SAI SRINIVAS BANALA

(Employee Absenteeism Project Report)

Contents

Chapter 1

Introduction

1.1. Problem Statement

XYZ is a courier company. As we appreciate that human capital plays an important role in collection, transportation and delivery. The company is passing through genuine issue of Absenteeism.

In which we have to predict the Employee absenteeism in hours using the given data in which 20 variables are independent variables and one dependent variable which we are going to predict.

1. What changes company should bring to reduce the number of absenteeism?

2. How much losses every month can we project in 2011 if same trend of absenteeism continues?

The Aim of this project is to resolve the above both issues.

1.2. Understanding of data

By looking into the data, we came to know that the dependent variable that which are going to predict is continuous, so it is not the classification model. We need to adopt Regression model for predicting the continuous variable which is count based on environmental and seasonal settings.

Out of 21 variables **Absenteeism time in hours** is the target variable.

Chapter 2

Methodology

2.1. Data Pre-processing:

This data pre-processing has many stages missing value analysis, outlier analysis, feature selection, feature scaling which the dimension reduction comes into existence.

For every data science problem this is the important stage that the concerned person should take care of. If the processing is perfect, then the model will be successful else we cannot predict the correct output.

Pre-processing of data is nothing but cleaning of the data removing the unwanted data and keeping the significant data which is useful for our model.

2.1.1. Data type Conversion

It means the data which we get from the client is raw data which may or may not contain proper shape so that we need to study the data and convert the data into the required data type like if the variables are continuous we need convert them to numerical variables and if there are classification variable then we need to convert into categories.

To do this firstly we need to import the data into the Working environment like R or python accordingly. Let’s look at the snippet of code in both the languages. Firstly, set the working directory and import the data.

df = read.xlsx("Absenteeism_at_work_Project.xls", sheetIndex = 1)- R language

there are different snippets for importing different kind of data like csv file, excel sheet etc. Snippet differs according to the data which we are going to import.

Now according to our dataset by studying on the data we came to know that

Numerical variables:

Transportation expense

Distance from Residence to Work

Service time

Age

Hit target

Son

Pet

Height

Weight

Absenteeism time in hours

Categorical variables:

Reason for absence

Month of absence

Day of the week

Seasons

Disciplinary failure

Education

Son

Social drinker

Social smoker

2.1.2. Missing value Analysis

The given dataset is having many values missing which we can detect using code below.

missing_value = data.frame(apply(data,2,function(x){sum(is.na(x))}))

Thus, here before proceeding we need to impute the missing values first using the best suited method of imputation.

After few hit and trials I find that KNN is slightly better than median and mean methods. However, median and KNN methods both are having the same imputed values in few cases.

I am finalizing KNN here to impute the missing values.

library(DMwR)
data = knnImputation(data,k=3)

2.1.3. Outlier Analysis:

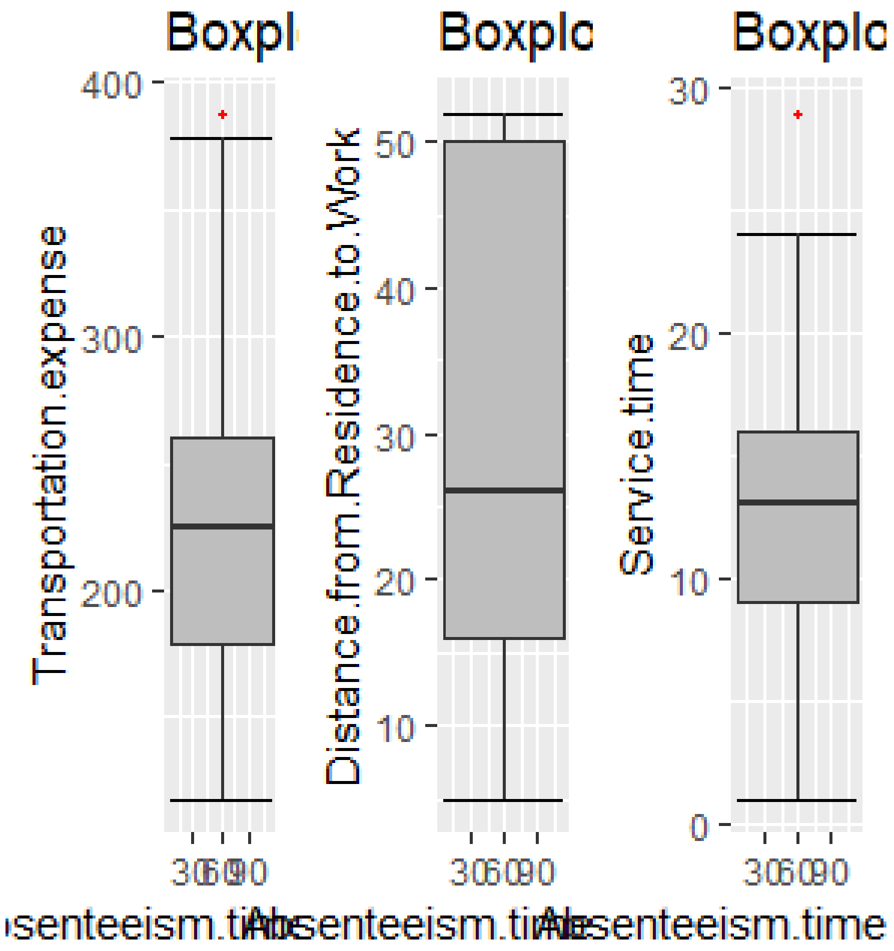
In statistics, an **outlier** is an observation point that is distant from other observations. An **outlier** may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An **outlier** can cause serious problems in statistical **analysis**.

