**PROJECT 3**

**PROJECT NAME :- EMPLOYEMEE ABSENTEEISM**

**ASHISH KUMAR SAW**

**29th AUGUST 2018**

**TABLE OF CONTENTS:-**

1. **INTRODUCTION**
2. **PROBLEM STATEMENT**
3. **HYPOTHESIS GENERATION**
4. **LOADING PACKAGES AND DATA**
5. **UNDERSTANDING THE DATA**
6. **EXPLORATORY DATA ANALYSIS**
7. **UNIVARIATE ANALYSIS**
8. **BIVARIATE ANALYSIS**
9. **DATA PREPERATION**
10. **MISSING VALUE ANALYSIS**
11. **FEATURE SELECTION**
12. **FEATURE SCALING**
13. **MODELING**

**CHAPTER 1:-INTRODUCTION**

**PROBLEM STATEMENT:-**XYZ is a courier company. As we appreciate that human capital plays an important rolein collection, transportation and delivery. The company is passing through genuineissue of Absenteeism.

The aim of the problem is to find:-

1. Changes that should company bring to reduce the number of absenteeism.
2. Build a Predictive Model that could find out losses every month can we project in 2011 if same trend of absenteeism continues.

**HYPOTHESIS GENERATION:-**This is a very important stage in any machine learning process. It involves understanding the problem in detail by brainstorming as many factors as possible which can impact the outcome. It is done by understanding the problem statement thoroughly and before looking at the data.

Absenteeism refers to an employee's intentional or habitual absence from work. While employers expect workers to miss a certain number of workdays each year, excessive absences can equate to decreased productivity and can have a major effect on company finances, morale and other factors. This article looks at the causes of absenteeism, the costs of lost productivity, and what employers can do to reduce absenteeism rates in the workplace.

Causes of Absenteeism:-

1. **Bullying and Harassment:-**Employees who are bullied or harassed by coworkers and/or bosses are more likely to call in sick to avoid the situation.
2. **Burnout, Stress and Low Moral:-**Heavy workloads, stressful meetings/presentations and feelings of being unappreciated can cause employees to avoid going into work. Personal stress (outside of work) can lead to absenteeism.
3. **Childcare and Eldercare:**-Employees may be forced to miss work in order to stay home and take care of a child/elder when normal arrangements have fallen through (for example, a sick caregiver or a snow day at school) or if the dependent is ill or hurt.
4. **Depression:**- According to the National Institute of Mental Health, the leading cause of absenteeism is depression. Depression can lead to substance abuse if people turn to drugs or alcohol to self-medicate their pain or anxiety.
5. **Disengagement:**-Employees who are not committed to their jobs, coworkers and/or the company are more likely to miss work simply because they have no motivation to go.
6. **Illness:**-Injuries, illness and medical appointments are the most commonly reported reasons for missing work (though not always the actual reason). Not surprisingly, each year during the cold and flu season, there is a dramatic spike in absenteeism rates for both full-time and part-time employees.
7. **Injuries:**-Accidents can occur on the job or outside of work, resulting in absences. In addition to acute injuries, chronic injuries such as back and neck problems are a common cause of absenteeism.
8. **Job Hunting:**-Employees may call in sick to attend a job interview, visit with a headhunter or work on their résumés/CVs.
9. **Partial Shifts:**-Arriving late, leaving early and taking longer breaks than allowed are considered forms of absenteeism and can affect productivity and workplace morale.

**LOADING PACKAGES AND DATA:-**In R and Python we include multiple Packages and Libraries which perform various functions on the Dataset. Here we include packages for Reading the data, Manipulation of the data, Visualizing the data and finally Modeling the data.

#Load the Libraries

library(dplyr) # used for data manipulation and joining

library(ggplot2) # used for ploting

library(caret) # used for modeling

library(corrplot) # used for making correlation plot

library(xgboost) # used for building XGBoost model

library(cowplot) # used for combining multiple plots

library(readxl) # used for reading xls file

library(VIM) # used to impute missing value

library(corrgram) # used to plot corgram plot

library(glmnet) # used for Ridge Regression

#Reading the Data

train = read\_excel("Absenteeism\_at\_work\_Project.xls")

**UNDERSTANDING THE DATA:-**Initially we should understand our raw data thoroughly, i.e., we should explore the no. of features/columns and rows, datatype of the features, feature names and so on.

#Understand the Data

#Dimension of the Data

dim(train)

## [1] 740 21

Train dataset has 740 rows and 21 features.

#Features of Data

names(train)

[1] "ID" "Reason for absence" "Month of absence"

[4] "Day of the week" "Seasons" "Transportation expense"

[7] "Distance from Residence to Work" "Service time" "Age"

[10] "Work load Average/day" "Hit target" "Disciplinary failure"

[13] "Education" "Son" "Social drinker"

[16] "Social smoker" "Pet" "Weight"

[19] "Height" "Body mass index" "Absenteeism time in hours"

#Structure of the data

str(train)

Classes ‘tbl\_df’, ‘tbl’ and 'data.frame': 740 obs. of 21 variables:

$ ID :num 11 36 3 7 11 3 10 20 14 1 ...

$ Reason for absence :num 26 0 23 7 23 23 22 23 19 22 ...

$ Month of absence :num 7 7 7 7 7 7 7 7 7 7 ...

$ Day of the week :num 3 3 4 5 5 6 6 6 2 2 ...

$ Seasons :num 1 1 1 1 1 1 1 1 1 1 ...

$ Transportation expense :num 289 118 179 279 289 179 NA 260 155 235 ...

$ Distance from Residence to Work: num 36 13 51 5 36 51 52 50 12 11 ...

$ Service time :num 13 18 18 14 13 18 3 11 14 14 ...

$ Age :num 33 50 38 39 33 38 28 36 34 37 ...

$ Work load Average/day :num 239554 239554 239554 239554 239554 ...

$ Hit target :num 97 97 97 97 97 97 97 97 97 97 ...

$ Disciplinary failure :num 0 1 0 0 0 0 0 0 0 0 ...

$ Education :num 1 1 1 1 1 1 1 1 1 3 ...

$ Son :num 2 1 0 2 2 0 1 4 2 1 ...

$ Social drinker :num 1 1 1 1 1 1 1 1 1 0 ...

$ Social smoker :num 0 0 0 1 0 0 0 0 0 0 ...

$ Pet :num 1 0 0 0 1 0 4 0 0 1 ...

$ Weight :num 90 98 89 68 90 89 80 65 95 88 ...

$ Height :num 172 178 170 168 172 170 172 168 196 172 ...

$ Body mass index :num 30 31 31 24 30 31 27 23 25 29 ...

$ Absenteeism time in hours :num 4 0 2 4 2 NA 8 4 40 8 ...

Here, all the variables are numerical/continuous variables.

**CHAPTER 3:- EXPLORATORY DATA ANALYSIS**

*Exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.*

Converting features into right data types:

train$Seasons=as.factor(train$Seasons)

train$`Disciplinary failure`=as.factor(train$`Disciplinary failure`)

train$Education=as.factor(train$Education)

train$Son=as.factor(train$Son)

train$`Social drinker`=as.factor(train$`Social drinker`)

train$`Social smoker`=as.factor(train$`Social smoker`)

train$Pet=as.factor(train$Pet)

**UNIVARIATE ANALYSIS:-** In Univariate analysis we explore the variables individually. For continous variable we use Histogram and for categorical variable we use Barplot.

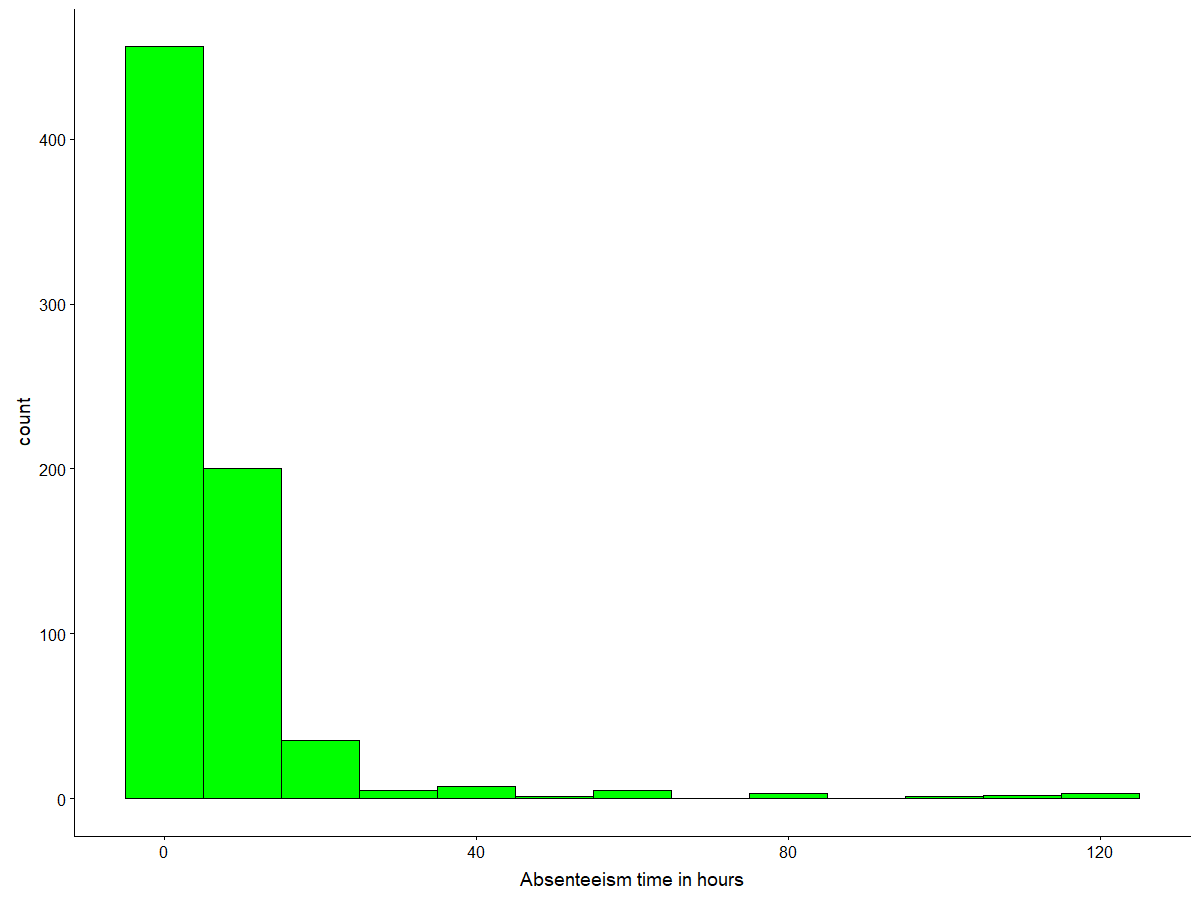
#Exploratory Data Analysis

#Univariate Analysis

#For targe variable

ggplot(data = train, aes(x = `Absenteeism time in hours`)) + geom\_histogram(color = "black",

fill = "green", binwidth = 10)



Here, the graph contains different distribution in the “Absent Time in Hours Graph”. Here, most of the employee are absent for hours between 0 to 25. The employee absent less for higher hours. It is Right Skewed.

#Now we will check Numerical independent variable

p1 = ggplot(data = train, aes(x = ID)) + geom\_histogram(color = "black",

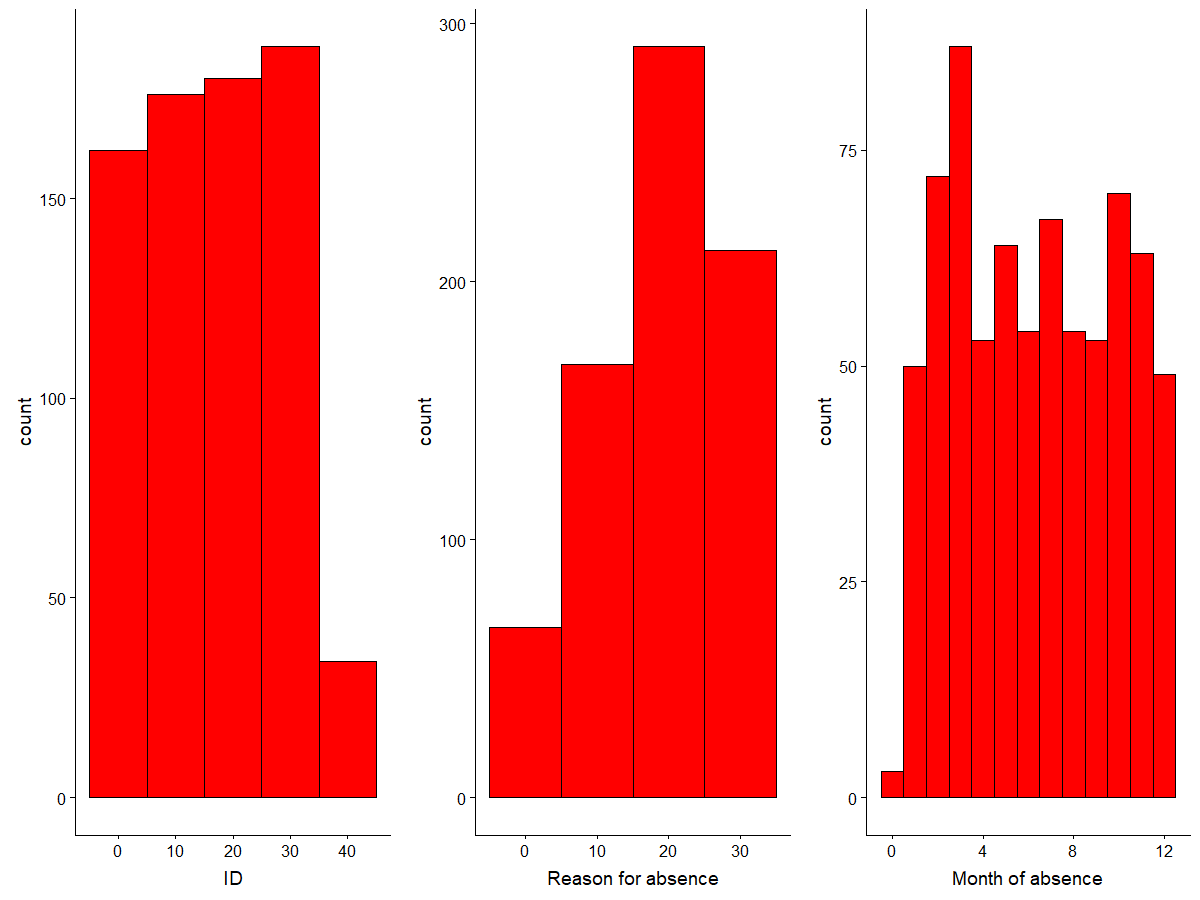
fill = "red", binwidth = 10)

p2 = ggplot(data = train, aes(x = `Reason for absence`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p3 = ggplot(data = train, aes(x = `Month of absence`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 1)



Here, ID is increasing till 35 and then decreases drasticallly, Reason of absence is increasing for higher values and Month of absence is uniformly distributed.

p4 = ggplot(data = train, aes(x = `Day of the week`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 1)

