EMPLOYEE ABSENTEEISM

Mayur Sharma

Data Scientist

Contents

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. The company has shared it dataset and requested to have an answer on the following areas:
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?

1.2 Dataset

Sample Dataset-

ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work
11	26.0	7.0	3	1	289.0	36.0
36	0.0	7.0	3	1	118.0	13.0
3	23.0	7.0	4	1	179.0	51.0
7	7.0	7.0	5	1	279.0	5.0
11	23.0	7.0	5	1	289.0	36.0
3	23.0	7.0	6	1	179.0	51.0

Service time	Age	Work load Average/day	Hit target	Disciplinary failure	Education	Son
13.0	33.0	239554.0	97.0	0.0	1.0	2.0
18.0	50.0	239554.0	97.0	1.0	1.0	1.0
18.0	38.0	239554.0	97.0	0.0	1.0	0.0
14.0	39.0	239554.0	97.0	0.0	1.0	2.0
13.0	33.0	239554.0	97.0	0.0	1.0	2.0
18.0	38.0	239554.0	97.0	0.0	1.0	0.0

Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
1.0	0.0	1.0	90.0	172.0	30.0	4.0
1.0	0.0	0.0	98.0	178.0	31.0	0.0
1.0	0.0	0.0	89.0	170.0	31.0	2.0
1.0	1.0	0.0	68.0	168.0	24.0	4.0
1.0	0.0	1.0	90.0	172.0	30.0	2.0
1.0	0.0	0.0	89.0	170.0	31.0	NaN

Dataset has 21 variables in which 20 variables are independent and 1 (Absenteeism time in hours) is dependent variable. Since target variable is continuous in nature, this is a regression problem.

Attribute Information:
1. Individual identification (ID)
2. Reason for absence (ICD).
Absences attested by the International Code of Diseases (ICD) stratified into 21 categories (I to XXI) as follows:
I Certain infectious and parasitic diseases
II Neoplasms
III Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
IV Endocrine, nutritional and metabolic diseases
V Mental and behavioural disorders
VI Diseases of the nervous system
VII Diseases of the eye and adnexa
VIII Diseases of the ear and mastoid process
IX Diseases of the circulatory system
X Diseases of the respiratory system
XI Diseases of the digestive system
XII Diseases of the skin and subcutaneous tissue
XIII Diseases of the musculoskeletal system and connective tissue
XIV Diseases of the genitourinary system
XV Pregnancy, childbirth and the puerperium
XVI Certain conditions originating in the perinatal period
XVII Congenital malformations, deformations and chromosomal abnormalities
XVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
XIX Injury, poisoning and certain other consequences of external causes
XX External causes of morbidity and mortality
XXI Factors influencing health status and contact with health services.
And 7 categories without (CID) patient follow-up **(22),** medical consultation **(23),** blood donation **(24)**, laboratory examination **(25)**, unjustified absence **(26)**, physiotherapy **(27),** dental consultation **(28)**.
3. Month of absence
4. Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))
5. Seasons (summer (1), autumn (2), winter (3), spring (4))
6. Transportation expense
7. Distance from Residence to Work (kilo meters)
8. Service time
9. Age
10. Work load Average/day
11. Hit target
12. Disciplinary failure (yes=1; no=0)
13. Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))
14. Son (number of children)

15. Social drinker (yes=1; no=0)
16. Social smoker (yes=1; no=0)
17. Pet (number of pet)
18. Weight
19. Height
20. Body mass index
21. Absenteeism time in hours (target)

1.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analysing data sets to summarize their main characteristics. In the given data set there are 21 variables and data types of all variables are either float64 or int64. There are 740 observations and 21 columns in our data set. Missing value is also present in our data.

```
ID                                int64
Reason for absence              float64
Month of absence                float64
Day of the week                   int64
Seasons                           int64
Transportation expense          float64
Distance from Residence to Work float64
Service time                    float64
Age                             float64
Work load Average/day           float64
Hit target                      float64
Disciplinary failure            float64
Education                       float64
Son                             float64
Social drinker                  float64
Social smoker                   float64
Pet                             float64
Weight                          float64
Height                          float64
Body mass index                 float64
Absenteeism time in hours       float64
```

From EDA we have concluded that there are 10 continuous variable and 11 categorical variable in nature.

Continuous variables in dataset-

- Transportation expense
- Distance from Residence to Work
- Service time
- Age
- Work load Average/day
- Hit target
- Weight
- Height
- Body mass index
- Absenteeism time in hours

Categorical variables in dataset-

- ID
- Reason for absence
- Month of absence
- Day of the week
- Seasons
- Disciplinary failure
- Education
- Son
- Social drinker
- Social smoker
- Pet

From EDA we have concluded the number of unique values in each variables.

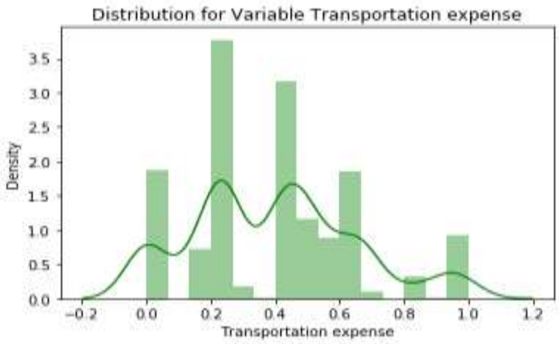
Reason for absence	**28**
Month of absence	**12**
Day of the week	**5**
Seasons	**4**
Transportation expense	**24**
Distance from Residence to Work	**25**
Service time	**18**
Age	**22**
Work load Average/day	**38**
Hit target	**13**
Disciplinary failure	**2**
Education	**4**
Son	**5**
Social drinker	**2**
Social smoker	**2**
Pet	**6**
Weight	**26**
Height	**14**
Body mass index	**17**
Absenteeism time in hours	**19**

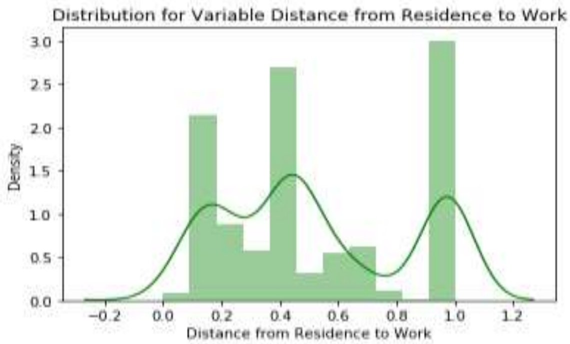
2. Methodology

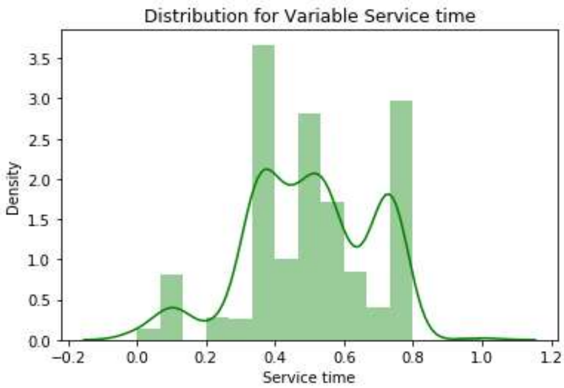
Before feeding the data to the model we need to clean the data and convert it to a proper format. It is the most crucial part of data science. In this we have to apply different pre-processing techniques to clean the data and to convert it into proper format.

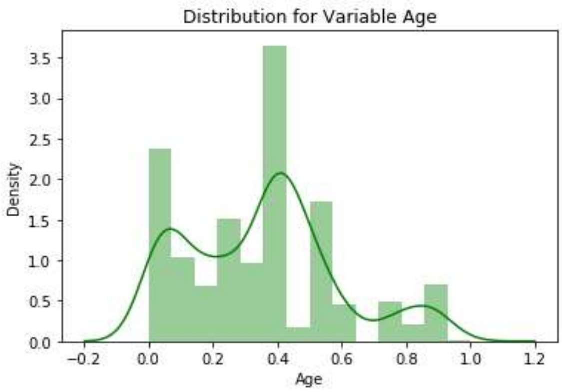
2.1 Data Pre Processing

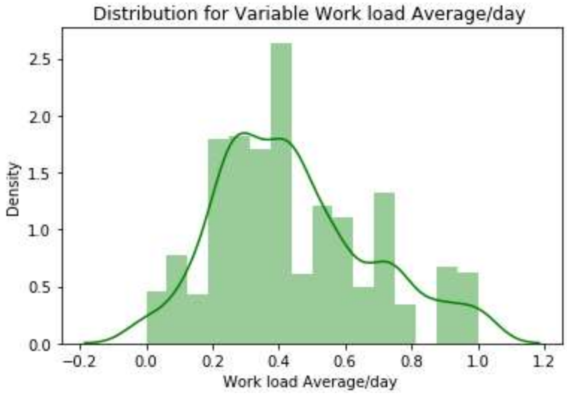
Any predictive modelling requires that we look at the data before we start modelling. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

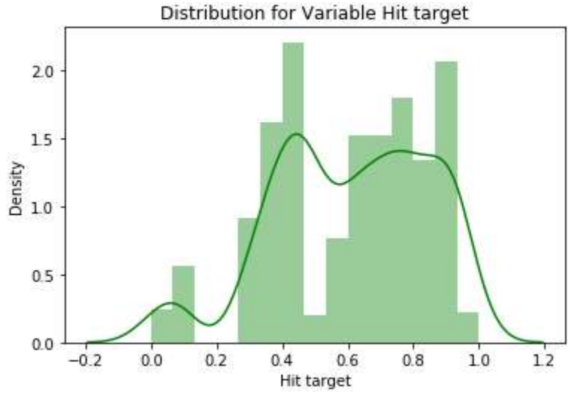


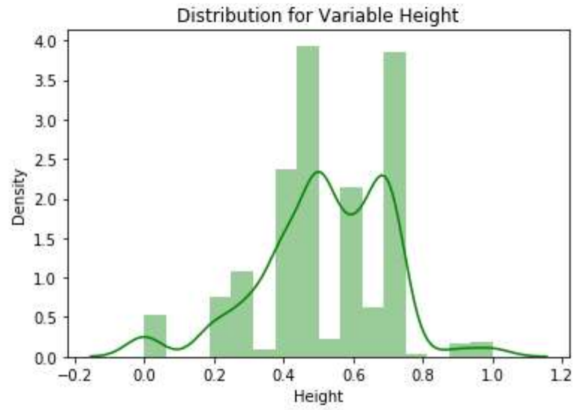


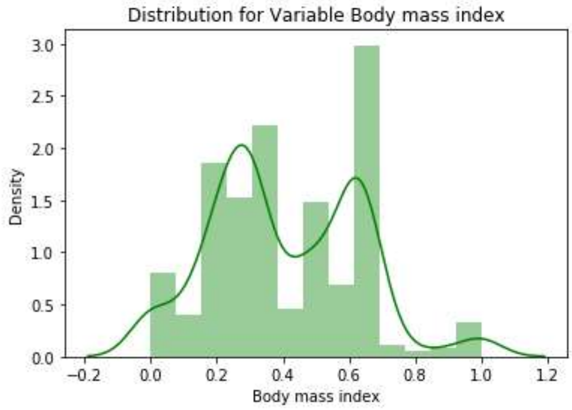


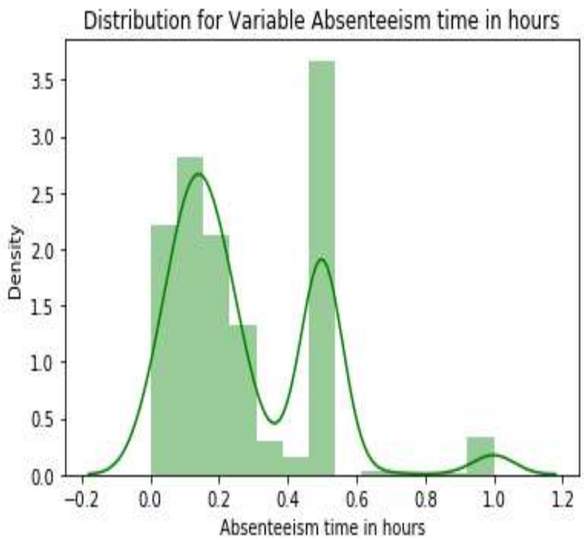












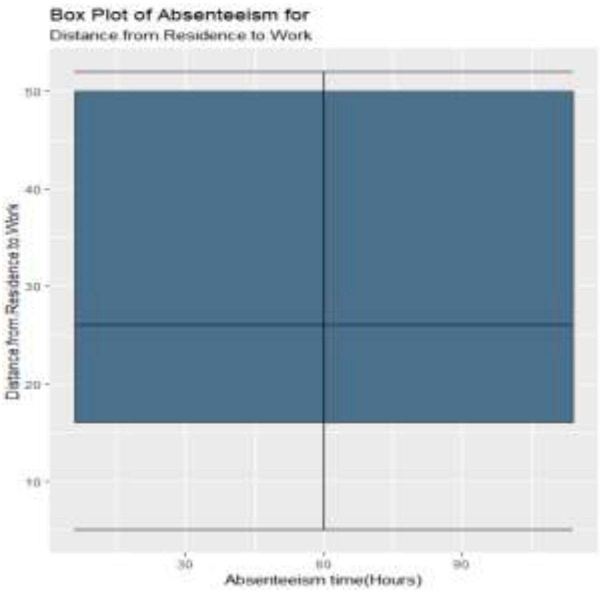
2.2.1 Missing Value Analysis

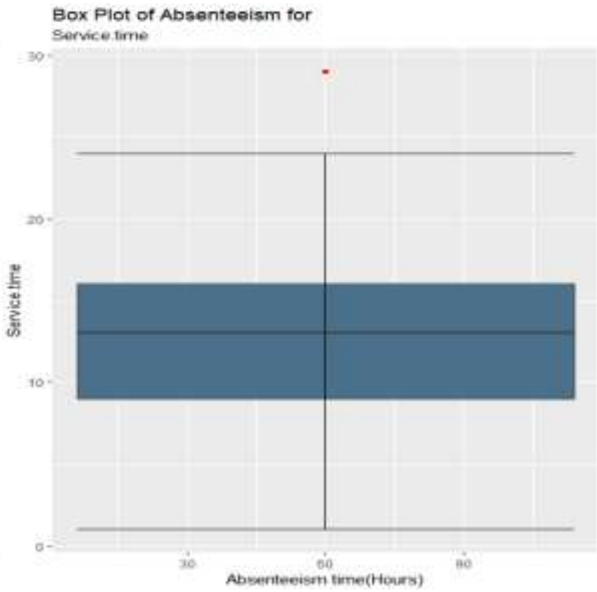
In statistics, *missing data*, or *missing values*, occur when no *data value* is stored for the variable in an observation. If a columns has more than 30% of data as missing value either we ignore the entire column or we ignore those observations. In the given data the maximum percentage of missing value is 4.189% for **body mass index** column. So we will compute missing value for all the columns.

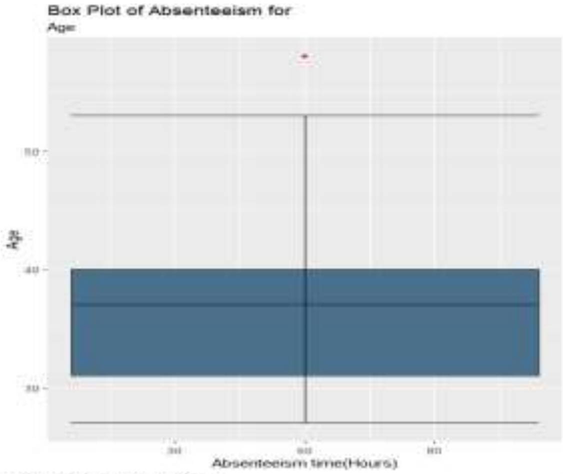
In our project we applied different-different techniques to impute the numeric missing value like- Mean, Median and KNN Imputation method, and select **KNN Imputation,** as we found it more accurate for continuous variable while testing on sample dataset as compare to others. For categorical variables we used mode method to impute the missing values. After applying mode for categorical and KNN for continuous variables we recheck the missing values, now data is free from missing values.

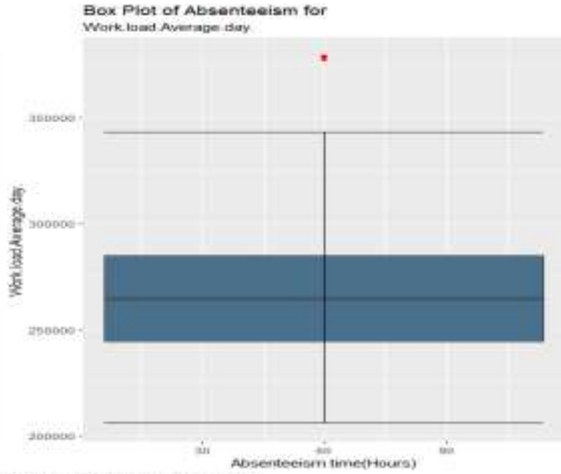
2.1.2 <u>Outlier Analysis</u>

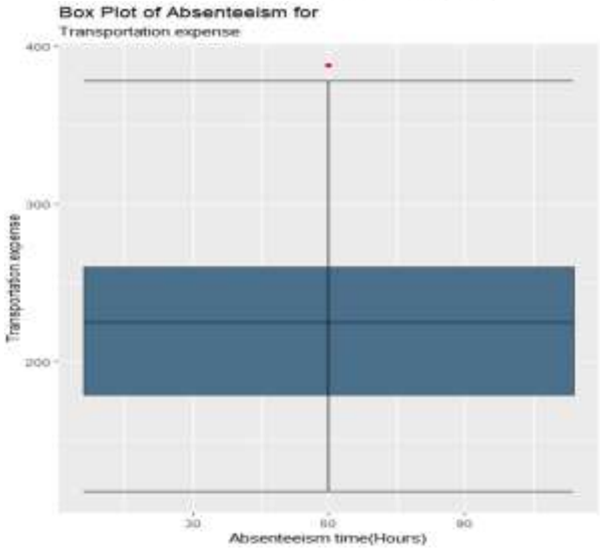
We can clearly observe from these probability distributions that most of the variables are skewed. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data. One of the other steps of pre-processing apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers. We visualize the outliers using boxplots.
In figure we have plotted the boxplots of the 11 predictor variables with respect to target variable **Absenteeism time in hour**, and detect the outliers by visualization.

