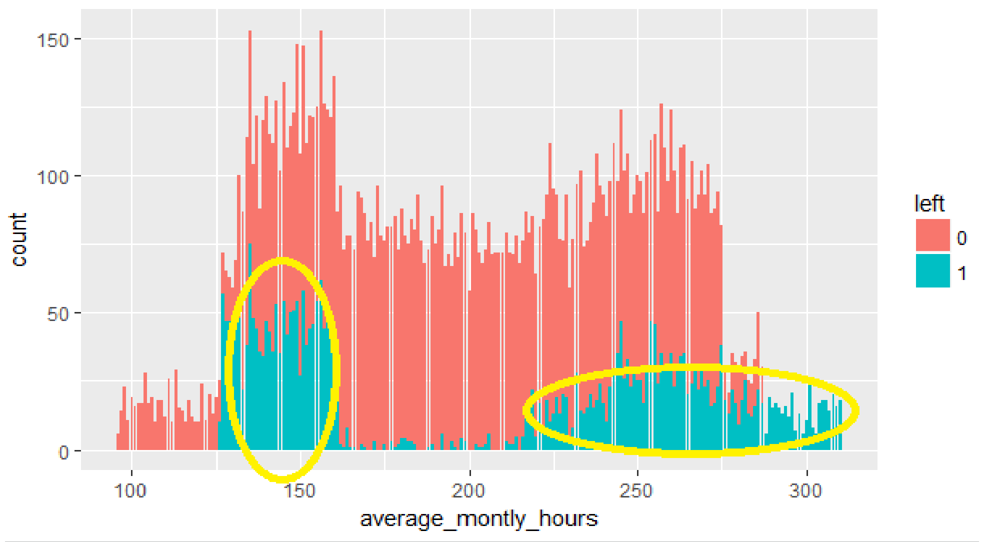


Figure 12.2 Plot for Average Monthly Hours

Code:

g3 <- ggplot(hr,aes(hr$last_evaluation,fill=left))+geom_bar()

plot(g3)

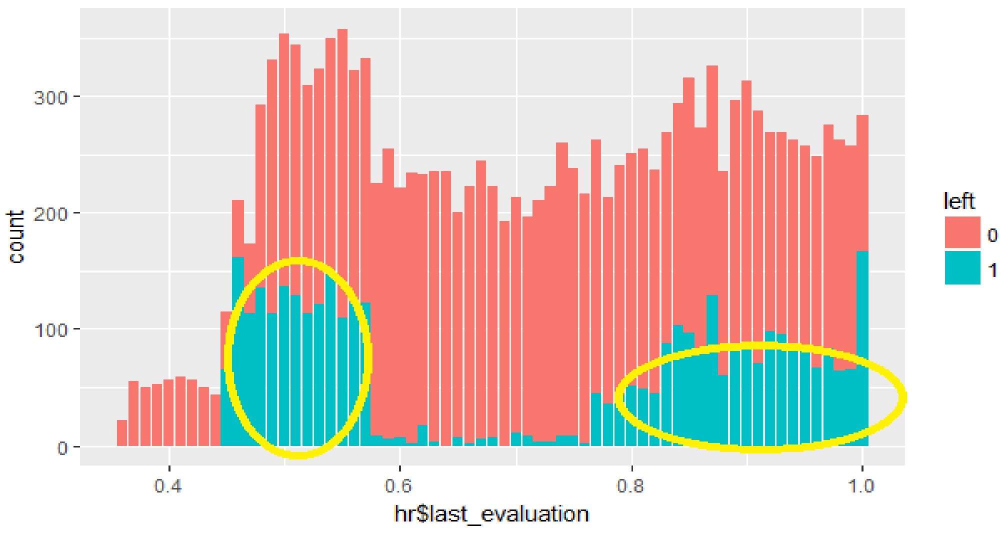


Figure 12.3 Plot last evaluation

Code:

g4 <- ggplot(hr,aes(hr$number_project,fill=left))+geom_bar()

prop.table(table(hr$number_project,hr$left))

plot(g4)