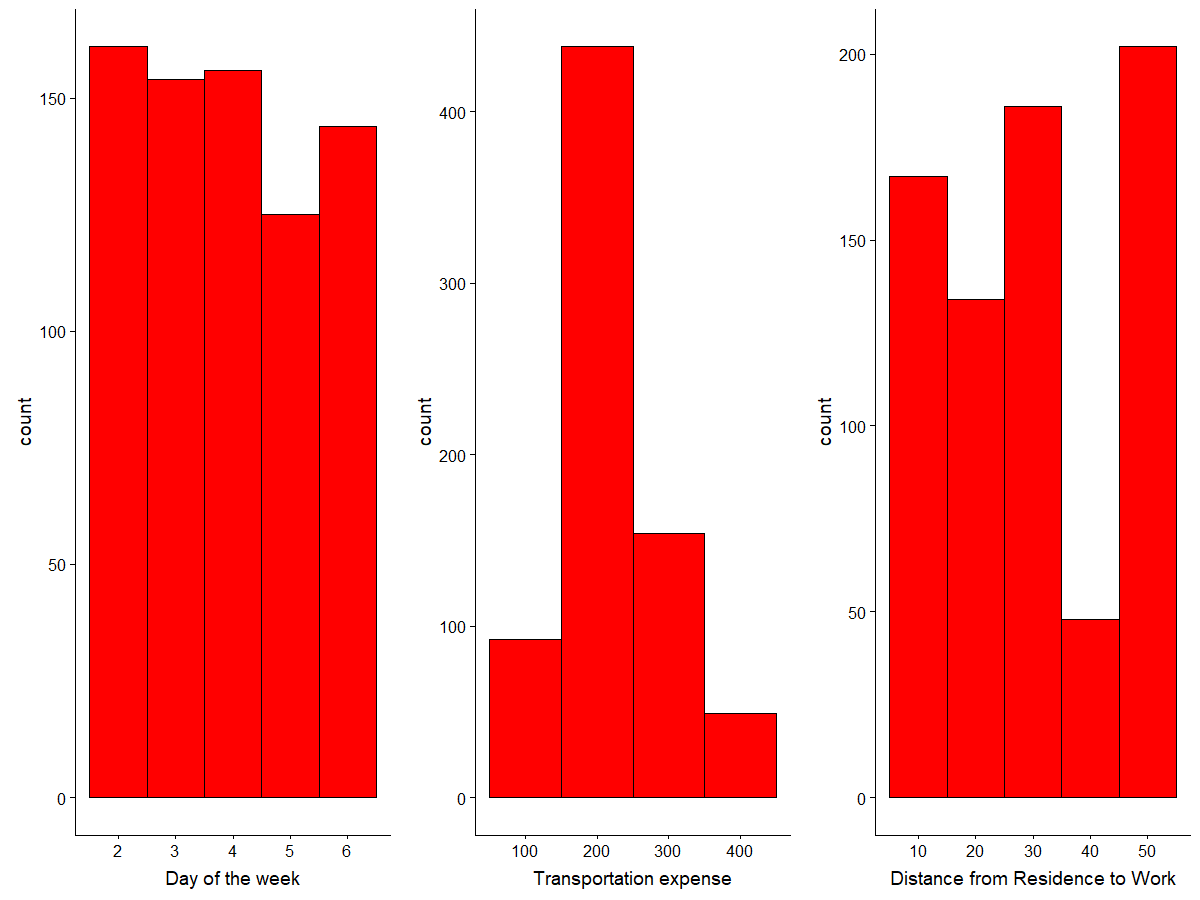
p5 = ggplot(data = train, aes(x = `Transportation expense`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 100)

p6 = ggplot(data = train, aes(x = `Distance from Residence to Work`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

plot\_grid(p4, p5, p6, nrow = 1)



Here, day of the week, Transportation expence and Distance from Residence to Work are uniformly distributed.

p7 = ggplot(data = train, aes(x = `Service time`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

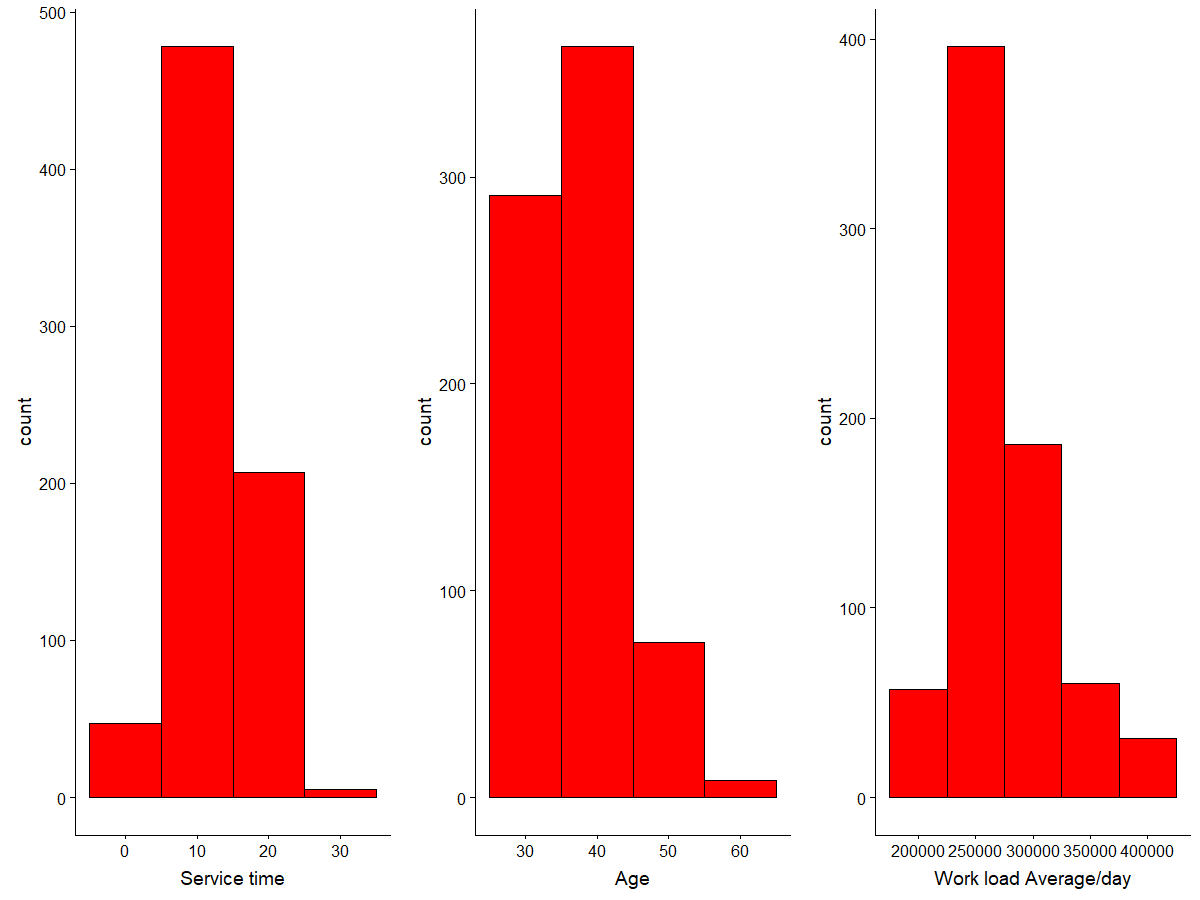
p8 = ggplot(data = train, aes(x = `Age`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p9 = ggplot(data = train, aes(x = `Work load Average/day`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 50000)

plot\_grid(p7, p8, p9, nrow = 1)



Here, Service time is uniformly distributed, Age variable is Right Skewed and Work load Average/day is almost uniformly distributed.

p10 = ggplot(data = train, aes(x = `Hit target`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 5)

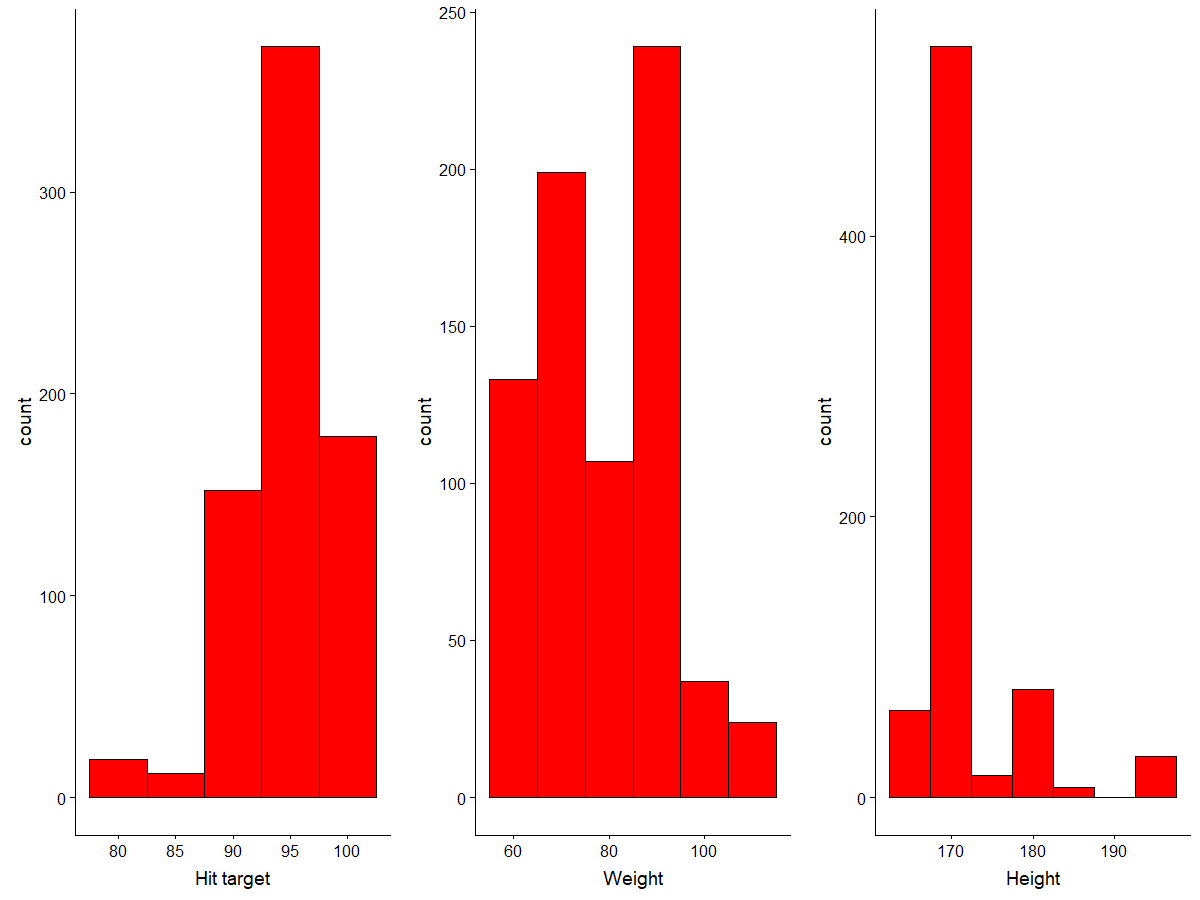
p11 = ggplot(data = train, aes(x = Weight)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p12 = ggplot(data = train, aes(x = Height)) + geom\_histogram(color = "black",

fill = "red", binwidth = 5)

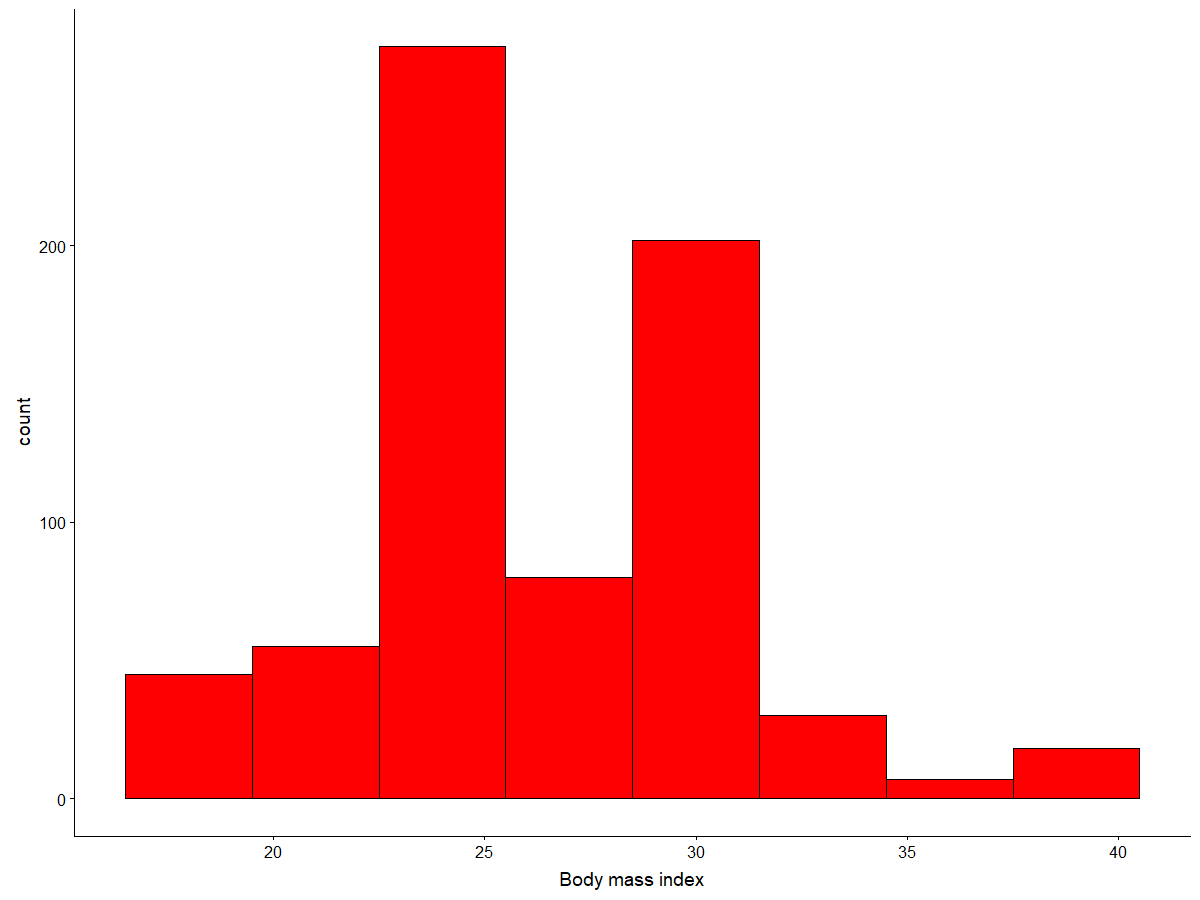
plot\_grid(p10, p11, p12, nrow = 1)



Here, Hit target is Left skewed, Weight is almost uniformly distributed and Height is non-uniformly distributed.

ggplot(data = train, aes(x = `Body mass index`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 3)

**

Here, Body mass index in uniformly distributed.