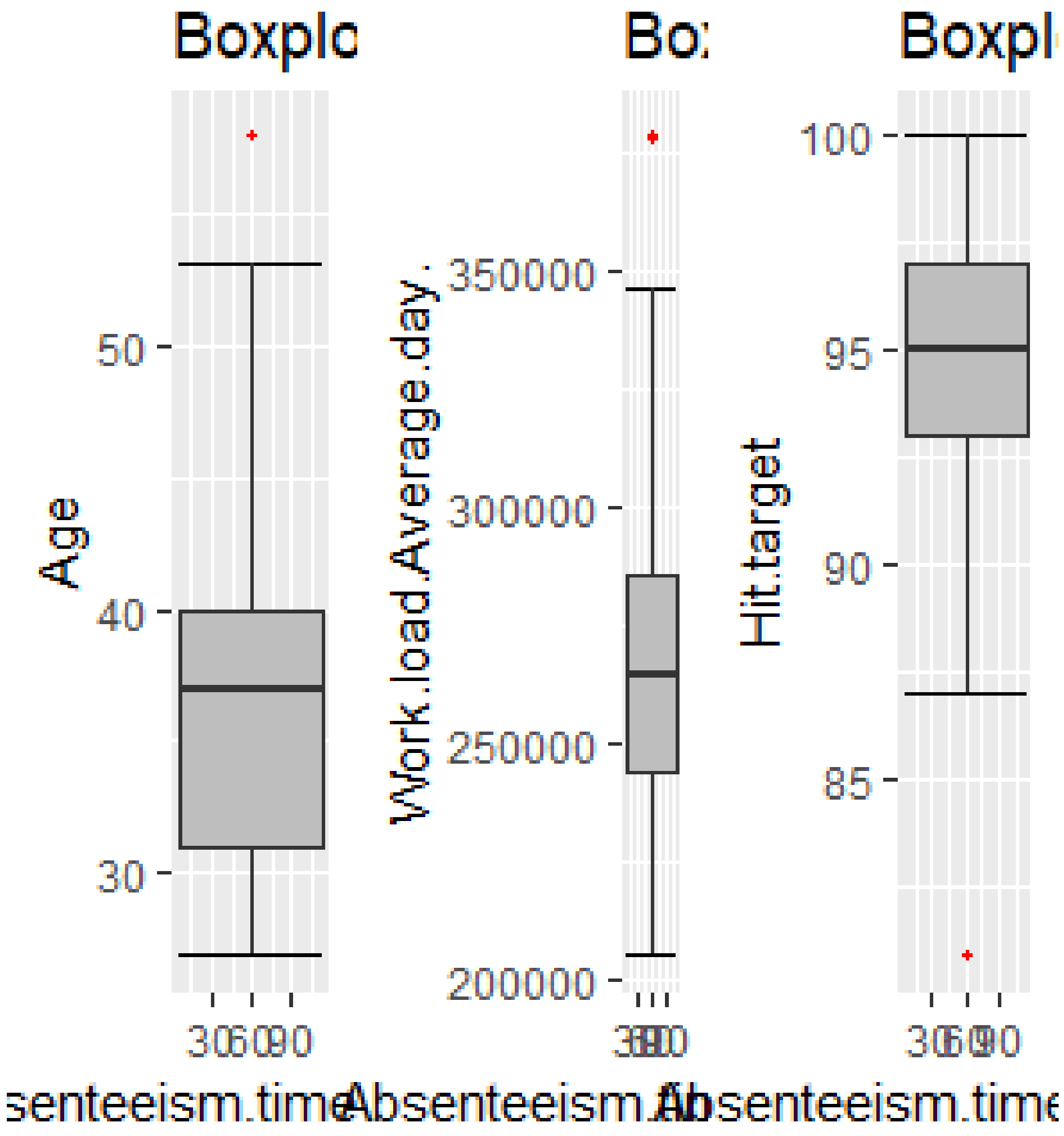
In simpler words outliers means taking all the observations of each column and finding the value which is falling away from the regular values.

