Configuration Manual

MSc Research Project

Sonali Pandhure
Student ID: x18137458

School of Computing
National College of Ireland

Supervisor: Dr.Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sonali Pandhure
Student ID:	x18137458
Programme:	MSc. Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	12/12/2019
Project Title:	Configuration Manual
Word Count:	232
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	12th December 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sonali Pandhure
x18137458

1 Introduction

In this given Configuration Manual all the installations like Windows 10, RStudio, and machine learning code is enlisted and explained in detailed. Section 1, Section 2 and Section 3 contains Windows installation, Rstudio installation and Code for all modules listed respectively.

1.1 Hardware Specification

Name of Device: SONALI
Processor Specification: Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
RAM Specification: 8.00 GB (7.98 GB Usable)
System type specification: 64-bit operating system, x64-based processor
Windows Edition: Windows 10 Pro The numbers starts at 1 with every call to the enumerate environment.

1.2 Software Specification

Languages Used: R Language is used to apply Machine learning models on London Stock Market dataset

2 Windows Installation

2.1 By the use of USB flash driver or DVD install windows 10 Figure1

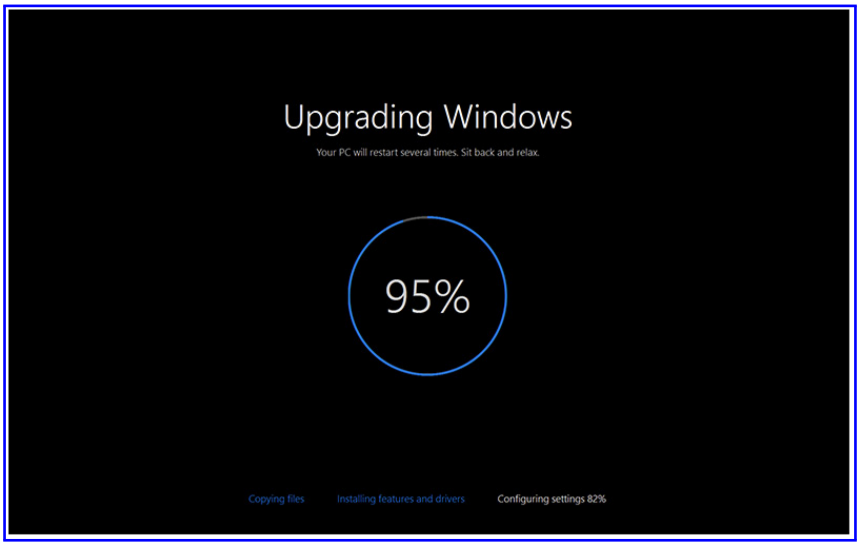


Figure 1: Windows installation

2.2 Select install now Figure2

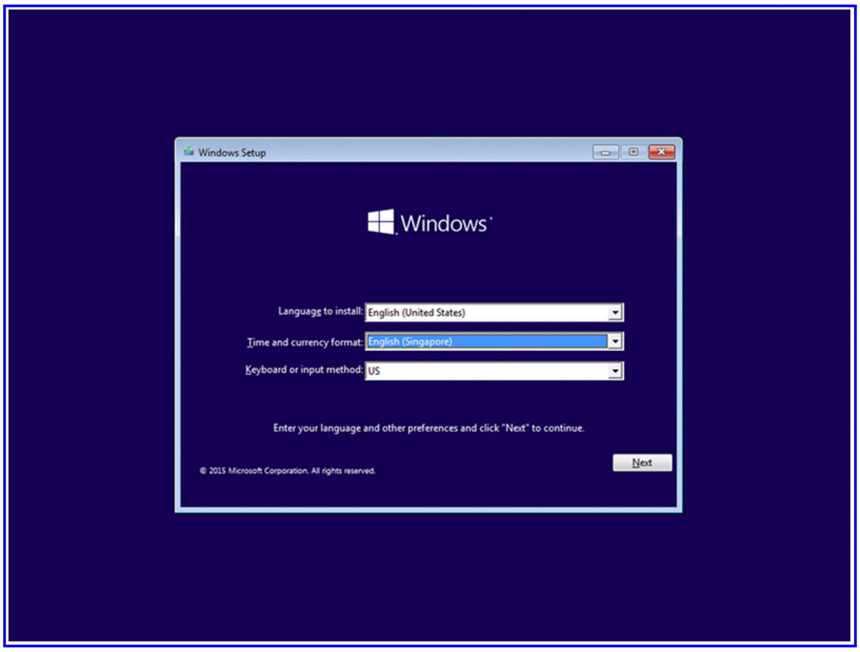


Figure 2:

2.3 Enter the product key Figure3

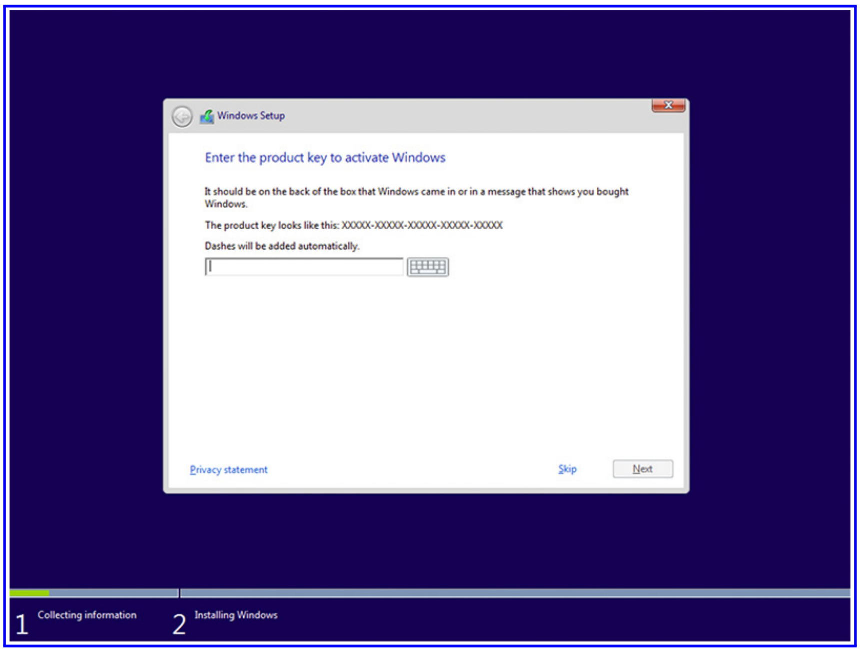


Figure 3:

2.4 Select accept on the user acceptance license Figure4

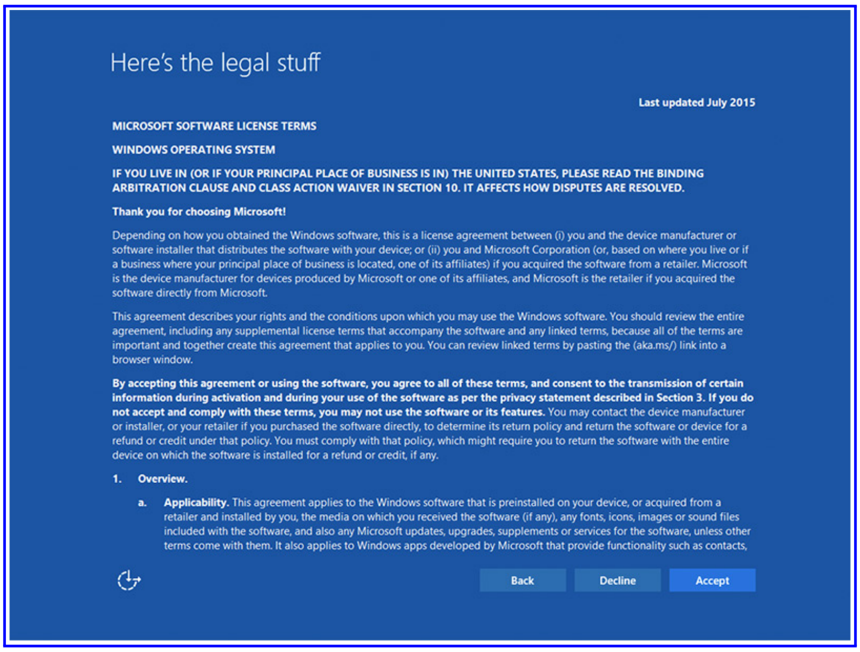


Figure 4:

2.5 Either upgradation of file can be done, or custom files can be installed based on the preferences Figure4

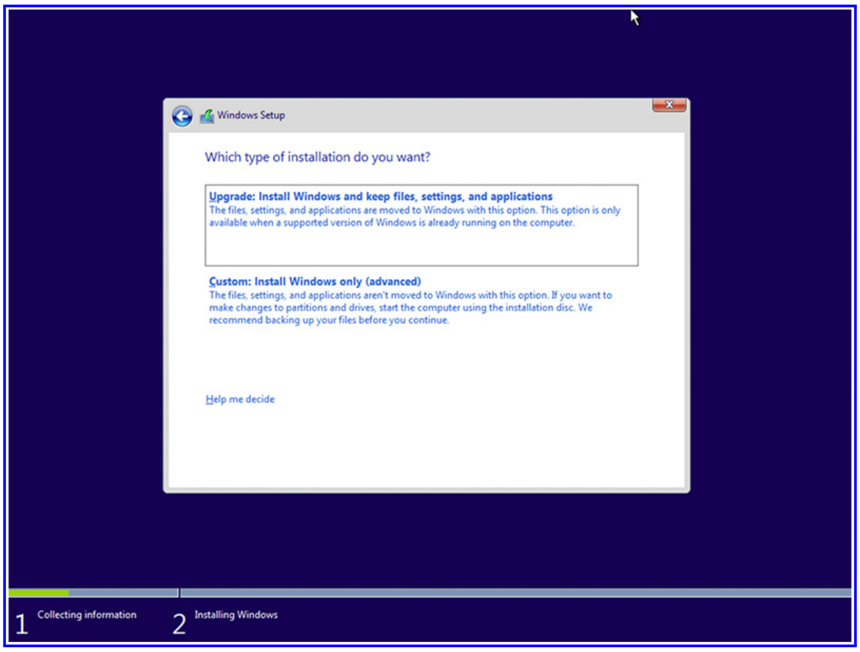


Figure 5:

2.6 Select windows 10 and formatting drive Figure5

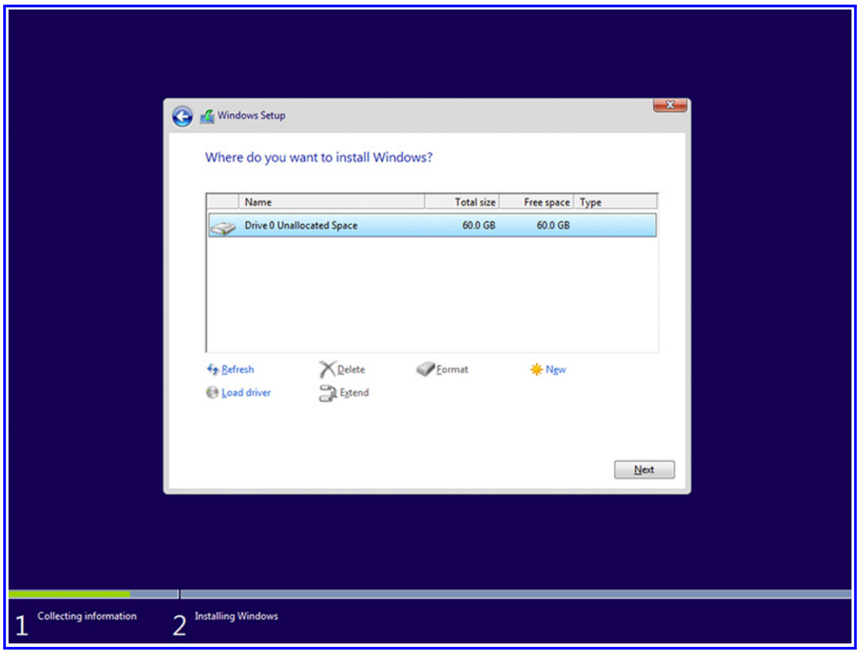


Figure 6:

2.7 Wait for the installation Figure6

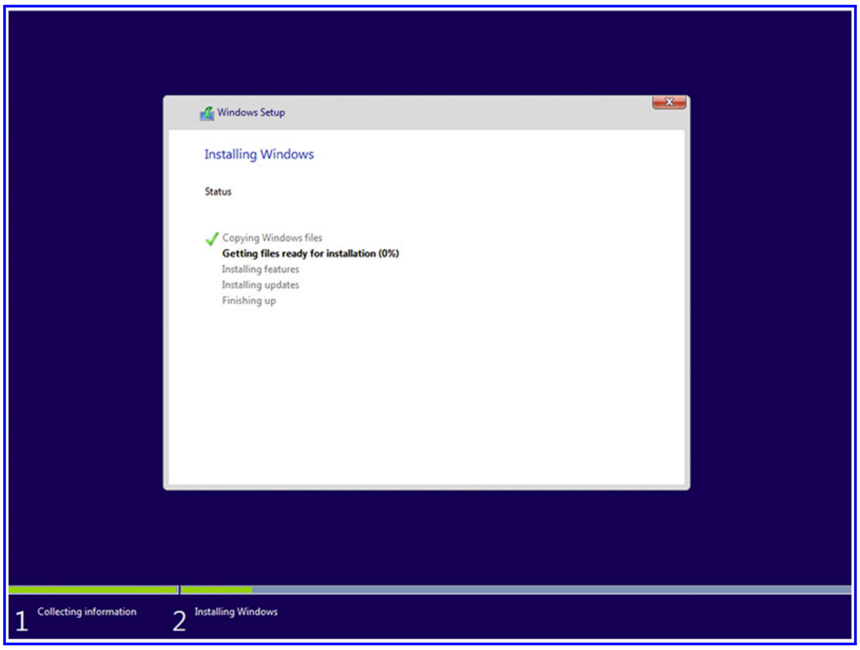


Figure 7:

2.8 Select the browser options Figure7

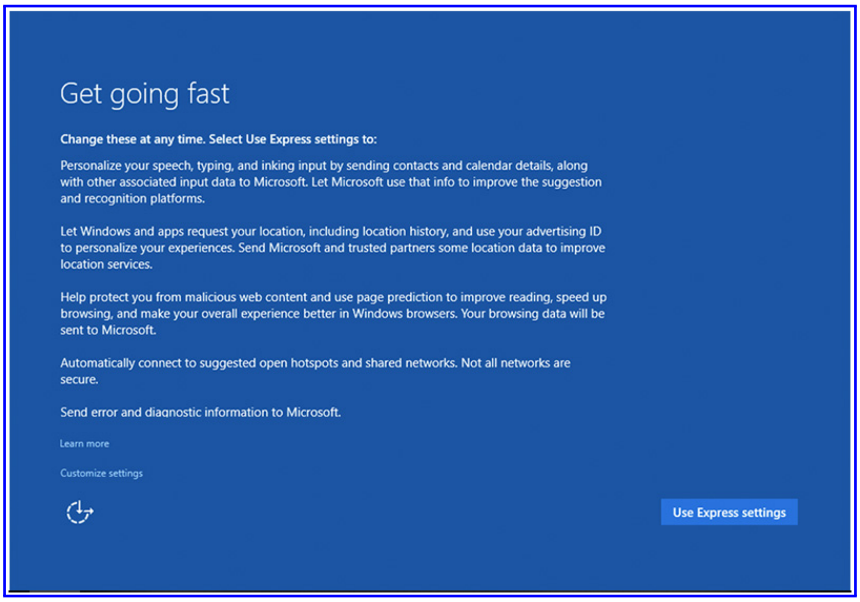


Figure 8:

2.9 Customize the calendar and inputs Figure8

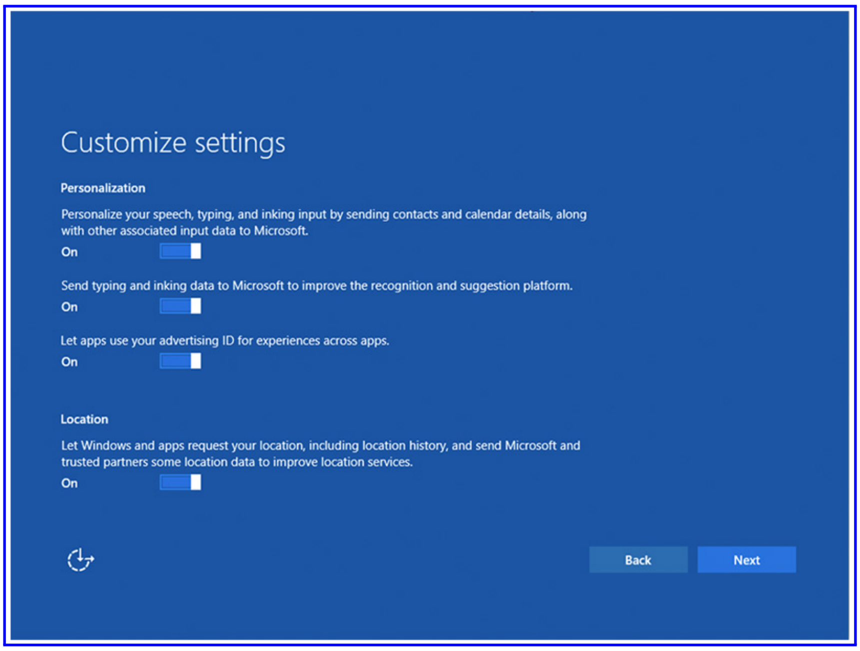


Figure 9:

2.10 Select the browser data and data connectivity options Figure9

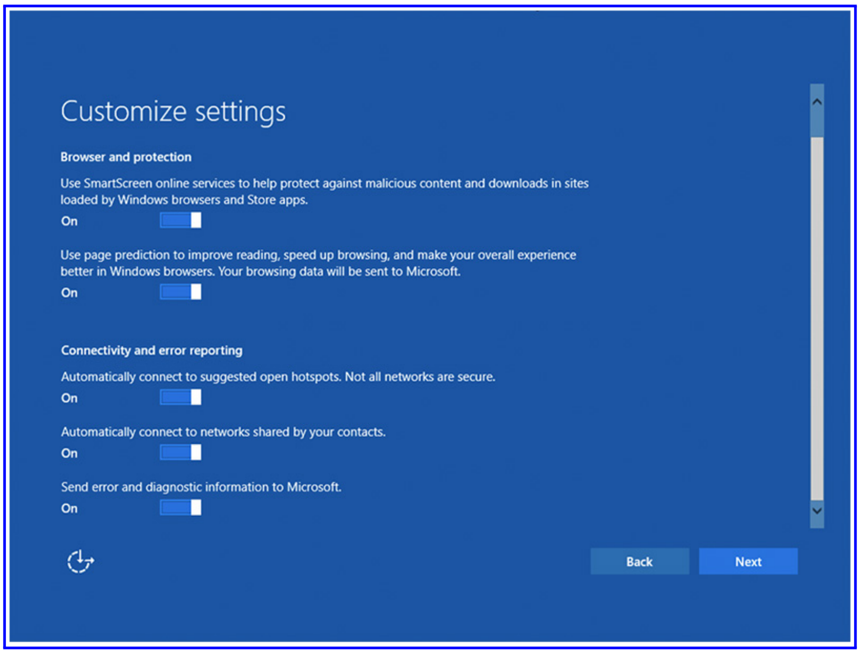


Figure 10:

2.11 Assign ID of the PC owner Figure10

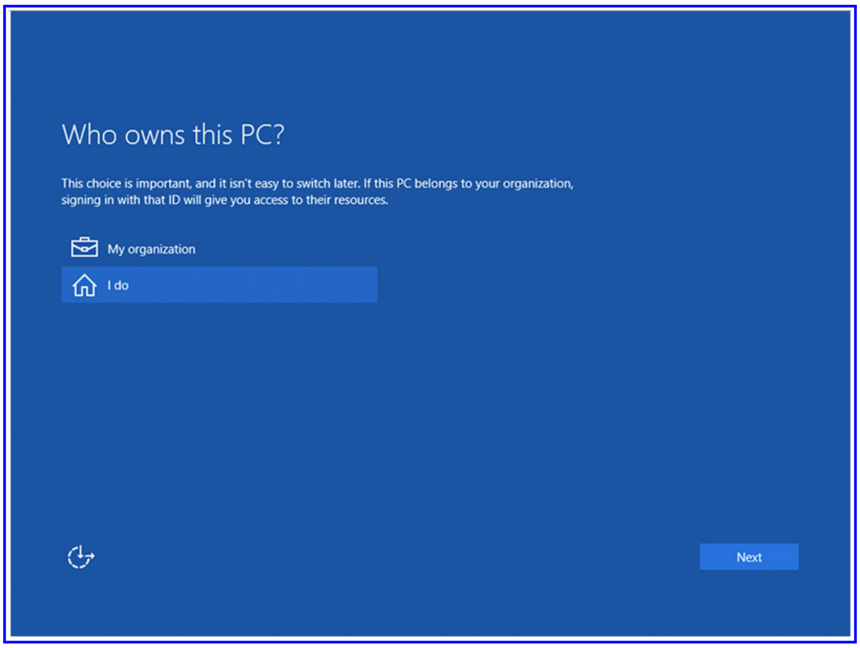
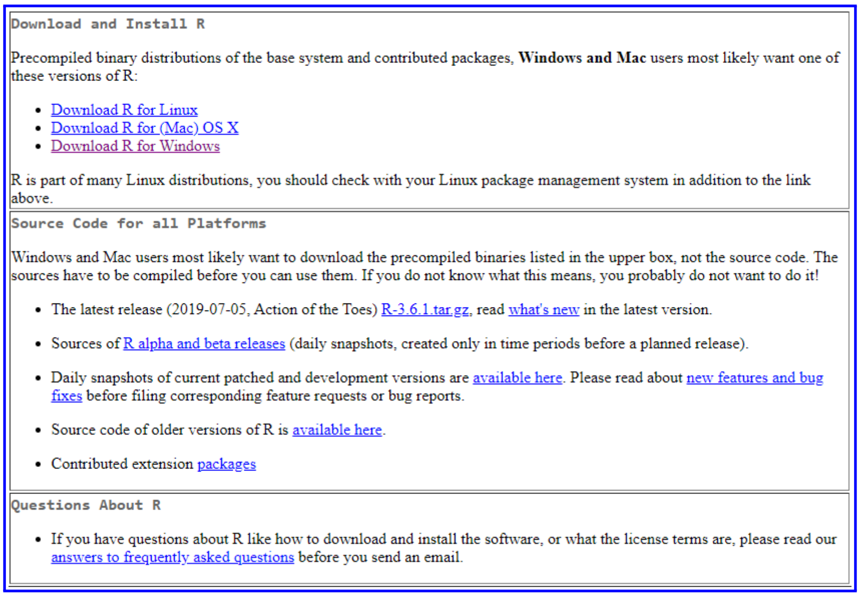