Univariate Analysis for Categorical Variables

#BarPlot for categorical variable

# Barplot for Seasons

p13 = ggplot(train, aes(x = Seasons)) + geom\_bar() +

labs(title = "Count of Seasons", x = "Different Seasons", y = "Count of different seasons") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

# Barplot for Disciplinary failure

p14 = ggplot(train, aes(x = `Disciplinary failure`)) + geom\_bar() +

labs(title = "Disciplinary Failure", x = "Different Disciplinary Failure", y = "Count of Disciplinary Failure") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

# Barplot for Education

p15 = ggplot(train, aes(x = Education)) + geom\_bar() +

labs(title = "Education", x = "Different Education", y = "Count of each Education") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

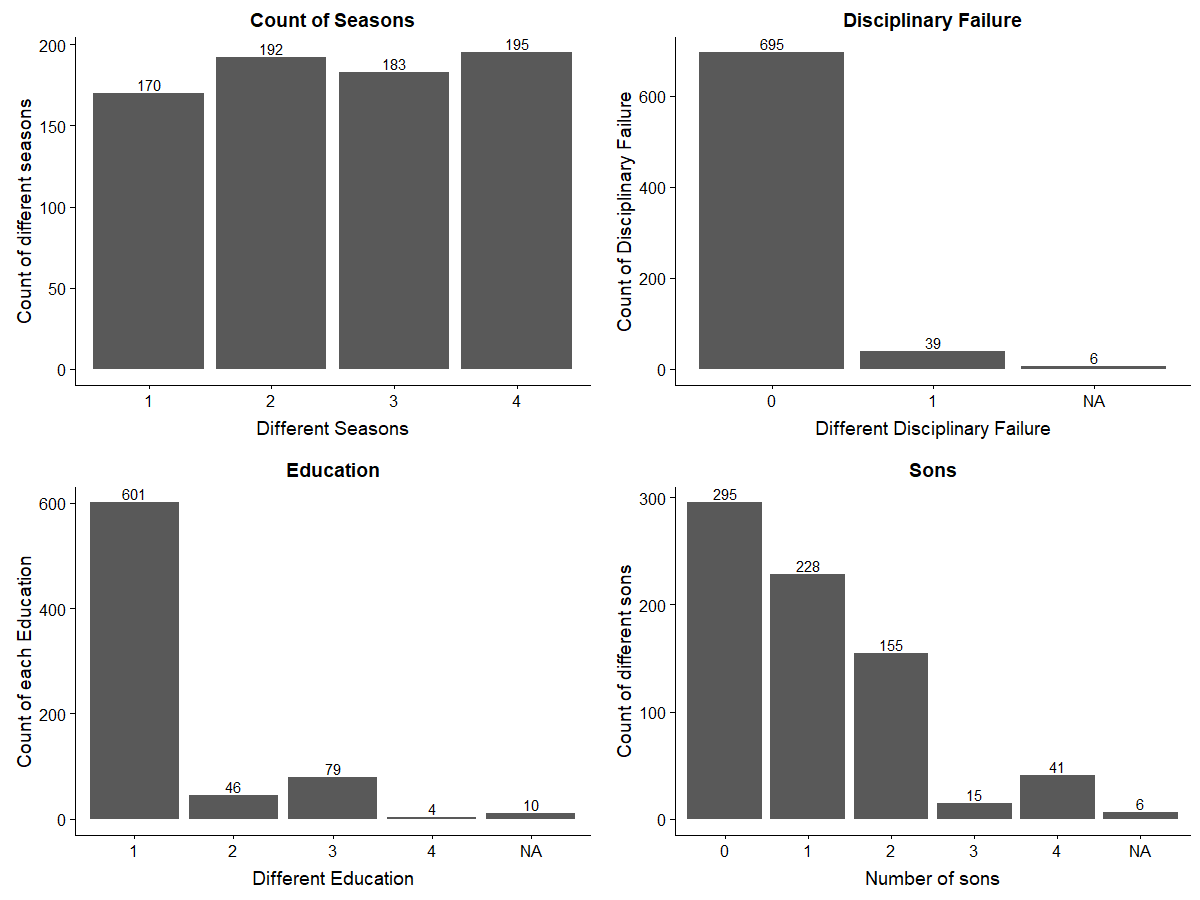
# Barplot for Son

p16 = ggplot(train, aes(x = Son)) + geom\_bar() +

labs(title = "Sons", x = "Number of sons", y = "Count of different sons") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

plot\_grid(p13, p14, p15,p16, nrow = 2)



Here, numbers for different seasons are almost same for different seasons. Disciplinary Failure for No is higher than yes value. Number of High school Education are more as compared to different Educations. Employee having more number for low number of sons.

#Barplot for Social Drinker

p17 = ggplot(train, aes(x = `Social drinker`)) + geom\_bar() +

labs(title = "Social Drinker", x = "Different Social Drinker", y = "Count of each Social Drinker") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

#Barplot for Social Smoker

p18 = ggplot(train, aes(x = `Social smoker`)) + geom\_bar() +

labs(title = "Social Smoker", x = "Different Social smoker", y = "Count of each Social smoker") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

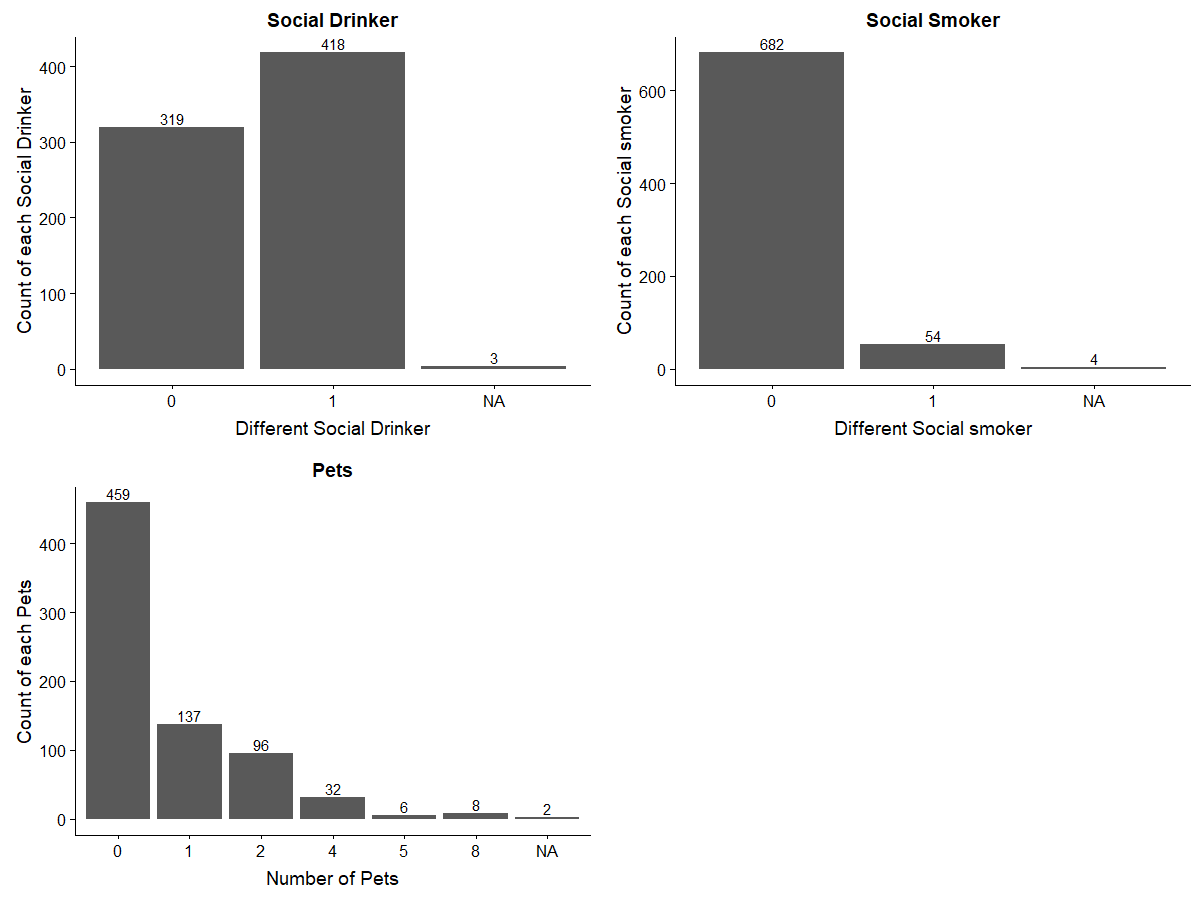
#Barplot for pets

p19 = ggplot(train, aes(x = Pet)) + geom\_bar() +

labs(title = "Pets", x = "Number of Pets", y = "Count of each Pets") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

plot\_grid(p17, p18, p19, nrow = 2)



Company has almost same number of Social drinker employee and Non Social drinker. Employee consuming less smoke are lower in number. Company havi more number of Employee having less Pets.

**BIVARIATE ANALYSIS:-** In Bivariate analysis we find the relation between two variables. For comparison between two variable I uses scatter plot and between continuous and categorical variable we use Box plot method.

#Bivariate Analysis

#Relation Between continous and categorical variable

#Relation between ID and Absenteeism time in hours

theme\_set(theme\_bw())

b1 = ggplot(train, aes(ID, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b2 = ggplot(train, aes(`Reason for absence`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

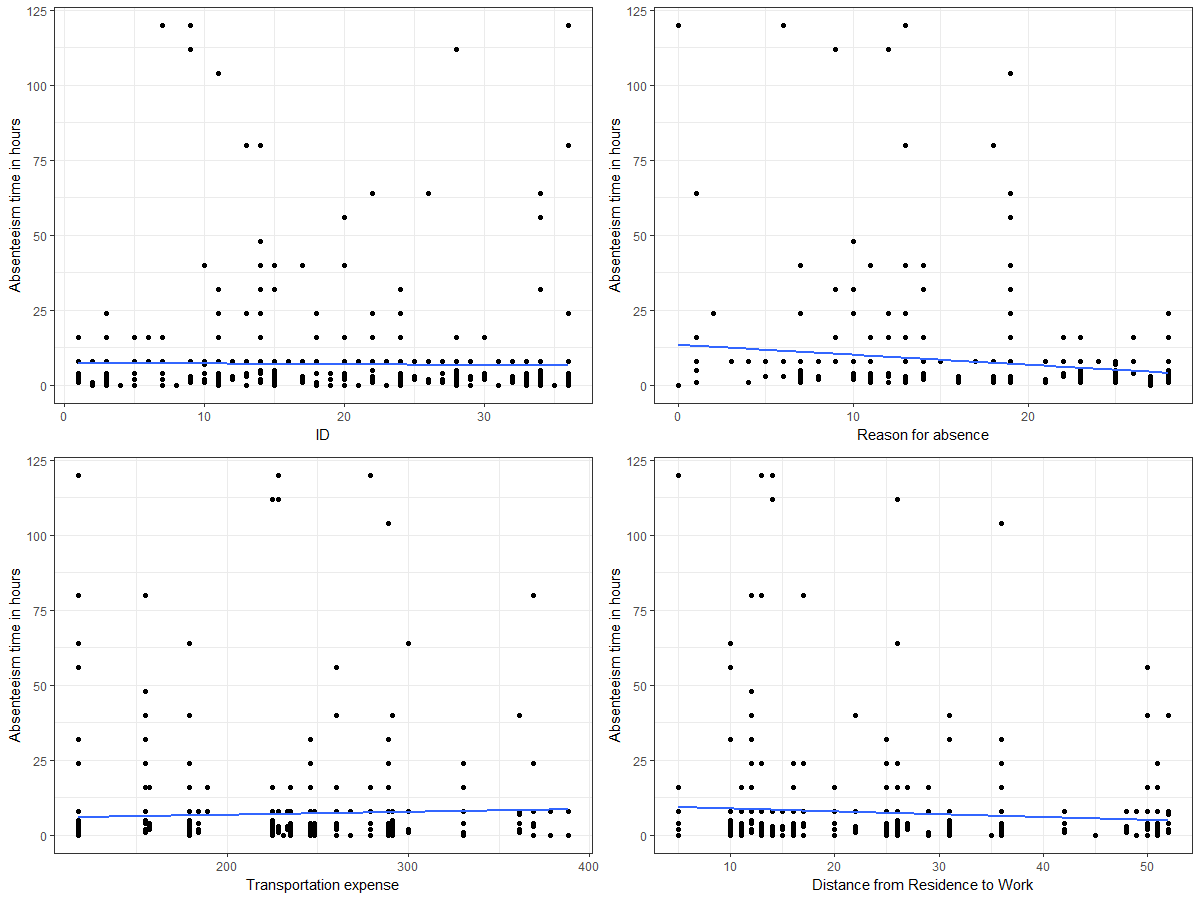
b3 = ggplot(train, aes(`Transportation expense`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b4 = ggplot(train, aes(`Distance from Residence to Work`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b1, b2, b3,b4, nrow = 2)



b5 = ggplot(train, aes(`Service time`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b6 = ggplot(train, aes(Age, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

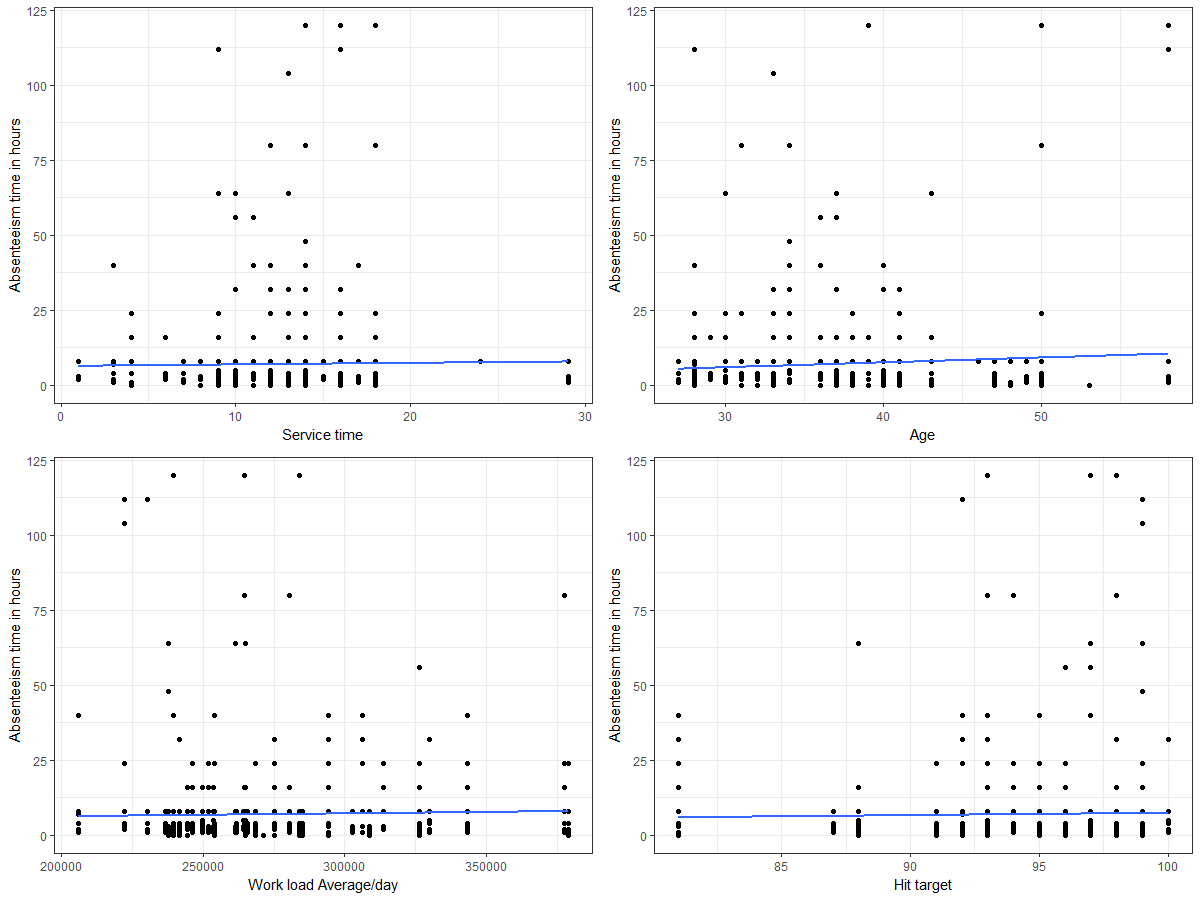
b7 = ggplot(train, aes(`Work load Average/day`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b8 = ggplot(train, aes(`Hit target`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b5, b6, b7, b8, nrow = 2)



b9 = ggplot(train, aes(Weight, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