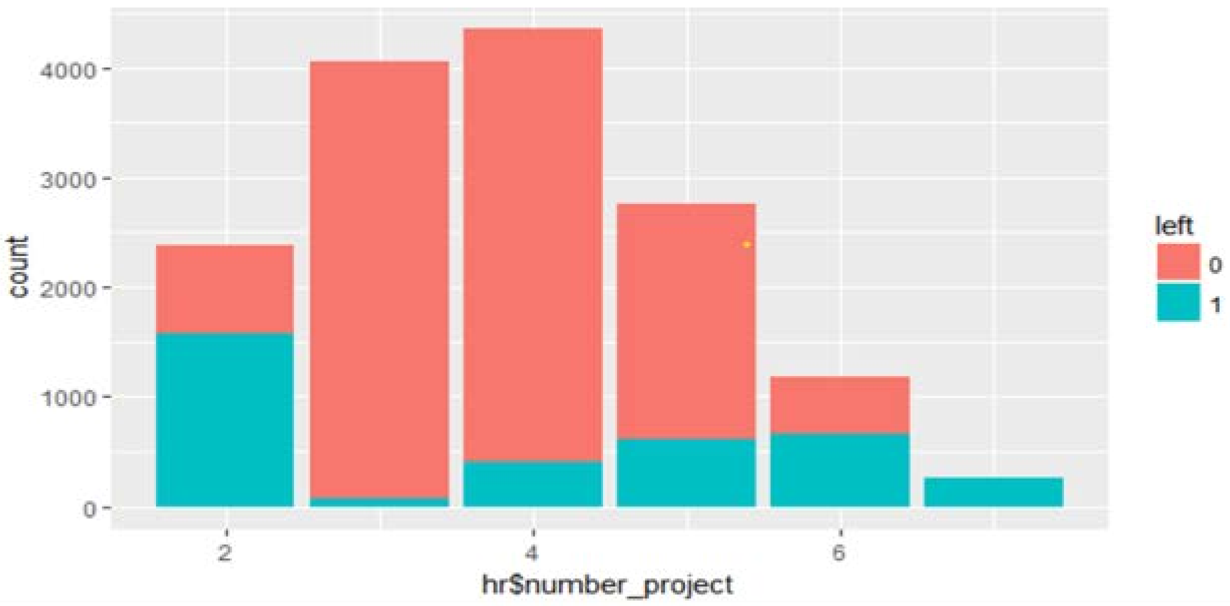


Figure 12.4 Plot for number project

Code:

```
g5 <- ggplot(hr,aes(hr$time_spend_company,fill=left))+geom_bar()
prop.table(table(hr$time_spend_company,hr$left))
plot(g5)
```

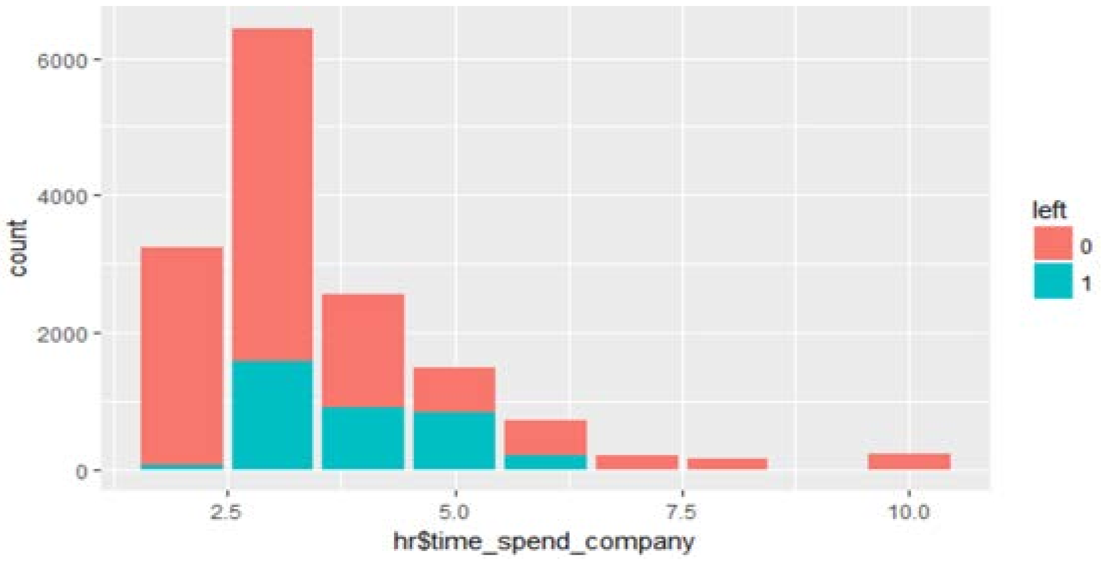


Figure 12.5 Plot for time spend company

Code:

```
g6 <- ggplot(hr,aes(hr$Work_accident,fill=left))+geom_bar()
prop.table(table(hr$Work_accident,hr$left))
plot(g6)
```