Figure 11:

3 RStudio Installation

Step by step RStudio installation is listed by the following steps,

3.1 Launch Firefox or Chrome to install RStudio Figure12



Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-07-05, Action of the Toes) R-3.6.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Figure 12:

3.2 Type install RStudio and follow the given link Figure13

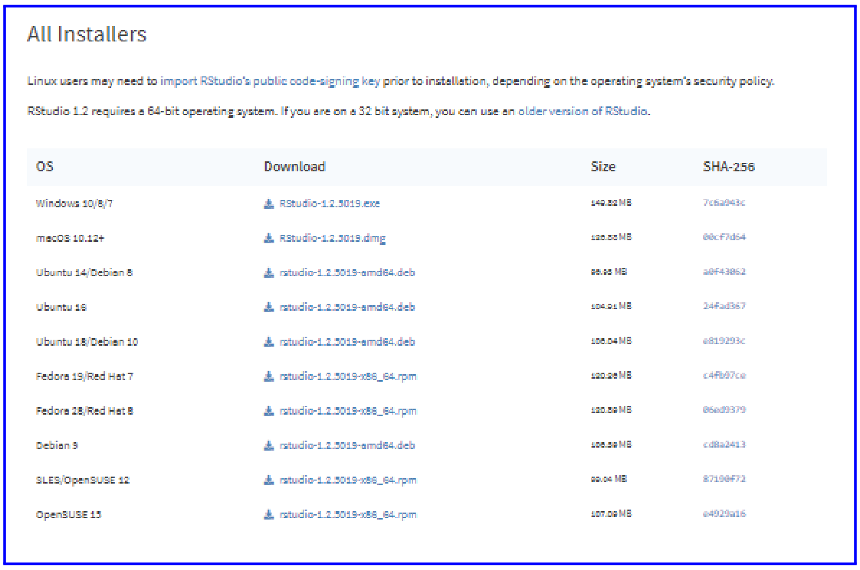


Figure 13:

[1]

3.3 Once it get installed, console window will promptFigure15

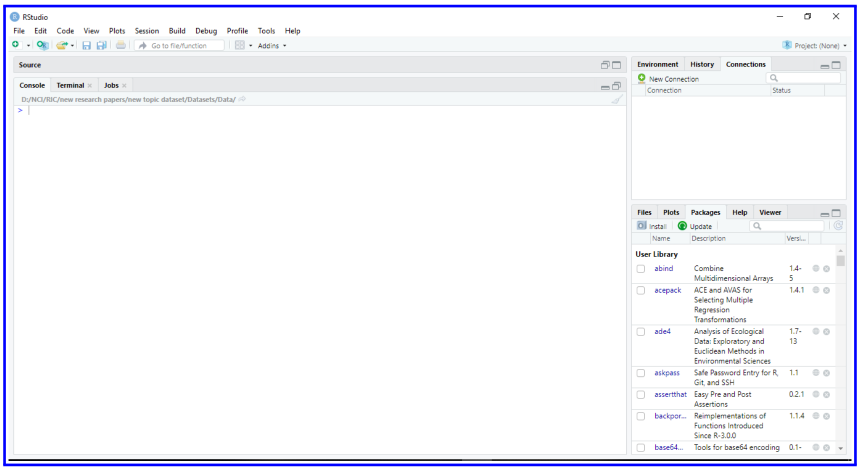


Figure 14:

[1]https://cran.r-project.org/

3.4 Section which are highlighted will let you to install R package's. It includes libraries which are required for coding and to apply models on datasetFigure16

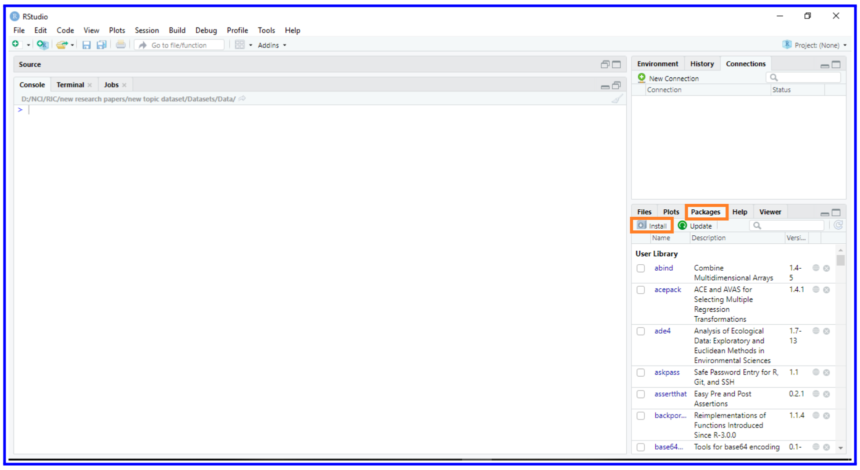


Figure 15:

3.5 Once clicking on install you can install any package by entering required package name Figure17

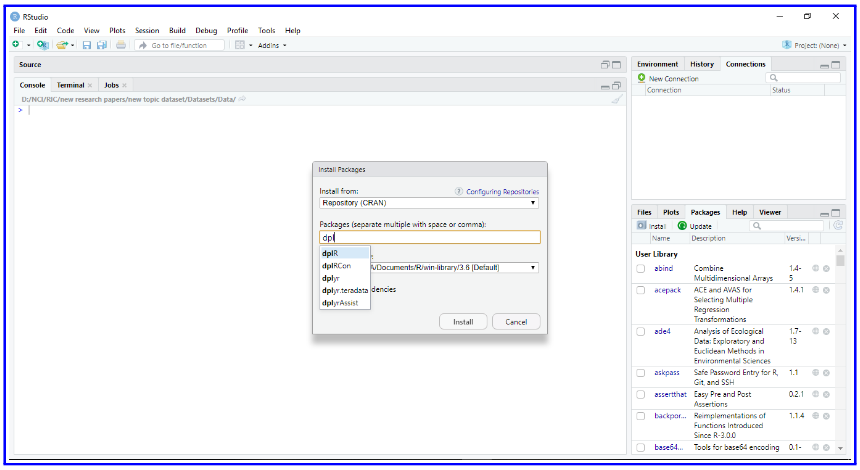


Figure 16:

3.6 Once package get installed it will show the following steps on console window, shown in Figure18

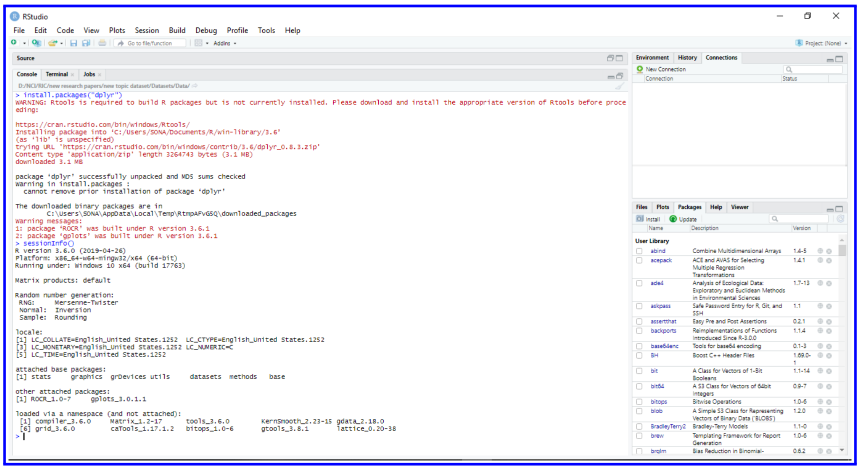
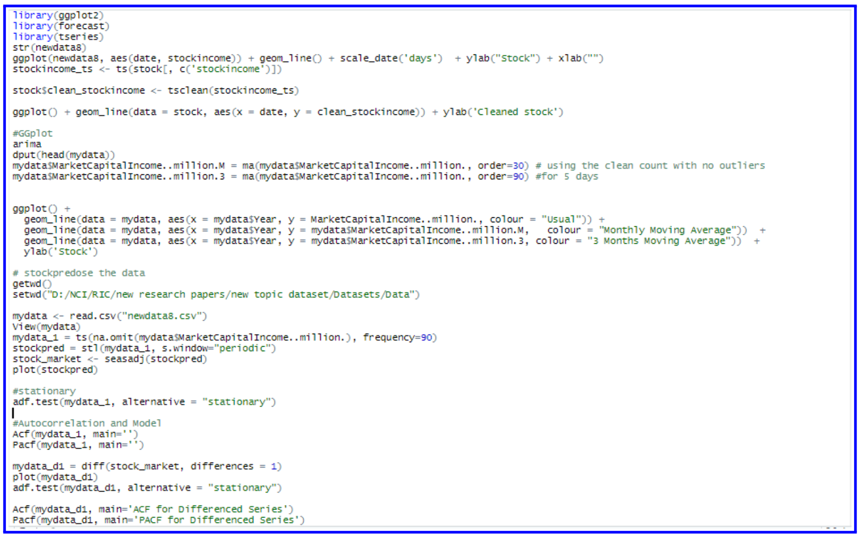


Figure 17:

4 Machine Learning algorithm Code of all Applied Models on Stock Market Data

Step by step machine learning code pictures are explained and enlisted below,

4.1 ARIMA Time series model is enlisted below, Figure19



```
library(ggplot2)
library(forecast)
library(tseries)
str(newdata8)
ggplot(newdata8, aes(date, stockincome)) + geom_line() + scale_date('days')  + ylab("Stock") + xlab("")
stockincome_ts <- ts(stock[, c('stockincome')])

stock$clean_stockincome <- tsclean(stockincome_ts)

ggplot() + geom_line(data = stock, aes(x = date, y = clean_stockincome)) + ylab('Cleaned stock')

#GGplot
arima
dput(head(mydata))
mydata$MarketCapitalIncome..million.M = ma(mydata$MarketCapitalIncome..million., order=30) # using the clean count with no outliers
mydata$MarketCapitalIncome..million.3 = ma(mydata$MarketCapitalIncome..million., order=90) #for 5 days


ggplot() +
  geom_line(data = mydata, aes(x = mydata$Year, y = MarketCapitalIncome..million., colour = "Usual")) +
  geom_line(data = mydata, aes(x = mydata$Year, y = mydata$MarketCapitalIncome..million.M,   colour = "Monthly Moving Average"))  +
  geom_line(data = mydata, aes(x = mydata$Year, y = mydata$MarketCapitalIncome..million.3, colour = "3 Months Moving Average"))  +
  ylab('Stock')

# stockpredose the data
getwd()
setwd("D:/NCI/RIC/new research papers/new topic dataset/Datasets/Data")

mydata <- read.csv("newdata8.csv")
View(mydata)
mydata_1 = ts(na.omit(mydata$MarketCapitalIncome..million.), frequency=90)
stockpred = stl(mydata_1, s.window="periodic")
stock_market <- seasadj(stockpred)
plot(stockpred)

#stationary
adf.test(mydata_1, alternative = "stationary")

#Autocorrelation and Model
Acf(mydata_1, main='')
Pacf(mydata_1, main='')

mydata_d1 = diff(stock_market, differences = 1)
plot(mydata_d1)
adf.test(mydata_d1, alternative = "stationary")

Acf(mydata_d1, main='ACF for Differenced Series')
Pacf(mydata_d1, main='PACF for Differenced Series')
```