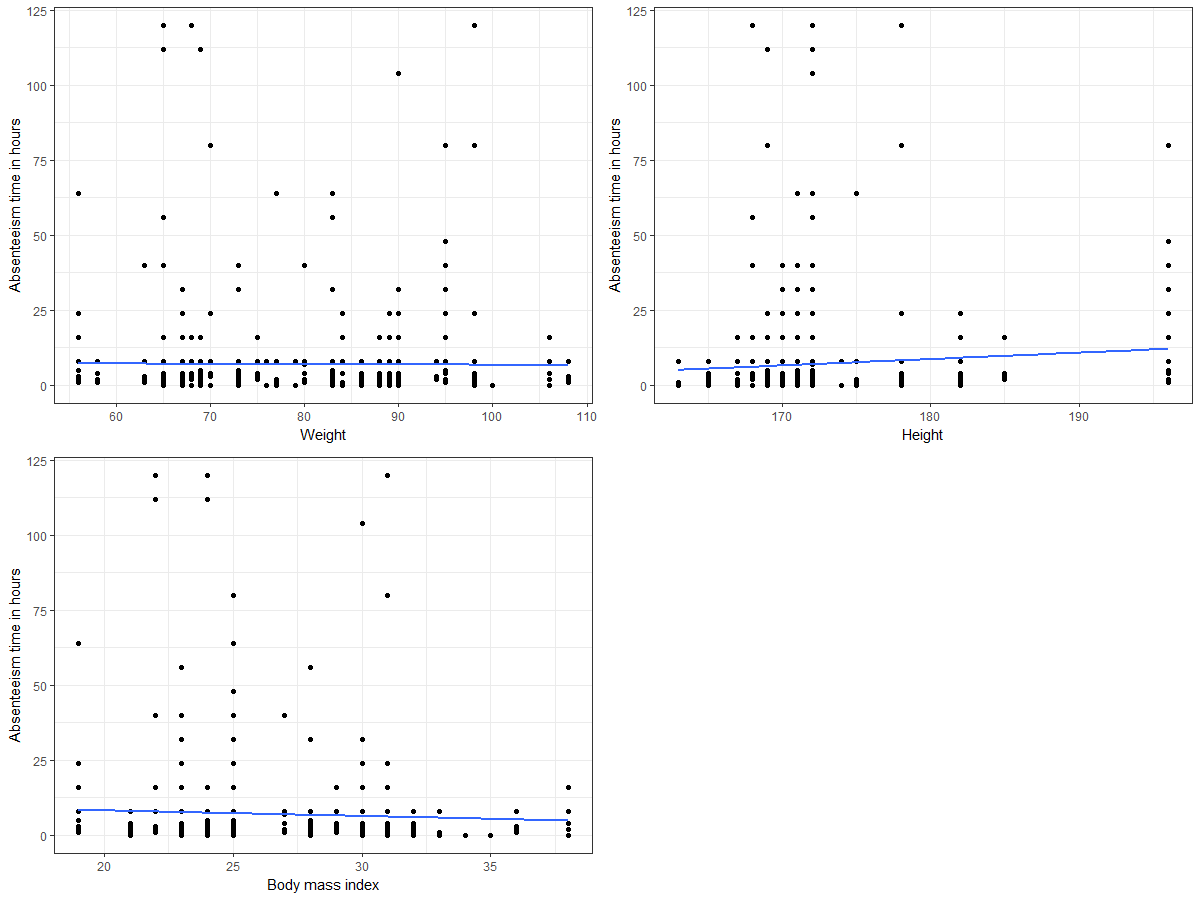
b10 = ggplot(train, aes(Height, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b11 = ggplot(train, aes(`Body mass index`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b9, b10, b11, nrow = 2)



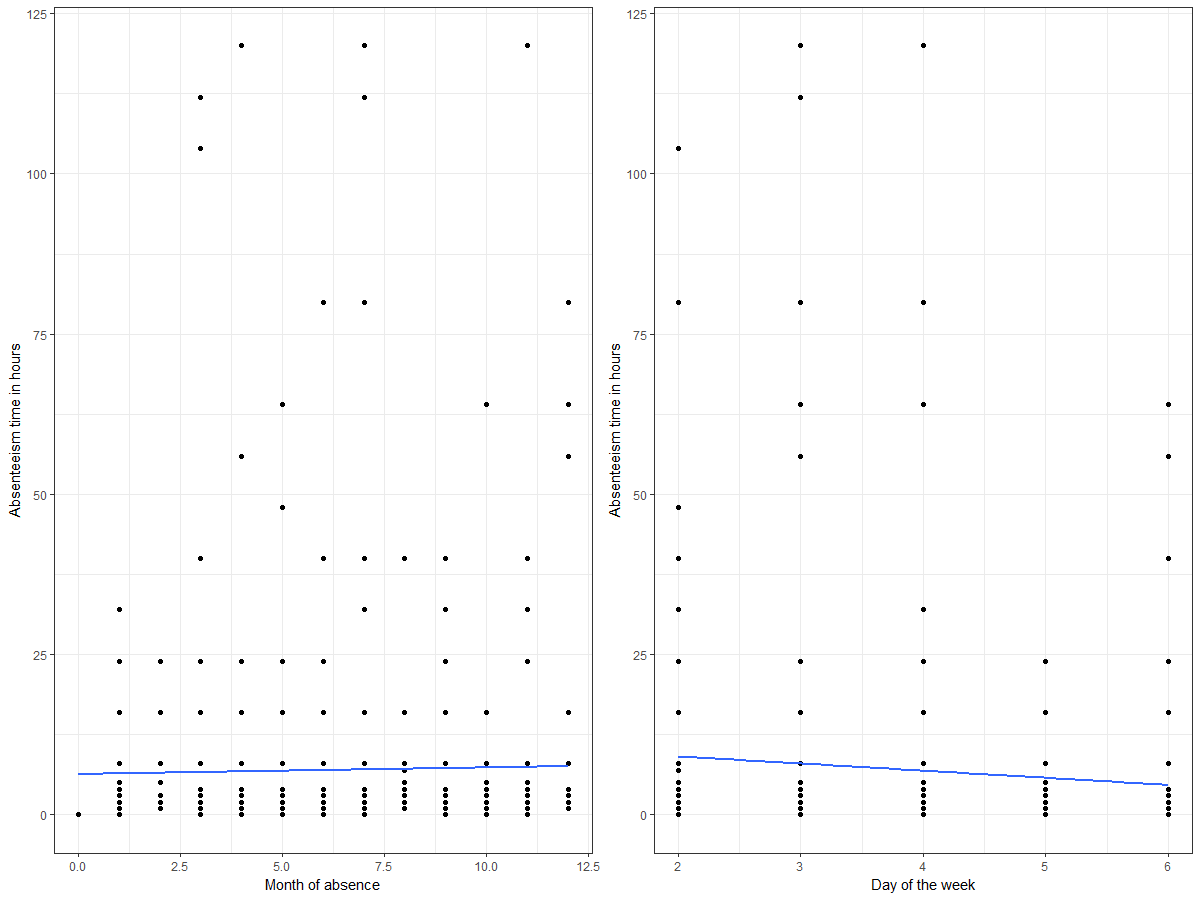
b12 = ggplot(train, aes(`Month of absence`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b13 = ggplot(train, aes(`Day of the week`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b12, b13, nrow = 1)



#Relation between categorical variable and continous variable

#Boxplot between Month of absence and Absenteeism time in hours

theme\_set(theme\_classic())

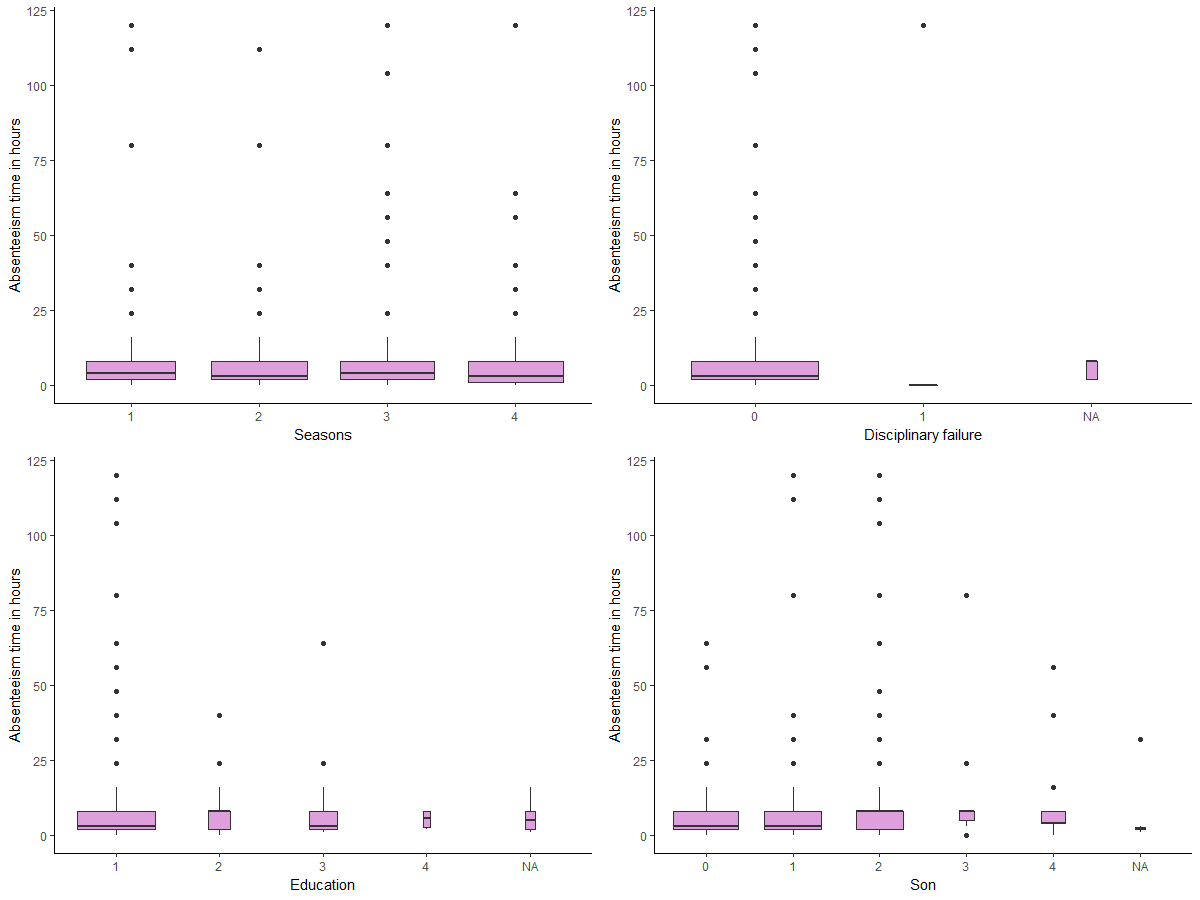
b14 = ggplot(train, aes(`Seasons`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b15 = ggplot(train, aes(`Disciplinary failure`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b16 = ggplot(train, aes(Education, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b17 = ggplot(train, aes(Son, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

plot\_grid(b14, b15, b16, b17, nrow = 2)

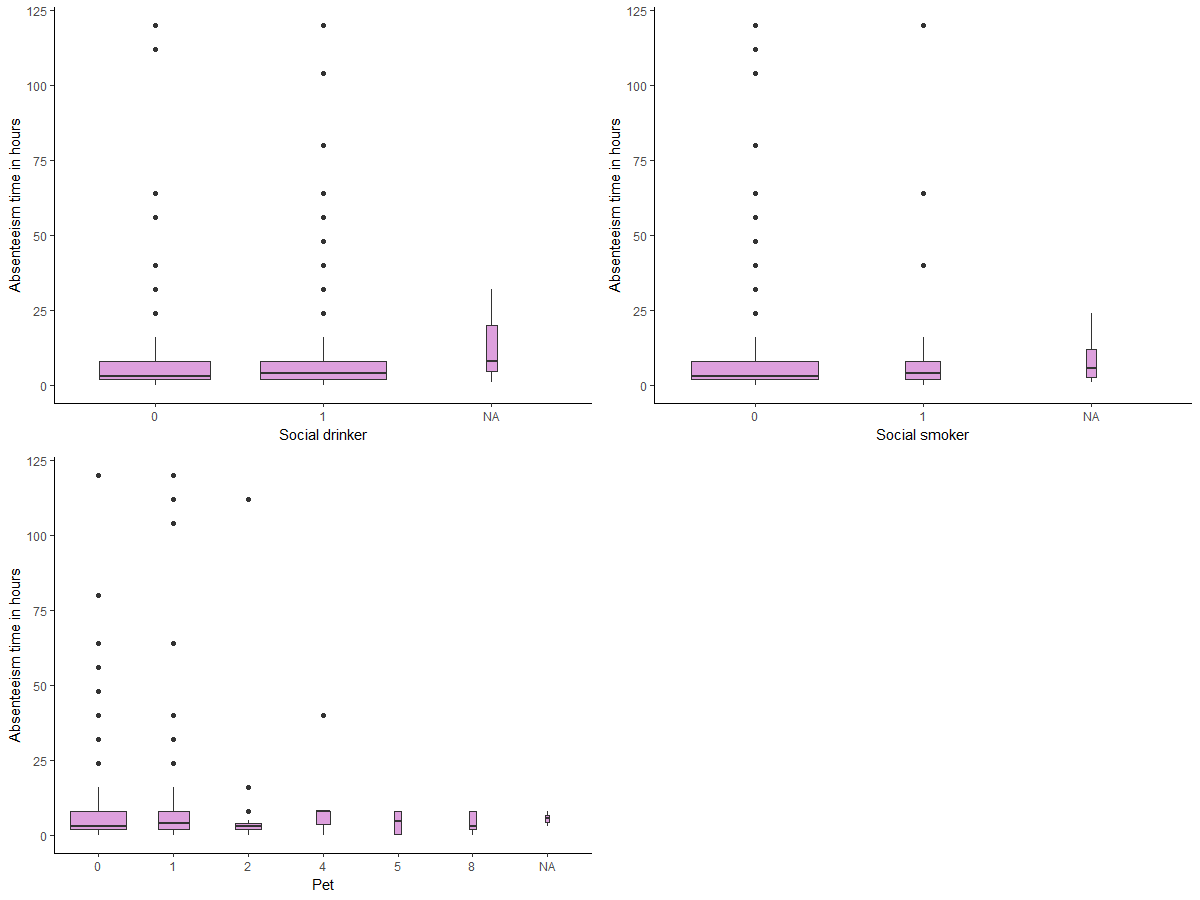


b18 = ggplot(train, aes(`Social drinker`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b19 = ggplot(train, aes(`Social smoker`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b20 = ggplot(train, aes(Pet, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

plot\_grid(b18, b19, b20, nrow = 2)



**CHAPTER 3:- DATA PREPERATION**

**MISSING VALUES:-** Missing data can have a severe impact on building predictive models because the missing values might be contain some vital information which could help in making better predictions. So, it becomes imperative to carry out missing data imputation. There are different methods to treat missing values based on the problem and the data. Some of the common techniques are as follows:

1. Deletion of Rows
2. Mean/Median/Mode Imputation
3. Building Predictive model

#MISSING VALUE ANALYSIS

missing\_val = data.frame(apply(train,2,function(x){sum(is.na(x))}))

missing\_val$Columns = row.names(missing\_val)

names(missing\_val)[1] = "Missing\_percentage"

missing\_val$Missing\_percentage = (missing\_val$Missing\_percentage/nrow(train)) \* 100

missing\_val = missing\_val[order(-missing\_val$Missing\_percentage),]

row.names(missing\_val) = NULL

missing\_val = missing\_val[,c(2,1)]

missing\_val

Columns Missing\_percentage

1 Body mass index 4.1891892

2 Absenteeism time in hours 2.9729730

3 Height 1.8918919

4 Work load Average/day 1.3513514

5 Education 1.3513514

6 Transportation expense 0.9459459

7 Hit target 0.8108108

8 Disciplinary failure 0.8108108

9 Son 0.8108108

10 Social smoker 0.5405405

11 Reason for absence 0.4054054

12 Distance from Residence to Work 0.4054054

13 Service time 0.4054054

14 Age 0.4054054

15 Social drinker 0.4054054

16 Pet 0.2702703

17 Month of absence 0.1351351

18 Weight 0.1351351

19 ID 0.0000000

20 Day of the week 0.0000000

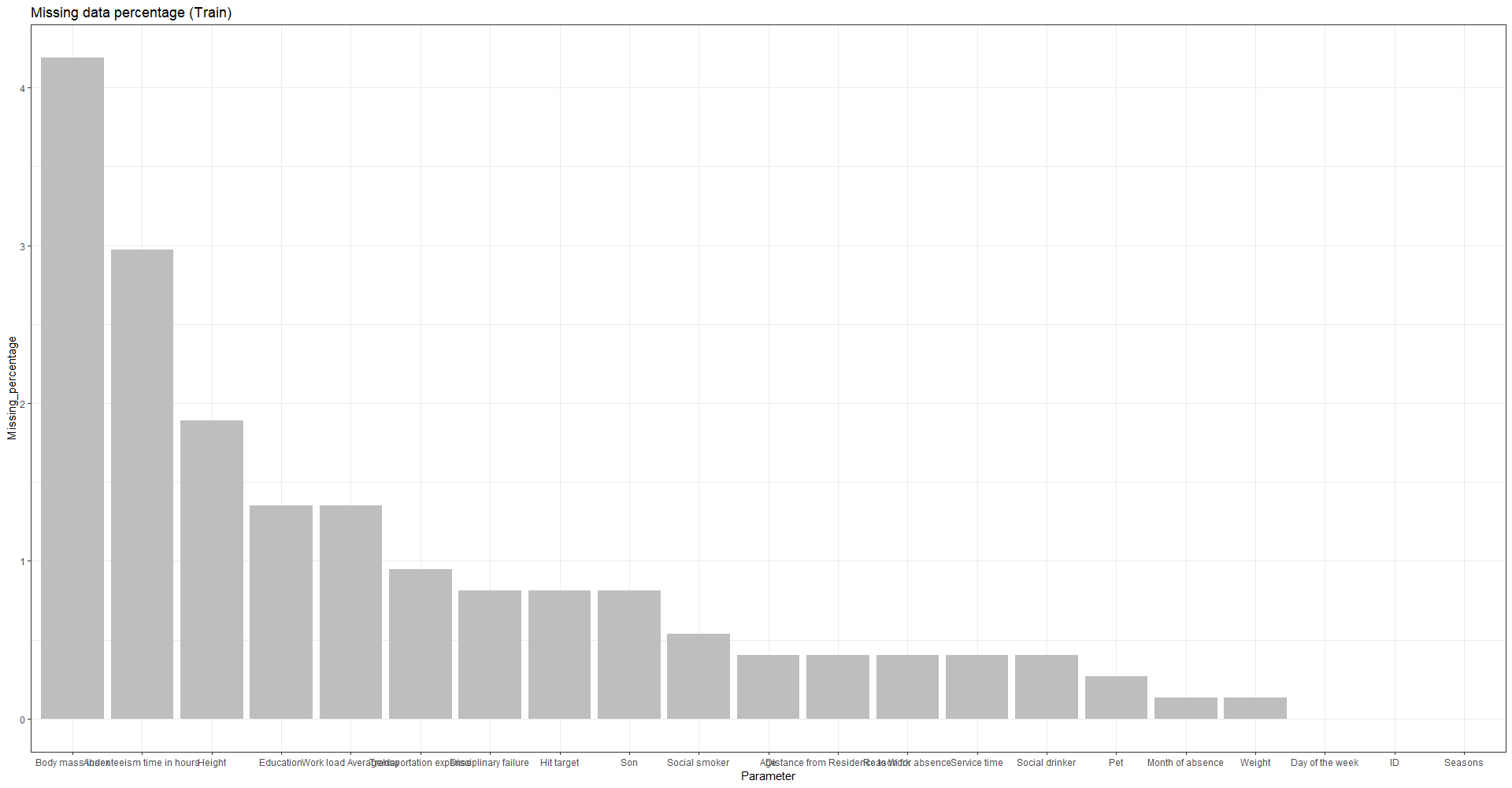
21 Seasons 0.0000000

#Visualization of Mssing Value

ggplot(data = missing\_val[,], aes(x=reorder(Columns, -Missing\_percentage),y = Missing\_percentage))+

geom\_bar(stat = "identity",fill = "grey")+xlab("Parameter")+

ggtitle("Missing data percentage (Train)") + theme\_bw()



I have used mean method to impute the missing value for Weight and Hieight.

#IMPUTING MISSING VALUE

#For Weight and Height

train$Weight[is.na(train$Weight)] = mean(train$Weight, na.rm = T)

train$Height[is.na(train$Height)] = mean(train$Height, na.rm = T)

Since, Body mass index = weight(in KG)/Height(in cm)^2

#For Body Mass Index

for(i in 1:nrow(train)){

if(is.na(train$`Body mass index`[i])){

train$`Body mass index`[i] = (train$Weight[i]\*10000)/train$Height[i]^2

}

}

Now I have used KNN method to impute the Missing Value for rest of the variables.

#KNN imputation to impute Missing Value

train = kNN(train, k=5)

Since 0 is not any month so 0 is replace by NA and then imputing it. Similarly, 0 is not any reason for absence so 0 is replace with NA and then imputing it.

#Since 0 is not the number for month so replace the data points with 0 value with NA and then imputing it

for(i in 1:nrow(train)){

if(train$`Month of absence`[i] == 0){

train$`Month of absence`[i] = NA

}

}

#Since 0 is not the reason for absence so here we first replace it with NA and then imputing it

for(i in 1:nrow(train)){

if(train$`Reason for absence`[i] == 0){

train$`Reason for absence`[i] = NA

}

}

#Again using KNN imputation for imputing the Missing value

train = kNN(train, variable = c("Month of absence","Reason for absence"), k = 3)

**FEATURE SELECTION:-** Feature selection is the method where we find the correlation between different variables and removing those variables which are highly correlated with other variables in Independent variables.

#FEATURE SELECTION

## Correlation Plot

numeric\_index = sapply(train, is.numeric)

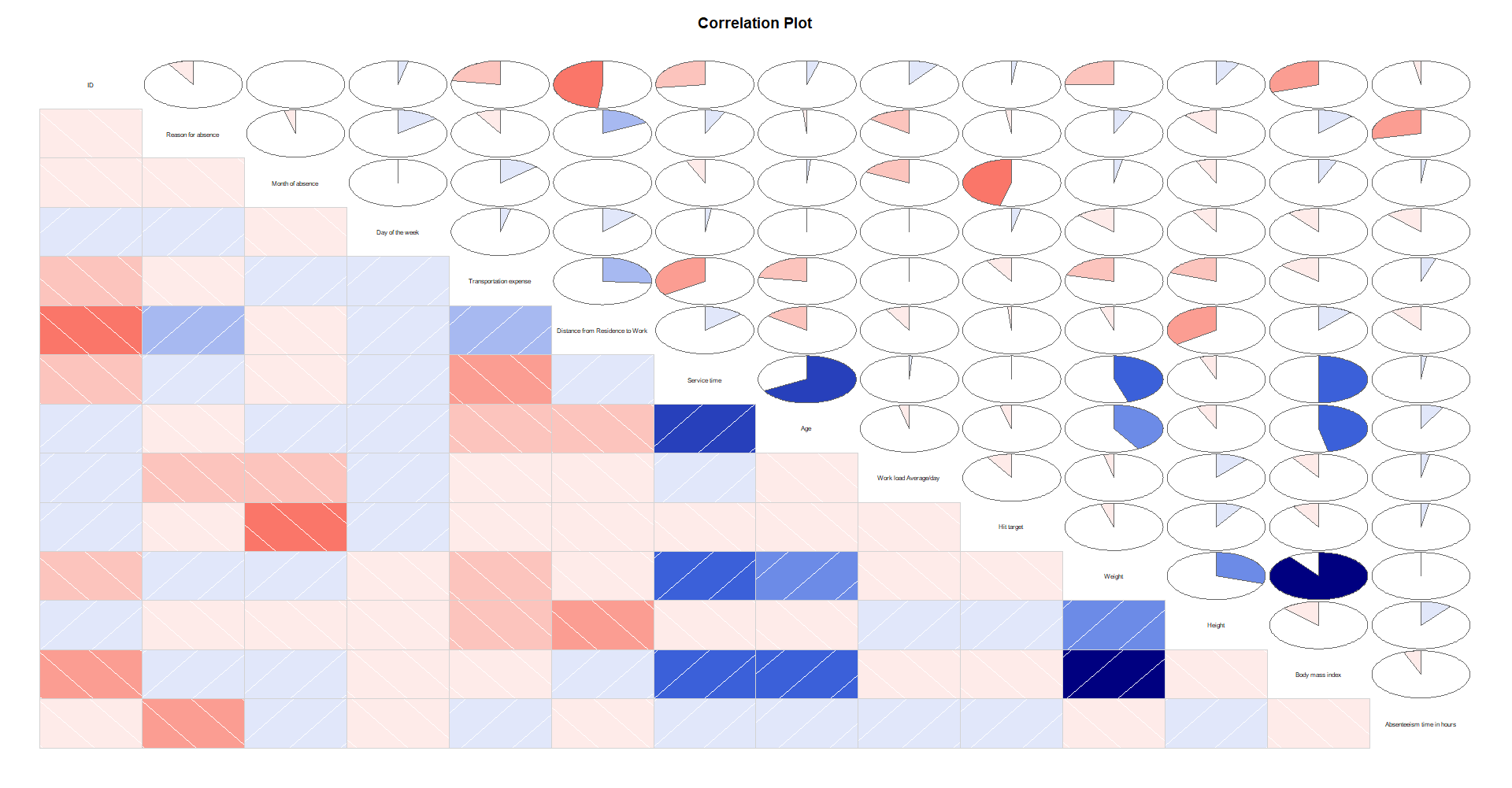
corrgram(train[,numeric\_index], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#DIMNESIONAL REDUCTION

train = subset(train,

select = -c(ID, `Service time`, Weight, `Work load Average/day`, `Hit target`,`Month of absence`, Pet))



**FEATURE SCALING:-** Feature scaling is the method to scale all the independent variables into appropriate form. Here, I have used normalization method to scale the variables. Normalization to bring all of the variables into proportion with one another.

#DIMNESIONAL REDUCTION

train = subset(train,

select = -c(ID, `Service time`, Weight, `Work load Average/day`, `Hit target`,`Month of absence`, Pet))

#Feature Scaling

cnames = c("Reason for absence","Distance from Residence to Work","Age","Transportation expense", "Height","Body mass index")

for(i in cnames){

print(i)

train[,i] = (train[,i] - min(train[,i]))/

(max(train[,i] - min(train[,i])))

}

**CHAPTER 4:- MODEL DEVELOPMENT AND EVALUATION**

There are various machine learning model we used which can predict the test case for best accuracy.

But, before this we will divide the train data into the two parts one is the train part and other is the validation part. Train part is to model the data and validation part is to make prediction and check the accuracy.

#Divide data into train and test using stratified sampling method

set.seed(1234)

train.index = createDataPartition(train$`Absenteeism time in hours`, p = .80, list = FALSE)

train = train[ train.index,]

test = train[-train.index,]

Here, I have used following models

* **LINEAR REGRESSION**
* **LASSO REGRESSION**
* **RIDGE REGRESSION**
* **RANDOM FOREST**
* **XBBOOST**

**EVALUATION METRICS FOR REGRESSION:-**

* **MEAN ABSOLUTE ERROR(MAE)**is the mean of absolute value of errors.

****

* **MEAN ABSOLUTE PERCENTAGE ERROR(MAPE)** is percentage errors.

****

* **ROOT MEAN SQYUARED ERRORS(RMSE)**is the square root of of the mean of the squared errors.

****

Here, we use RMSE as the evaluation metrics.

VIF:- Variation Inflanation Factor, it defines multicollinearity between the continous variables. If VIF<=4, means no multicollinarity whereas if VIF>=10, means high multicollinearity.

#Check multicollinearity

vifcor(train[, -c(2,3,7,8,9,10,11)], th = 0.9)

No variable from the 7 input variables has collinearity problem.

The linear correlation coefficients ranges between:

min correlation ( Age ~ Reason for absence ): 0.00927738

max correlation ( Body mass index ~ Age ): 0.4851592

---------- VIFs of the remained variables --------

Variables VIF

1 Reason for absence 1.156199

2 Transportation expense 1.184697

3 Distance from Residence to Work 1.303712

4 Age 1.439269

5 Height 1.204846

6 Body mass index 1.394149

7 Absenteeism time in hours 1.129629

1. **Linear regression:- Linear regression** is the simplest and most widely used statistical technique for predictive modeling. Given below is the linear regression equation:

Ypredicted = b0 + b1\*x1 + b2\*x2 + b3\*x3 + b4\*x4

where X1, X2,…,Xn are the independent variables, Y is the target variable and all b are the coefficients. Magnitude of a coefficient wrt to the other coefficients determines the importance of the corresponding independent variable.

#Linear Regression Model

model1 = lm(`Absenteeism time in hours` ~ ., data = train)

summary(model1)

Call:

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

Residuals:

Min 1Q Median 3Q Max

-12.863 -4.422 -1.413 1.365 106.168

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 19.97725 3.14633 6.349 4.40e-10 \*\*\*

`Reason for absence` -13.92833 1.85723 -7.500 2.45e-13 \*\*\*

`Day of the week` -0.70354 0.32736 -2.149 0.03204 \*

Seasons2 -0.07481 1.33695 -0.056 0.95540

Seasons3 -0.27906 1.34303 -0.208 0.83547

Seasons4 -0.62341 1.34601 -0.463 0.64343

`Transportation expense` -0.54896 2.47680 -0.222 0.82467

`Distance from Residence to Work` -2.00656 2.47842 -0.810 0.41850

Age 2.87003 3.01460 0.952 0.34148

`Disciplinary failure`1 -6.68465 2.06753 -3.233 0.00129 \*\*

Education2 -1.48382 2.25034 -0.659 0.50992

Education3 -1.37360 1.76802 -0.777 0.43753

Education4 -6.55179 6.52118 -1.005 0.31547

Son1 1.22400 1.28655 0.951 0.34181

Son2 2.80631 1.54002 1.822 0.06894 .

