Employee Absenteeism

PROJECT REPORT

Abhinav Singh

4th January 2019

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 Data

We need to build models which can help the company in solving absenteeism issue so as to optimize the work of the company and predict the losses if the issue continues. From the dataset, we can observe there are 21 variables in our data which can be further identified as 20 independent variables and 1 dependent variable. Here our target variable i.e. "Absenteeism time in hours" is continuous in nature. From the dataset provided to us we have picked up 5 observations as sample which is given below :

Absenteeism at Work (Column 1 - 12)

ID	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
11	26	7	3	1	289	36	13	33	239,554	97	0
36	0	7	3	1	118	13	18	50	239,554	97	1
3	23	7	4	1	179	51	18	38	239,554	97	0
7	7	7	5	1	279	5	14	39	239,554	97	0
11	23	7	5	1	289	36	13	33	239,554	97	0

Absenteeism at Work (Column 13 - 21)

Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
1	2	1	0	1	90	172	30	4

1	1	1	0	0	98	178	31	0
1	0	1	0	0	89	170	31	2
1	2	1	1	0	68	168	24	4

Variables 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 behavioral 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 (kilometers)

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 a way to analyze data sets and get a better understanding about the data. In the given data set there are 21 variables and data types of all variables are in numerical format. We have 21 variables and 740 observations in the data.

Variables and their number of unique values -

```
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
```

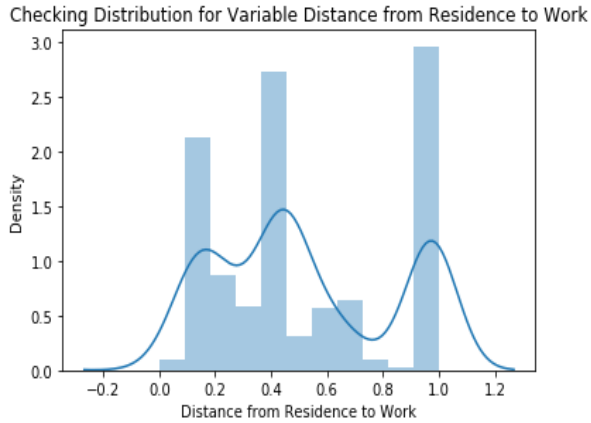
```
Hit target                        13
Disciplinary failure               2
Education                          4
Son                                5
Social drinker                     2
Social smoker                      2
Pet                                6
Weight                            26
Height                            14
Body mass index                   17
Absenteeism time in hours         19
```

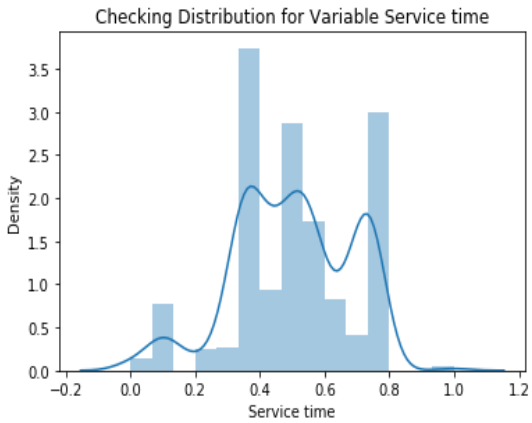
After Exploratory Data Analysis, we can observe that there are 10 continuous variables and 11 categorical variables in the dataset.