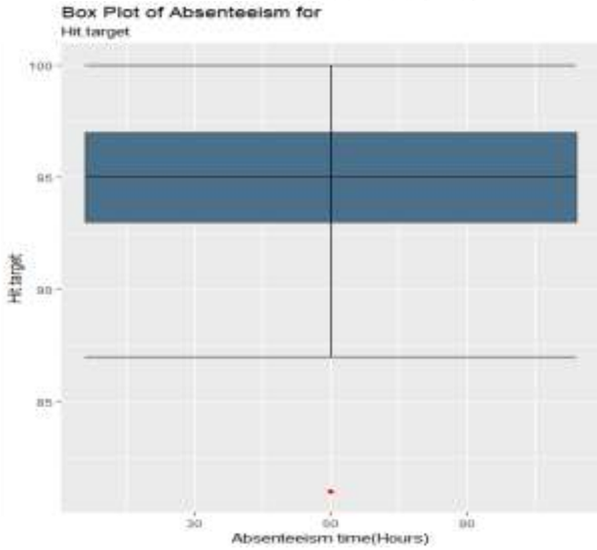


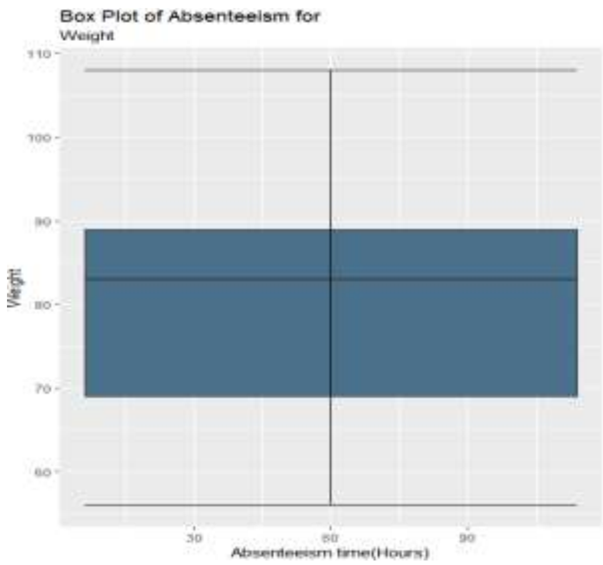


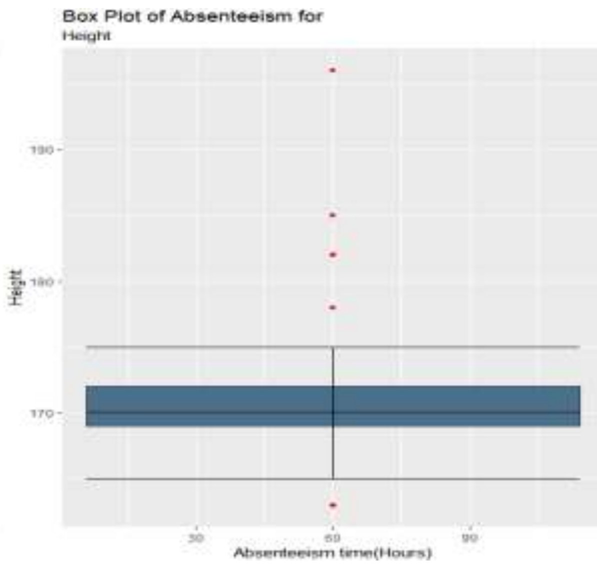

Box Plot of Absenteeism for
Age
Age
Absenteeism time(Hours)

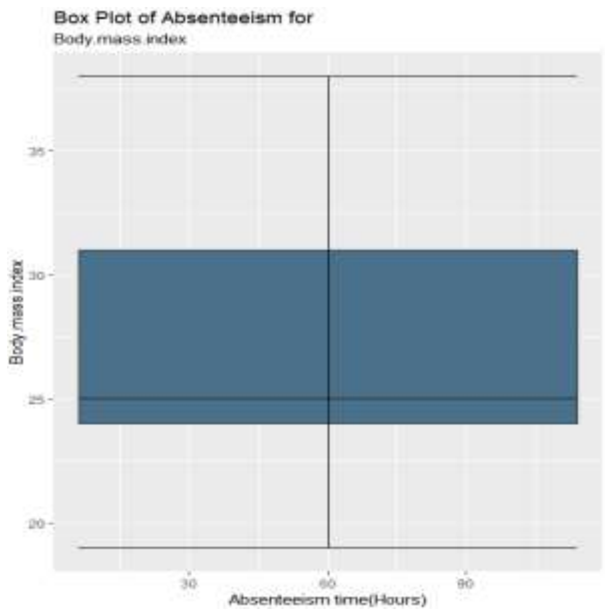

Box Plot of Absenteeism for
Work.load.Average.day
Work.load.Average.day
Absenteeism time(Hours)

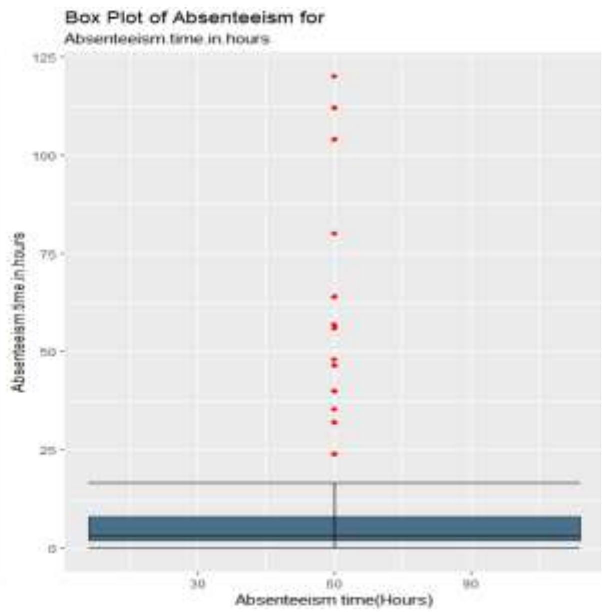

Box Plot of Absenteeism for
Transportation.expense
Transportation.expense
400
300
200
30
60
90
Absenteeism time(Hours)


Box Plot of Absenteeism for
Hit.target
Hit.target
100
95
90
85
30
60
90
Absenteeism time(Hours)


Box Plot of Absenteeism for
Weight
Weight
110
100
90
80
70
60
30
60
90
Absenteeism time(Hours)


Box Plot of Absenteeism for
Height
Height
190
180
170
30
60
90
Absenteeism time(Hours)



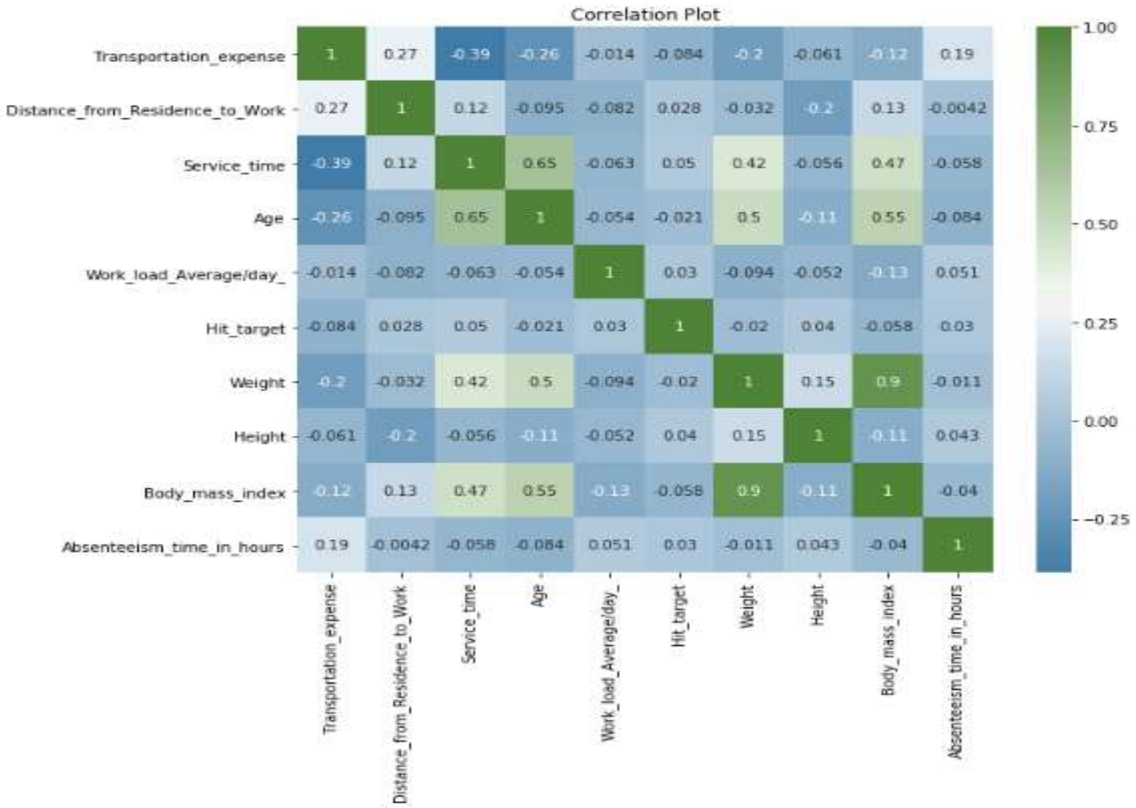


From the boxplot almost all the variables **except "Distance from residence to work", "Weight" and "Body mass index"** consists of outliers. From the boxplot visualization we also can see Maximum outliers are present in variables **"Height**" and **"Absenteeism time in hours"** .We have converted the outliers (data beyond minimum and maximum values) as NA i.e. missing values and fill them by **KNN** imputation method.

2.1.3 <u>Feature Selection</u>

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of prediction. Selecting subset of relevant columns for the model construction is known as Feature Selection. We cannot use all the features because some features may be carrying the same information or irrelevant information which can increase overhead. To reduce overhead we adopt feature selection technique to extract meaningful features out of data. This in turn helps us to avoid the problem of multi collinearity. In this project we have selected **Correlation Analysis** for numerical variable and **ANOVA** (Analysis of variance) for categorical variables.

Correlation Analysis plot for continuous variables-



ANOVA Analysis for categorical variables-

	sum_sq	df	F	PR(>F)
Reason_for_absence	204.282325	1	17.915574	0.000026
Residual	8164.189531	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Month_of_absence	0.292084	1	0.024991	0.874433
Residual	8368.179772	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Day_of_the_week	54.172342	1	4.665143	0.031112
Residual	8314.299514	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Seasons	29.53959	1	2.536337	0.111694
Residual	8338.932266	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Disciplinary_failure	668.371022	1	62.149011	1.19E-14
Residual	7700.100834	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Education	3.693486	1	0.316151	0.574106
Residual	8364.77837	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Son	218.839218	1	19.226496	0.000013
Residual	8149.632638	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Social_drinker	70.901908	1	6.118149	0.013611

Residual	8297.569948	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Social_smoker	19.623439	1	1.682913	0.194956
Residual	8348.848417	716	NaN	NaN
	sum_sq	df	F	PR(>F)
Pet	4.589346	1	0.392876	0.530991
Residual	8363.88251	716	NaN	NaN

From correlation analysis we have found that **Weight** and **Body mass index** has high correlation (>0.7), so we have excluded the **Weight** column, and from ANOVA analysis we have found that in categorical variables **Pet**, **Social smoker**, **Education** and **Month of absence** have the pr(>0.05), so we excluded them.
After Correlation Analysis we have remaining variables are-

Continuous variables in dataset-

- Transportation expense
- Distance from Residence to Work
- Service time
- Age
- Work load Average/day
- Hit target
- Height
- Body mass index
- Absenteeism time in hours

Categorical variables in dataset-

- Reason for absence
- Day of the week
- Disciplinary failure
- Son
- Social drinker

2.1.4 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Since our data is not uniformly distributed we will use **Normalization** as Feature Scaling Method.

2.2 Model Development

After Data pre-processing the next step is to develop a model using a train or historical data Which can perform to predict accurate result on test data or new data. Here we have tried with different model and will choose the model which will provide the most accurate values.

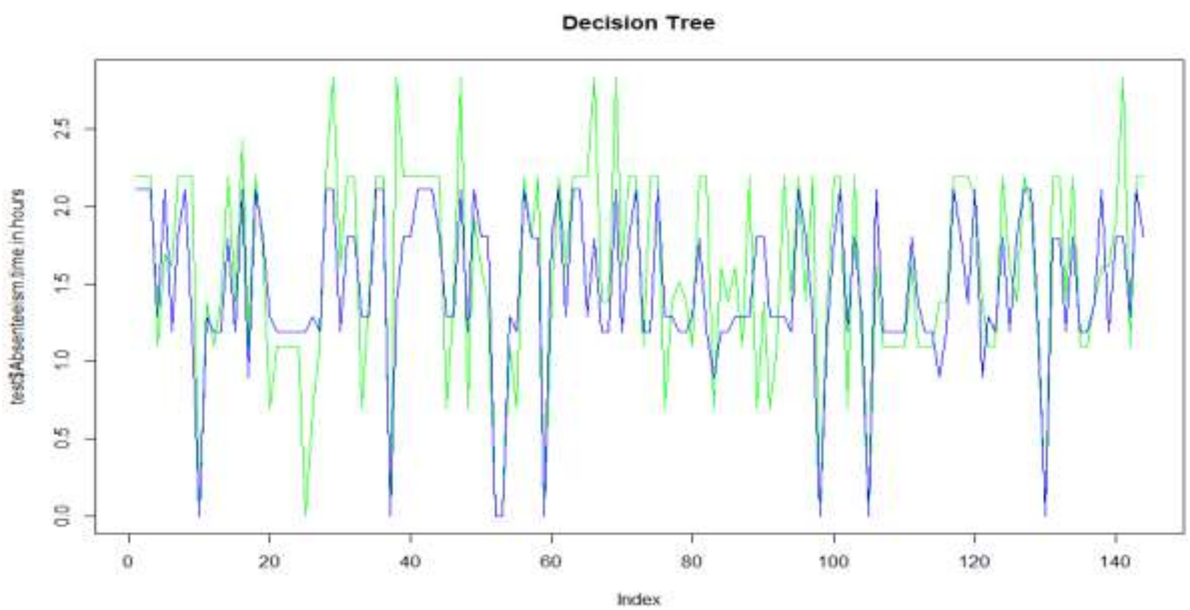
2.2.1 Decision Tree

Decision Tree is a supervised machine learning algorithm, which is used to predict the data for classification and regression. It accepts both continuous and categorical variables. A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each branch connects nodes with “and” and multiple branches are connected by “or”. Extremely easy to understand by the business users. It provides its output in the form of rule, which can easily understood by a non-technical person also.