Son3 6.16194 4.10918 1.500 0.13428

Son4 1.13982 2.33243 0.489 0.62525

`Social drinker`1 2.25113 1.51798 1.483 0.13863

`Social smoker`1 -2.32582 2.18108 -1.066 0.28671

Height 2.10372 3.62946 0.580 0.56240

`Body mass index` -4.87756 2.93411 -1.662 0.09699 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.99 on 573 degrees of freedom

Multiple R-squared: 0.163, Adjusted R-squared: 0.1338

F-statistic: 5.579 on 20 and 573 DF, p-value: 2.937e-13

Here, the most important variables are Reasons for absence, Day of the week, and Disciplinary failure. The Adjusted R-squared is 0.1338 which means that only 13% of variance of train data is defined by the independent features, which is very low. So, linear regression is not working well.

Making Prediction on test data:-

#Predictting Test Data

predictions\_LR = predict(model1, test[,1:13])

Calculating RMSE

#Calculate RMSE

RMSE = function(m, o){

sqrt(mean((m - o)^2))

}

RMSE(test[,14], predictions\_LR)

[1] 7.47

Here, the RMSE is 7.47.

To check how robust our model is to unseen data, we’ll use **Cross Validation**. It is a technique which involves reserving a particular sample of a dataset on which we do not train the model. Later, we test your model on this sample before finalizing it. Some of the common methods for cross validation are listed below:

1. The Validation Set approach
2. K-folds cross validation
3. Leave one out cross validation(LOOCV)

k-folda cross validation:-

Next, we will be using the k-fold cross validation (CV) to test our models. The steps followed for k-fold CV are as follows:

1. Randomly split the data into k”folds”.
2. For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold.
3. Record the error you see on each of the predictions.
4. Repeat this until each of the k-folds has served as the test set

The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model.

**REGULARIZED LINEAR REGRESSION:- Regularised regression models** can handle the correlated independent variables well and helps in overcoming overfitting. **Ridge** penalty shrinks the coefficients of correlated predictors towards each other, while the **Lasso** tends to pick one of a pair of correlated features and discard the other. The tuning parameter **lambda** controls the strength of the penalty.

**LASSO REGRESSION:-**

#Lasso Regression

X = data.matrix(train)

Y = data.matrix(test)

set.seed(1235)

my\_control = trainControl(method="cv", number=5)

Grid = expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0002))

lasso\_linear\_reg\_mod = train(x = X[, 1:13], y = X[,14],

method='glmnet', trControl= my\_control, tuneGrid = Grid)

#Prediction

prediction\_LL = predict(lasso\_linear\_reg\_mod, Y[, -14])

#Calculate RMSE

RMSE(Y[,14], prediction\_LL)

[1] 7.36

Here, the RMSE is 7.36.

**RIDGE REGRESSION:-**

#Ridge Regression

set.seed(1236)

my\_control = trainControl(method="cv", number=5)

Grid = expand.grid(alpha = 0, lambda = seq(0.001,0.1,by = 0.0002))

ridge\_linear\_reg\_mod = train(x = X[, 1:13], y = X[, 14],

method='glmnet', trControl= my\_control, tuneGrid = Grid)

#Prediction

prediction\_RR = predict(ridge\_linear\_reg\_mod, Y[, -14])

#Calculate RMSE

RMSE(Y[, 14], prediction\_RR)

1] 7.39

**RANDOM FOREST:- RandomForest** is a tree based bootstrapping algorithm wherein a certain number of weak learners (decision trees) are combined to make a powerful prediction model. For every individual learner, a random sample of rows and a few randomly chosen variables are used to build a decision tree model. Final prediction can be a function of all the predictions made by the individual learners. In case of a regression problem, the final prediction can be mean of all the predictions.

We will now build a RandomForest model with 400 trees. The other tuning parameters used here are mtry — no. of predictor variables randomly sampled at each split, and min.node.size — minimum size of terminal nodes (setting this number large causes smaller trees and reduces overfitting).

#Random Forest

set.seed(1237)

my\_control = trainControl(method="cv", number=5) # 5-fold CV

tgrid = expand.grid(

.mtry = c(1:13),

.splitrule = "variance",

.min.node.size = c(10,15,20)

)

rf\_mod = train(x = train[, 1:13],

y = train$`Absenteeism time in hours`,

method='ranger',

trControl= my\_control,

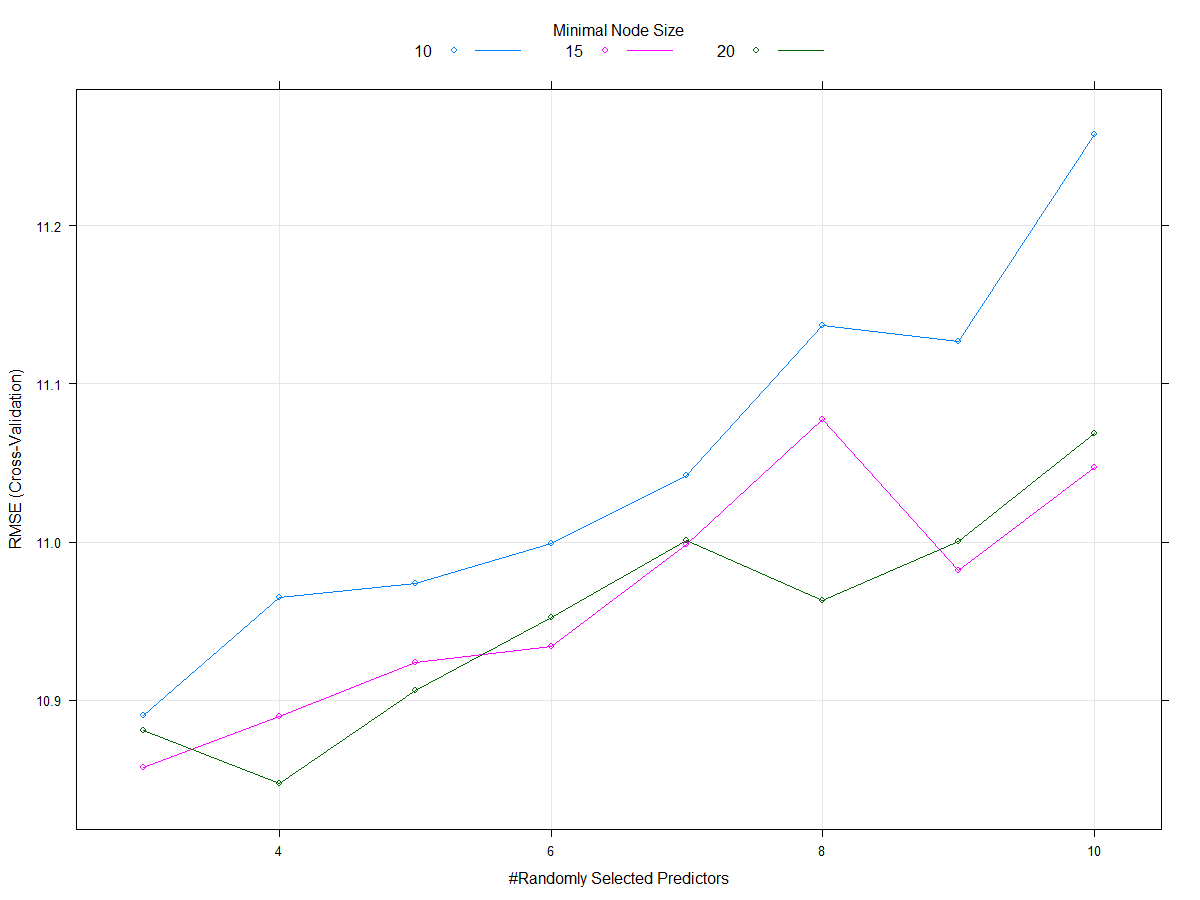
tuneGrid = tgrid,

num.trees = 400,

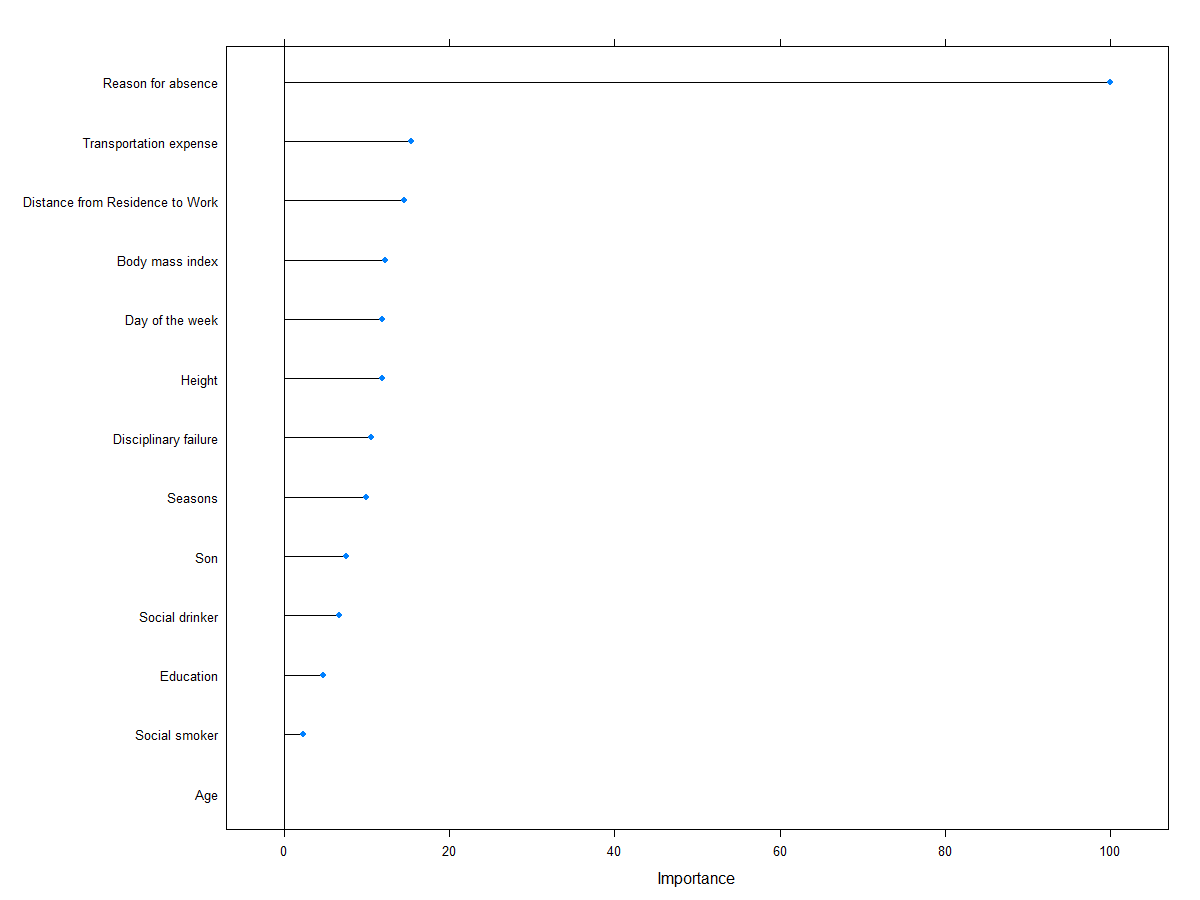
importance = "permutation")

plot(rf\_mod)

plot(varImp(rf\_mod))



From the graph the best score is achived as mtry = 4 and min.node.size = 20.

**

Here, Reason for absence is the most important variable for training the data.

#Predictions

prediction\_RF = predict(rf\_mod, test[, -14])

#Calculate RMSE

RMSE(test[, 14], prediction\_RF)

[1] 5.17

Here, the RMSE is 5.17 which is best one.

**XGBOOST:- XGBoost** is a fast and efficient algorithm and has been used to by the winners of many data science competitions. It’s a boosting algorithm and you may refer this article to know more about boosting. XGBoost works only with numeric variables and we have already replaced the categorical variables with numeric variables. There are many tuning parameters in XGBoost which can be broadly classified into General Parameters, Booster Parameters and Task Parameters.

* **General parameters** refer to which booster we are using to do boosting. The commonly used are tree or linear model
* **Booster parameters** depend on which booster you have chosen
* **Learning Task parameters** that decide on the learning scenario, for example, regression tasks may use different parameters with ranking tasks.

Let’s have a look at the parameters that we are going to use in our model.

1. **eta**: It is also known as the learning rate or the shrinkage factor. It actually shrinks the feature weights to make the boosting process more conservative. The range is 0 to 1. Low eta value means the model is more robust to overfitting.
2. **gamma**: The range is 0 to ∞. Larger the gamma more conservative the algorithm is.
3. **max\_depth**: We can specify maximum depth of a tree using this parameter.
4. **subsample**: It is the proportion of rows that the model will randomly select to grow trees.
5. **colsample\_bytree**: It is the ratio of variables randomly chosen to build each tree in the model.

#XGBoost

param\_list = list(

objective = "reg:linear",

eta=0.01,

gamma = 1,

max\_depth=6,

subsample=0.8,

colsample\_bytree=0.5

)

dtrain = xgb.DMatrix(data = as.matrix(X[,-14]), label= X[,14])

dtest = xgb.DMatrix(data = as.matrix(Y[,]))

### Cross Validation

We are going to use the xgb.cv() function for cross validation. This function comes with the xgboost package itself. Here we are using cross validation for finding the optimal value of nrounds.

set.seed(112)

xgbcv = xgb.cv(params = param\_list,

data = dtrain,

nrounds = 1000,

nfold = 5,

print\_every\_n = 10,

early\_stopping\_rounds = 30,

maximize = F)

[1] train-rmse:13.256390+0.506809 test-rmse:13.110989+2.145465

Multiple eval metrics are present. Will use test\_rmse for early stopping.

Will train until test\_rmse hasn't improved in 30 rounds.