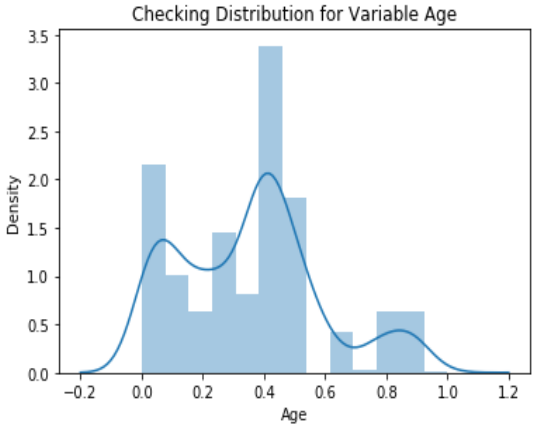
2. Methodology

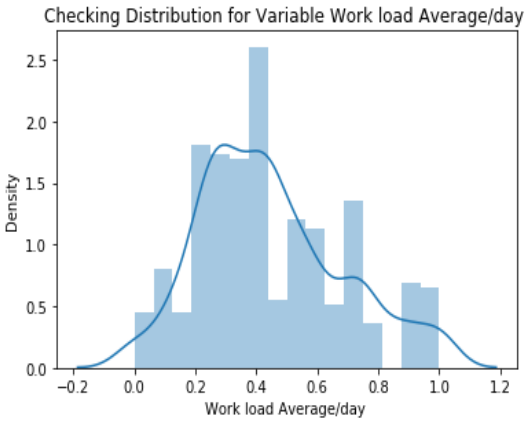
2.1 Pre Processing

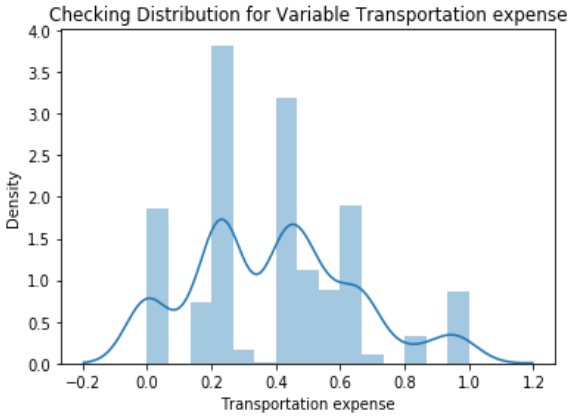
Data in real world is not clean and can't be used until we pre-process the data i.e. transforming the data into a suitable format before feeding it to our machine leaning model or algorithm. With the help data mining techniques which involves transforming raw data into a machine learning model understandable format. In the process we clean the data, and work on data integration, transformation etc. It is the most crucial part of data science project and almost 80% of time is spent on it. This is known as Exploratory Data Analysis. We first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. Here, we visualize and explore the data which will help us to build hypothesis, selecting proper machine learning algorithm etc.