Figure 18:

4.2 How ARIMA function works is shown below in Figure20

```
#stationary
adf.test(mydata_1, alternative = "stationary")

#Autocorrelation and Model
Acf(mydata_1, main='')
Pacf(mydata_1, main='')

mydata_d1 = diff(stock_market, differences = 1)
plot(mydata_d1)
adf.test(mydata_d1, alternative = "stationary")

Acf(mydata_d1, main='ACF for Differenced Series')
Pacf(mydata_d1, main='PACF for Differenced Series')

#Fitting ARIMA model
auto.arima(stock_market, seasonal=FALSE)

#Evaluate
fit<-auto.arima(stock_market, seasonal=FALSE)
tsdisplay(residuals(fit), lag.max=200, main='(1,1,1) Model Residuals')

fit2 = arima(stock_market, order=c(1,1,7))
fit2
tsdisplay(residuals(fit2), lag.max=200, main='Monthly stock Model Residuals')

#Forecast
fcast <- forecast(fit2, h=2)
plot(fcast)
fcast

hold <- window(ts(stock_market), start=0.1)

fit_no_holdout = arima(ts(stock_market), order=c(1,1,7))

fcast_no_holdout <- forecast(fit_no_holdout,h=7)

plot(fcast_no_holdout, main=" ")
lines(ts(stock_market))


fit_w_seasonality = auto.arima(stock_market, seasonal=TRUE)
     fit_w_seasonality

monthly_stocks <- forecast(fit_w_seasonality, h=30)
plot(monthly_stocks)

#---------------------------------------------------
```

Figure 19:

4.3 Logistic regression code for year 2010 to 2019 is listed below and shown in Figure21

```
##Logistic
getwd()
setwd("D:/NCI/RIC/new research papers/new topic dataset/Datasets/Data")
newdata2019 <- read.csv("newdata8.csv")
View(newdata2019)
mylogistic <- newdata2019
str(mylogistic)
mylogistic$X <- as.factor(mylogistic$X)
mylogistic$List.Date <- as.numeric(mylogistic$List.Date)
mylogistic$Company <- as.numeric(mylogistic$Company)
mylogistic$Sector <- as.numeric(mylogistic$Sector)
mylogistic$X <- as.factor(mylogistic$X)
mylogistic$Country.of.Incorporation <- as.numeric(mylogistic$Country.of.Incorporation)
mylogistic$Market <- as.numeric(mylogistic$Market)
mylogistic$MarketCapitalIncome..million. <- as.numeric(mylogistic$MarketCapitalIncome..million.)
mylogistic$Year <- as.numeric(mylogistic$Year)
mylogistic$Sub.Sector <- as.numeric(mylogistic$Sub.Sector)

xtabs(~X + Company, data = mylogistic)

set.seed(1234)
ind <- sample(2, nrow(mylogistic), replace=T, prob = c(0.8, 0.2))
train <- mylogistic[ind==1,]
test <- mylogistic[ind==2,]

mymodel <- glm(X ~ Sub.Sector + Year + MarketCapitalIncome..million. + Market + Country.of.Incorporation
               + Sector + Company + List.Date , data=train, family='binomial')
summary(mymodel)

p1 <- predict(mymodel, train, type= 'response')
head(p1)
head(train)
p1

pred1 <- ifelse(p1>0.5,1,0)
pred1
tab1 <- table(predicted = pred1, Actual = train$X)
tab1

sum(diag(tab1))/sum(tab1)
1-sum(diag(tab1))/sum(tab1)

table(mylogistic$X)
p2 <- predict(mymodel, test, type = 'response')
pred2 <- ifelse(p2>0.5, 1, 0)
tab2 <- table(Predicted = pred2, Actual = test$X)
tab2
1-sum(diag(tab2))/sum(tab2)
```

Figure 20:

4.4 How Logistic Regression function works for data from year 2010 to 2019 is shown below in Figure22

```
pred1 <- ifelse(p1>0.5,1,0)
pred1
tab1 <- table(predicted = pred1, Actual = train$X)
tab1

sum(diag(tab1))/sum(tab1)
1-sum(diag(tab1))/sum(tab1)

table(mylogistic$X)
p2 <- predict(mymodel, test, type = 'response')
pred2 <- ifelse(p2>0.5, 1, 0)
tab2 <- table(Predicted = pred2, Actual = test$X)
tab2
1-sum(diag(tab2))/sum(tab2)

with(mymodel, pchisq(null.deviance - deviance, df.null-df.residual, lower.tail = F))


library(gplots)
library(ROCR)

head(p2)


pred4 <- predict(mymodel, mylogistic, type = "response")
head(mylogistic)
histogram(pred4)
pred4 <- prediction(pred4, mylogistic$X)
eval <- performance(pred4, "acc")
plot(eval)
abline(h=0.8, v=0.75)
max <- which.max(slot(eval, "y.values")[[1]])
max
acc <- slot(eval, "y.values")[[1]][max]
acc
cut <- slot(eval, "x.values")[[1]][max]
cut
print(c(Accuracy=acc, Cutoff = cut))
```