We have prepared a model by using decision tree algorithm and calculate RMSE value and R^2 value for our project in R and Python are –

Decision Tree	R	PYTHON
RMSE Train	0.4303573	0.078849
RMSE Test	0.4250704	0.513122
R^2 Train	0.5594418	0.985408
R^2 Test	0.6366973	0.425600

Visualization for test and predicted test in Decision Tree-

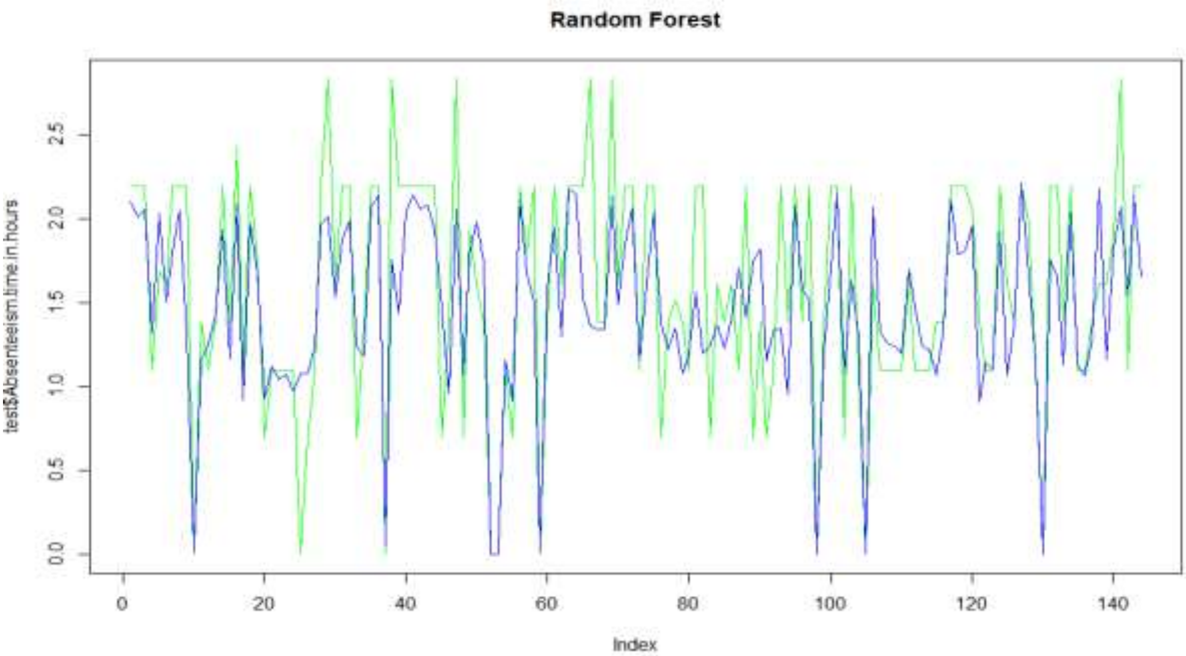


2.2.2 Random Forest

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations. It means to build each decision tree on random forest we are not going to use the same data. The higher no of trees in the random forest will give higher no of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important. The RMSE value and R^2 value for our project in R and Python are –

Random Forest	R	PYTHON
RMSE Train	0.2386952	0.180293
RMSE Test	0.404051	0.383689
R^2 Train	0.8821434	0.923709
R^2 Test	0.6826756	0.678831

Visualization for test and predicted test in Random Forest-



2.2.3 Liner Regression

Linear Regression is one of the statistical method of prediction. It is most common predictive analysis algorithm. It uses only for regression, means if the target variable is continuous than we can use linear regression machine learning algorithm. The RMSE value and R^2 value for our project in R and Python are –

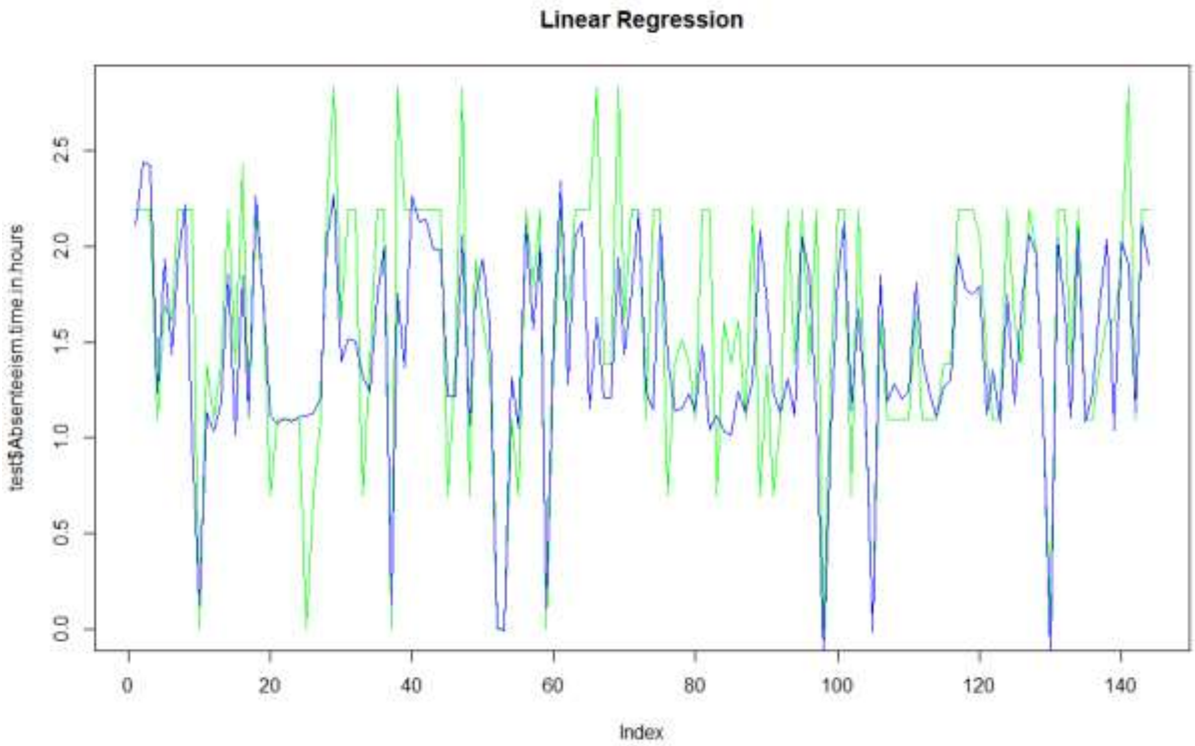
Linear Regression	R	PYTHON
RMSE Train	0.4235679	0.419084
RMSE Test	0.4375786	0.420187
R^2 Train	0.5732328	0.587794
R^2 Test	0.6218227	0.614823

2.2.4 Gradient Boosting

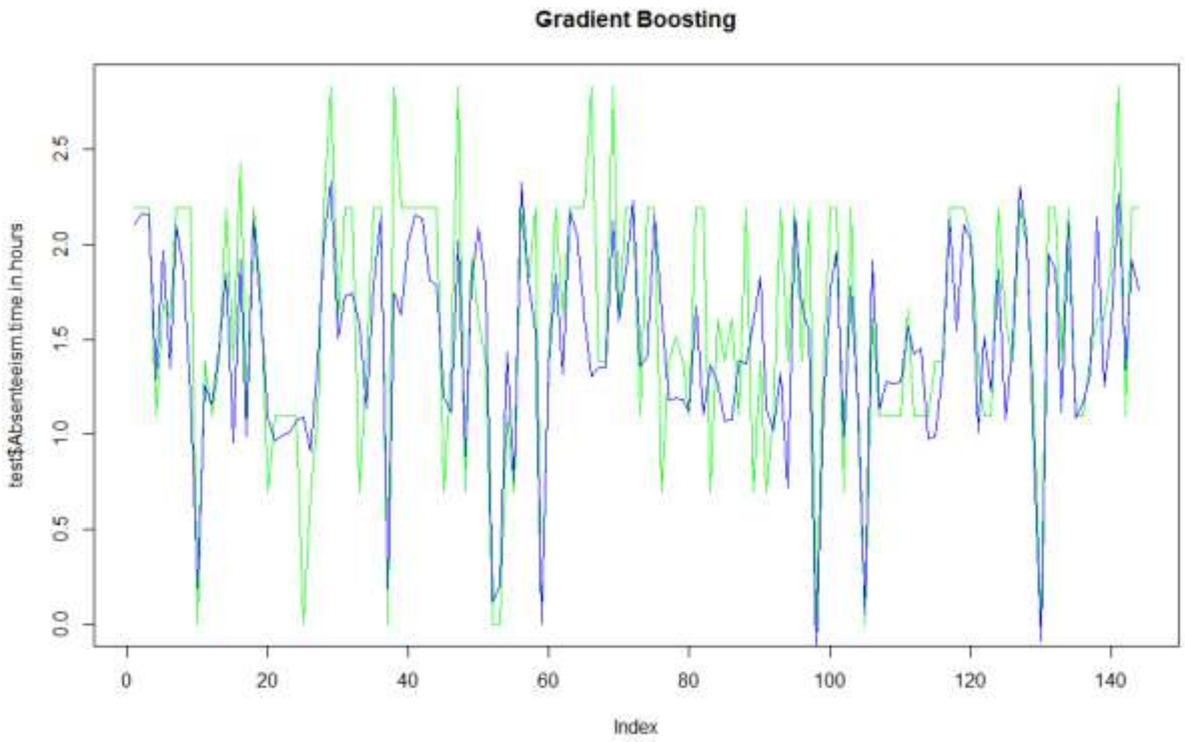
Gradient boosting is a machine learning technique for regression and classification problems, It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The RMSE value and R^2 value for our project in R and Python are –

Gradient Boosting	R	PYTHON
RMSE Train	0.3577656	0.358427
RMSE Test	0.4150737	0.399007
R^2 Train	0.7010977	0.698483
R^2 Test	0.6648008	0.652676

Visualization for test and predicted test in Linear Regression-



Visualization for test and predicted test in Gradient Boosting-



3. <u>Conclusion</u>

In methodology we have done data cleaning and then applied different-different machine learning algorithms on the data set to check the performance of each model, now in conclusion we will finalize the model of Employee Absenteeism dataset.

3.1 <u>Model Evaluation</u>

In the previous chapter we have applied four algorithms on our dataset and calculate the Root Mean Square Error (RMSE) and R-Squared Value for all the models. Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. RMSE is an absolute measure of fit. RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. R-squared is a relative measure of fit. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. Value of R-squared between 0-1, where 0 means independent variable unable to explain the target variable and

1 means target variable is completely explained by the independent variable. So, Lower values of RMSE and higher value of R-Squared Value indicate better fit of model.

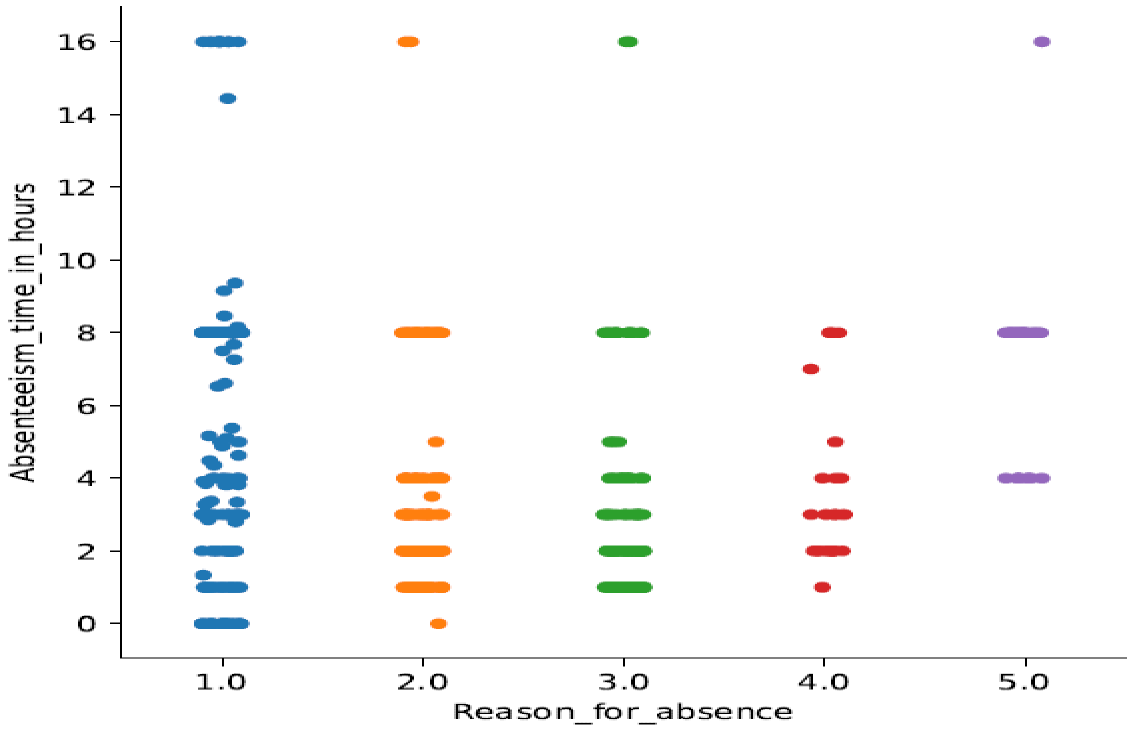
3.2 Model Selection

From the observation of all **RMSE Value** and **R-Squared** Value we have concluded that **Random Forest** has minimum value of RMSE (**0.404051)** and it's **R-Squared** Value is also maximum (**0.68**).Means, By Random forest algorithm predictor are explain 68% to the target variable on the test data. The RMSE value of Test data and Train does not differs a lot this implies that it is not the case of overfitting.

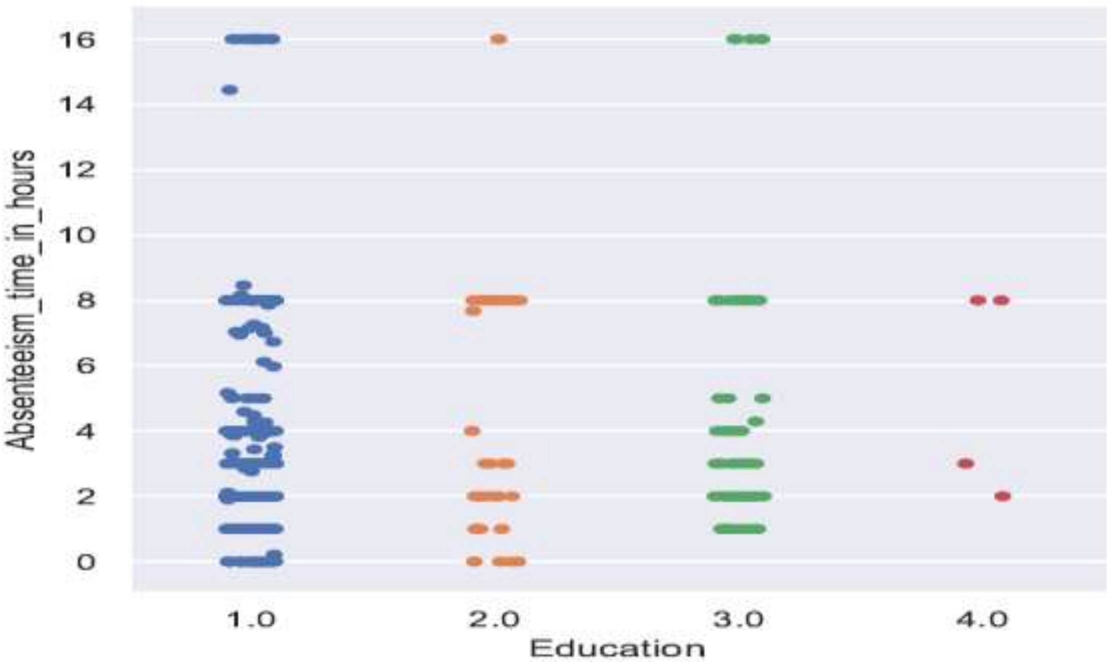
3.3 Answers of asked questions

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

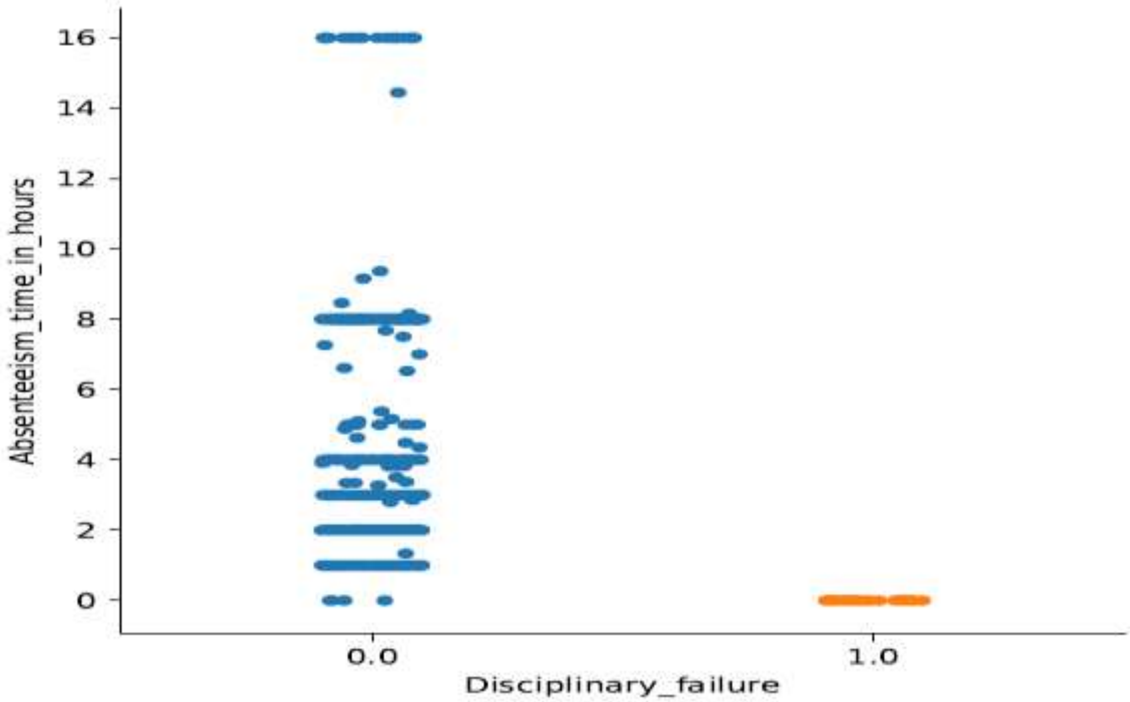
For the above query, we analysed the data properly and plot some visualization to find the answer these question and we found that the maximum tend of absenteeism people as education wise are those who have only high school degree, when we studied the reason for absence we observed that absenteeism is frequently used for category 1, which is code of Diseases. we also observed that that people with no children or no pets tend to be absent more than people who have children or pets, it shows that these people are far from their home town and they are unmarried. We also observe from visualization that the people who are social drinker tend to be absent more as compare to non drinker. Company should focus on these people. There is no such impact of day of week or month or season because absenteeism trend is same for all these variables. We noticed that People with disciplinary failure have maximum absenteeism as compare to non disciplinary failure. From the predictors also observed that Employee who have work load between 240 to 300 minutes and age below 30 and service time is below 8, these employee tend to absent more frequently. These are the main reason for absenteeism, company should focus on these reasons and find out appropriate solution for these. Here below are the visualization by which we observed the data.