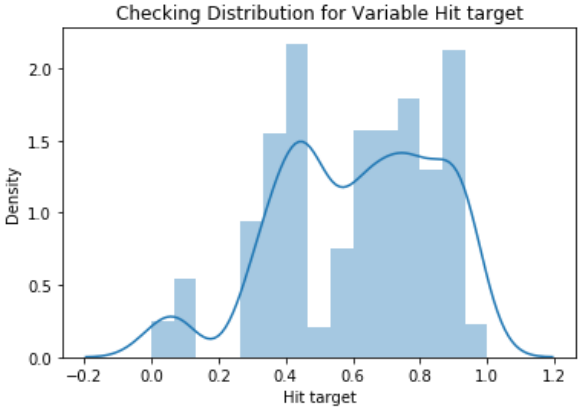


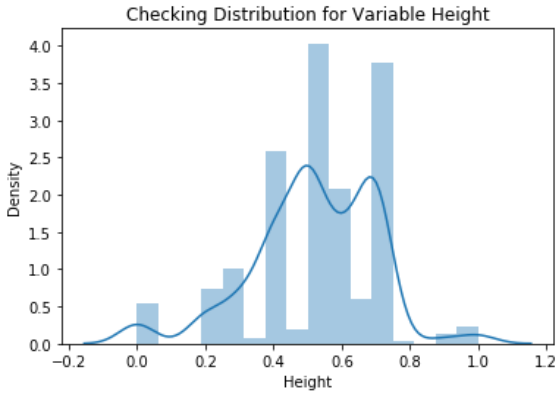


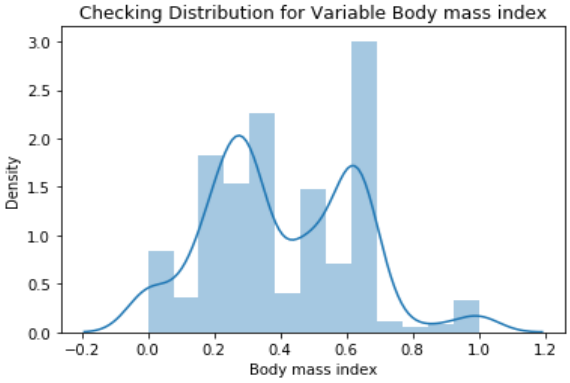








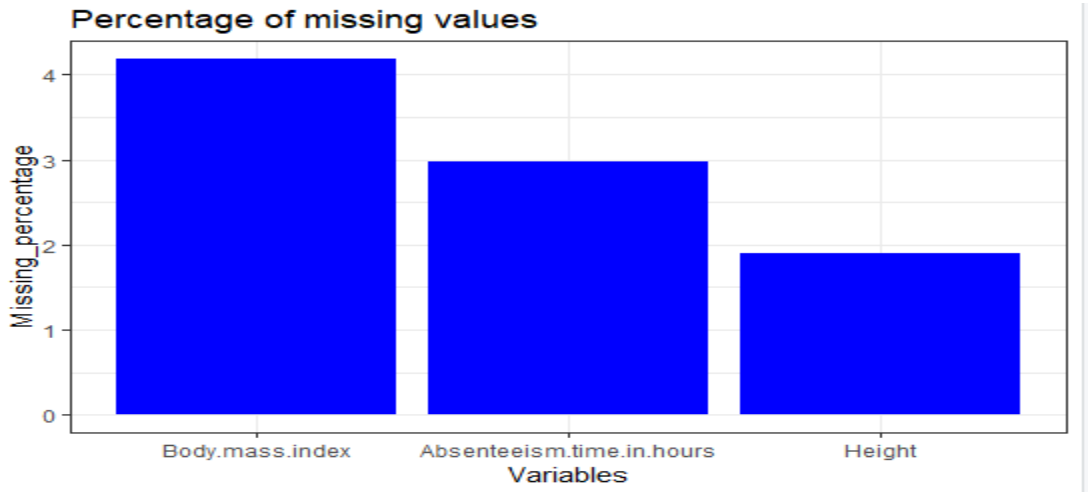




2.2.1 Missing Value Analysis

In a dataset, if we find no value in an observation for a particular variable or variables we consider them as missing values. Most data sets have missing values for various reasons like human error, refuse to answer while surveying, optional box in questionare etc. On encountering missing values, we can either choose to impute them or ignore them. Depending on conditions we can either choose to drop the entire variable or entire observation or even

conisder imputing the missing values. As per industry standards, we consider the variables to impute whose missing values is less than 30%. We plotted top 3 variables for missing values and the maximum percentage of missing value is 4.189% for “Body Mass Index” variable .So we will check for missing values in all variables and consider imputing them. We will impute the values with KNN imputation method as it works good on this dataset.