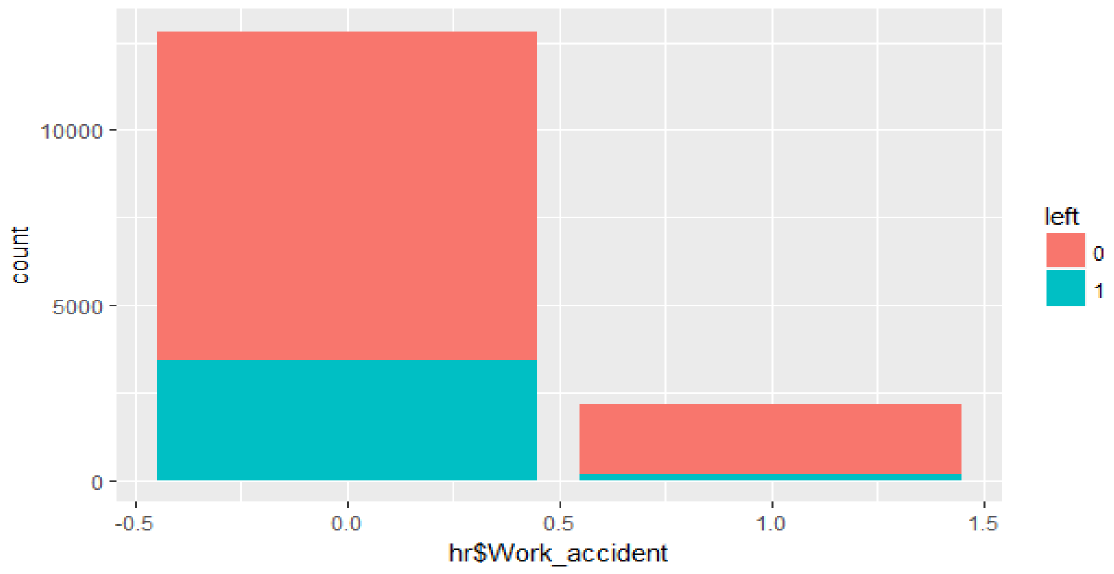


Figure 12.6 Plot for Work Accident

Code:

g7 <- ggplot(hr,aes(hr$promotion_last_5years,fill=left))+geom_bar()

prop.table(table(hr$promotion_last_5years,hr$left))

plot(g7)