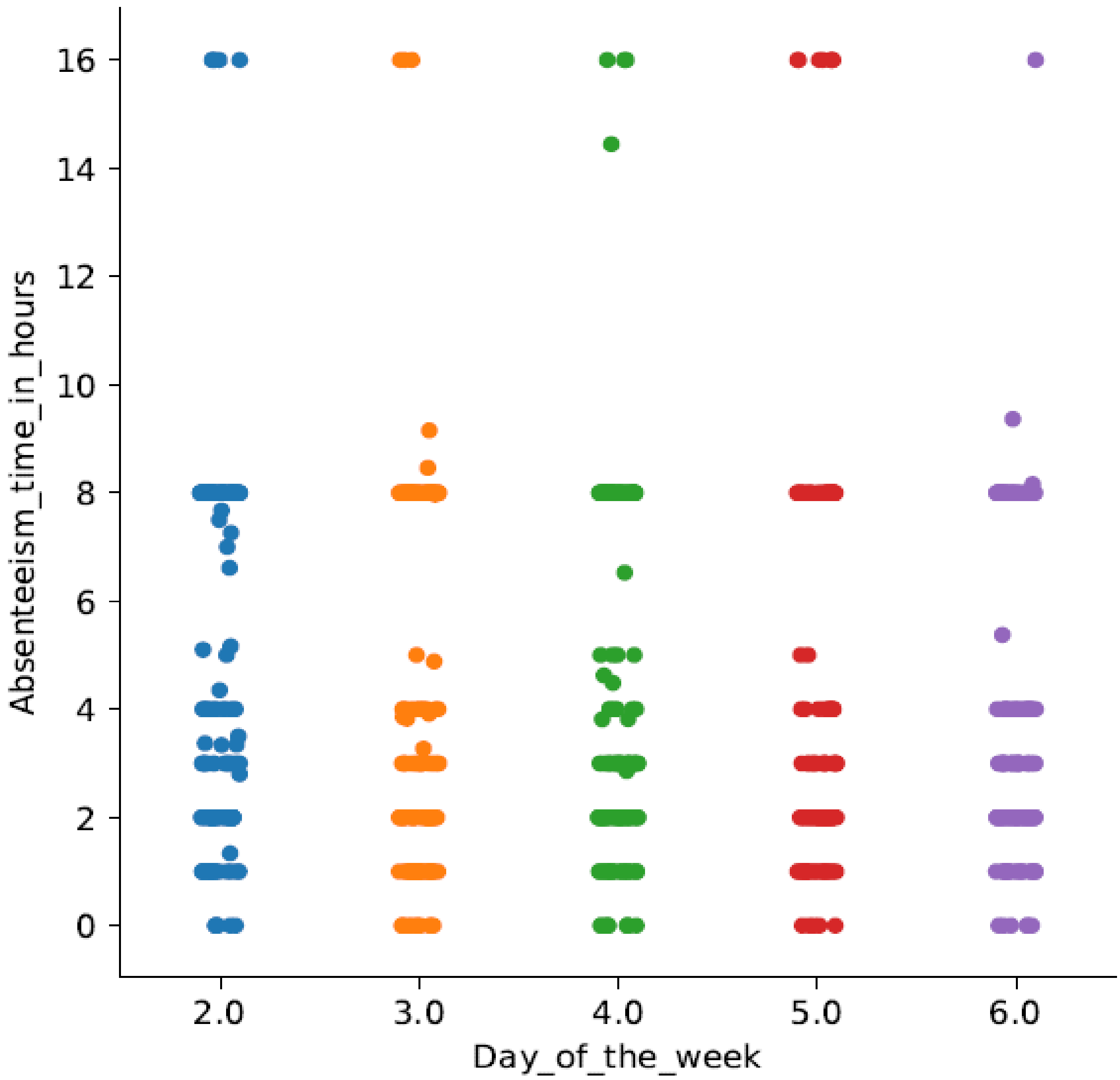
16
14
12
10
8
6
4
2
0
Absenteeism_time_in_hours
1.0
2.0
3.0
4.0
5.0
Reason_for_absence



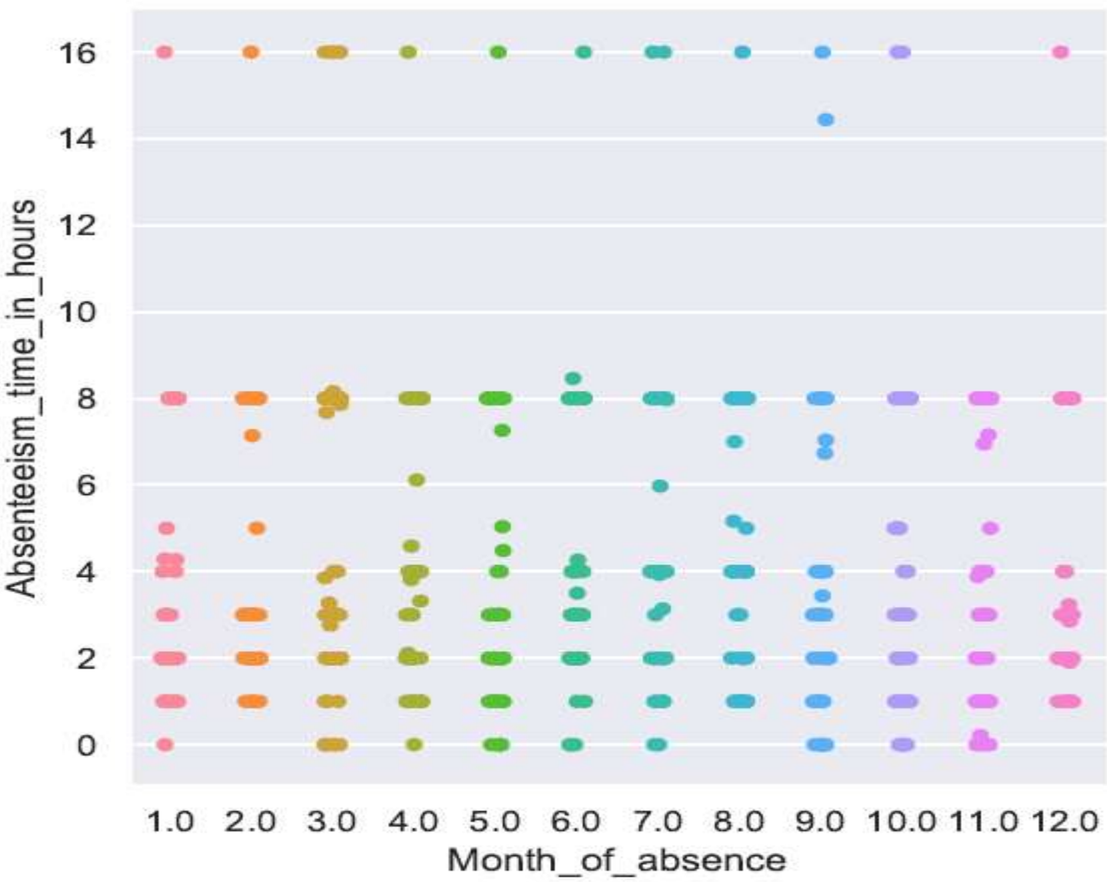
16
14
12
10
8
6
4
2
0
Absenteeism_time_in_hours
1.0
2.0
3.0
4.0
Education



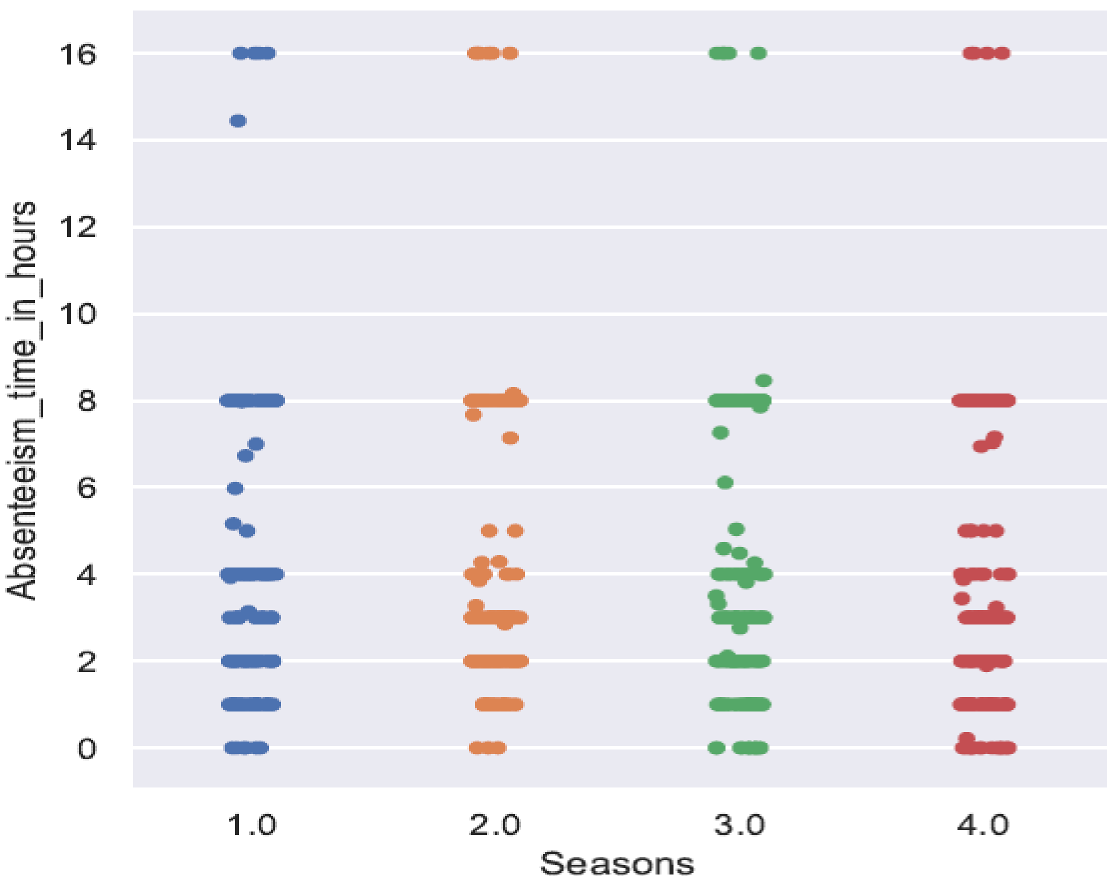
Absenteeism_time_in_hours
Disciplinary_failure
0.0
1.0
0
2
4
6
8
10
12
14
16



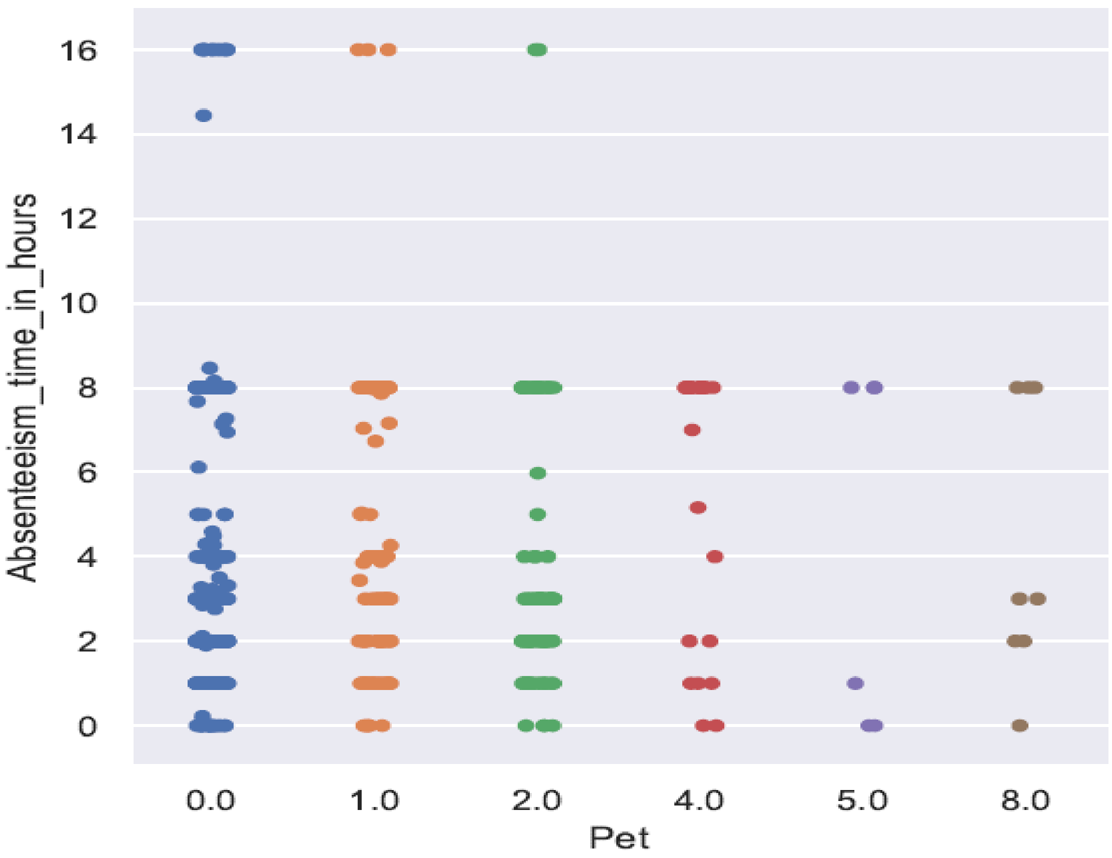
Absenteeism_time_in_hours
Day_of_the_week
2.0
3.0
4.0
5.0
6.0
0
2
4
6
8
10
12
14
16