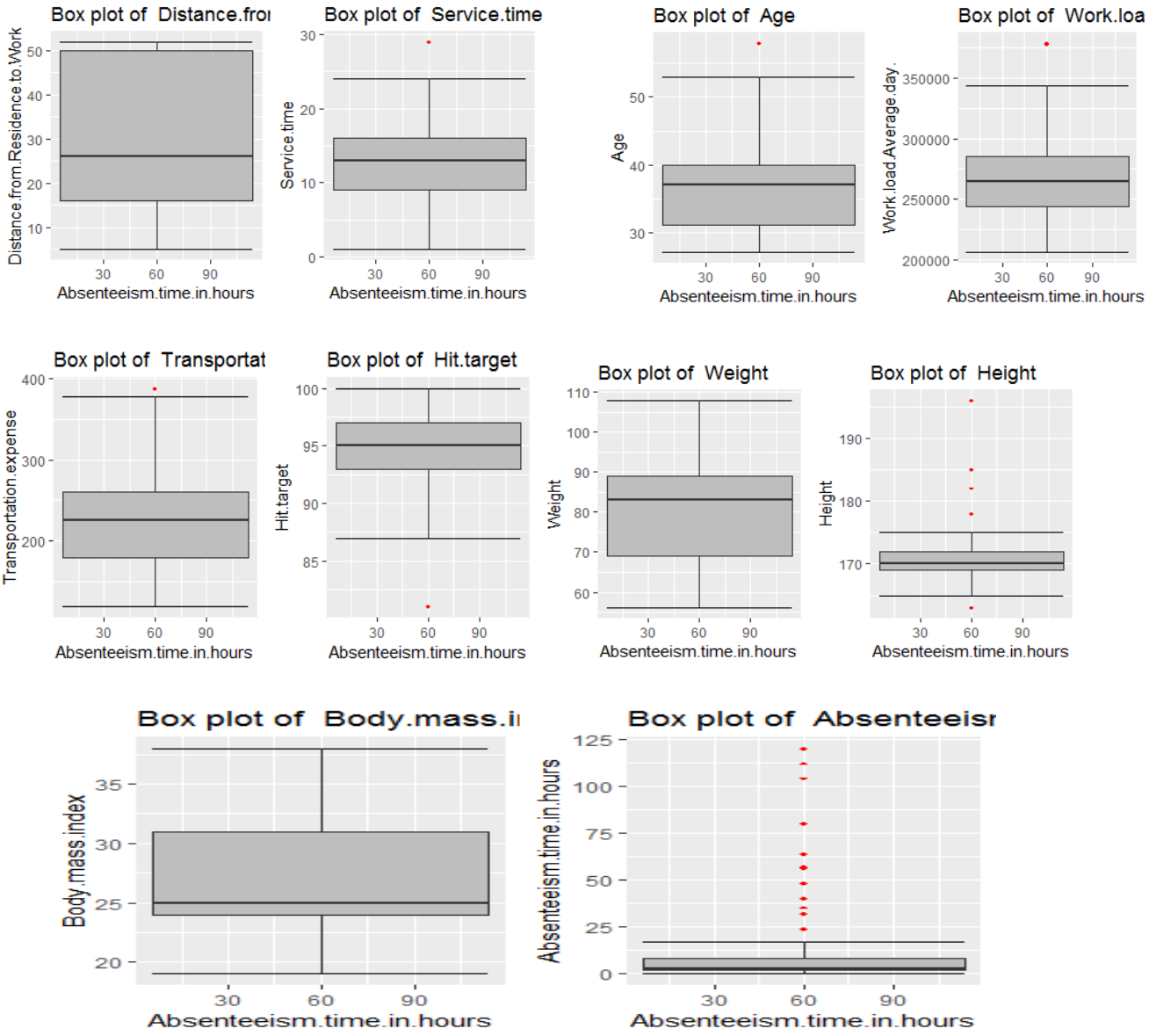
Below figure shows the exact missing value percentages in data.

	Variables	Missing_percentage
0	Body mass index	4.189189
1	Absenteeism time in hours	2.972973
2	Height	1.891892
3	Work load Average/day	1.351351
4	Education	1.351351
5	Transportation expense	0.945946
6	Son	0.810811
7	Disciplinary failure	0.810811
8	Hit target	0.810811
9	Social smoker	0.540541
10	Age	0.405405
11	Reason for absence	0.405405
12	Service time	0.405405
13	Distance from Residence to Work	0.405405
14	Social drinker	0.405405
15	Pet	0.270270
16	Weight	0.135135
17	Month of absence	0.135135
18	Seasons	0.000000
19	Day of the week	0.000000
20	ID	0.000000