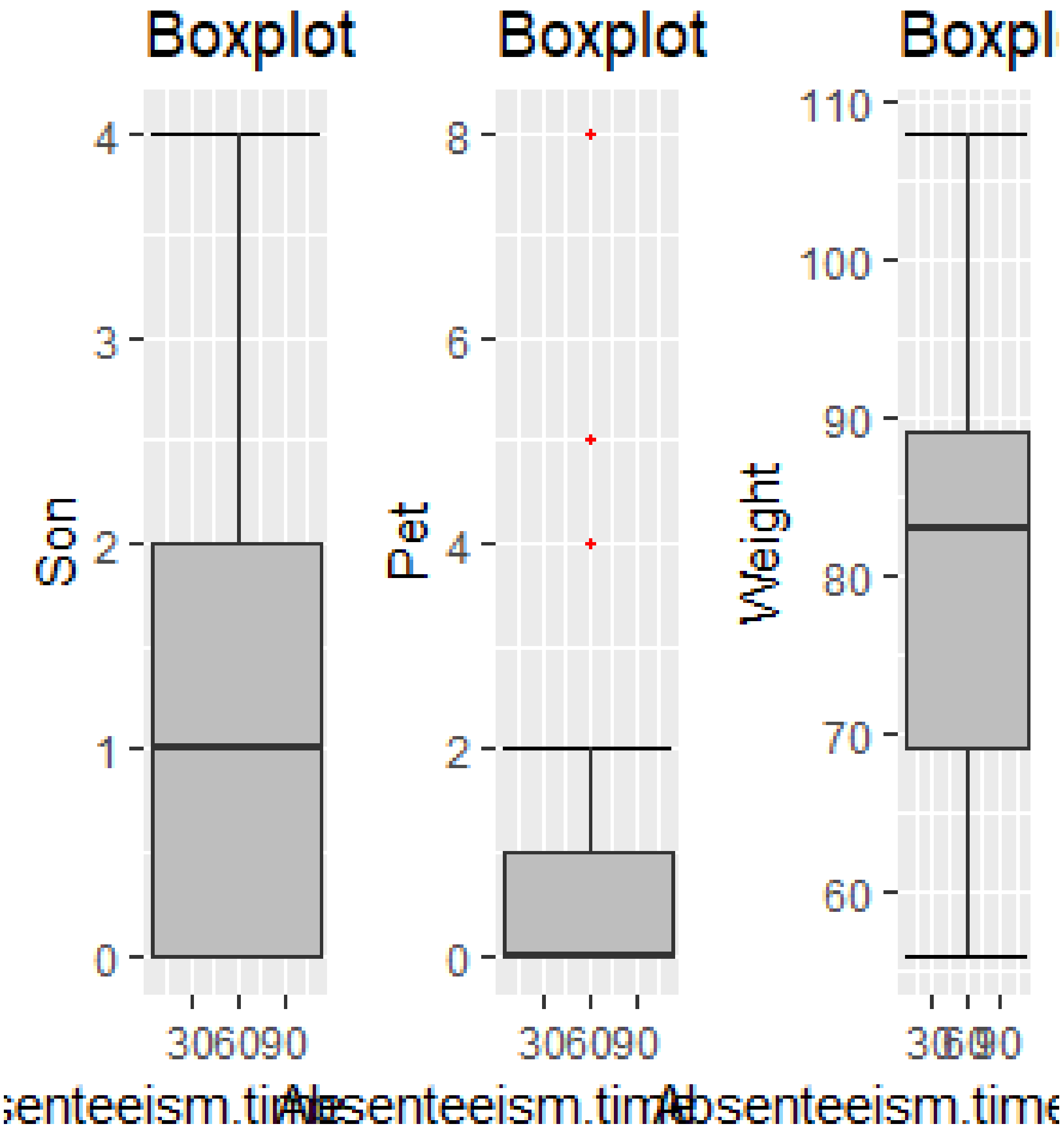
Generally, outliers are detected using boxplot method so that we can visualize clearly using this method, by passing some color coding in snippet we can easily identify the outliers.

Actually there are two ways to deal with the outliers one is imputing the outlier using mean, mode, median (central tendency) and another one is KNN imputation.