[11] train-rmse:12.771642+0.503187 test-rmse:12.772489+2.151049

[21] train-rmse:12.330284+0.510343 test-rmse:12.478205+2.148854

[31] train-rmse:11.922196+0.508113 test-rmse:12.222584+2.146650

[41] train-rmse:11.553433+0.514227 test-rmse:12.015456+2.142077

[51] train-rmse:11.233632+0.523082 test-rmse:11.848348+2.134037

[61] train-rmse:10.960147+0.522272 test-rmse:11.708746+2.137584

[71] train-rmse:10.689543+0.504658 test-rmse:11.579383+2.136515

[81] train-rmse:10.415945+0.490758 test-rmse:11.467626+2.129484

[91] train-rmse:10.160701+0.486304 test-rmse:11.380099+2.144459

[101] train-rmse:9.944064+0.475260 test-rmse:11.317652+2.143232

[111] train-rmse:9.724943+0.487323 test-rmse:11.258428+2.137466

[121] train-rmse:9.524959+0.499362 test-rmse:11.206800+2.123922

[131] train-rmse:9.331632+0.512269 test-rmse:11.160928+2.120826

[141] train-rmse:9.147028+0.492848 test-rmse:11.134575+2.139170

[151] train-rmse:8.983242+0.468819 test-rmse:11.118892+2.149388

[161] train-rmse:8.820854+0.464150 test-rmse:11.109893+2.144987

[171] train-rmse:8.660690+0.453592 test-rmse:11.092318+2.148560

[181] train-rmse:8.511471+0.472849 test-rmse:11.077529+2.145710

[191] train-rmse:8.364746+0.476257 test-rmse:11.068827+2.137994

[201] train-rmse:8.227077+0.478003 test-rmse:11.061200+2.134705

[211] train-rmse:8.108365+0.481132 test-rmse:11.068377+2.141708

[221] train-rmse:7.983736+0.481960 test-rmse:11.084970+2.141057

[231] train-rmse:7.848698+0.481798 test-rmse:11.094667+2.149223

Stopping. Best iteration:

[205] train-rmse:8.179140+0.472501 test-rmse:11.060620+2.138039

### Model Training

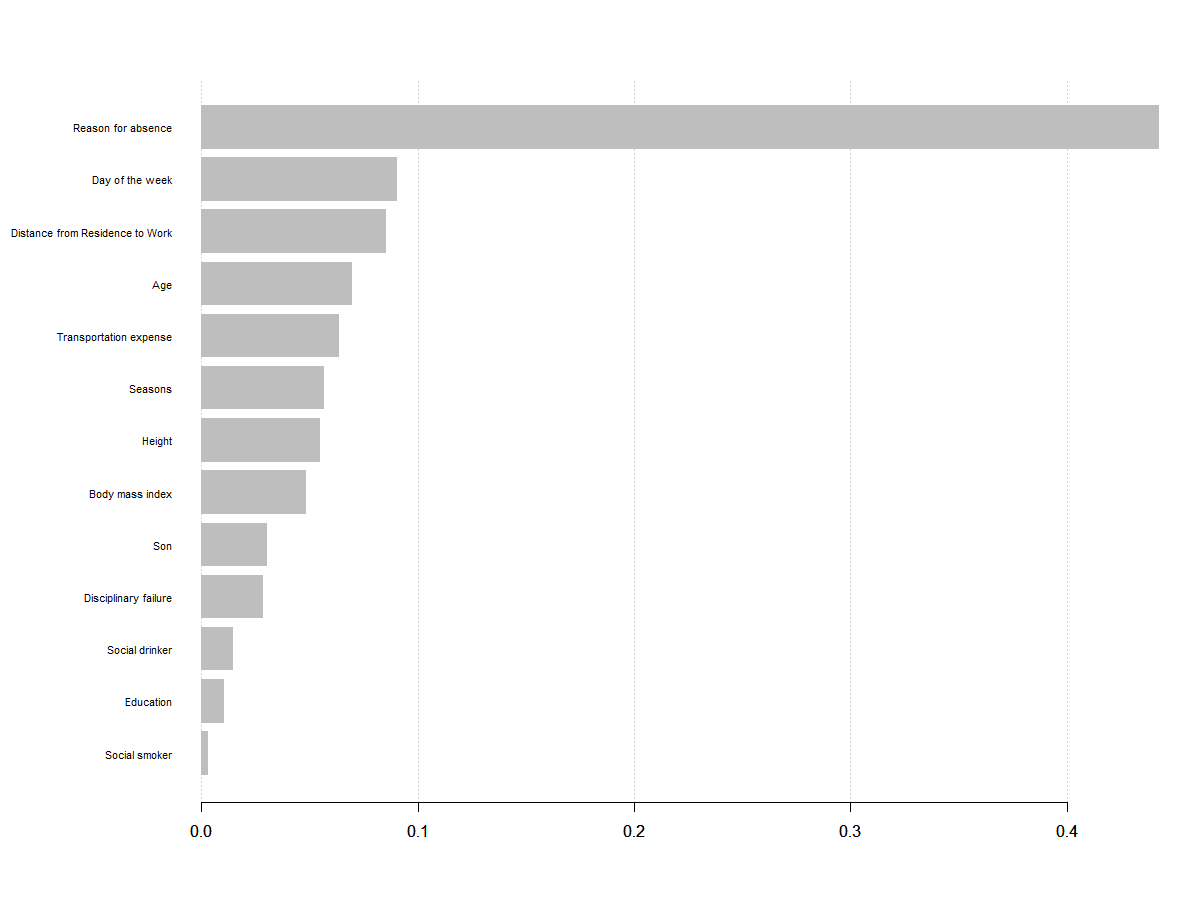
As per the verbose above, we got the best validation/test score at the 430th iteration. Hence, we will use nrounds = 430 for building the XGBoost model.

xgb\_model = xgb.train(data = dtrain, params = param\_list, nrounds = 205)

var\_imp = xgb.importance(feature\_names = setdiff(names(train), c("Absenteeism time in hours")),

model = xgb\_model)

xgb.plot.importance(var\_imp)



This model has even outperformed the RandomForest model. Here, the RMSE is 5.15 whih is lowest.

Here, the most important feature is Reason for absence.

**CONCLUSION:-**

* 1. The Absenteeism occurs mainly due to the Reasons given by the Employee like some employee are suffering Diseases in nerves, skin, stomach, circulitary syatem, Digestive system, Brain hamarage. Whereas there are some employee are having problems like they are going outdoor for patient follow-up, medical consultation, blood donation, laboratory examination, unjustified absence, physiotherapy, dental consultation. These are the main reasons for Absenteeism for the employee.

So, to reduce the number of Absenteeism the company should provide regular Medical treatment for the employee and the employee who are making absent for unavoidable reasons sholud be taken into considerations and should take strict actions for them like payment. Instead of strict actions the company should make them busy with the interesting projects, motivating them and teching them them about the new technologies, and also be given exciting benefits so that the work can attract them and the should not getting absent more.

Besides, the Reason there are the other factors which also affect the Absenteeism like day of the week, Distance from Residence to Work, Age, Transportation expense.

* 1. As in the XGBoost Model is performing well on the dataset. We have seen that its RMSE is 5.15. So, if the same trend of Absenteeism continues then we can say that 95% of same Absenteeism occurs in the upcomig years.

**R CODE:-**

#remove all the objects stored

rm(list = ls())

#set current working directory

setwd("E:/")

#Load Libraries

library(dplyr) # used for data manipulation and joining

library(ggplot2) # used for ploting

library(caret) # used for modeling

library(corrplot) # used for making correlation plot

library(xgboost) # used for building XGBoost model

library(cowplot) # used for combining multiple plots

library(readxl) # used to read excel file

library(VIM) # used to impute missing value

library(corrgram) # used to plot corgram plot

library(glmnet) # used for Ridge Regression

library(MASS)

library(DAAG)

library(ridge)

library(rpart)

library(ranger)

#Reading the Data

train = read\_excel("Absenteeism\_at\_work\_Project.xls")

#Understand the Data

#Dimension of the Data

dim(train)

#Features of Data

names(train)

#Structure of the data

str(train)

#converting the variable to its appropriate data type

train$Seasons=as.factor(train$Seasons)

train$`Disciplinary failure`=as.factor(train$`Disciplinary failure`)

train$Education=as.factor(train$Education)

train$Son=as.factor(train$Son)

train$`Social drinker`=as.factor(train$`Social drinker`)

train$`Social smoker`=as.factor(train$`Social smoker`)

train$Pet=as.factor(train$Pet)

#Exploratory Data Analysis

#Univariate Analysis

#For targe variable

ggplot(data = train, aes(x = `Absenteeism time in hours`)) + geom\_histogram(color = "black",

fill = "green", binwidth = 10)

#Now we will check Numerical independent variable

p1 = ggplot(data = train, aes(x = ID)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p2 = ggplot(data = train, aes(x = `Reason for absence`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p3 = ggplot(data = train, aes(x = `Month of absence`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 1)

plot\_grid(p1, p2, p3, nrow = 1) # plot\_grid() from cowplot package

p4 = ggplot(data = train, aes(x = `Day of the week`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 1)

p5 = ggplot(data = train, aes(x = `Transportation expense`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 100)

p6 = ggplot(data = train, aes(x = `Distance from Residence to Work`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

plot\_grid(p4, p5, p6, nrow = 1)

p7 = ggplot(data = train, aes(x = `Service time`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p8 = ggplot(data = train, aes(x = `Age`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p9 = ggplot(data = train, aes(x = `Work load Average/day`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 50000)

plot\_grid(p7, p8, p9, nrow = 1)

p10 = ggplot(data = train, aes(x = `Hit target`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 5)

p11 = ggplot(data = train, aes(x = Weight)) + geom\_histogram(color = "black",

fill = "red", binwidth = 10)

p12 = ggplot(data = train, aes(x = Height)) + geom\_histogram(color = "black",

fill = "red", binwidth = 5)

plot\_grid(p10, p11, p12, nrow = 1)

ggplot(data = train, aes(x = `Body mass index`)) + geom\_histogram(color = "black",

fill = "red", binwidth = 3)

#BarPlot for categorical variable

# Barplot for Seasons

p13 = ggplot(train, aes(x = Seasons)) + geom\_bar() +

labs(title = "Count of Seasons", x = "Different Seasons", y = "Count of different seasons") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

# Barplot for Disciplinary failure

p14 = ggplot(train, aes(x = `Disciplinary failure`)) + geom\_bar() +

labs(title = "Disciplinary Failure", x = "Different Disciplinary Failure", y = "Count of Disciplinary Failure") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

# Barplot for Education

p15 = ggplot(train, aes(x = Education)) + geom\_bar() +

labs(title = "Education", x = "Different Education", y = "Count of each Education") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

# Barplot for Son

p16 = ggplot(train, aes(x = Son)) + geom\_bar() +

labs(title = "Sons", x = "Number of sons", y = "Count of different sons") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

plot\_grid(p13, p14, p15,p16, nrow = 2)

#Barplot for Social Drinker

p17 = ggplot(train, aes(x = `Social drinker`)) + geom\_bar() +

labs(title = "Social Drinker", x = "Different Social Drinker", y = "Count of each Social Drinker") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

#Barplot for Social Smoker

p18 = ggplot(train, aes(x = `Social smoker`)) + geom\_bar() +

labs(title = "Social Smoker", x = "Different Social smoker", y = "Count of each Social smoker") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

#Barplot for pets

p19 = ggplot(train, aes(x = Pet)) + geom\_bar() +

labs(title = "Pets", x = "Number of Pets", y = "Count of each Pets") +

geom\_text(stat = 'count', aes(label = ..count..), vjust = -0.25)

plot\_grid(p17, p18, p19, nrow = 2)

#Bivariate Analysis

#Relation Between continous and categorical variable

#Relation between ID and Absenteeism time in hours

theme\_set(theme\_bw())

b1 = ggplot(train, aes(ID, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b2 = ggplot(train, aes(`Reason for absence`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b3 = ggplot(train, aes(`Transportation expense`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b4 = ggplot(train, aes(`Distance from Residence to Work`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b1, b2, b3,b4, nrow = 2)

b5 = ggplot(train, aes(`Service time`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b6 = ggplot(train, aes(Age, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b7 = ggplot(train, aes(`Work load Average/day`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b8 = ggplot(train, aes(`Hit target`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b5, b6, b7, b8, nrow = 2)

b9 = ggplot(train, aes(Weight, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b10 = ggplot(train, aes(Height, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b11 = ggplot(train, aes(`Body mass index`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b9, b10, b11, nrow = 2)

b12 = ggplot(train, aes(`Month of absence`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

b13 = ggplot(train, aes(`Day of the week`, `Absenteeism time in hours`)) + geom\_point() +

geom\_smooth(method = "lm", se = F)

plot\_grid(b12, b13, nrow = 1)

#Relation between categorical variable and continous variable

#Boxplot between Month of absence and Absenteeism time in hours

theme\_set(theme\_classic())

b14 = ggplot(train, aes(`Seasons`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b15 = ggplot(train, aes(`Disciplinary failure`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b16 = ggplot(train, aes(Education, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b17 = ggplot(train, aes(Son, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

plot\_grid(b14, b15, b16, b17, nrow = 2)

b18 = ggplot(train, aes(`Social drinker`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b19 = ggplot(train, aes(`Social smoker`, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

b20 = ggplot(train, aes(Pet, `Absenteeism time in hours`)) + geom\_boxplot(varwidth = T, fill = "plum")

plot\_grid(b18, b19, b20, nrow = 2)

#MISSING VALUE ANALYSIS

missing\_val = data.frame(apply(train,2,function(x){sum(is.na(x))}))

missing\_val$Columns = row.names(missing\_val)

names(missing\_val)[1] = "Missing\_percentage"

missing\_val$Missing\_percentage = (missing\_val$Missing\_percentage/nrow(train)) \* 100

missing\_val = missing\_val[order(-missing\_val$Missing\_percentage),]

row.names(missing\_val) = NULL

missing\_val = missing\_val[,c(2,1)]

missing\_val

#Visualization of Mssing Value

ggplot(data = missing\_val[,], aes(x=reorder(Columns, -Missing\_percentage),y = Missing\_percentage))+

geom\_bar(stat = "identity",fill = "grey")+xlab("Parameter")+

ggtitle("Missing data percentage (Train)") + theme\_bw()

#IMPUTING MISSING VALUE

#For Weight and Height

train$Weight[is.na(train$Weight)] = mean(train$Weight, na.rm = T)

train$Height[is.na(train$Height)] = mean(train$Height, na.rm = T)

#For Body Mass Index

for(i in 1:nrow(train)){

if(is.na(train$`Body mass index`[i])){

train$`Body mass index`[i] = (train$Weight[i]\*10000)/train$Height[i]^2

}

}

#KNN imputation to impute Missing Value

train = kNN(train, k=5)

#Since 0 is not the number for month so replace the data points with 0 value with NA and then imputing it

for(i in 1:nrow(train)){

if(train$`Month of absence`[i] == 0){

train$`Month of absence`[i] = NA

}

}

#Since 0 is not the reason for absence so here we first replace it with NA and then imputing it

for(i in 1:nrow(train)){

if(train$`Reason for absence`[i] == 0){

train$`Reason for absence`[i] = NA

}

}

#Again using KNN imputation for imputing the Missing value

train = kNN(train, variable = c("Month of absence","Reason for absence"), k = 3)

#Dropping the Useless variable

train = subset(train,

select = -c(ID\_imp,`Reason for absence\_imp`,`Month of absence\_imp`,`Day of the week\_imp`,Seasons\_imp,

`Transportation expense\_imp`,`Distance from Residence to Work\_imp`,`Service time\_imp`,

Age\_imp,`Work load Average/day\_imp`,`Hit target\_imp`,`Disciplinary failure\_imp`,Education\_imp,

Son\_imp,`Social drinker\_imp`,`Social smoker\_imp`,Pet\_imp,Weight\_imp,Height\_imp,

`Body mass index\_imp`,`Absenteeism time in hours\_imp`))

#FEATURE SELECTION

## Correlation Plot

numeric\_index = sapply(train, is.numeric)

corrgram(train[,numeric\_index], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#DIMNESIONAL REDUCTION

train = subset(train,

select = -c(ID, `Service time`, Weight, `Work load Average/day`, `Hit target`,`Month of absence`, Pet))

#Feature Scaling

cnames = c("Reason for absence","Distance from Residence to Work","Age","Transportation expense", "Height","Body mass index")

for(i in cnames){

print(i)

train[,i] = (train[,i] - min(train[,i]))/

(max(train[,i] - min(train[,i])))

}

#MODEL DEVELOPMENT

#Divide data into train and test using stratified sampling method

set.seed(1234)

train.index = createDataPartition(train$`Absenteeism time in hours`, p = .80, list = FALSE)

train = train[ train.index,]

test = train[-train.index,]

#Check multicollinearity

vifcor(train[, -c(2,3,7,8,9,10,11)], th = 0.9)

#Linear Regression Model

model1 = lm(`Absenteeism time in hours` ~ ., data = train)

summary(model1)

#Predictting Test Data

predictions\_LR = predict(model1, test[,1:13])

#Calculate RMSE

RMSE = function(m, o){

sqrt(mean((m - o)^2))

}

RMSE(test[,14], predictions\_LR)

#K-folda cross validation

model2 = cv.lm(data = train, model1, m=5) # 5 fold cross-validation

#Lasso Regression

X = data.matrix(train)

Y = data.matrix(test)

set.seed(1235)

my\_control = trainControl(method="cv", number=5)

Grid = expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0002))

lasso\_linear\_reg\_mod = train(x = X[, 1:13], y = X[,14],

method='glmnet', trControl= my\_control, tuneGrid = Grid)