2.1.2 Outlier Analysis

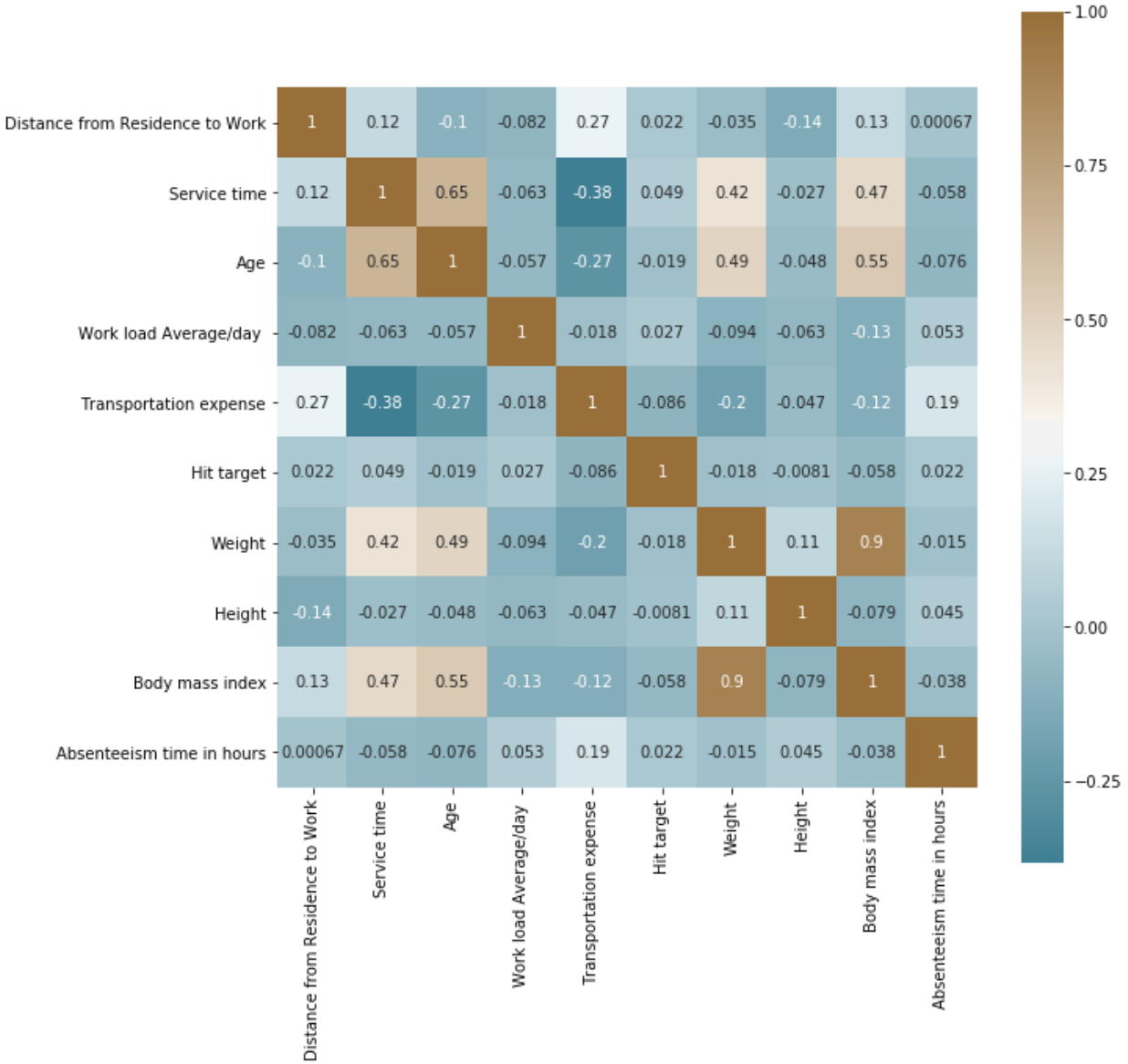
The shown boxplot refers outliers on the predictors variables, we can see various outliers associated with the features. Even though, the data has considerable amount of outliers, the approach is to retain every outlier and grab respective behavior of all employees. As shown there are significant amount of outliers present in the target variable, which indicates a trend on Employee ' behavior, there can be pattern , we need to treat those outliers.

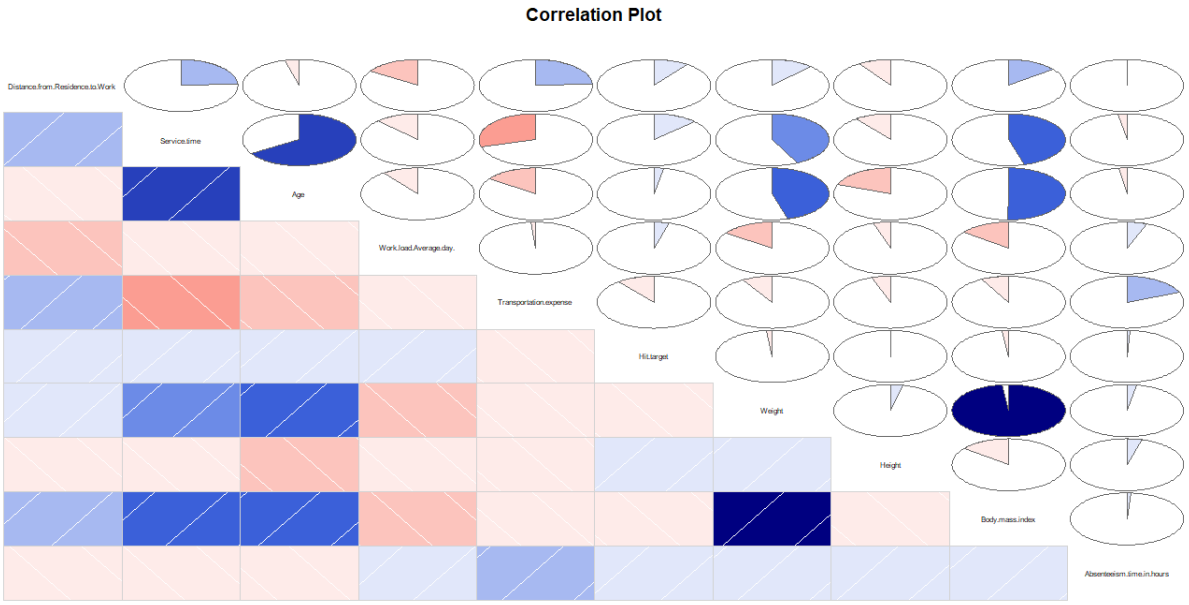
Below are the boxplots of the 11 independent continuous variables with respect to our target variable "Absenteeism time in hour**"**. These plots can help us with a lot of data understanding and give us knowledge how to manipulate the data. Most of the variables except "Distance from residence to work", "Weight" and "Body mass index" consists of outliers. We converted the outliers as NA i.e. missing values and filled them by **KNN** imputation method.