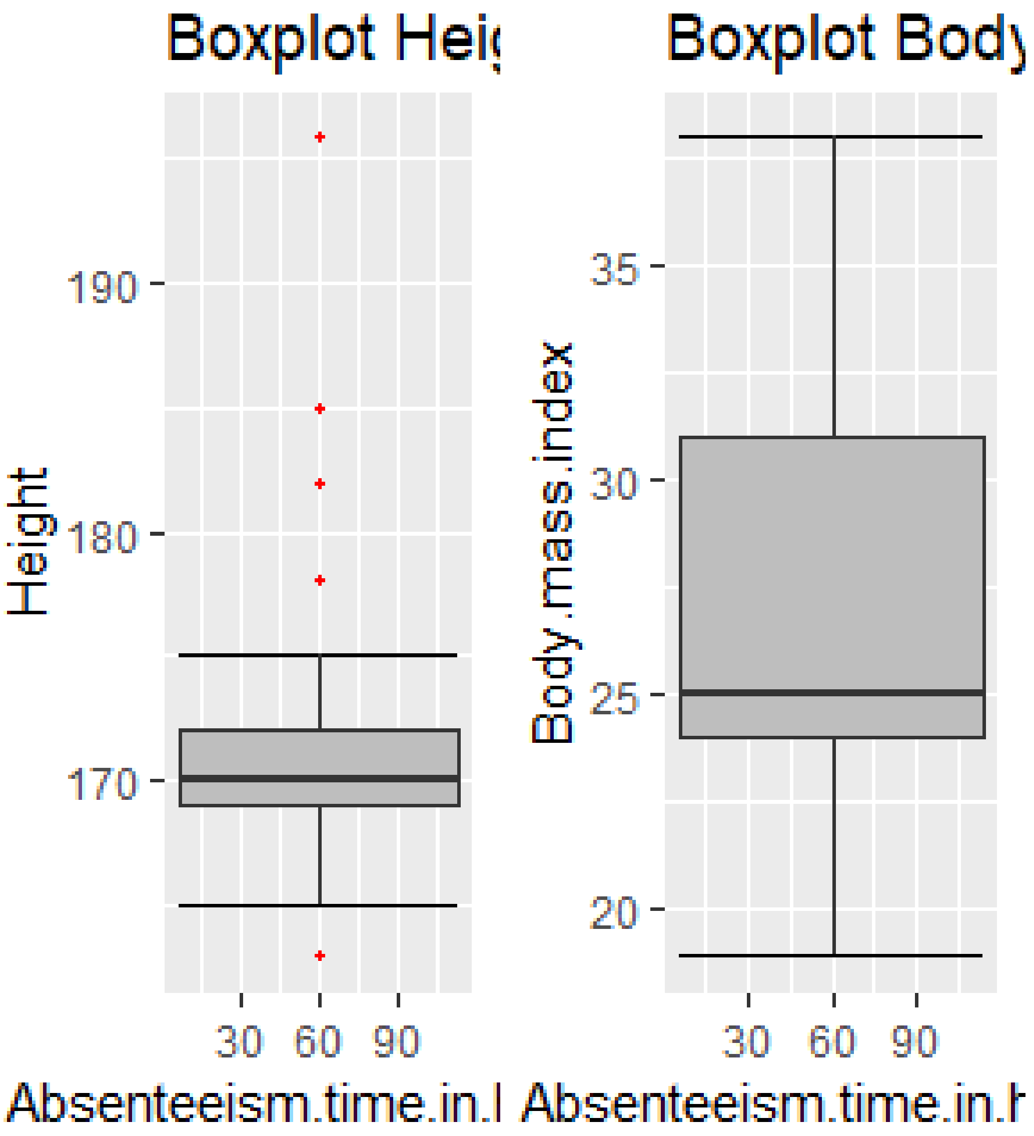
I have imputed the outliers using KNN imputation method by following the accuracy given by remaining two methods.




Boxplo
Bo:
Boxpl
Age
Work.load.Average.day.
Hit.target
50
40
30
350000
300000
250000
200000
100
95
90
85
306090
300
306090
senteeism.time
Absenteeism.ti
Absenteeism.time



Boxplot
Boxplot
Boxpl
Son
Pet
Weight
4
3
2
1
0
8
6
4
2
0
110
100
90
80
70
60
306090
306090
306090
senteeism.time
Absenteeism.time
Absenteeism.time



Boxplot Heig
Height
190
180
170
30
60
90
Absenteeism.time.in.l
Boxplot Body
Body.mass.index
35
30
25
20
30
60
90
Absenteeism.time.in.h

Here the box plot displays outliers of each variable. Now we have to remove these outliers.

In order to remove the outliers first we need to replace each of the outlier values with NA and then impute using already selected most suitable method of imputation KNN.

Having a look at this data I feel that outliers available in Absenteeism time in hours must not be treated. This is because it may be possible that a single person remain absent throughout the month and considering the abruptally high absent time as outlier may make the model less efficient.

So, I only resolved the outliers of rest of the variables and not the target variable.

2.1.4. Feature Selection

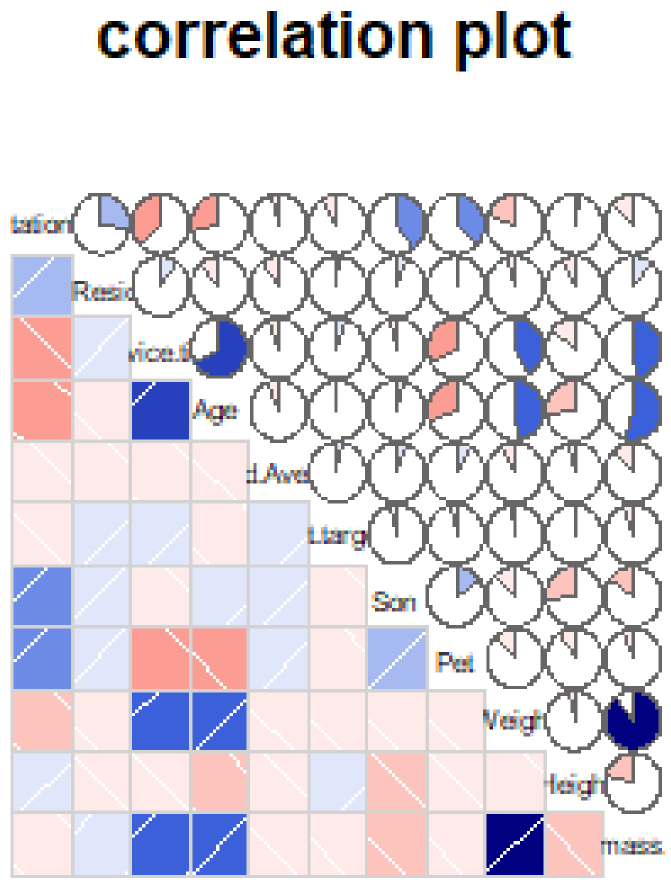
Feature selection is nothing but selection the independent variables which are more significant for implementing and developing the model. Technically, finding the significant variables and finding the highly correlated variables either negatively or positively.