#Prediction

prediction\_LL = predict(lasso\_linear\_reg\_mod, Y[, -14])

#Calculate RMSE

RMSE(Y[,14], prediction\_LL)

#Ridge Regression

set.seed(1236)

my\_control = trainControl(method="cv", number=5)

Grid = expand.grid(alpha = 0, lambda = seq(0.001,0.1,by = 0.0002))

ridge\_linear\_reg\_mod = train(x = X[, 1:13], y = X[, 14],

method='glmnet', trControl= my\_control, tuneGrid = Grid)

#Prediction

prediction\_RR = predict(ridge\_linear\_reg\_mod, Y[, -14])

#Calculate RMSE

RMSE(Y[, 14], prediction\_RR)

#Random Forest

set.seed(1237)

my\_control = trainControl(method="cv", number=5) # 5-fold CV

tgrid = expand.grid(

.mtry = c(3:10),

.splitrule = "variance",

.min.node.size = c(10,15,20)

)

rf\_mod = train(x = train[, 1:13],

y = train$`Absenteeism time in hours`,

method='ranger',

trControl= my\_control,

tuneGrid = tgrid,

num.trees = 400,

importance = "permutation")

plot(rf\_mod)

plot(varImp(rf\_mod))

#Predictions

prediction\_RF = predict(rf\_mod, test[, -14])

#Calculate RMSE

RMSE(test[, 14], prediction\_RF)

#XGBoost

param\_list = list(

objective = "reg:linear",

eta=0.01,

gamma = 1,

max\_depth=6,

subsample=0.8,

colsample\_bytree=0.5

)

dtrain = xgb.DMatrix(data = as.matrix(X[,-14]), label= X[,14])

dtest = xgb.DMatrix(data = as.matrix(Y[,-14]), label = Y[,14])

set.seed(112)

xgbcv = xgb.cv(params = param\_list,

data = dtrain,

nrounds = 1000,

nfold = 5,

print\_every\_n = 10,

early\_stopping\_rounds = 30,

maximize = F)

xgb\_model = xgb.train(data = dtrain, params = param\_list, nrounds = 205)

prediction\_XG = predict(xgb\_model, dtest)

#Calculate RMSE

RMSE(test[, 14], prediction\_XG)

var\_imp = xgb.importance(feature\_names = setdiff(names(train), c("Absenteeism time in hours")),

model = xgb\_model)

xgb.plot.importance(var\_imp)

**PYTHON CODE:-**

import os

os.chdir("E:")

Importing Library

import pandas as pd

import numpy as np # For mathematical calculations

import seaborn as sns # For data visualization

import matplotlib as mpl

print('Matplotlib version: ', mpl.\_\_version\_\_) # >= 2.0.0

import matplotlib.pyplot as plt # For plotting graphs

%matplotlib inline

import seaborn as sns

from fancyimpute import KNN

import warnings # To ignore any warnings

warnings.filterwarnings("ignore")

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_absolute\_error

from sklearn.ensemble import RandomForestRegressor

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import Imputer

from sklearn.model\_selection import cross\_val\_score

from sklearn import linear\_model

import xgboost as xgb

from sklearn import preprocessing

from sklearn.metrics import mean\_squared\_error

Reading File

train = pd.read\_excel("Absenteeism\_at\_work\_Project.xls")

Understanding the Data

train.shape

train.info()

train.head()

CONVERT THE VARIABLE TO ITS APPROPRIATE DATA TYPE

train['Month of absence'] = train['Month of absence'].astype(object)

train['Day of the week'] = train['Day of the week'].astype(object)

train['Seasons'] = train['Seasons'].astype(object)

train['Education'] = train['Education'].astype(object)

train['Son'] = train['Son'].astype(object)

train['Social drinker'] = train['Social drinker'].astype(object)

train['Social smoker'] = train['Social smoker'].astype(object)

train['Pet'] = train['Pet'].astype(object)

UNIVARIATE ANALYSIS

Target Variable

count,bin\_edges=np.histogram(train['Absenteeism time in hours'].dropna())

train['Absenteeism time in hours'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

INDEPENDENT CONTINOUS VARIABLE

count,bin\_edges=np.histogram(train['ID'])

train['ID'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Reason for absence'].dropna())

train['Reason for absence'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Transportation expense'].dropna())

train['Transportation expense'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Distance from Residence to Work'].dropna())

train['Distance from Residence to Work'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Distance from Residence to Work'].dropna())

train['Distance from Residence to Work'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Service time'].dropna())

train['Service time'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Age'].dropna())

train['Age'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Work load Average/day'].dropna())

train['Work load Average/day'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Hit target'].dropna())

train['Hit target'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Weight'].dropna())

train['Weight'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Height'].dropna())

train['Height'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['Body mass index'].dropna())

train['Body mass index'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

INDEPENDENT CATEGORICAL VARIABLE

train['Month of absence'].value\_counts(normalize=True).plot.bar()

train['Day of the week'].value\_counts(normalize=True).plot.bar()

train['Seasons'].value\_counts(normalize=True).plot.bar()

train['Disciplinary failure'].value\_counts(normalize=True).plot.bar()

train['Education'].value\_counts(normalize=True).plot.bar()

train['Son'].value\_counts(normalize=True).plot.bar()

train['Social drinker'].value\_counts(normalize=True).plot.bar()

train['Social smoker'].value\_counts(normalize=True).plot.bar()

train['Pet'].value\_counts(normalize=True).plot.bar()

BIVARIATE ANALYSIS

#correlation between ID and Absenteeism time in hours

sns.regplot(x = "ID", y = "Absenteeism time in hours", data = train)

train['ID'].corr(train['Absenteeism time in hours'])

#correlation between Reason for absence and Absenteeism time in hours

sns.regplot(x = "Reason for absence", y = "Absenteeism time in hours", data = train)

train['Reason for absence'].corr(train['Absenteeism time in hours'])

#correlation between Transportation expence and Absenteeism time in hours

sns.regplot(x = "Transportation expense", y = "Absenteeism time in hours", data = train)

train['Transportation expense'].corr(train['Absenteeism time in hours'])

#correlation between Distance from residence to Work and Absenteeism time in hours

sns.regplot(x = "Distance from Residence to Work", y = "Absenteeism time in hours", data = train)

train['Distance from Residence to Work'].corr(train['Absenteeism time in hours'])

#correlation between Service time and Absenteeism time in hour

sns.regplot(x = "Service time", y = "Absenteeism time in hours", data = train)

train['Service time'].corr(train['Absenteeism time in hours'])

#correlation between Age and Absenteeism time in hours

sns.regplot(x = "Age", y = "Absenteeism time in hours", data = train)

train['Age'].corr(train['Absenteeism time in hours'])

#correlation between work load Average/day and Absenteeism time in hours

sns.regplot(x = "Work load Average/day", y = "Absenteeism time in hours", data = train)

train['Work load Average/day'].corr(train['Absenteeism time in hours'])

#Correlation between Hit target and Absenteeism time in hours

sns.regplot(x = "Hit target", y = "Absenteeism time in hours", data = train)

train['Hit target'].corr(train['Absenteeism time in hours'])

#correlation between Weight and Absenteeism time in hours

sns.regplot(x = "Weight", y = "Absenteeism time in hours", data = train)

train['Weight'].corr(train['Absenteeism time in hours'])

#correlation between Height and Absenteeism time in hours

sns.regplot(x = "Height", y = "Absenteeism time in hours", data = train)

train['Height'].corr(train['Absenteeism time in hours'])

#correlation between Body Mass index and Absenteeism time in hours

sns.regplot(x = "Body mass index", y = "Absenteeism time in hours", data = train)

train['Body mass index'].corr(train['Absenteeism time in hours'])

#Box plot for continous and categorical variables

sns.boxplot(x = "Month of absence", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Day of the week", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Seasons", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Seasons", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Disciplinary failure", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Education", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Son", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Social drinker", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Social smoker", y = "Absenteeism time in hours", data = train)

sns.boxplot(x = "Pet", y = "Absenteeism time in hours", data = train)

MISSING VALUE ANALYSIS

#Create dataframe with missing percentage

missing\_val = pd.DataFrame(train.isnull().sum())

#Reset index

missing\_val = missing\_val.reset\_index()

#Rename variable

missing\_val = missing\_val.rename(columns = {'index': 'Variables', 0: 'Missing\_percentage'})

#Calculate percentage

missing\_val['Missing\_percentage'] = (missing\_val['Missing\_percentage']/len(train))\*100

#descending order

missing\_val = missing\_val.sort\_values('Missing\_percentage', ascending = False).reset\_index(drop = True)

missing\_val

#Imputing Weight and Height with mean for missing values

train['Weight'] = train['Weight'].fillna(train['Weight'].mean())

train['Height'] = train['Height'].fillna(train['Height'].mean())

#Imputing Body Mass Index

for i in range(len(train)):

if(train['Body mass index'].iloc[i] == np.nan):

train['Body mass index'].iloc[i] = (train['Weight'].iloc[i]) \* 10000/(train['Height'].iloc[i])\*\*2

#Replacing 0 value of month and Reason for Absence ti NA

for i in range(len(train)):

if(train['Month of absence'].iloc[i] == 0):

train['Month of absence'].iloc[i] = np.nan

for i in range(len(train)):

if(train['Reason for absence'].iloc[i] == 0):

train['Reason for absence'].iloc[i] = np.nan

#KNN imputation

#Assigning levels to the categories

lis = []

for i in range(0, train.shape[1]):

if(train.iloc[:,i].dtypes == 'object'):

train.iloc[:,i] = pd.Categorical(train.iloc[:,i])

train.iloc[:,i] = train.iloc[:,i].cat.codes

train.iloc[:,i] = train.iloc[:,i].astype('object')

lis.append(train.columns[i])

#replace -1 with NA to impute

for i in range(0, train.shape[1]):

train.iloc[:,i] = train.iloc[:,i].replace(-1, np.nan)

#Apply KNN imputation algorithm

train = pd.DataFrame(KNN(k = 3).complete(train), columns = train.columns)

#Convert into proper datatypes

for i in lis:

train.loc[:,i] = train.loc[:,i].round()

train.loc[:,i] = train.loc[:,i].astype('object')

FEATURE SELEACTION

#Saving the numerical data

cnames = ["ID","Reason for absence","Transportation expense","Service time","Work load Average/day","Hit target","Weight",

"Height","Body mass index","Age"]

##Correlation analysis

#Correlation plot

df\_corr = train.loc[:,cnames]

#Set the width and hieght of the plot

f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix

corr = df\_corr.corr()

#Plot using seaborn library

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),

square=True, ax=ax)

train = train.drop(['ID','Service time','Weight','Work load Average/day','Hit target','Month of absence','Pet'], axis=1)

FEATURE SCALING

cnames = ["Reason for absence","Distance from Residence to Work","Age","Transportation expense","Height","Body mass index"]

#Nomalisation

for i in cnames:

print(i)

train[i] = (train[i] - min(train[i]))/(max(train[i]) - min(train[i]))

# MODEL DEVELOPMENT

features = ["Reason for absence","Day of the week","Seasons","Transportation expense","Distance from Residence to Work","Age","Disciplinary failure",

"Education","Son","Social drinker","Social smoker","Height","Body mass index"]

X = train[features]

Y = train['Absenteeism time in hours']

#Decision Tree Model

# split data into training and validation data, for both features and target

# The split is based on a random number generator. Supplying a numeric value to

# the random\_state argument guarantees we get the same split every time we

# run this script.

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, Y, random\_state = 0)

# Define model

model1 = DecisionTreeRegressor()

# Fit model

model1.fit(train\_X, train\_y)

# get predicted prices on validation data

val\_predictions = model1.predict(val\_X)

print(mean\_absolute\_error(val\_y, val\_predictions))

lm = linear\_model.LinearRegression()

model = lm.fit(train\_X,train\_y)

predictions = lm.predict(val\_X)

print(mean\_absolute\_error(val\_y, predictions))

#Tunning the Decision Tree Model

def get\_mae(max\_leaf\_nodes, train\_X, val\_X, train\_y, val\_y):

model2 = DecisionTreeRegressor(max\_leaf\_nodes=max\_leaf\_nodes, random\_state=0)

model2.fit(train\_X, train\_y)

preds\_val = model2.predict(val\_X)

mae = mean\_absolute\_error(val\_y, preds\_val)

return(mae)

# compare MAE with differing values of max\_leaf\_nodes

for max\_leaf\_nodes in [5, 50, 500, 5000]:

my\_mae = get\_mae(max\_leaf\_nodes, train\_X, val\_X, train\_y, val\_y)

print("Max leaf nodes: %d \t\t Mean Absolute Error: %d" %(max\_leaf\_nodes, my\_mae))

#Random Forest Model

forest\_model = RandomForestRegressor(random\_state=1)

forest\_model.fit(train\_X, train\_y)

preds = forest\_model.predict(val\_X)

print(mean\_absolute\_error(val\_y, preds))

#Pipeline

my\_pipeline = make\_pipeline(Imputer(), RandomForestRegressor())

my\_pipeline.fit(train\_X, train\_y)

predictions = my\_pipeline.predict(val\_X)

print(mean\_absolute\_error(val\_y, predictions))

my\_imputer = Imputer()

my\_model = RandomForestRegressor()

imputed\_train\_X = my\_imputer.fit\_transform(train\_X)

imputed\_val\_X = my\_imputer.transform(val\_X)

my\_model.fit(imputed\_train\_X, train\_y)

predictions = my\_model.predict(imputed\_val\_X)

print(mean\_absolute\_error(val\_y, predictions))

#Cross-validation

my\_pipeline = make\_pipeline(Imputer(), RandomForestRegressor())

scores = cross\_val\_score(my\_pipeline, X, Y, scoring='neg\_mean\_absolute\_error')

print(scores)

print('Mean Absolute Error %2f' %(-1 \* scores.mean()))

#XGBoost Model

for f in X.columns:

if X[f].dtype=='object':

lbl = preprocessing.LabelEncoder()

lbl.fit(list(X[f].values))

X[f] = lbl.transform(list(X[f].values))

X.fillna((-999), inplace=True)

X=np.array(X)

Y=np.array(Y)

X = X.astype(float)

Y = Y.astype(float)

data\_dmatrix = xgb.DMatrix(data=X,label=Y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=123)

xg\_reg = xgb.XGBRegressor(objective ='reg:linear', colsample\_bytree = 0.3, learning\_rate = 0.1,

max\_depth = 5, alpha = 10, n\_estimators = 10)

xg\_reg.fit(X\_train,y\_train)

preds = xg\_reg.predict(X\_test)

rmse = np.sqrt(mean\_squared\_error(y\_test, preds))

print("RMSE: %f" % (rmse))

#K-folds cross validation

params = {"objective":"reg:linear",'colsample\_bytree': 0.3,'learning\_rate': 0.1,

'max\_depth': 5, 'alpha': 10}

cv\_results = xgb.cv(dtrain=data\_dmatrix, params=params, nfold=3,

num\_boost\_round=50,early\_stopping\_rounds=10,metrics="rmse", as\_pandas=True, seed=123)

cv\_results.head()

#Extract and print the final boosting round metric

print((cv\_results["test-rmse-mean"]).tail(1))

#Visualize Boosting Trees and Feature Importance

xg\_reg = xgb.train(params=params, dtrain=data\_dmatrix, num\_boost\_round=30)

xgb.plot\_tree(xg\_reg,num\_trees=0)

plt.rcParams['figure.figsize'] = [50, 10]

plt.show()

xgb.plot\_importance(xg\_reg)

plt.rcParams['figure.figsize'] = [5, 5]

plt.show()