Figure 21:

4.5 Logistic regression code for Brexit Discussion Month October 2019 is listed below in Figure23

```
table(mylogistic$X)
p2 <- predict(mymodel, test, type = 'response')
pred2 <- ifelse(p2>0.5, 1, 0)
tab2 <- table(Predicted = pred2, Actual = test$X)
tab2
1-sum(diag(tab2))/sum(tab2)

with(mymodel, pchisq(null.deviance - deviance, df.null-df.residual, lower.tail = F))


library(gplots)
library(ROCR)

head(p2)


pred4 <- predict(mymodel, mylogistic, type = "response")
head(mylogistic)
histogram(pred4)
pred4 <- prediction(pred4, mylogistic$X)
eval <- performance(pred4, "acc")
plot(eval)
abline(h=0.8, v=0.75)
max <- which.max(slot(eval, "y.values")[[1]])
max
acc <- slot(eval, "y.values")[[1]][max]
acc
cut <- slot(eval, "x.values")[[1]][max]
cut
print(c(Accuracy=acc, Cutoff = cut))
```

Figure 22:

4.6 Random forest model and how it function's is enlisted below and shown in Figure24

```r
mynewdata <- mydata

str(mynewdata)
mynewdata <- na.omit(mynewdata)

mynewdata$List.Date <- as.numeric(mynewdata$List.Date)
mynewdata$Company <- as.numeric(mynewdata$Company)
mynewdata$Sector <- as.numeric(mynewdata$Sector)
mynewdata$X <- as.factor(mynewdata$X)
mynewdata$Country.of.Incorporation <- as.numeric(mynewdata$Country.of.Incorporation)
mynewdata$Market <- as.numeric(mynewdata$Market)
mynewdata$MarketCapitalIncome..million. <- as.numeric(mynewdata$MarketCapitalIncome..million.)
mynewdata$Year <- as.numeric(mynewdata$Year)
mynewdata$Sub.Sector <- as.numeric(mynewdata$Sub.Sector)
library(caret)
set.seed(18129633)
na.omit(mynewdata)


set.seed(100)
train <- sample(nrow(mynewdata), 0.7*nrow(mynewdata), replace = FALSE)
TrainSet <- mynewdata[train,]
ValidSet <- mynewdata[-train,]
summary(TrainSet)
summary(ValidSet)

#sample <- createDataPartition(mynewdata$X, p = .75, list = FALSE)
#train <- mynewdata[sample, ]
#test <- mynewdata[-sample, ]

#names(mynewdata) <- make.names(mynewdata)


head(mynewdata)
library(randomForest)
model1 <- randomForest(X ~ ., data = TrainSet, importance = TRUE)
model1
#rfModel = randomForest(mynewdata$X~., data=train)
varImpPlot(model1)

library(e1071)
confusionMatrix(predict(model1,ValidSet), ValidSet$X)
```

Figure 23:

4.7 Naive Bayes model and its implementation is listed below in Figure24

```r
#creating training and testing sets
ind <- sample(2, nrow(data), replace =T, prob = c(0.8, 0.2))
train_nb <- data[ind == 1,]
test_nb <- data[ind == 2,]
summary(train_nb)
names(train_nb)
str(train_nb)
typeof(train_nb)

#Naive bayes models
model_nav <- naive_bayes(X~.,  data = train_nb)
head(model_nav)

#p1
p1_n <- predict(model_nav, train_nb)
head(cbind(p1_n, train_nb))
(tab1_n <- table(p1_n, train_nb$X))
1- sum(diag(tab1_n)) / sum(tab1_n)
p5 <- predict(model_nav, train_nb, type= "response")
p5 <- predict(model_nav, train_nb = AbsTest, type="prob")
head(p1)

#p2
p2_n <- predict(model_nav, test_nb)
(tab2 <- table(p2_n, test_nb$X))
1- sum(diag(tab2)) / sum(tab2)

#confusion matrix
mat <- confusionMatrix(p1_n, train_nb$X)
mat
#checking accuracy, precison, recall
(accuracy <- sum(diag(mat)) / sum(mat))
accuracy
(precision <- diag(mat) / rowSums(mat))
precision
(recall <- diag(mat) / colSums(mat))
recall

#AUC/ROC
n <- ROCR::plot(X ~ p1_n, data = train_nb)
#plot(n)

#sensitivity, Specificity, F1 score
sensitivity(tab1_n)
specificity(tab1_n)
posPredValue(tab1_n)
negPredValue(tab1_n)
```

Figure 24:

4.8 Multiple Regression model and how it function's shown below in Figure24

```
#Tidyverse library for visualization and data manipulation
library(tidyverse)
#fetching dataset which contains market income so for that datarium package is installed
newstock <- read.csv("newdata8.csv")
model <- lm(MarketCapitalIncome..million. ~ X, data = newstock)

summary(model)
summary(model)$coefficient
model <- lm(MarketCapitalIncome..million. ~ X, data = newstock)
summary(model)
confint(model)
#Residual Standard Error (RSE), or sigma
sigma(model)/mean(newstock$MarketCapitalIncome..million.)
```

Figure 25:

References

1. https://cran.r-project.org/
2. https://rstudio.com/products/rstudio/download/
3. https://www.microsoft.com/en-gb/software-download/windows10
4. https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/
5. https://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/
6. https://www.statmethods.net/stats/regression.html
7. https://www.r-bloggers.com/understanding-naive-bayes-classifier-using-r/