There should be highly correlation between dependent and independent variable and no correlation between independent variables. For this there are different methods like correlation plot, chi-squared test of independence, ANOVA test.

Correlation plot: It is used for finding the highly correlated numerical variables. It is also used to find whether there is any multicollinearity problem.

Chi-Squared Test: It is used to find the independence between one categorical variable and numerical variable.

ANOVA: It is used to find the significance of variables against target variable



The correlation plot shows how the variables are correlated. Here we can observe that Weight and BMI are highly positively correlated and thus they can incur Multicollinearity in the model if both the variables are feed into the model.

Thus it is required to remove one of the variable. I am deleting BMI here and keeping weight as relevant feature because weight is a basic property whereas BMI is a derived value based on weight and height.

Now we are done with the feature selection of numerical variable.

Coming to the selection of Categorical variable I have used ANOVA test over the features.

```
library("lsr")

anova_test = aov(Absenteeism.time.in.hours ~ ID + Day.of.the.week + Education +
Social.smoker + Social.drinker + Reason.for.absence + Seasons +
Month.of.absence + Disciplinary.failure, data = data)

summary(anova_test)
```

Anova uses one categorical and one numerical variable to calculate the relevancy of that particular variable.

Using the pr probability value generated by ANOVA test we can select those variables which are having p value less than 0.05.

Here one point must be considered. We have to do our calculation according to month in which employees are making absence. Then I went through the ANOVA test to find the significant categorical variable against the target variable. I came to know that every variable is significant.

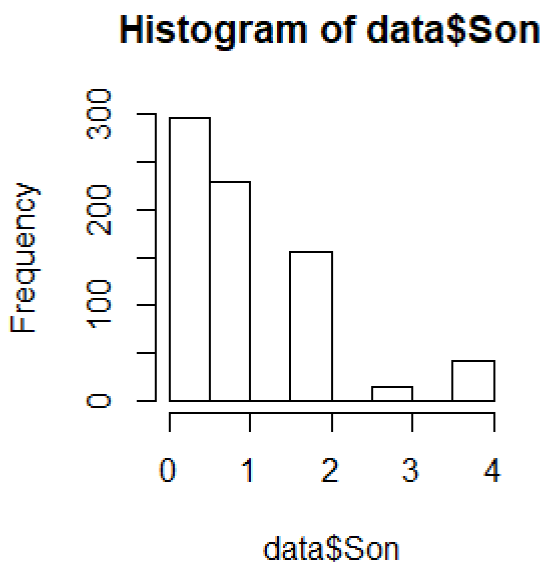
2.1.5. Feature Scaling:

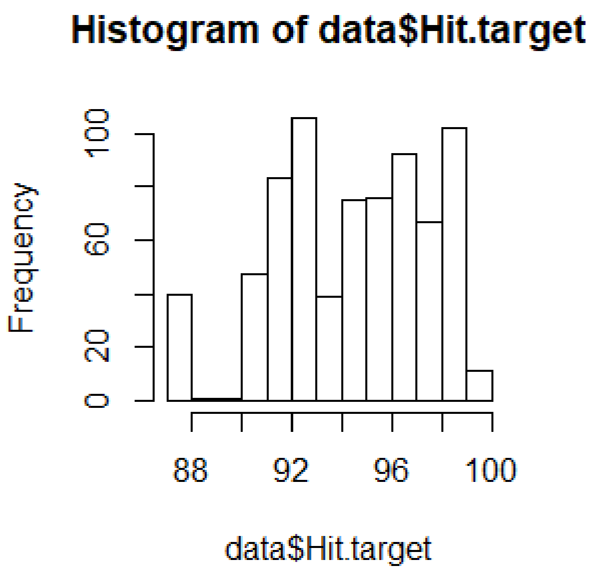
Feature scaling is a technique in which the dataset having quite different range of values are subjected to scale.

In order to scale the features there are usually two methods:-

1. Standardization

2. Normalization





The above two randomly chosen variable's histogram plot is indicating that the data distributed is not normal.

In this case we have to use Normalization to scale our features.

```
num_names = c("Transportation.expense","Distance.from.Residence.to.Work","Service.time","Age","Work.load.Average.day.","Hit.target","Son","Pet","Weight","Height")


for (i in num_names) {
    print(i)
    data_selected[,i] = ((data_selected[,i] - min(data_selected[,i])) /
                  (max(data_selected[,i]) - min(data_selected[,i])))
}
```

2.1.6. Dimension Reduction

Dimension reduction is nothing but removing the unwanted variables which are not required for the model that which we have extracted from correlation plot and ANOVA test.

I removed ID,Body.mass.index, Education, Social.smoker, Social.drinker, Seasons, Disciplinary.failurevariables which the data has been reduced to 15 variables. They I came to know that they were insignificant for model development

2.2. Modelling

2.2.1. Decision Tree :

Decision tree builds regression or classification 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**.

A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target.

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.