2.1.3 **Feature Selection**

Feature Election is described as the knowledge extraction process, where important features are selected using domain knowledge to make a machine learning algorithm work. There can be features that aren't relevant for the analysis, we can remove such variables using numerous ways. All the features may not be useful as they might be carrying the same information or irrelevant information and may cause multi- collinearity. Here, we have used Correlation Analysis for continuous variable and ANOVA (Analysis of variance) for categorical variable since our target variable is continuous.





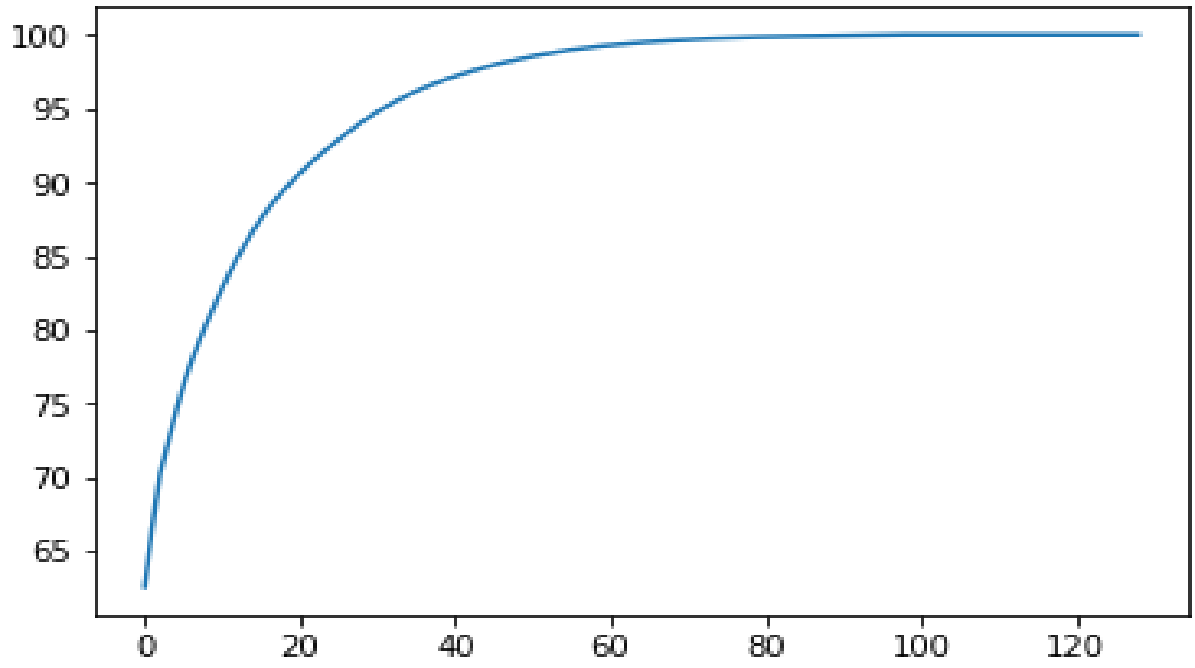
Weight and Body mass index have high correlation and hence we decided to drop the weight.