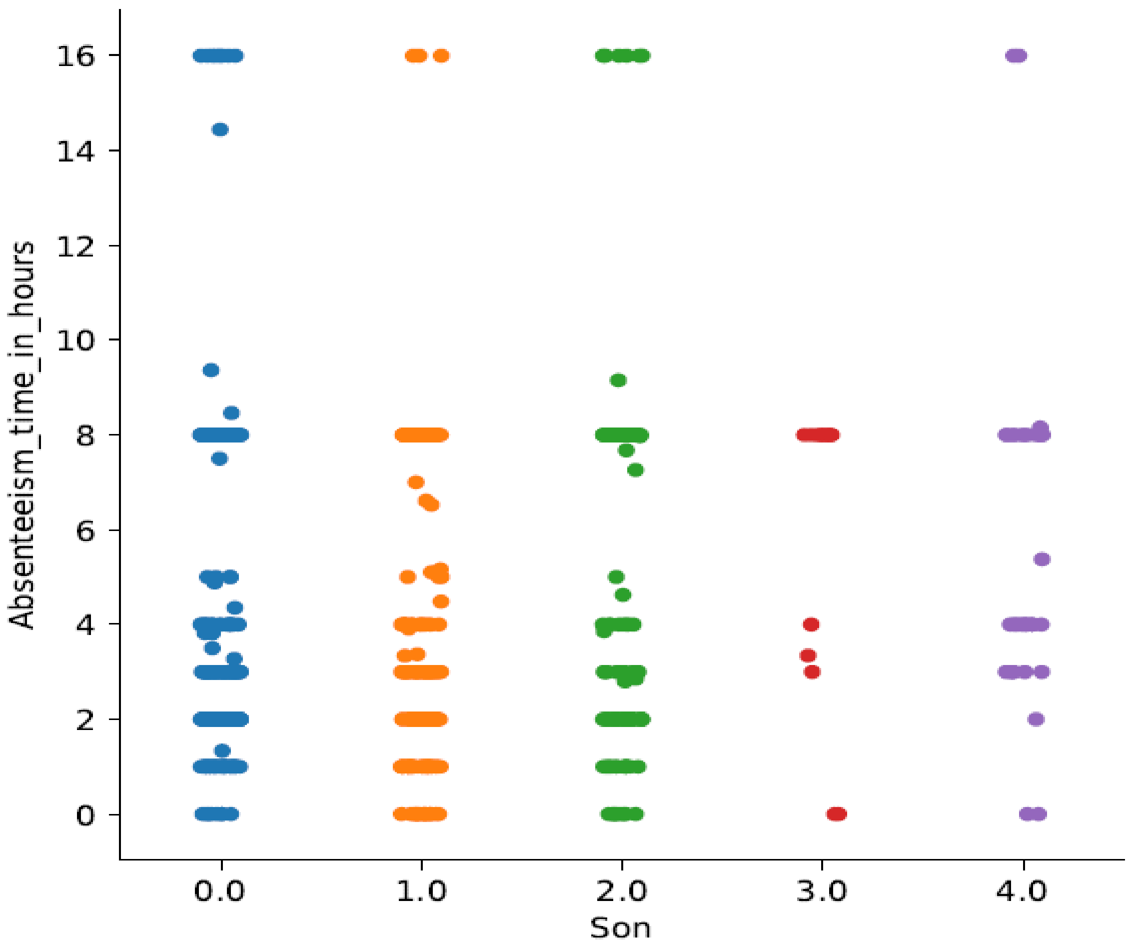


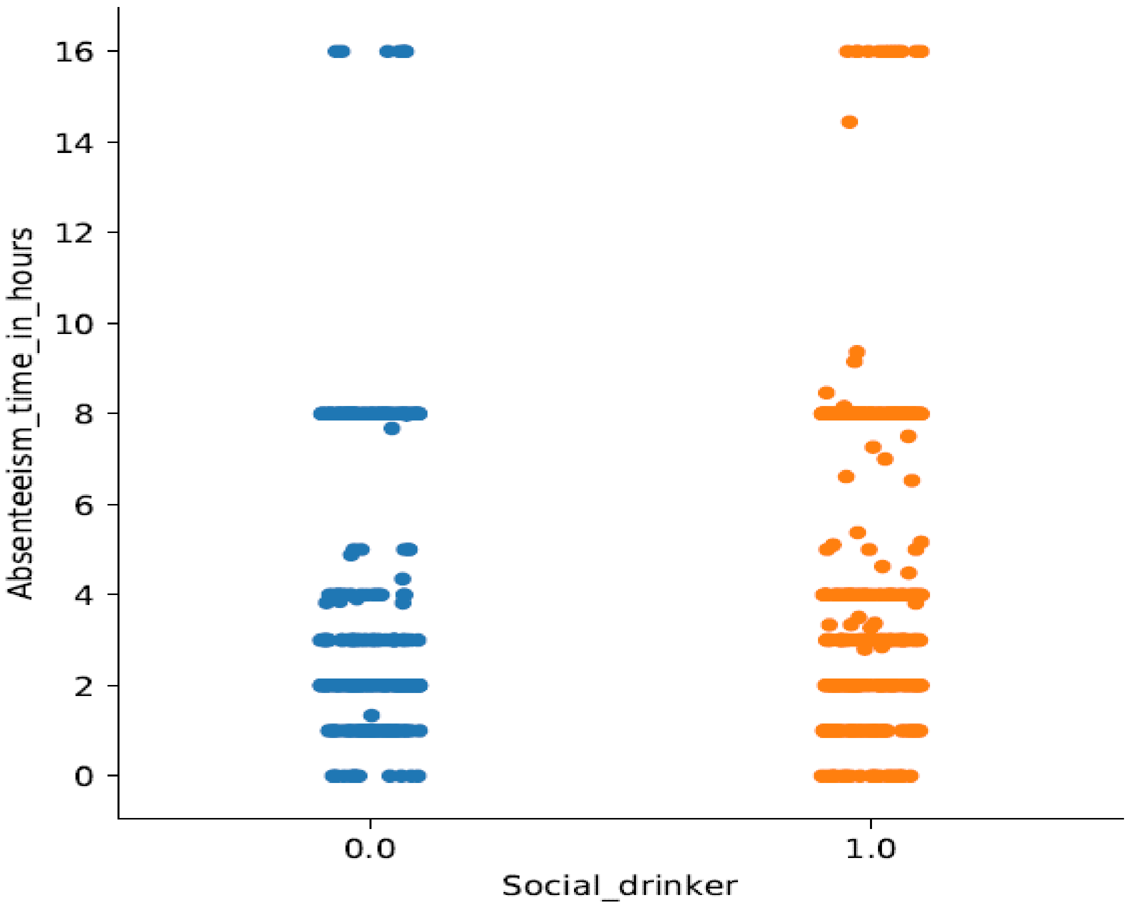
16
14
12
10
8
6
4
2
0
Absenteeism_time_in_hours
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0
Month_of_absence



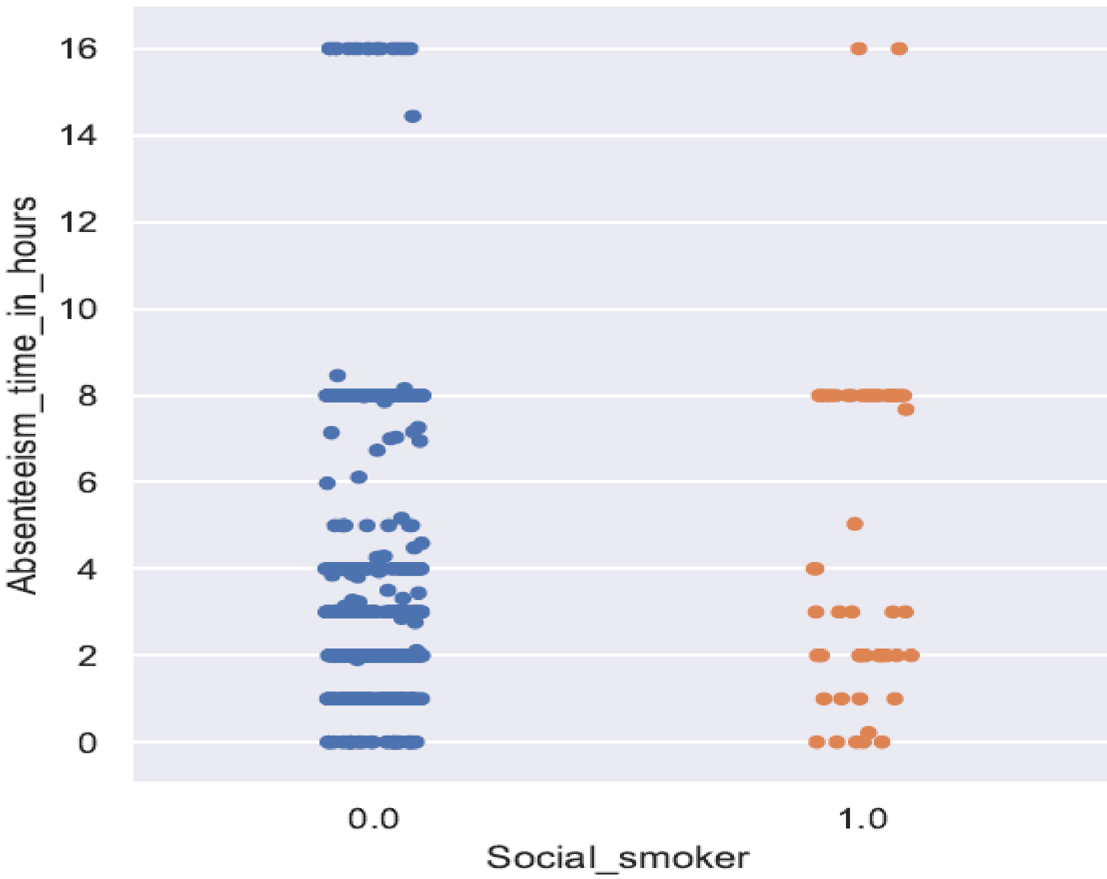
16
14
12
10
8
6
4
2
0
Absenteeism_time_in_hours
1.0 2.0 3.0 4.0
Seasons


16
14
12
10
8
6
4
2
0
Absenteeism_time_in_hours
0.0
1.0
2.0
4.0
5.0
8.0
Pet

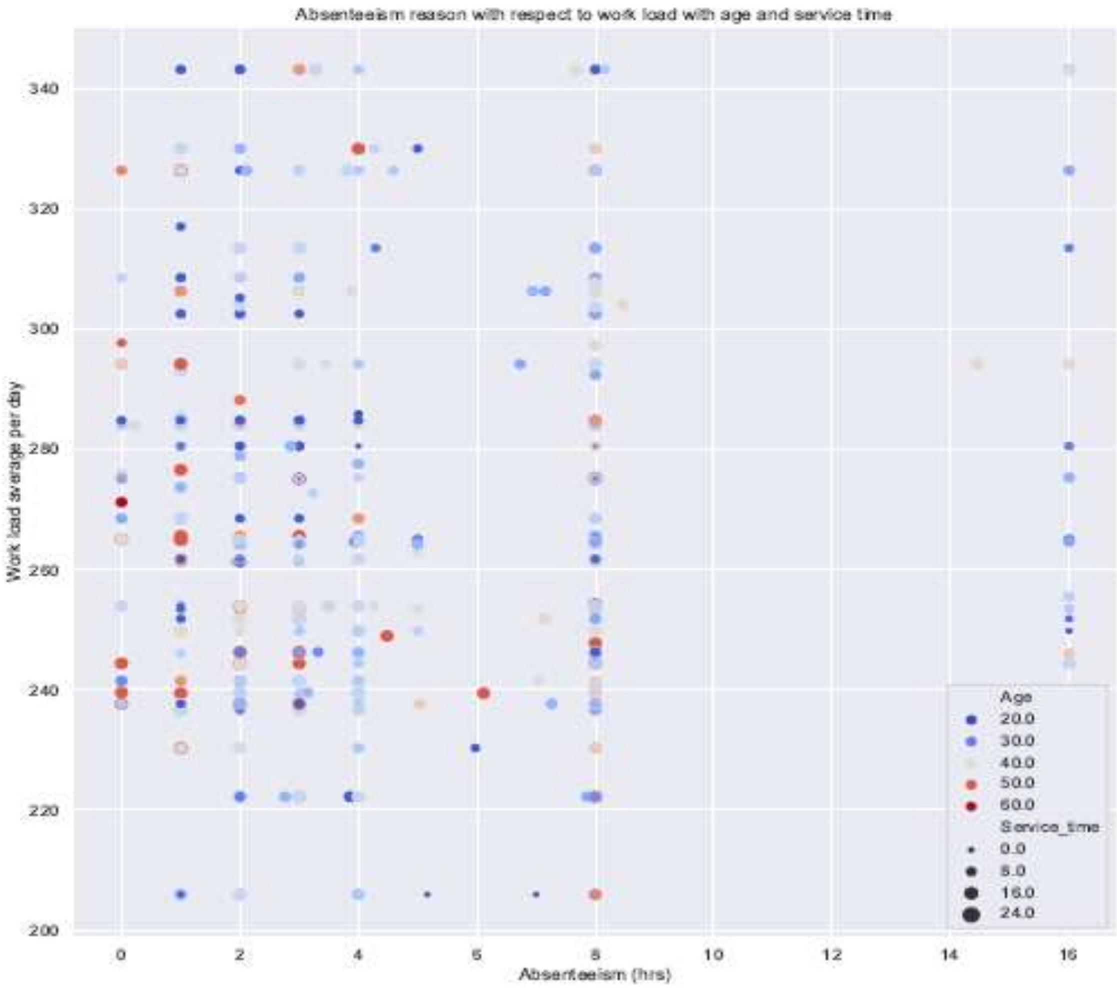

16
14
12
10
8
6
4
2
0
Absenteeism_time_in_hours
0.0
1.0
2.0
3.0
4.0
Son



Absenteeism_time_in_hours
16
14
12
10
8
6
4
2
0
0.0
1.0
Social_drinker



Absenteeism_time_in_hours
16
14
12
10
8
6
4
2
0
0.0
1.0
Social_smoker



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

As we don't have the data for 2011, we shall use the given data to calculate the loss in 2011 assuming the trend remains same. For this, we shall calculate the loss of work in time (hrs) which would be the total sum of absenteeism time in hours for each month respectively. We shall also calculate the loss in work load. Assuming work load average per day is the target workload for that day we shall calculate its loss due to absenteeism time in hours by the formula given below.

$$\textbf{Work loss} = \frac{\textbf{workload per day* Absenteeism\_in\_hours}}{\textbf{24}}$$

By using above formula we can calculate monthly work loss in time (hrs.). Here, in below index we have calculated the work loss monthly by assuming that the trend will be same for the year 2011.

Month_of_absence	Absenteeism time/month(hrs.)	Work loss per month
1.0	171.685945	2258.468119
2.0	279.364511	3167.553023
3.0	443.587342	5202.503497
4.0	239.915715	2732.079510
5.0	259.744293	2651.602016
6.0	240.571077	2710.274163
7.0	370.688157	3914.799765
8.0	237.163987	2332.985366
9.0	186.884730	2118.879953
10.0	281.000014	3152.253693
11.0	245.691372	2915.497810
12.0	199.865272	2159.711668

4. Coding

In this section we are attaching the coding of R and Python which we developed for our model.