2.2.2. Random Forest:

The Random Forest is one of the most effective machine learning models for predictive analytics, making it an industrial workhorse for machine learning.

The **random forest** model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x)=f0(x)+f1(x)+f2(x)+...$$

where the final model g is the sum of simple base models fi. Here, each base classifier is a simple **decision tree**. This broad technique of using multiple models to obtain better predictive performance is called **model ensembling**. In random forests, all the base models are constructed independently using a **different subsample** of the data.

2.2.3. Model Selection:

Modelling is nothing but applying the machine learning algorithms according to the data and extract the patterns from the data. When the new data comes we feed the data into the model we implemented and predict the target values.

By observing the we came to that we need to predict the **Absenteeism in Hours** variable which is continuous variable therefore, the problem statement does not come under the classification model it comes under regression model.

After applying various regression models, we need to find the error metrics for each model and for which model the error is less we need to freeze that model. Technically, it is called as performance evaluation of the model.

Firstly, we divide the given data into parts train data and test data.

Train data: 80% of the total data is called as train data from which the hidden patterns are extracted by applying machine learning algorithms.

Test data: Remaining 20% of the data is called as train data. test data is fed into the model to predict the target variables and compared with the actual target variable and the error metrics are calculated, to find the accuracy of the model.

After applying several regression models like Decision tree, Random forest and Random regression models, I concluded that Random Forest with 100 trees is giving better performance with less error compared to remaining regression models.

Therefore, I am freezing Random Forest Regression model.

Chapter 3

Model Evaluation

As the problem statement belongs to time series prediction I have chosen RMSE as the error metric for model evaluation.

Decision tree:

```
library("DMwR")
regr.eval(test[,13], predicted_data, stats = 'rmse')
```

Thus in Decision tree regression model the error is 9.10 which tells that our model is 90.9% accurate

RandomForest Model

```
regr.eval(test[,13], RF_prediction, stats = 'rmse')
```

Thus the error rate in Random Forest Model is 8.78% and the accuracy of the model is 100-8.78 = 91.22%

After applying Linear Regression, Random Forest and Decision Tree I found that Random Forest with number of trees = 100 is giving the best accuracy of the model

Loss Prediction

I feel that loss is the total amount of unit of work which is not done or remains pending. This pending work might have been easily achieved if the employees were regular.