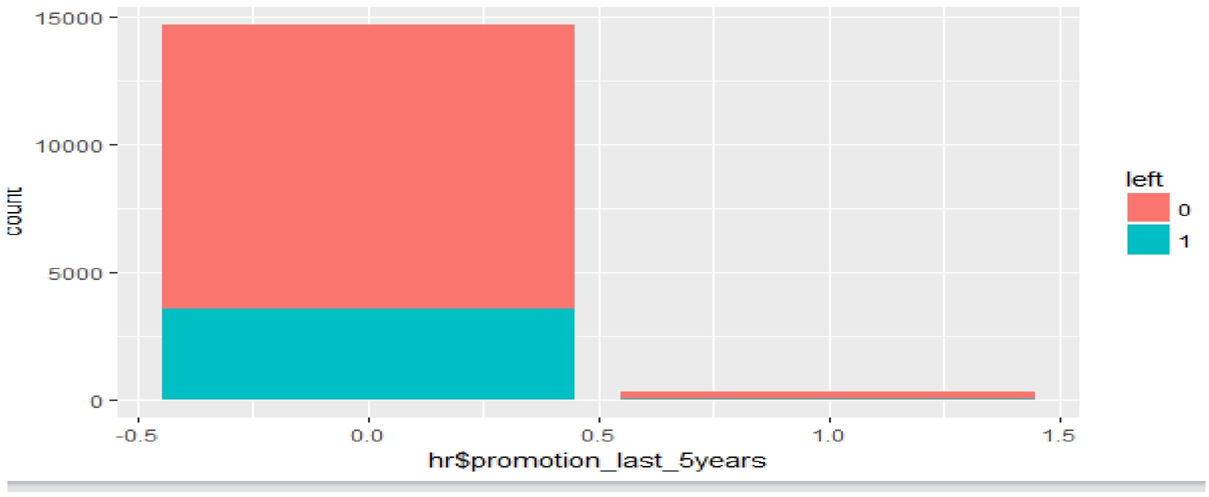


Figure 12.7 Plot for Promotion last 5 years

Code:

g8 <- ggplot(hr,aes(hr$salary,fill=left))+geom_bar()

prop.table(table(hr$salary,hr$left))

plot(g8)

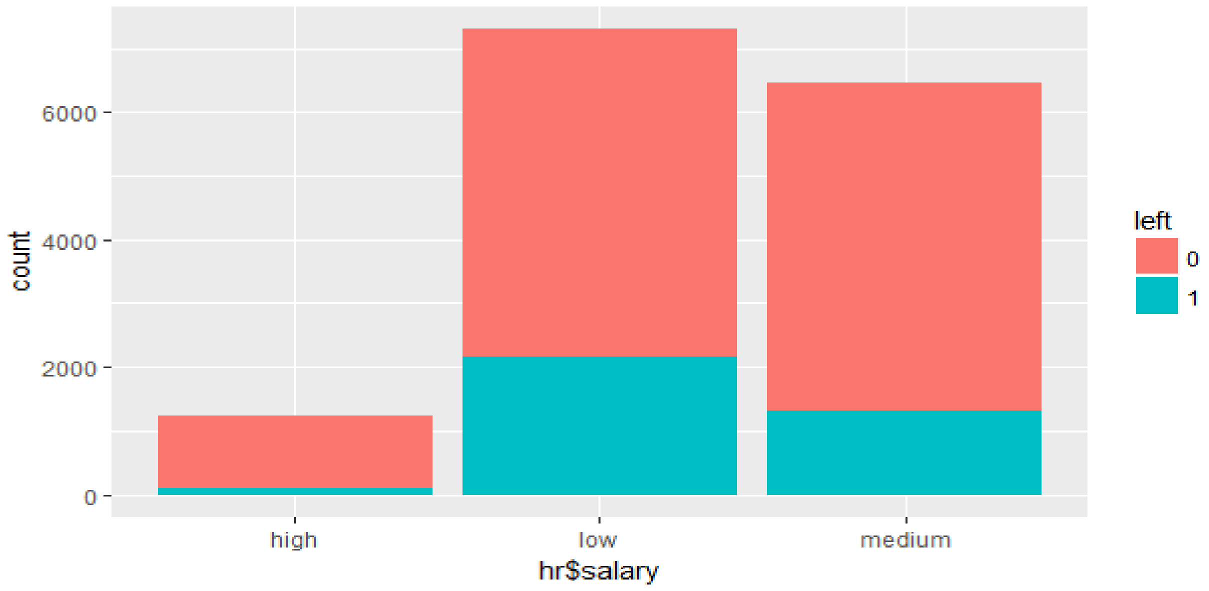


Figure 12.8 Plot for Salary

Code:

g9 <- ggplot(hr,aes(hr$satisfaction_level,fill=left))+geom_bar()

plot(g9)