4.1 R Coding

```
#Remove the existing Environment-
rm(list=ls())

#Set working directory-
setwd("D:/R-programming/1.Project- Employee Absenteeism")

#Check the working directory-
getwd()

#Load the Data-
library(xlsx)    #Library for read excel file
data= read.xlsx("Absenteeism_at_work_Project.xls",sheetIndex = 1,header=TRUE)

#---------------------------Exploratory Data Analysis-----------------------------------

head(data)
class(data)
dim(data)
str(data)
names(data)
summary(data)

#from the data summary we can see variable ID, which is not useful variable
#in modeling further, so here removing variable ID from dataset.
data= subset(data,select=-c(ID))

#since month variable can contain maximum 12 values, so here replace 0 with NA-
data$Month.of.absence[data$Month.of.absence %in% 0]= NA

#Dividing Work_load_Average/day_ by 1000 (As told by the support team)
data$Work.load.Average.day.= data$Work.load.Average.day./1000
```

```
#Extract column names of numeric and categorical variables-
cnames = c('Distance.from.Residence.to.Work', 'Service.time', 'Age',
                    'Work.load.Average.day.', 'Transportation.expense',
                    'Hit.target', 'Weight', 'Height',
                    'Body.mass.index', 'Absenteeism.time.in.hours')

cat_cnames = c('Reason.for.absence','Month.of.absence','Day.of.the.week',
                    'Seasons','Disciplinary.failure', 'Education', 'Social.drinker',
                    'Social.smoker', 'Son', 'Pet')


#============================Data Pre-processing========================================

#---------------------------Missing Value Analysis-------------------------------------
#Total missing values in dataset-
sum(is.na(data))

#check missing values in target variable-
sum(is.na(data$Absenteeism.time.in.hours))
#remove the observations in which target varia le have missing value-
data= data[(!data$Absenteeism.time.in.hours %in% NA),]

#remaining missing values in data-
sum(is.na(data))

#Calculate missing values in dataset-
missing_value= data.frame(apply(data,2,function(x)sum(is.na(x))))
missing_value$variable= row.names(missing_value)
row.names(missing_value)=NULL
names(missing_value)[1]="missing_precentage"
missing_value= missing_value[,c(2,1)]
missing_value$missing_precentage= (missing_value$missing_precentage/nrow(data))*100
missing_value= missing_value[order(-missing_value$missing_precentage),]
write.csv(missing_value,"missing_value.csv", row.names = FALSE)

#Missing value imputation for categorical variables-
#Mode method-
mode=function(v){
  uniqv=unique(v)
  uniqv[which.max(tabulate(match(v,uniqv)))]
}

for(i in cat_cnames){
  print(i)
  data[,i][is.na(data[,i])] = mode(data[,i])
}

#Now check the remaining missing values-
sum(is.na(data))
#missing values= 72

#Missing value imputation for numeric variables-
#Lets take one sample data for referance-

data$Body.mass.index[11]

#Actual value= 23
#Mean= 26.68
#Median= 25
#KNN= 23

#Mean method-
#repalce the sample data with NA to check the accuracy to select the final method-
data$Body.mass.index[11]=NA
data$Body.mass.index[is.na(data$Body.mass.index)] = mean(data$Body.mass.index,
                                                        na.rm=TRUE)

data$Body.mass.index[11]
#Mean = 26.68
```

```
#Median Method- #reload the data first
data$Body.mass.index[11]=NA
data$Body.mass.index[is.na(data$Body.mass.index)]= median(data$Body.mass.index,na.rm=TR
data$Body.mass.index[11]
#Median= 25

#KNN Imputation- #reload the data first
data$Body.mass.index[11]=NA
library(DMwR)  #Library for KNN
data= knnImputation(data,k=3)
data$Body.mass.index[11]
#KNN=23

#-> From all above method we have seen that KNN is more accurate then all other method,
#So we will take KNN Imputation for missing value imputation.
sum(is.na(data))

#-> Now, here data is free from missing values.

#-----------------------------Outlier Analysis-----------------------------------------
#save data for reference-
df= data
data=df

#Create box-plot for outlier analysis-
library(ggplot2)    #Library for visualization
for(i in 1:length(cnames))
  assign(paste0("AB",i),ggplot(aes_string(y=(cnames[i]),x="Absenteeism.time.in.hours"),
                              d=subset(data))
  +geom_boxplot(outlier.colour = "Red",outlier.shape = 18,outlier.size = 2,
                fill="skyblue4")+theme_gray()
  +stat_boxplot(geom = "errorbar", width=0.5)
  +labs(y=cnames[i],x="Absenteeism time(Hours)")
  +ggtitle("Box Plot of Absenteeism for",cnames[i]))
```

```
gridExtra::grid.arrange(AB1,AB2,ncol=2)
gridExtra::grid.arrange(AB3,AB4,ncol=2)
gridExtra::grid.arrange(AB5,AB6,ncol=2)
gridExtra::grid.arrange(AB7,AB8,ncol=2)
gridExtra::grid.arrange(AB9,AB10,ncol=2)

#Remove outliers from dataset-
for(i in cnames){
  print(i)
  outlier= data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
  print(length(outlier))
  data=data[which(!data[,i] %in% outlier),]
}

#Replace outliers with NA and impute using KNN method-
for(i in cnames){
  print(i)
  outlier= data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
  print(length(outlier))
  data[,i][data[,i] %in% outlier]=NA
}
sum(is.na(data))

#KNN-
data= knnImputation(data,k=3)
sum(is.na(data))
```

```r
#--------------------------------Feature Selection------------------------------------

df=data
data=df

#Correlation Analysis for continuous variables-
library(corrgram)    #Library for correlation plot

corrgram(data[,cnames],order=FALSE,upper.panel = panel.pie,
           text.panel = panel.txt,font.labels =1,
           main="Correlation plot for Absenteeism")

#Correlated variable= weight & Body mass index.

#Anova Test for categorical variable-

for(i in cat_cnames){
  print(i)
  Anova_result= summary(aov(formula = Absenteeism.time.in.hours~data[,i],data))
  print(Anova_result)
}

#redudant categorical variables- Pet,Social.smoker,Education,Seasons,Month.of.absence

#Dimensionity Reduction
data= subset(data,select=-c(Weight,Pet,Social.smoker,Education,Seasons,Month.of.absence
dim(data)

#---------------------------------Feature Scaling-------------------------------------

df= data
data=df
```

```
#update the continuous variable-
cnames= c('Distance.from.Residence.to.Work', 'Service.time', 'Age',
            'Work.load.Average.day.', 'Transportation.expense',
            'Hit.target','Height', 'Body.mass.index','Absenteeism.time.in.hours')

#update the categorical variable-
cat_cnames = c('Reason.for.absence','Day.of.the.week',
               'Disciplinary.failure','Social.drinker','Son')


#summary of data to check min and max values of numeric variables-
summary(data)

#Skewness of numeric variables-
library(propagate)

for(i in cnames){
  skew = skewness(data[,i])
  print(i)
  print(skew)
}

#log transform
data$Absenteeism.time.in.hours = log1p(data$Absenteeism.time.in.hours)

#Normality check-
hist(data$Absenteeism.time.in.hours,col="Yellow",main="Histogram of Absenteeism ")
hist(data$Distance.from.Residence.to.Work,col="Red",main="Histogram of
     distance between work and residence")
hist(data$Transportation.expense,col="Green",main="Histogram of Transportation Expense")

#From all above histogram plot we can say that data is not uniformaly distributed,
#So best method for scaling will be normalization-

#Normalization-
for(i in cnames){
  if(i !='Absenteeism.time.in.hours'){
    print(i)
  data[,i]= (data[,i]-min(data[,i]))/(max(data[,i]-min(data[,i])))
  print(data[,i])
 }
}

#Summary of data after all preprocessing-
summary(data)

write.csv(data,"Absenteeism_Pre_processed_Data.csv",row.names=FALSE)


#================================Model Devlopment========================================

#Clean the Environment-
library(DataCombine)
rmExcept("data")

#Data Copy for refrance-
df=data
data=df

cat_cnames = c('Reason.for.absence','Day.of.the.week',
               'Disciplinary.failure','Social.drinker','Son')

#create dummy variable for categorical variables-
library(dummies)
data = dummy.data.frame(data, cat_cnames)

dim(data)
```

```
#Divide the data into train and test-
set.seed(6789)
train_index= sample(1:nrow(data),0.8*nrow(data))
train= data[train_index,]
test= data[-train_index,]

#--------------------------Decision Tree for Regression-------------------------------

#Model devlopment for train data-
library(rpart)    #Library for regression model
DT_model= rpart(Absenteeism.time.in.hours~.,train,method="anova")
DT_model

#Prediction for train data-
DT_train=predict(DT_model,train[-51])

#Prediction for test data-
DT_test=predict(DT_model,test[-51])

#Error metrics to calculate the performance of model-
rmse= function(y,y1){
  sqrt(mean(abs(y-y1)^2))
}

#RMSE calculation for train data-
rmse(train[,51],DT_train)
#RMSE_train=  0.4303573

#RMSE calculation for test data-
rmse(test[,51],DT_test)
#RMSE_test= 0.4250704
```

```
#r-square calculation-
#function for r-square-
rsquare=function(y,y1){
  cor(y,y1)^2
}

#r-square calculation for train data-
rsquare(train[,51],DT_train)
#r-square_train= 0.5594418

#r-square calculation for test data-
rsquare(test[,51],DT_test)
#r-square_test= 0.6366973

#Visulaization to check the model performance on test data-
plot(test$Absenteeism.time.in.hours,type="l",lty=1.8,col="Green",main="Decision Tree")
lines(DT_test,type="l",col="Blue")

#Write rule into drive-
write(capture.output(summary(DT_model)),"Decision_Tree_Model.txt")

#---------------------------Random Forest for Regression----------------------------

library(randomForest)  #Library for randomforest machine learning algorithm
library(inTrees)       #Library for intree transformation
RF_model= randomForest(Absenteeism.time.in.hours~.,train,ntree=300,method="anova")

#transform ranfomforest model into treelist-
treelist= RF2List(RF_model)

#Extract rules-
rules= extractRules(treelist,train[-51])
rules[1:5,]
#covert rules into redable format-
readable_rules= presentRules(rules,colnames(train))
```

```
readable_rules[1:5,]
#Get Rule metrics-
rule_metrics= getRuleMetric(rules,train[-51],train$Absenteeism.time.in.hours)
rule_metrics= presentRules(rule_metrics,colnames(train))
rule_metrics[1:10,]
summary(rule_metrics)

#Check model performance on train data-
RF_train= predict(RF_model,train[-51])

#Check model performance on test data-
RF_test= predict(RF_model,test[-51])

#RMSE calculation for train data-
rmse(train[,51],RF_train)
#RMSE_train=  0.2386952

#RMSE calculation for test data-
rmse(test[,51],RF_test)
#RMSE_test=  0.404051

#r-square calculation for train data-
rsquare(train[,51],RF_train)
#r-square= 0.8821434

#r-square calculation for test data-
rsquare(test[,51],RF_test)
#r-square= 0.6826756

#Visulaization to check the model performance on test data-
plot(test$Absenteeism.time.in.hours,type="l",lty=1.8,col="Green",main="Random Forest")
lines(RF_test,type="l",col="Blue")

#write rule into drive-
write(capture.output(summary(rule_metrics)),"Random_Forest_Model.txt")
#--------------------------------Linear Regression--------------------------------------

#recall numeric variables to check the VIF-
numeric_index1= c("Transportation.expense","Distance.from.Residence.to.Work","Service.t
                  "Age","Work.load.Average.day.","Hit.target","Height",
                  "Body.mass.index","Absenteeism.time.in.hours")
numeric_data1= data[,numeric_index1]
cnames1= colnames(numeric_data1)
cnames1

library(usdm)  #Library for VIF(Variance Infleation factor)
vif(numeric_data1)
vifcor(numeric_data1,th=0.7) #VIF calculation for numeric variables

#Linear regression model-
lr_model= lm(Absenteeism.time.in.hours~.,train)
summary(lr_model)

#check model performance on train data-
lr_train= predict(lr_model,train[-51])

#check model performance on test data-
lr_test= predict(lr_model,test[-51])

#RMSE calculation for train data-
rmse(train[,51],lr_train)
#RMSE_train=0.4235679

#RMSE calculation for test data-
rmse(test[,51],lr_test)
#RMSE_test=0.4375786

#r-square calculation for train data-
rsquare(train[,51],lr_train)
#r-square_train=0.5732328
```

```
#r-square calculation for test data-
rsquare(test[,51],lr_test)
#r-square_test=0.6218227

#Visulaization to check the model performance on test data-
plot(test$Absenteeism.time.in.hours,type="l",lty=1.8,col="Green",main="Linear Regressio
lines(lr_test,type="l",col="Blue")

write(capture.output(summary(lr_model)),"Linear_Regression_Model.txt")

#---------------------------------Gradient Boosting-------------------------------------
library(gbm)

#Develop Model
GB_model = gbm(Absenteeism.time.in.hours~., data = train, n.trees = 500, interaction.de

#check model performance on train data-
GB_train = predict(GB_model, train,n.trees = 500)

#check model performance on test data-
GB_test = predict(GB_model, test, n.trees = 500)

#RMSE calculation for train data-
rmse(train[,51],GB_train)
#RMSE_train=0.3577656

#RMSE calculation for test data-
rmse(test[,51],GB_test)
#RMSE_test=0.4150737

#r-square calculation for train data-
rsquare(train[,51],GB_train)
#r-square_train=0.7010977

#r-square calculation for test data-
rsquare(test[,51],GB_test)
#r-square_test=0.6648008

#Visulaization to check the model performance on test data-
plot(test$Absenteeism.time.in.hours,type="l",lty=1.8,col="Green",
     main="Gradient Boosting")
lines(GB_test,type="l",col="Blue")

#####################################Thank You########################################
```

4.2 Python Coding

In [1]:
```
#Load Libraries-
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from fancyimpute import KNN
from random import randrange,uniform
from ggplot import *
from scipy.stats import chi2_contingency
from sklearn.metrics import r2_score
from scipy import stats
```

```
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issu
bdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\ggplot\utils.py:81: FutureWarning: pandas.tslib is deprecated and will be rem
oved in a future version.
You can access Timestamp as pandas.Timestamp
  pd.tslib.Timestamp,
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core.datetools mod
ule is deprecated and will be removed in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
```

In [2]:
```
#Set working directory-
os.chdir("D:/Python-programming/1.Project-Employee Absenteeism")
os.getcwd()
```

Out[2]: 'D:\\Python-programming\\1.Project-Employee Absenteeism'

In [157]:
```
#Load data-
data= pd.read_excel("Absenteeism_at_work_Project.xls")
```

In [184]:
```
data.iloc[0:6:,14:21]
```

Out[184]:

	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
0	1.0	0.0	1.0	90.0	172.0	30.0	4.0
1	1.0	0.0	0.0	98.0	178.0	31.0	0.0
2	1.0	0.0	0.0	89.0	170.0	31.0	2.0
3	1.0	1.0	0.0	68.0	168.0	24.0	4.0
4	1.0	0.0	1.0	90.0	172.0	30.0	2.0
5	1.0	0.0	0.0	89.0	170.0	31.0	NaN

Exploratory Data Analysis

In [53]:
```
data.head()
data.shape
data.dtypes
```

Out[53]:
```
ID                                  int64
Reason for absence                float64
Month of absence                  float64
Day of the week                     int64
Seasons                             int64
Transportation expense            float64
Distance from Residence to Work   float64
Service time                      float64
Age                               float64
Work load Average/day             float64
Hit target                        float64
Disciplinary failure              float64
Education                         float64
Son                               float64
Social drinker                    float64
Social smoker                     float64
Pet                               float64
Weight                            float64
Height                            float64
Body mass index                   float64
Absenteeism time in hours         float64
dtype: object
```

In [54]:
```
# Replacing the white spaces " " in the feature name with "_"
for i in data.columns:
    data = data.rename(index=str, columns={i: i.replace(" ", "_")})
```

```
In [55]: data.nunique()

Out[55]: ID                                 36
         Reason_for_absence                 28
         Month_of_absence                   13
         Day_of_the_week                     5
         Seasons                             4
         Transportation_expense             24
         Distance_from_Residence_to_Work    25
         Service_time                       18
         Age                                22
         Work_load_Average/day_             38
         Hit_target                         13
         Disciplinary_failure                2
         Education                           4
         Son                                 5
         Social_drinker                      2
         Social_smoker                       2
         Pet                                 6
         Weight                             26
         Height                             14
         Body_mass_index                    17
         Absenteeism_time_in_hours          19
         dtype: int64

In [56]: #since month variable can contain maximum 12 values, so here replace 0 with NA-
         data['Month_of_absence']= data['Month_of_absence'].replace(0,np.nan)

         data.nunique()

Out[56]: ID                                 36
         Reason_for_absence                 28
         Month_of_absence                   12
         Day_of_the_week                     5
         Seasons                             4
         Transportation_expense             24
         Distance_from_Residence_to_Work    25
         Service_time                       18
         Age                                22
         Work_load_Average/day_             38
         Hit_target                         13
         Disciplinary_failure                2
         Education                           4
         Son                                 5
         Social_drinker                      2
         Social_smoker                       2
         Pet                                 6
         Weight                             26
         Height                             14
         Body_mass_index                    17
         Absenteeism_time_in_hours          19
         dtype: int64

In [57]: #remove redudant variable (ID variable does not carry meanningful information,so remove it)
         data= data.drop(['ID'],axis=1)

In [58]: # Dividing Work_load_Average/day_ by 1000 (As told by the support team)
         data['Work_load_Average/day_'] = data['Work_load_Average/day_']/1000

In [59]: data.columns

Out[59]: Index(['Reason_for_absence', 'Month_of_absence', 'Day_of_the_week', 'Seasons',
                'Transportation_expense', 'Distance_from_Residence_to_Work',
                'Service_time', 'Age', 'Work_load_Average/day_', 'Hit_target',
                'Disciplinary_failure', 'Education', 'Son', 'Social_drinker',
                'Social_smoker', 'Pet', 'Weight', 'Height', 'Body_mass_index',
                'Absenteeism_time_in_hours'],
               dtype='object')

In [60]: #Extract numeric and categorical varaibles
         cnames=['Transportation_expense', 'Distance_from_Residence_to_Work','Service_time', 'Age',
                 'Work_load_Average/day_', 'Hit_target','Weight', 'Height', 'Body_mass_index',
                 'Absenteeism_time_in_hours']

         cat_cnames= ['Reason_for_absence', 'Month_of_absence', 'Day_of_the_week', 'Seasons',
                      'Disciplinary_failure', 'Education', 'Son', 'Social_drinker','Social_smoker', 'Pet']
```

Data Pre-processing

Missing value Analysis-

In [61]:
```
#Missing values in each variable-
data.isnull().sum()
```

Out[61]:
```
Reason_for_absence                  3
Month_of_absence                    4
Day_of_the_week                     0
Seasons                             0
Transportation_expense              7
Distance_from_Residence_to_Work     3
Service_time                        3
Age                                 3
Work_load_Average/day_             10
Hit_target                          6
Disciplinary_failure                6
Education                          10
Son                                 6
Social_drinker                      3
Social_smoker                       4
Pet                                 2
Weight                              1
Height                             14
Body_mass_index                    31
Absenteeism_time_in_hours          22
dtype: int64
```

In [62]:
```
# Droping observation in which "Absenterism time in hours" has missing value-
data = data.drop(data[data['Absenteeism_time_in_hours'].isnull()].index, axis=0)
print(data.shape)
```

```
(718, 20)
```

In [63]:
```
##Missing value analysis-

#Creat dataframe with missing value present in each variable-
missing_value= pd.DataFrame(data.isnull().sum()).reset_index()

#Rename variable-
missing_value= missing_value.rename(columns={'index':'variable',0:'missing_precentage'})

#Missing value precentage-
missing_value['missing_precentage']= (missing_value['missing_precentage']/len(data))*100

#Sorting missing value-
missing_value= missing_value.sort_values('missing_precentage',ascending=False).reset_index(drop=True)

#Write missing data into drive-
missing_value.to_csv("missing_value.csv",index=False)
```

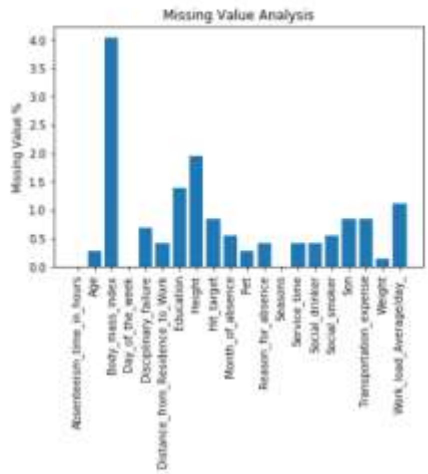
In [64]:
```
missing_value
```

Out[64]:

	variable	missing_precentage
0	Body_mass_index	4.038997
1	Height	1.949861
2	Education	1.392758
3	Work_load_Average/day_	1.114206
4	Transportation_expense	0.835655
5	Son	0.835655
6	Hit_target	0.835655
7	Disciplinary_failure	0.696379
8	Social_smoker	0.557103
9	Month_of_absence	0.557103
10	Social_drinker	0.417827
11	Reason_for_absence	0.417827
12	Service_time	0.417827
13	Distance_from_Residence_to_Work	0.417827
14	Age	0.278552
15	Pet	0.278552
16	Weight	0.139276
17	Seasons	0.000000
18	Day_of_the_week	0.000000
19	Absenteeism_time_in_hours	0.000000

```
In [167]: #Missing value analysis by visualization-
plt.bar(missing_value['variable'],missing_value['missing_precentage'])
plt.ylabel('Missing Value %')
plt.title('Missing Value Analysis')
plt.savefig('missing_value.pdf')
plt.xticks(rotation=90)
```

```
Out[167]: ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
 <a list of 20 Text xticklabel objects>)
```




```
In [66]: #Missing value imputation for categorical varibles-
for i in cat_cnames:
    print(i)
    data[i] = data[i].fillna(data[i].mode()[0])
    print(data[i])
```

```
Reason_for_absence
0      26.0
1       0.0
2      23.0
3       7.0
4      23.0
6      22.0
7      23.0
8      19.0
9      22.0
10      1.0
11      1.0
12     11.0
13     11.0
14     23.0
16     23.0
17     21.0
18     11.0
19     23.0
21     11.0
```

```
In [67]: ##Missing value imputation for numeric variables-

#Lets take one sample data for referance-
data['Body_mass_index'][29]
#Actual Value= 29.0
#Mean= 26.703488372093023
#Median= 25.0
#KNN= 29.816416946640054

data['Body_mass_index'][29]=np.nan     #Replace sample data with NA for check the accuracy of imputation method.
```

```
In [24]: #Mean  method-
data['Body_mass_index']= data['Body_mass_index'].fillna(data['Body_mass_index'].mean())

data['Body_mass_index'][29]
#Mean = 26.703488372093023
```

```
Out[24]: 26.703488372093023
```

```
In [44]: #Median method-
data['Body_mass_index'][29]=np.nan
data['Body_mass_index']=data['Body_mass_index'].fillna(data['Body_mass_index'].median())

data['Body_mass_index'][29]
#Median=25
```

```
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
Out[44]: 25.0
```

```
In [68]: #KNN method-
data['Body_mass_index'][29]=np.nan
data= pd.DataFrame(KNN(k=3).fit_transform(data),columns=data.columns)

data['Body_mass_index'][29]
#KNN=29.816416946640054
```

```
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

Outlier Analysis-

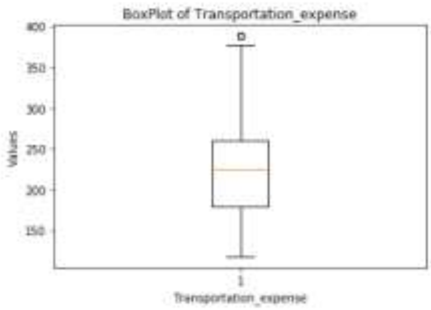
In [70]:
```
#copy of data-