Formula that I used for loss calculation

Loss = absenteeism time * work load average/day

R CODE

```
 #Clean the R environment

rm(list = ls())


#Set working Directory in which the data exist

setwd("D:/Project")

getwd()


#installing all the required packages for model development and preprocessing

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","pRO
C","randomForest","inTrees"

     ,"usdm","Metrics")


#Load XLSX library

#install.packages("xlsx")

library("xlsx")


#loading the data into R environment

df = read.xlsx("Absenteeism_at_work_Project.xls", sheetIndex = 1)
```

```
#observe the class of df object

class(df)


#Observe the dimension of df

dim(df)


#get the names of variable

colnames(df)


#checking the data types of data

str(df)


#EXPLORATORY DATA ANALYSIS


#note that all the variables of df are of numeric type


#Now we have to observe and explore which are the variables that are continuous and
which are categorical variables.


#conversion of required data types into categorical variables a/c to the data

df$ID = as.factor(as.character(df$ID))

df$Day.of.the.week = as.factor(as.character(df$Day.of.the.week))

df$Education = as.factor(as.character(df$Education))
```

```
df$Social.drinker = as.factor(as.character(df$Social.drinker))

df$Social.smoker = as.factor(as.character(df$Social.smoker))

df$Reason.for.absence = as.factor(as.character(df$Reason.for.absence))

df$Seasons = as.factor(as.character(df$Seasons))

df$Month.of.absence = as.factor(as.character(df$Month.of.absence))

df$Disciplinary.failure = as.factor(as.character(df$Disciplinary.failure))


#MISSING VALUE ANALYSIS

sum(is.na(df$Transportation.expense))

missing_value = data.frame(apply(df,2,function(x){sum(is.na(x))}))

names(missing_value)[1] = "Missing_data"

missing_value$Variables = row.names(missing_value)

row.names(missing_value) = NULL


#Rearange the columns

missing_value = missing_value[, c(2,1)]


#place the variables according to their number of missing values.

missing_value = missing_value[order(-missing_value$Missing_data),]


#Calculate the missing value percentage

missing_value$percentage = (missing_value$Missing_data/nrow(df) )* 100
```

```
#Store the missing value information in a csv file

write.csv(missing_value,"Project_Missing_value.csv", row.names = F)


#Since the calculated missing percentage is less than 5%

#Thus there is no need to drop any variable due to less number of data

#Now generate a missing value in the dataset and try various imputation methods on it

#make a backup for data object


#df_back = df


## create a missing value in any variable


#*******

#df[79,6] = 361

#ACTUTAL vALUE  = 361

#Median = 225

#KNN = 315

#Mean  = 220



##Now apply mean method to impute this generated value and observe the result

#df$Transportation.expense[is.na(df$Transportation.expense)]  =
mean(df$Transportation.expense, na.rm = T)
```

```
#sum(is.na(df))


##Now apply median method to impute the missing value

#df$Transportation.expense[is.na(df$Transportation.expense)] =
median(df$Transportation.expense, na.rm = T)


#now apply KNN method to impute

#install.packages('DMwR')

library(DMwR)

df = knnImputation(df,k=3)

sum(is.na(df))


#Both median and KNN are giving same result we can choose any one for furthur
implementation

#On analysis of different data using median and KNN i find that KNN is more
accurate than median for imputation


#Now load the data again and impute the missing value by KNN

#Check presence of missing values once to confirm


apply(df,2, function(x){sum(is.na(x))})


#NO missing value is found
```

```
#create subset of the dataset which have only numeric varaiables


numeric_index = sapply(df, is.numeric)

numeric_data = df[,numeric_index]

numeric_data = as.data.frame(numeric_data)


n_data = colnames(numeric_data)[-12]

str(df)


#draw the boxplot to detect outliers


#install.packages('ggplot2')

library("ggplot2")


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

  assign(paste0("gn",i), ggplot(aes_string( y = (n_data[i]), x= 
"Absenteeism.time.in.hours") , data = subset(df)) +

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

           geom_boxplot(outlier.color = "red", fill = "grey", outlier.shape = 20, 
outlier.size = 1, notch = FALSE)+

           theme(legend.position = "bottom")+

           labs(y = n_data[i], x= "Absenteeism.time.in.hours")+

           ggtitle(paste("Boxplot" , n_data[i])))

  #print(i)
```

```
}

options(warn = -1)

#Now plotting the plots

gridExtra::grid.arrange(gn1, gn2,gn3, ncol=3)
gridExtra::grid.arrange(gn4,gn5,gn6, ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9, ncol =3)
gridExtra::grid.arrange(gn10,gn11, ncol =2 )

#calculating the outliers found in each variable and printing them.

for (i in n_data) {
  print(i)
  val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out]
  print(length(val))
  print(val)
}

#Making each outlier as NA for imputation
for (i in n_data) {
  val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out]
  df[,i][df[,i] %in% val] = NA
}


#Check number of missing values
sum(is.na(df))


#Impute the values using KNN method
df = knnImputation(df, k=3)


#Check again for missing value if present in case
sum(is.na(df))


#Correlation plot
#install.packages("corrgram")
library(corrgram)
corrgram(na.omit(df))
dim(df)
corrgram(df[,n_data],order = F, upper.panel = panel.pie, text.panel = panel.txt, main =
"correlation plot" )


#Here we can see in correlation plot that Body Mass Index and weight are highly
positively correlated
#I am removing the Body Mass Index variable because weight is a basic variable.
```

```
# Select the relevant numerical features

num_var = c("Transportation.expense", "Distance.from.Residence.to.work", 
"Service.time",
            "Age", "Work.load.Average.day","Hit.target", "Son", "Pet", "Weight", 
"Height")

#install.packages("lsr")

library("lsr")

anova_test = aov(Absenteeism.time.in.hours ~ ID + Day.of.the.week + Education +
                  Social.smoker + Social.drinker + Reason.for.absence + Seasons + 
Month.of.absence + Disciplinary.failure, data = df)
summary(anova_test)

#Dimension reduction
data = subset(df,select = -c(ID,Body.mass.index, Education, Social.smoker, 
Social.drinker, Seasons, Disciplinary.failure))

#draw histogram on random variables to check if the distributions are normal

hist(data$Hit.target)
hist(data$Work.load.Average.day.)
hist(data$Son)

hist(data$Weight)