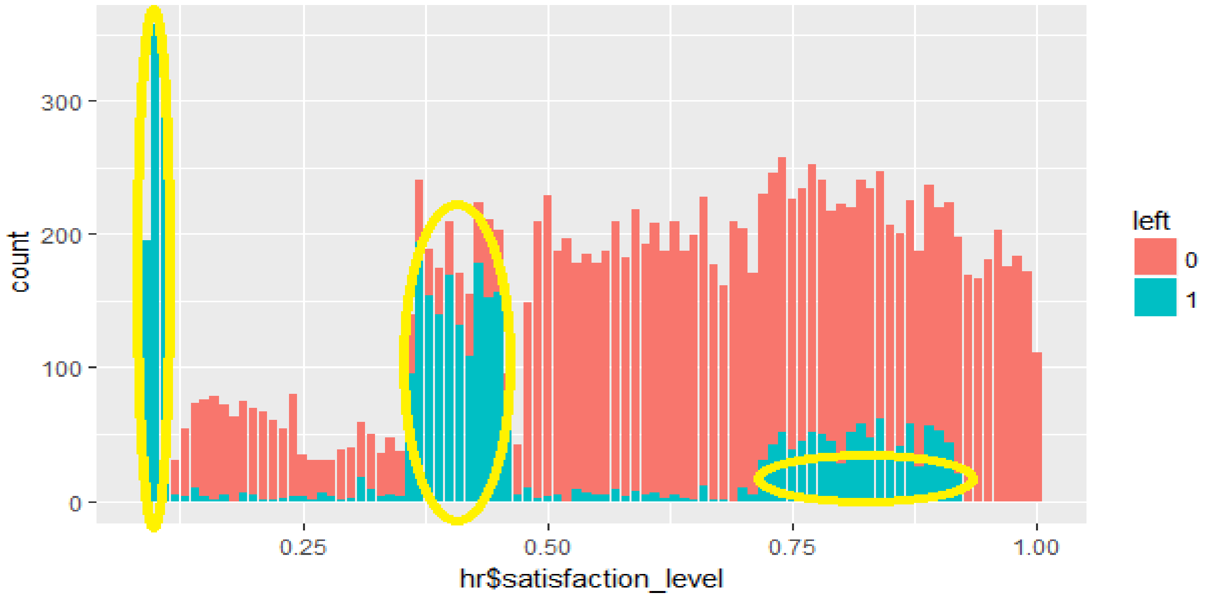


Figure 12.9 Plot for Satisfaction Level

Code:

f1<- ggplot(hr,aes(satisfaction_level,last_evaluation,fill=left,color=left))+geom_jitter()

plot(f1)

f2<-ggplot(hr,aes(satisfaction_level,average_montly_hours,fill=left,color=left))

+geom_jitter() plot(f2)

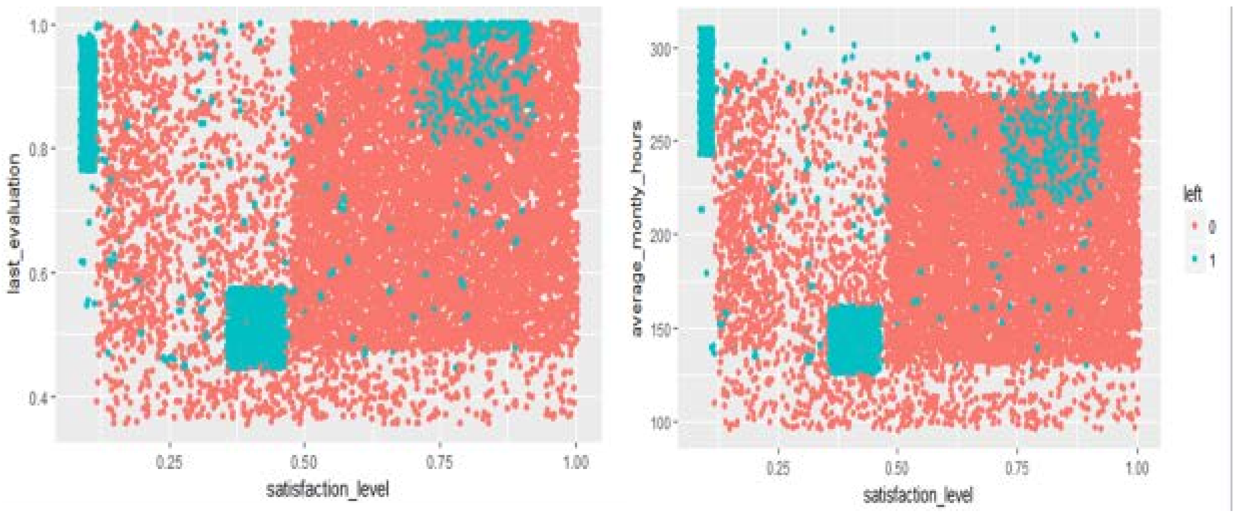


Figure 12.10 Plot for satisfaction level vs last evaluation & average_monthly_hours