#df= data.copy()
data= df.copy()
```

In [71]:
```
cnames
```

Out[71]:
```
['Transportation_expense',
 'Distance_from_Residence_to_Work',
 'Service_time',
 'Age',
 'Work_load_Average/day_',
 'Hit_target',
 'Weight',
 'Height',
 'Body_mass_index',
 'Absenteeism_time_in_hours']
```

In [72]:
```
##Plot boxplot to visularie outliers-
for i in cnames:
    print(i)
    plt.boxplot(data[i])
    plt.xlabel(i)
    plt.ylabel('Values')
    plt.title("BoxPlot of "+i)
    plt.show()
```

Transportation_expense



In [73]:
```
##Calculate iqr, lower fence and upper fence-
for i in cnames:
    print(i)
    q75,q25= np.percentile(data.loc[:,i],[75,25])
    iqr= q75-q25
    minimum= q25-(iqr*1.5)
    maximum= q75+(iqr*1.5)
    print(minimum)
    print(maximum)
    print(iqr)

#Replace outliers with NA-
    data.loc[data[i]< minimum,i] = np.nan
    data.loc[data[i]> maximum,i] = np.nan
```

Feature Selection-

In [76]:
```
df= data.copy()
data= df.copy()
```

In [77]:
```
cat_cnames
```

Out[77]:
```
['Reason_for_absence',
 'Month_of_absence',
 'Day_of_the_week',
 'Seasons',
 'Disciplinary_failure',
 'Education',
 'Son',
 'Social_drinker',
 'Social_smoker',
 'Pet']
```

In [78]:
```
##Correlation analysis for numeric variables-

#extract only numeric variables in dataframe for correlation-
df_corr= data.loc[:,cnames]

#Generate correlation matrix-
corr_matrix= df_corr.corr()
```

In [79]:

```
corr_matrix
```

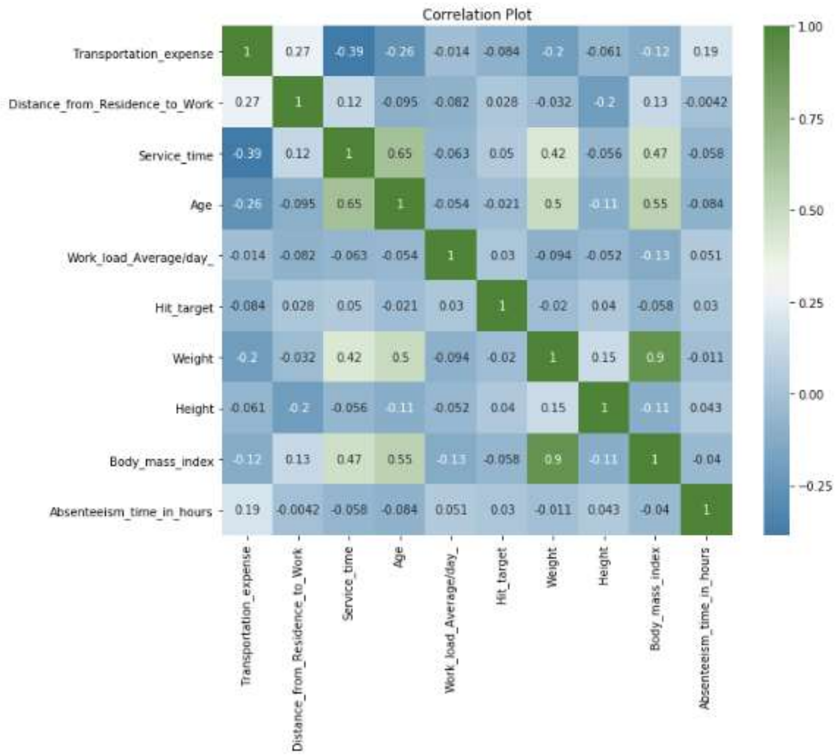
Out[79]:

	Transportation_expense	Distance_from_Residence_to_Work	Service_time	Age	Work_load_Average/day_	Hit_target	V
Transportation_expense	1.000000	0.266940	-0.385904	-0.261317	-0.014478	-0.084040	-0.1
Distance_from_Residence_to_Work	0.266940	1.000000	0.121563	-0.095117	-0.081580	0.028050	-0.0
Service_time	-0.385904	0.121563	1.000000	0.646516	-0.063252	0.049683	0.4
Age	-0.261317	-0.095117	0.646516	1.000000	-0.054495	-0.020749	0.4
Work_load_Average/day_	-0.014478	-0.081580	-0.063252	-0.054495	1.000000	0.030202	-0.0
Hit_target	-0.084040	0.028050	0.049683	-0.020749	0.030202	1.000000	-0.0
Weight	-0.198477	-0.031948	0.423628	0.498177	-0.093550	-0.019882	1.0
Height	-0.061436	-0.196777	-0.056136	-0.105807	-0.051549	0.040103	0.1
Body_mass_index	-0.120465	0.131062	0.471658	0.553311	-0.125038	-0.058425	0.9
Absenteeism_time_in_hours	0.192174	-0.004184	-0.058076	-0.083672	0.050705	0.029598	-0.0

In [173]:

```
#Corrleation plot-

#Set height and width of page-
f,ax= plt.subplots(figsize=(10,8))

#Plot-
sns.heatmap(corr_matrix,mask=np.zeros_like(corr_matrix,dtype=np.bool),cmap=sns.diverging_palette(240,120,as_cmap=True),
            square=True,ax=ax,annot=True)
plt.title("Correlation Plot")
```

Out[173]: `Text(0.5,1,'Correlation Plot')`



In [81]:
```
##Anova test for categorical predictor and numeric target variable-

import statsmodels.api as sm
from statsmodels.formula.api import ols

label = 'Absenteeism_time_in_hours'
for  i in cat_cnames:
    frame = label + ' ~ ' + i
    model = ols(frame,data=data).fit()
    anova = sm.stats.anova_lm(model, typ=2)
    print(anova)
```

```
                       sum_sq     df          F    PR(>F)
Reason_for_absence   204.282325    1.0  17.915574  0.000026
Residual            8164.189531  716.0        NaN       NaN
                      sum_sq     df         F    PR(>F)
Month_of_absence     0.292084    1.0  0.024991  0.874433
Residual          8368.179772  716.0       NaN       NaN
                     sum_sq     df         F    PR(>F)
Day_of_the_week   54.172342    1.0  4.665143  0.031112
Residual        8314.299514  716.0       NaN       NaN
              sum_sq     df         F    PR(>F)
Seasons    29.539590    1.0  2.536337  0.111694
Residual  8338.932266  716.0       NaN       NaN
                          sum_sq     df          F        PR(>F)
Disciplinary_failure   668.371022    1.0  62.149011  1.188029e-14
Residual              7700.100834  716.0        NaN           NaN
               sum_sq     df         F    PR(>F)
Education    3.693486    1.0  0.316151  0.574106
Residual  8364.778370  716.0       NaN       NaN
              sum_sq     df          F    PR(>F)
Son       218.839218    1.0  19.226496  0.000013
Residual  8149.632638  716.0        NaN       NaN
                    sum_sq     df         F    PR(>F)
Social_drinker   70.901908    1.0  6.118149  0.013611
Residual       8297.569948  716.0       NaN       NaN
                   sum_sq     df         F    PR(>F)
Social_smoker   19.623439    1.0  1.682913  0.194956
Residual      8348.848417  716.0       NaN       NaN
              sum_sq     df         F    PR(>F)
Pet         4.589346    1.0  0.392876  0.530991
Residual  8363.882510  716.0       NaN       NaN
```

In [82]:
```
##Dimensionality reduction (Droping redundant variable) on behalf of

data = data.drop(["Weight","Pet","Social_smoker","Education","Seasons","Month_of_absence"],axis=1)
```

In [83]:
```
data.shape
```

Out[83]:
```
(718, 14)
```

Feature Scaling-

In [84]:
```
#df= data.copy()
data= df.copy()
```

In [85]:
```
#updating continuous variables-
cnames= ['Transportation_expense', 'Distance_from_Residence_to_Work','Service_time', 'Age',
         'Work_load_Average/day_', 'Hit_target','Height', 'Body_mass_index',
        'Absenteeism_time_in_hours']

#updating categorical variables-
cat_cnames= ['Reason_for_absence','Day_of_the_week','Disciplinary_failure','Son', 'Social_drinker']
```

In [86]:
```
#Skewness of numeric variables-

for i in cnames:
    skewness = stats.describe(data.loc[:,i])
    print("statistical properities of :"+str(i))
    print(skewness)
    print("*************************************************************")
```

```
statistical properities of :Transportation_expense
DescribeResult(nobs=718, minmax=(118.0, 378.0), mean=219.96885818314786, variance=4267.295223992289, skewness=0.362760680839818
7, kurtosis=-0.33354147214041374)
************************************************************
statistical properities of :Distance_from_Residence_to_Work
DescribeResult(nobs=718, minmax=(5.0, 52.0), mean=29.54874651572311, variance=218.28702076207122, skewness=0.3205003614151142,
kurtosis=-1.238708485370826)
************************************************************
statistical properities of :Service_time
DescribeResult(nobs=718, minmax=(1.0, 24.0), mean=12.473537606081669, variance=17.21756933851194, skewness=-0.3395611779215922
4, kurtosis=-0.17000453283733075)
************************************************************
statistical properities of :Age
DescribeResult(nobs=718, minmax=(27.0, 53.0), mean=36.159532159681824, variance=37.208692540082055, skewness=0.482362503527695,
kurtosis=-0.25572276239363756)
************************************************************
statistical properities of :Work_load_Average/day_
DescribeResult(nobs=718, minmax=(205.917, 343.253), mean=267.2818537094081, variance=1044.2774030970302, skewness=0.55254486867
48119, kurtosis=-0.22139504428714307)
************************************************************
statistical properities of :Hit_target
DescribeResult(nobs=718, minmax=(87.0, 100.0), mean=94.94744096662974, variance=9.503157769997522, skewness=-0.452957743293498
7, kurtosis=-0.3902600452996592)
************************************************************
statistical properities of :Height
DescribeResult(nobs=718, minmax=(165.0, 175.00000871636007), mean=170.26298572455133, variance=3.759182898285231, skewness=-0.5
097400994461557, kurtosis=0.7250946645345899)
************************************************************
statistical properities of :Body_mass_index
DescribeResult(nobs=718, minmax=(19.0, 38.0), mean=26.703399579177084, variance=18.36098460542446, skewness=0.2790603274810965
5, kurtosis=-0.3375966638205714)
************************************************************
statistical properities of :Absenteeism_time_in_hours
DescribeResult(nobs=718, minmax=(0.0, 16.0), mean=4.395769382082745, variance=11.671508864660172, skewness=1.1166027772140417,
kurtosis=1.3378043972288554)
************************************************************
```

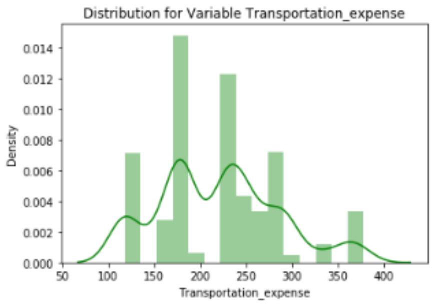
In [87]:

```
#since skewness of target variable is high, apply log transform to reduce the skewness-
data['Absenteeism_time_in_hours'] = np.log1p(data['Absenteeism_time_in_hours'])
```

In [88]:

```
#Normality check to check data is uniformly distributed or not-

for i in cnames:
    print(i)
    sns.distplot(data[i],bins='auto',color='green')
    plt.title("Distribution for Variable "+i)
    plt.ylabel("Density")
    plt.show()    #From below lot its showing data is not uniformaly distributed, so we will do normalization for dataset.
```

```
Transportation_expense
```




```
Distance_from_Residence_to_Work
```

In [89]:

```
#Normalization-
for i in cnames:
    if i== 'Absenteeism_time_in_hours':
        continue
    print(i)
    data[i]= (data[i]-min(data[i]))/(max(data[i])-min(data[i]))
    print(data[i])
```

In [91]:
```
data.describe()
```

Out[91]:

	Reason_for_absence	Day_of_the_week	Transportation_expense	Distance_from_Residence_to_Work	Service_time	Age	Work_load_Average/day_
count	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000
mean	19.409471	3.899721	0.392188	0.522314	0.498849	0.352290	0.446823
std	8.279768	1.419519	0.251248	0.314352	0.180409	0.234611	0.235301
min	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	13.000000	3.000000	0.234615	0.234043	0.347826	0.153846	0.280116
50%	23.000000	4.000000	0.411538	0.446809	0.521739	0.384615	0.424739
75%	26.000000	5.000000	0.546154	0.957447	0.652174	0.500000	0.574766
max	28.000000	6.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [92]:
```
#write Normalized data into drive-
data.to_csv("Absenteeism_Pre_processed_Data.csv",index=False)
```

Machine Learning Model Devlopment-

Train-Test Split-

In [93]:
```
#Import Libraries-
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
from sklearn.metrics import mean_squared_error
```

```
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprecate
d in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also no
te that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [134]:
```
#make copy of data for reference-
#df=data
data=df.copy()
```

In [135]:
```
#Convert categorical to dummy variable-
data = pd.get_dummies(data,columns=cat_cnames)

#split data for predictor and target seperatly-
X= data.drop(['Absenteeism_time_in_hours'],axis=1)
y= data['Absenteeism_time_in_hours']

#Divide data into train and test-
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.20,random_state=220192)
```

Decision Tree-

In [96]:
```
#import libraries-
from sklearn.tree import DecisionTreeRegressor

#Decision tree for regression-
DT_model= DecisionTreeRegressor().fit(X_train,y_train)

#model prediction on train data-
DT_train= DT_model.predict(X_train)

#model prediction on test data-
DT_test= DT_model.predict(X_test)

#RMSE for train data-
RMSE_train=np.sqrt(mean_squared_error(y_train,DT_train))

#RMSE for test data-
RMSE_test=np.sqrt(mean_squared_error(y_test, DT_test))

#r2 value for train data-
r2_train= r2_score(y_train,DT_train)

#r2 value for test data-
r2_test=r2_score(y_test,DT_test)

print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
Root Mean Square Rate for train data=0.0788495202003774
Root Mean Square Rate for test data=0.5131222099616964
R^2_score for train data=0.985408233172478
R^2_score for test data=0.42560025979778693
```

Random Forest

In [141]:
```
#import Libraris-
from sklearn.ensemble import RandomForestRegressor

#Random Forest for regression-
RF_model= RandomForestRegressor(n_estimators=300).fit(X_train,y_train)

#model prediction on train data-
RF_train= RF_model.predict(X_train)

#model prediction on test data-
RF_test= RF_model.predict(X_test)

#RMSE for train data-
RMSE_train=np.sqrt(mean_squared_error(y_train, RF_train))

#RMSE for test data-
RMSE_test=np.sqrt(mean_squared_error(y_test, RF_test))

#r2 value for train data-
r2_train= r2_score(y_train,RF_train)

#r2 value for test data-
r2_test=r2_score(y_test,RF_test)

print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
Root Mean Square Rate for train data=0.18029356384203085
Root Mean Square Rate for test data=0.3836894649958583
R^2_score for train data=0.9237095427126606
R^2_score for test data=0.6788319325296582
```