hist(data$Transportation.expense)

hist(data$Distance.from.Residence.to.Work)

hist(data$Service.time)

hist(data$Age)

hist(data$Pet)


#THE variables are not seems as normally distributed thus we have to perform
Normalisation instead of standardisation

#Now select numerical variables from data_selected Object to perform normalization


print(sapply(data ,is.numeric))


rm(gn1,gn2,gn3,gn4,gn5,gn6,gn7,gn8,gn9,gn10,gn11)




#Feature Scaling

#num_names object contains all the numerical features of selected features

#Normalisation

rm(cnames)

cnames = c("Transportation.expense",
"Distance.from.Residence.to.Work","Service.time","Age","Work.load.Average.day.",

        "Hit.target","Son","Pet","Weight","Height")
```

```
for(i in cnames){
  print(i)
  data[,i] = (data[,i] - min(data[,i]))/
    (max(data[,i] - min(data[,i])))
}

#Till here we are having our data being normalised and stored in data_selected object
#Write this cleaned data in csv form
write.csv(data,"processed.csv", row.names = F)

#_____________________________________________________________________
____________
#install.packages("DataCombine")
library(DataCombine)
rmExcept("data")
colnames(data)
#View(data)

#___________ MODEL DEVELOPMENT _______________________

## Divide the data into train and test data using simple random sampling
#install.packages("rpart")
```

```
library("rpart")

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


train = data[train_index,]

test = data[-train_index,]


#Building the Decision Tree Regression Model


regression_model = rpart(Absenteeism.time.in.hours ~. , data = train, method =
"anova")


#########  predicting for the test case

predicted_data = predict(regression_model , test[,-14])


##Calculate RMSE to analyze performance of Decision tree regression model

#RMSE is used here as the data is of time series


library("DMwR")

regr.eval(test[,13], predicted_data, stats = 'rmse')


##__ Thus in Decision tree regression model the error is 9.10 which tells that our
model is 90.9% accurate
```

```
##_________________RANDOM FOREST MODEL
____________________________________##

#install.packages("randomForest")

library("randomForest")

RF_model = randomForest(Absenteeism.time.in.hours~. , train,  ntree=100)


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

#install.packages("inTrees")

library("inTrees")

rules_list = RF2List(RF_model)


#Extract rules from rules_list

rules = extractRules(rules_list, train[,-14])

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[,-14], train$Absenteeism.time.in.hours)

rule_metric[1:2,]
```

```
#Prediction of the target variable data using the random Forest model

RF_prediction = predict(RF_model,test[,-14])

regr.eval(test[,13], RF_prediction, stats = 'rmse')


#Thus the error rate in Random Forest Model is 8.78% and the accuracy of the model
is 100-8.78 = 91.22%


###________________ LINEAR REGRESSION
_________________________________

install.packages("usdm")

library("usdm")

LR_data_select = subset(data,select = -c(Reason.for.absence,Day.of.the.week))

colnames(LR_data_select)

vif(LR_data_select[,-11])

vifcor(LR_data_select[,-11], th=0.9)


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

lr_model = lm(Absenteeism.time.in.hours~. , data = train)


summary(lr_model)

colnames(test)


#Predict the data

LR_predict_data = predict(lr_model, test[,1:13])
```

```
regr.eval(test[,13], LR_predict_data, stats = 'rmse')



##________ Till here we have implemented Decision Tree, Random Forest and Linear
Regression. Among all of these,

##________ Random Forest is having highest accuracy.



#_________ Now we have to predict loss for each month _____________


#__ To calculate loss month wise we need to include month of absence variable again
in our data set

# LOSS = Work.load.average.per.day * Absenteeism.time.in.hours


colnames(data)

data$loss = data$Work.load.Average.day. * data$Absenteeism.time.in.hours


i=1


#__________________________________________________________________________
___________


# NOW calculate Month wise loss encountered due to absenteeism of employees
```

```
#Calculate loss in january

loss_jan = as.data.frame(data$loss[data$Month.of.absence %in% 1])

names(loss_jan)[1] = "Loss"

sum(loss_jan[1])

write.csv(loss_jan,"jan_loss.csv", row.names = F)


#Calculate loss in febreuary

loss_feb = as.data.frame(data$loss[data$Month.of.absence %in% 2])

names(loss_feb)[1] = "Loss"

sum(loss_feb[1])

write.csv(loss_feb,"feb_loss.csv", row.names = F)


#Calculate loss in march


loss_march = as.data.frame(data$loss[data$Month.of.absence %in% 3])

names(loss_march)[1] = "Loss"

sum(loss_march[1])

write.csv(loss_march,"march_loss.csv", row.names = F)


#Calculate loss in april


loss_apr = as.data.frame(data$loss[data$Month.of.absence %in% 4])
```

```
names(loss_apr)[1] = "Loss"
sum(loss_apr[1])
write.csv(loss_apr,"apr_loss.csv", row.names = F)


#calculate loss in may

loss_may = as.data.frame(data$loss[data$Month.of.absence %in% 5])
names(loss_may)[1] = "Loss"
sum(loss_may[1])
write.csv(loss_may,"may_loss.csv", row.names = F)


#calculate in june

loss_jun = as.data.frame(data$loss[data$Month.of.absence %in% 6])
names(loss_jun)[1] = "Loss"
sum(loss_jun[1])
write.csv(loss_jun,"jun_loss.csv", row.names = F)

#Calculate loss in july

loss_jul = as.data.frame(data$loss[data$Month.of.absence %in% 7])
```

```
names(loss_jul)[1] = "Loss"

sum(loss_jul[1])

write.csv(loss_jul,"jul_loss.csv", row.names = F)


#calculate loss in august


loss_aug = as.data.frame(data$loss[data$Month.of.absence %in% 8])

names(loss_aug)[1] = "Loss"

sum(loss_aug[1])

write.csv(loss_aug,"aug_loss.csv", row.names = F)


#Calculate loss in september


loss_sep = as.data.frame(data$loss[data$Month.of.absence %in% 9])

names(loss_sep)[1] = "Loss"

sum(loss_sep[1])

write.csv(loss_sep,"sep_loss.csv", row.names = F)


#calculate loss in october


loss_oct = as.data.frame(data$loss[data$Month.of.absence %in% 10])

names(loss_oct)[1] = "Loss"

sum(loss_oct[1])
```

```
write.csv(loss_oct,"oct_loss.csv", row.names = F)


#calculate loss in november


loss_nov = as.data.frame(data$loss[data$Month.of.absence %in% 2])
names(loss_nov)[1] = "Loss"
sum(loss_nov[1])
write.csv(loss_nov,"nov_loss.csv", row.names = F)


#calculate loss in december
loss_dec = as.data.frame(data$loss[data$Month.of.absence %in% 2])
names(loss_dec)[1] = "Loss"
sum(loss_dec[1])
write.csv(loss_dec,"dec_loss.csv", row.names = F)
```