Code:

```
f3<- ggplot(hr,aes(salary,last_evaluation,fill=left,color=left))+geom_jitter()
plot(f3)
f4<- ggplot(hr,aes(salary,satisfaction_level,fill=left,color=left))+geom_jitter()
plot(f4)
```

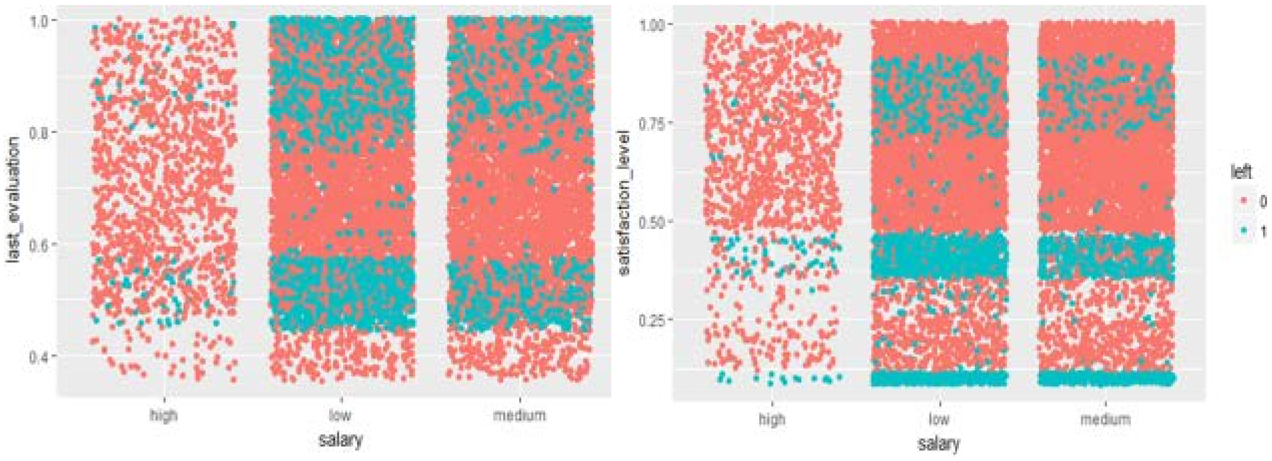


Figure 12.11 Plot for Salary vs last evaluation and satisfaction_level

Code:

```
f5<-
ggplot(hr,aes(last_evaluation,average_montly_hours,fill=left,color=left))+geom_jitter()
plot(f5)
f6<- ggplot(hr,aes(last_evaluation,Work_accident,fill=left,color=left))+geom_jitter()
plot(f6)
```