Linear Regression

In [110]:
```
#import Libraries-
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

#Linear Regression model for regression-
LR_model= sm.OLS(y_train,X_train).fit()
print(LR_model.summary())
```

```
                            OLS Regression Results
==================================================================================
Dep. Variable:     Absenteeism_time_in_hours   R-squared:                   0.588
Model:                                   OLS   Adj. R-squared:              0.553
Method:                        Least Squares   F-statistic:                 16.73
Date:                       Mon, 25 Mar 2019   Prob (F-statistic):       3.38e-75
Time:                               17:48:25   Log-Likelihood:            -315.27
No. Observations:                        574   AIC:                         722.5
Df Residuals:                            528   BIC:                         922.8
Df Model:                                 45
Covariance Type:                   nonrobust
=================================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
Transportation_expense          0.3280      0.116      2.833      0.005       0.101       0.555
Distance_from_Residence_to_Work -0.1533     0.099     -1.553      0.121      -0.347       0.041
Service_time                    0.2733      0.187      1.463      0.144      -0.094       0.641
Age                            -0.2745      0.133     -2.069      0.039      -0.535      -0.014
Work_load_Average/day_         -0.0638      0.083     -0.772      0.440      -0.226       0.098
Hit_target                     -0.0599      0.081     -0.738      0.461      -0.219       0.100
Height                         -0.1633      0.115     -1.418      0.157      -0.390       0.063
Body_mass_index                 0.0746      0.116      0.642      0.521      -0.154       0.303
Reason_for_absence_0.0         -1.4031      0.307     -4.565      0.000      -2.007      -0.799
Reason_for_absence_1.0          0.4022      0.127      3.173      0.002       0.153       0.651
Reason_for_absence_2.0         -0.2112      0.432     -0.489      0.625      -1.060       0.638
Reason_for_absence_3.0          0.5805      0.430      1.350      0.178      -0.264       1.425
Reason_for_absence_4.0         -0.2637      0.309     -0.852      0.395      -0.872       0.344
Reason_for_absence_5.0          0.3412      0.250      1.362      0.174      -0.151       0.833
Reason_for_absence_6.0          0.2934      0.196      1.494      0.136      -0.092       0.679
Reason_for_absence_7.0         -0.0072      0.127     -0.057      0.955      -0.257       0.242
Reason_for_absence_8.0          0.1431      0.179      0.800      0.424      -0.208       0.494
Reason_for_absence_9.0          0.8711      0.251      3.466      0.001       0.377       1.365
Reason_for_absence_10.0         0.3507      0.111      3.150      0.002       0.132       0.569
Reason_for_absence_11.0         0.0463      0.113      0.409      0.683      -0.176       0.269
```

```
Reason_for_absence_12.0         -0.2889      0.167     -1.733      0.084      -0.616       0.039
Reason_for_absence_13.0          0.1064      0.077      1.383      0.167      -0.045       0.258
Reason_for_absence_14.0         -0.0166      0.120     -0.138      0.890      -0.253       0.220
Reason_for_absence_15.0          0.3380      0.306      1.103      0.271      -0.264       0.940
Reason_for_absence_16.0         -0.6387      0.252     -2.534      0.012      -1.134      -0.144
Reason_for_absence_17.0          0.5909      0.432      1.368      0.172      -0.257       1.439
Reason_for_absence_18.0          0.2914      0.112      2.601      0.010       0.071       0.512
Reason_for_absence_19.0          0.2195      0.088      2.496      0.013       0.047       0.392
Reason_for_absence_21.0          0.0883      0.179      0.493      0.622      -0.264       0.440
Reason_for_absence_22.0          0.3907      0.096      4.090      0.000       0.203       0.578
Reason_for_absence_23.0         -0.4466      0.059     -7.523      0.000      -0.563      -0.330
Reason_for_absence_24.0          0.3209      0.308      1.042      0.298      -0.284       0.926
Reason_for_absence_25.0         -0.2778      0.103     -2.698      0.007      -0.480      -0.076
Reason_for_absence_26.0          0.2831      0.098      2.892      0.004       0.091       0.475
Reason_for_absence_27.0         -0.5181      0.079     -6.600      0.000      -0.672      -0.364
Reason_for_absence_28.0         -0.4826      0.065     -7.401      0.000      -0.611      -0.354
Day_of_the_week_2.0              0.2224      0.046      4.830      0.000       0.132       0.313
Day_of_the_week_3.0              0.2259      0.048      4.669      0.000       0.131       0.321
Day_of_the_week_4.0              0.2077      0.047      4.427      0.000       0.116       0.300
Day_of_the_week_5.0              0.2172      0.053      4.104      0.000       0.113       0.321
Day_of_the_week_6.0              0.2299      0.051      4.465      0.000       0.129       0.331
Disciplinary_failure_0.0         0.7238      0.127      5.677      0.000       0.473       0.974
Disciplinary_failure_1.0         0.3792      0.225      1.688      0.092      -0.062       0.821
Son_0.0                          0.2654      0.060      4.418      0.000       0.147       0.383
Son_1.0                          0.1471      0.055      2.682      0.008       0.039       0.255
Son_2.0                          0.2558      0.061      4.170      0.000       0.135       0.376
Son_3.0                         -0.0204      0.150     -0.136      0.892      -0.315       0.274
Son_4.0                          0.4550      0.096      4.741      0.000       0.266       0.644
Social_drinker_0.0               0.4997      0.073      6.848      0.000       0.356       0.643
Social_drinker_1.0               0.6033      0.087      6.896      0.000       0.431       0.775
==============================================================================
Omnibus:                       20.476   Durbin-Watson:                   1.985
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               45.586
Skew:                           0.120   Prob(JB):                     1.26e-10
Kurtosis:                       4.360   Cond. No.                     1.24e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.45e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [111]:

```
#model prediction on train data-
LR_train= LR_model.predict(X_train)

#model prediction on test data-
LR_test= LR_model.predict(X_test)

#RMSE for train data-
RMSE_train=np.sqrt(mean_squared_error(y_train,LR_train))

#RMSE for test data-
RMSE_test=np.sqrt(mean_squared_error(y_test,LR_test))

#r2 for train data-
r2_train=r2_score(y_train,LR_train)

#r2 for test data-
r2_test=r2_score(y_test,LR_test)

print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
Root Mean Square Rate for train data=0.4190847663653237
Root Mean Square Rate for test data=0.4201877072548572
R^2_score for train data=0.587794748509313б
R^2_score for test data=0.6148239329765861
```

Gradient Boosting

In [112]:

```
#import libraries-
from sklearn.ensemble import GradientBoostingRegressor

#Gradient Boosting for regression-
GB_model = GradientBoostingRegressor().fit(X_train, y_train)

#model prediction on train data-
GB_train= GB_model.predict(X_train)

#model prediction on test data-
GB_test= GB_model.predict(X_test)

#RMSE for train data-
RMSE_train=np.sqrt(mean_squared_error(y_train,GB_train))

#RMSE for test data-
RMSE_test=np.sqrt(mean_squared_error(y_test, GB_test))

#r2 value for train data-
r2_train= r2_score(y_train,GB_train)

#r2 value for test data-
r2_test=r2_score(y_test,GB_test)

print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
Root Mean Square Rate for train data=0.35842713783278973
Root Mean Square Rate for test data=0.3990074797536028
R^2_score for train data=0.6984831830720599
R^2_score for test data=0.6526760889868312
```

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

```
In [120]: #save data copy for refrence-
          #df2=df1.copy()
          df1=df2.copy()
```

```
In [121]: # Combining similar groups in Reason for absence for batter visualization-
          df1['Reason_for_absence'] = df1['Reason_for_absence'].replace({0:1,1:1,2:1,3:1,4:1,5:1,6:1,7:1,8:1,9:1,10:1,11:1,12:1,
                                                                          13:1,14:1,15:1,16:1,17:1,18:1,19:1,20:1,21:2,22:2,23:3,
                                                                          24:3,25:4,26:5,27:2,28:2})
```

```
In [115]: df1.head()
```

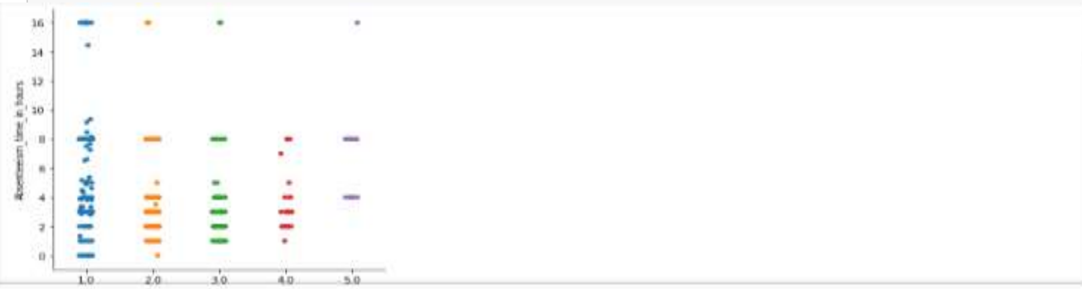
Out[115]:

	Reason_for_absence	Month_of_absence	Day_of_the_week	Seasons	Transportation_expense	Distance_from_Residence_to_Work	Service_time	Age	Work_l
0	5.0	7.0	3.0	1.0	289.0	36.0	13.0	33.0	
1	1.0	7.0	3.0	1.0	118.0	13.0	18.0	50.0	
2	3.0	7.0	4.0	1.0	179.0	51.0	18.0	38.0	
3	1.0	7.0	5.0	1.0	279.0	5.0	14.0	39.0	
4	3.0	7.0	5.0	1.0	289.0	36.0	13.0	33.0	

```
In [116]: df1.columns
```

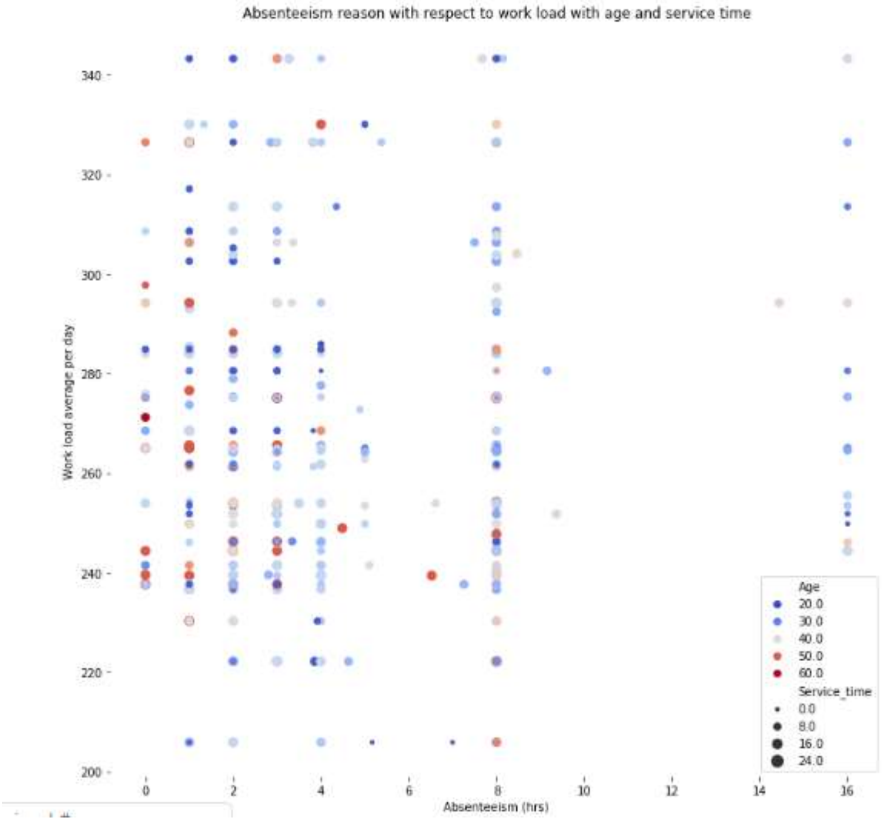
```
Out[116]: Index(['Reason_for_absence', 'Month_of_absence', 'Day_of_the_week', 'Seasons',
                 'Transportation_expense', 'Distance_from_Residence_to_Work',
                 'Service_time', 'Age', 'Work_load_Average/day_', 'Hit_target',
                 'Disciplinary_failure', 'Education', 'Son', 'Social_drinker',
                 'Social_smoker', 'Pet', 'Weight', 'Height', 'Body_mass_index',
                 'Absenteeism_time_in_hours'],
                dtype='object')
```

```
In [117]: for i in cat_cnames:
              sns.catplot(x=i, y="Absenteeism_time_in_hours", data=df1)
              fname = str(i)+'.pdf'
              plt.savefig(fname)
```



From the above plots we can say that people with only a high school degree are absent more often. The reson frequently used is 1 in absenteeism hrs.which is code of Diseases.we can see that people with no children or no pets tend to be absent more ofte than people who have children or pets. we can also see that the people who are social drinker tend to be absent more as comapre to non drinker. Absenteeism through months and days of week and seasons are almost constant. People with disciplinary failure o have maximum absenteeism.

```
In [122]: #Absenteeism visualization-
          f, ax = plt.subplots(figsize=(12, 12))
          sns.despine(f, left=True, bottom=True)
          ax5 = sns.scatterplot(x="Absenteeism_time_in_hours", y="Work_load_Average/day_",
                          hue="Age", size="Service_time",
                          palette="coolwarm",
                          sizes=(10, 100), linewidth=0,
                          data=df1, ax=ax)
          ax5.set_title("Absenteeism reason with respect to work load with age and service time")
          ax5.set_ylabel("Work load average per day")
          ax5.set_xlabel("Absenteeism (hrs)")
          plt.savefig('Absenteeism_reason1.pdf')
```



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

In [123]:
```
#data-
predict_loss= df1
```

In [124]:
```
predict_loss.head()
```

Out[124]:

	Reason_for_absence	Month_of_absence	Day_of_the_week	Seasons	Transportation_expense	Distance_from_Residence_to_Work	Service_time	Age	Work_l
0	5.0	7.0	3.0	1.0	289.0	36.0	13.0	33.0	
1	1.0	7.0	3.0	1.0	118.0	13.0	18.0	50.0	
2	3.0	7.0	4.0	1.0	179.0	51.0	18.0	38.0	
3	1.0	7.0	5.0	1.0	279.0	5.0	14.0	39.0	
4	3.0	7.0	5.0	1.0	289.0	36.0	13.0	33.0	

In [125]:
```
#work loss per month-
predict_loss['work_loss_average/day'] = 0
for i  in range(len(predict_loss)):
    predict_loss['work_loss_average/day'].loc[i] = ((predict_loss['Work_load_Average/day_'].loc[i])/24)*predict_loss['Absenteeism
```

```
C:\Users\Mayur Sharma\Anaconda3\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

In [126]:
```
predict_loss
```

Out[126]:

	Reason_for_absence	Month_of_absence	Day_of_the_week	Seasons	Transportation_expense	Distance_from_Residence_to_Work	Service_time	Age
0	5.0	7.0	3.0	1.0	289.000000	36.0	13.0	33.000000
1	1.0	7.0	3.0	1.0	118.000000	13.0	18.0	50.000000
2	3.0	7.0	4.0	1.0	179.000000	51.0	18.0	38.000000
3	1.0	7.0	5.0	1.0	279.000000	5.0	14.0	39.000000
4	3.0	7.0	5.0	1.0	289.000000	36.0	13.0	33.000000
5	2.0	7.0	6.0	1.0	360.999993	52.0	3.0	28.000000
6	3.0	7.0	6.0	1.0	260.000000	50.0	11.0	36.000000
7	1.0	7.0	2.0	1.0	155.000000	12.0	14.0	34.000000
8	2.0	7.0	2.0	1.0	235.000000	11.0	14.0	37.000000
9	1.0	7.0	2.0	1.0	260.000000	50.0	11.0	36.000000
10	1.0	7.0	3.0	1.0	260.000000	50.0	11.0	36.000000
11	1.0	7.0	4.0	1.0	260.000000	50.0	11.0	36.000000
12	1.0	7.0	4.0	1.0	179.000000	51.0	18.0	38.000000
13	3.0	7.0	4.0	1.0	179.000000	51.0	18.0	38.000000

In [127]:

```
#total absenteeism per month-
Absenteeism_hours_monthly = predict_loss.groupby('Month_of_absence').sum()
```

In [128]:

```
Absenteeism_hours_monthly= Absenteeism_hours_monthly[['Absenteeism_time_in_hours','work_loss_average/day']]
```

In [129]:

```
Monthly_loss= Absenteeism_hours_monthly.rename(columns={'Absenteeism_time_in_hours': 'Absenteeism time/month(hrs.)',
                                  'work_loss_average/day': 'Work loss per month'})
```

In [130]:

```
Monthly_loss
```

Out[130]:

	Absenteeism time/month(hrs.)	Work loss per month
Month_of_absence		
1.0	171.685945	2258.468119
2.0	279.384511	3167.553023
3.0	443.587342	5202.503497
4.0	239.915715	2732.079510
5.0	259.744293	2651.602016
6.0	240.571077	2710.274163
7.0	370.688157	3914.799765
8.0	237.163987	2332.985366
9.0	186.884730	2118.879953
10.0	281.000014	3152.253693
11.0	245.691372	2915.497810
12.0	199.865272	2159.711668

Thank You

References-

1. For Data Cleaning and Model Development - https://edwisor.com/career-data-scientist

2. For Visualization – https://www.udemy.com/python-for-data-science-and-machine-learning-bootcamp/

THANK YOU