R Coding:

```
#cleaning the R environment

rm(list = ls())


#installing the required packages used for model development and preprocessing techniques

install.packages(c("ggplot2","lsr","corrgram","rpart","DataCombine","DMwR","rattle","mltools"
,"pROC","randomForest","inTrees"

                   ,"usdm","Metrics"))

x =
c("ggplot2","lsr","corrgram","rpart","DataCombine","DMwR","rattle","mltools","pROC","rando
mForest","inTrees"

     ,"usdm","Metrics")


#x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",

#      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')


#cross checking whether all the packages are installed or not

lapply(x, require, character.only = TRUE)


#removing the values

rm(x)
```

```
#setting the working directory in which our data set is present
setwd("D:/data science/project")

#checking the working directory
getwd()

#loading the data set into R environment
data_frame = read.csv("day.csv")

#checking the data types of data
str(data_frame)

#conversion of required data types into numeric a/c data
data_frame$instant = as.numeric(data_frame$instant)
data_frame$temp = as.numeric(data_frame$temp)
data_frame$atemp = as.numeric(data_frame$atemp)
data_frame$hum = as.numeric(data_frame$hum)
data_frame$windspeed = as.numeric(data_frame$windspeed)
data_frame$casual = as.numeric(data_frame$casual)
data_frame$registered = as.numeric(data_frame$registered)
data_frame$cnt = as.numeric(data_frame$cnt)

#converting into categorical variables
data_frame$season = as.factor(as.character(data_frame$season))
```

```
data_frame$yr = as.factor(as.character(data_frame$yr))

data_frame$mnth = as.factor(as.character(data_frame$mnth))

data_frame$holiday = as.factor(as.character(data_frame$holiday))

data_frame$workingday = as.factor(as.character(data_frame$workingday))

data_frame$weathersit = as.factor(as.character(data_frame$weathersit))

data_frame$weekday = as.factor(as.character(data_frame$weekday))

data_frame$dteday = as.Date(data_frame$dteday)

str(data_frame)


#checking is there any missing values in the data

sum(is.na(data_frame))


#no missing values found in the given data


#Outliers detection on numerical variables

num_var = c("instant","temp","atemp","hum","windspeed","casual","registered","cnt")


for (i in 1:length(num_var)) {

  assign(paste0("gn",i), ggplot(aes_string(y = (num_var[i]), x = "cnt"),data =
subset(data_frame))+

        stat_boxplot(geom = "errorbar" , width = 0.5) +

        geom_boxplot(outlier.color="red", fill = "grey" , outlier.shape=18, outlier.size=1,
notch=FALSE) +

        theme(legend.position="bottom")+

        labs(y=num_var[i],x="cnt")+
```

```
        ggtitle(paste("Box Plot of responded",num_var[i])))
  print(i)
  print(num_var[i])
}
options(warn = -1)
#plotting for clear  vision of outliers
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,ncol = 2)


#--------Getting the outliers data from each numerical variable----------


for (i in num_var) {
  print(i)
  val = data_frame[,i][data_frame[,i] %in% boxplot.stats(data_frame[,i])$out]
  print(length(val))
  print(val)
}
#Remove all the rows which contains outliers because less outliers were observed and it might not impact the model after deletion of rows
for (i in num_var) {
  val = data_frame[,i][data_frame[,i] %in% boxplot.stats(data_frame[,i])$out]
  data_frame = data_frame[which(!data_frame[,i] %in% val),]
```

```
}


#checkingany missing value found

sum(is.na(data_frame))


#checking any outlier found

for (i in num_var) {

  val = data_frame[,i][data_frame[,i] %in% boxplot.stats(data_frame[,i])$out]

}


length(val)


#Correlation plot for detecting the insignificant numeric variables which are highly correlated

library(corrgram)

corrgram(na.omit(data_frame))

dim(data_frame)

corrgram(data_frame[,num_var],order = F, upper.panel = panel.pie, text.panel = panel.txt, main = "correlation plot" )


#now we are going for feature selection means which variable is most significant in predicted the dependent variable

cat_var = c("season","yr","mnth","holiday","workingday","weathersit","weekday","dteday")

#performing ANOVA test on categorical variable against dependent variable

av_test = aov(cnt ~   season + yr + mnth + holiday + workingday + weekday  +weathersit , data = data_frame)
```

```
summary(av_test)


#by performing anova we came to know that every categorical variable is significant for us and we need not remove any variable


#Dimension reduction(selecting the data required for our model)


data = subset(data_frame,select = -c(instant,casual,registered,temp))

#column names of processed data

names(data)


#writing the processed data into hard disk

write.csv(data,"processed_data.csv", row.names = F)



#-----------MODEL ----------------------------------------

#removing all the objects from R environment except processed data

rmExcept("data")

#DIviding the dataset into train and test data using sampling


train_index = sample(1:nrow(data), 0.8* nrow(data))

train = data[train_index,]

test = data[-train_index,]
```

```
##__  Decision tree regression model development


fit = rpart(cnt ~. , data = train, method = "anova")


#########  predict results for the test case dataset


predictions_DT_reg = predict(fit , test[,-12])


#names(test)


library("DMwR")

library("mltools")


#Error coefficient method used here is RMSLE Root Mean Square Log Error


rmsle( predictions_DT_reg,test[,12]) #0.25



library("rattle")

fancyRpartPlot(fit)


#install.packages("pROC")

library("pROC")
```

```
##_______ RANDOM FOREST MODEL DEVELOPMENT  _____________________#


#Random Forest Model

library("randomForest")



RandomForest_model = randomForest(cnt~., train, ntree = 100)

str(data)


as.Date(data$dteday)



#Extract the rules generated as a result of random Forest model

library("inTrees")

rules_list = RF2List(RandomForest_model)


#Extract rules from rules_list

rules = extractRules(rules_list, train[,-12])

rules[1:2,]


#Convert the rules in readable format

read_rules = presentRules(rules,colnames(train))

read_rules[1:2,]
```

```
#Determining the rule metric
rule_metric = getRuleMetric(rules, train[,-12], train$cnt)
rule_metric[1:2,]

#Prediction of the target variable data using the random Forest model
RandomForest_prediction = predict(RandomForest_model,test[,-12])
regr.eval(test[,12], RandomForest_prediction, stats = 'rmse')
rmsle( RandomForest_prediction , test[,12]) #0.17




####----------STATISTICAL MODEL----------#########

## DEVELOPMENT OF LINEAR REGRESSION MODEL


library("usdm")
LR_data_select = subset(data, select = -(dteday))
colnames(LR_data_select)
vif(LR_data_select[,-12])
vifcor(LR_data_select[,-12], th=0.9)
```

```
####Execute the linear regression model over the data

linearRegression_model = lm(cnt~. , data = train)

summary(linearRegression_model)


colnames(test)


#Predict the data

linearRegression_model_predict_data = predict(linearRegression_model, test[,1:12])

install.packages("Metrics")

library("Metrics")

rmsle(linearRegression_model_predict_data,test[,12])
```