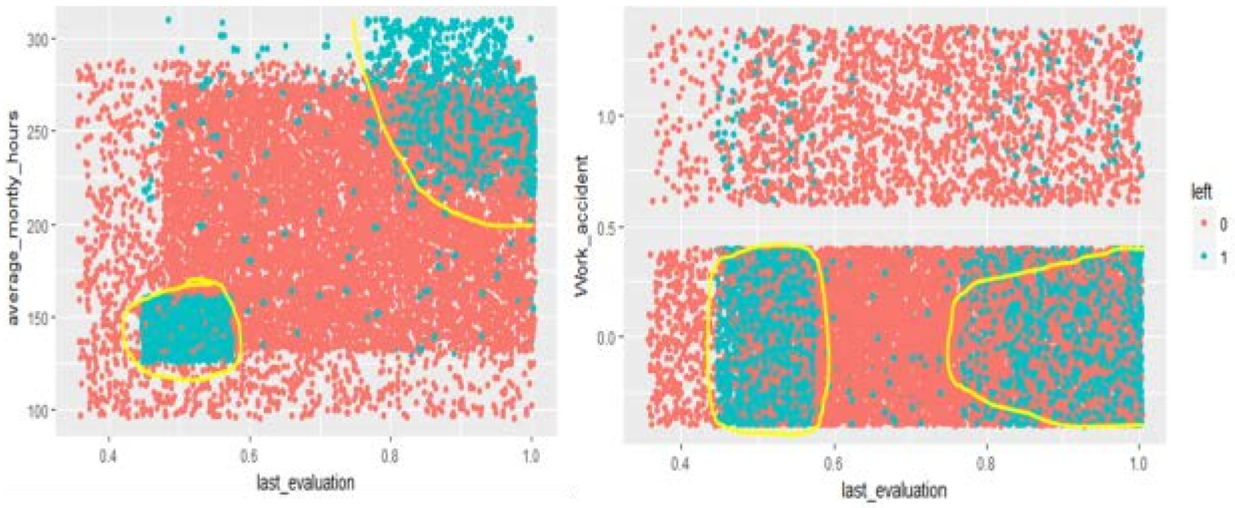


Figure 12.12 Plot for last evaluation, average monthly hours & Work_accident

Code:

fit<- rpart(left ~ satisfaction_level +last_evaluation + number_project + average_montly_hours + promotion_last_5years + salary , method="class", data = hr)

plot(fit, uniform=TRUE)

text(fit, use.n=TRUE, all=TRUE)

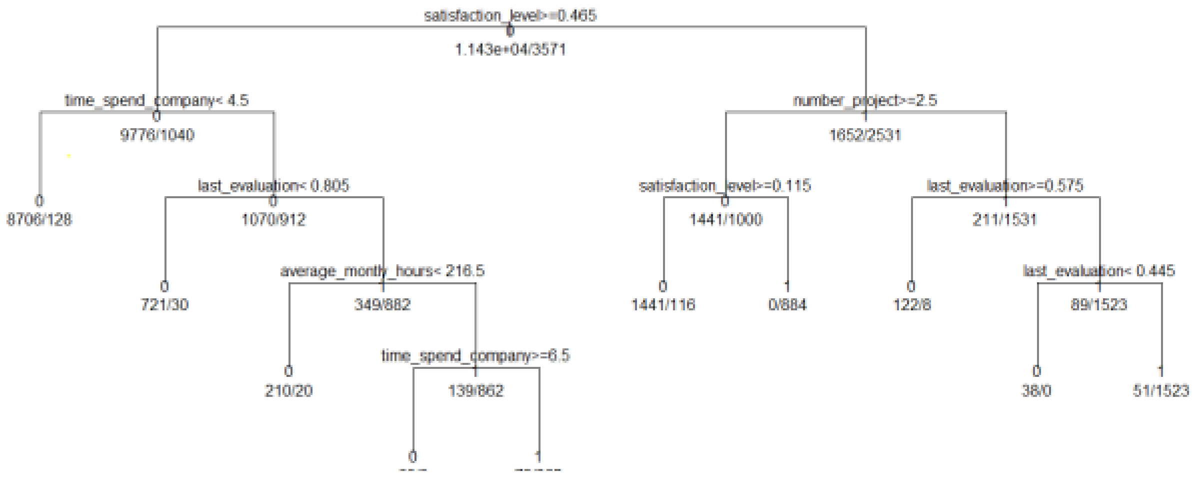


Figure 12.13 Classification Decision Tree

13. CONCLUSION

The given dataset possesses a massive amount of employee’s data of an organization. By using Data visualization to this dataset, we found the reason behind the employees leaving the organization. By applying predictive analysis to this data, we were be able to a stratergies that relies on proven and data-driven predictive models, instead of relying on gut feeling and soft science. HR predictive analytics enabled us to forecast the impact of people policies on employee attrition in the organization.

Predictive modeling allowed us to know organizations to identify employees, build profiles of those most likely to leave or stay, and understand how to retain maximum no of experienced employees distributed throughout the organization.

Common traits of Good people leaving

- Experienced
- Very low satisfaction level
- Spend more time at work

Possible Reasons for people leaving:

- Experienced people may not be finding any challenges in work. Hence they leave.
- Work to Pay ratio may be high (because we find clear correlation only in low and medium salary ranges.

REFERENCES

Book:

[1] O'Rielly R in a nutshell

[2] O'Rielly R Cookbook

[3] Big data Hadoop 3rd edition by Apache (Pdf Version)

[4] R for Dummies.

Website:

[5] https://www.r-project.org (current date)

[6] https://cran.r-project.org(current date)

[7] datacamp.org (current date)

[8] Edx.org (current date)

[9] https://www.analyticsvidhya.com (current date)

[10] https://en.wikipedia.org/wiki/R_(programming_language)(current date)