2.2.4 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.

2.2.5 Principal Component Analysis

Principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. With fewer variables, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher dimensional data. After creating dummy variable of categorical variables the shape of our data became 107 columns and 714 observations, this high number of columns leads to bad accuracy.



We have applied PCA algorithm on our data and from the above graph we have concluded that 45 variables out of 107 explains more than 95% of data. So we have selected only those 45 variables to feed our models.

2.2 Modeling

No we will apply some regression models on our processed data to predict the target variable. As we can see, we have applied all the possible preprocessing analysis to our dataset to make it suitable for calculation. We have also removed the missing values and outliers. Now since our data is a regression model, we have applied suitable models such as decision tree and random forest. The error metric results of both the models are as follows

2.2.1 Decision Tree

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”. It can be used for classification and regression. It is a supervised machine learning algorithm. Accept continuous and categorical variables as independent variables. Extremely easy to understand by the business users. Split of decision tree is seen in the below tree. The RMSE value and R^2 value for our project in R and Python are –

Decision Tree	R	PYTHON
RMSE Train	0.410	0.573

RMSE Test	0.363	0.568
R^2 Test	0.98	0.96

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 to build each decision tree. It means to build each decision tree on random forest we are not going to use the same data. The RMSE value and R^2 value for our project in R and Python are –

Random Forest	R	PYTHON
RMSE Train	0.369	0.033
RMSE Test	0.614	0.029
R^2 Test	0.96	0.99

2.2.3 Liner Regression

Linear Regression is one of the statistical methods of prediction. It is applicable only on continuous data. To build any model we have some assumptions to put on data and model. Here are the assumptions to the linear regression model.

Linear Regression	R	PYTHON
RMSE Train	0.001	4.34e-15
RMSE Test	0.001	4.14e-15
R^2 Test	0.99	1

2.2.4 Gradient boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Gradient boosting	R	PYTHON
RMSE Train	2.276	0.001
RMSE Test	1.818	0.001
R^2	0.91	0.99

Chapter 3
Conclusion

Here we will analyse our models and try to answer the questions asked.

3.1 Model Evaluation

We calculated the Root Mean Square Error (RMSE) and R-Squared Value of different 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. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, 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. Lower values of RMSE and higher value of R-Squared Value indicate better fit.

3.2 Model Selection

Here according to the problem statement, we are supposed to predict the loss incurred by the company if the same pattern of absenteeism continues. Hence we are selection the following two models,

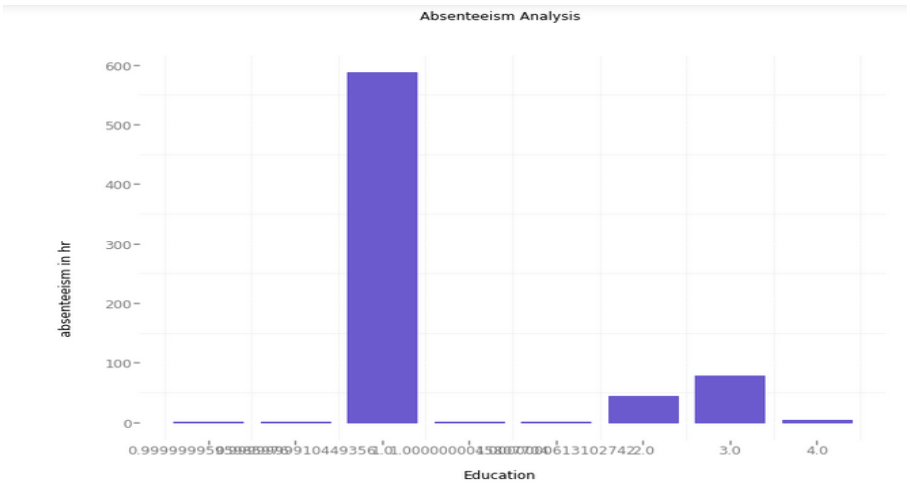
1. Decision tree
2. Random forest model

Both training models Decision tree and random forest were implemented in R and python. After building an initial model, performance tuning was done using hyper parameter tuning for optimized parameters.

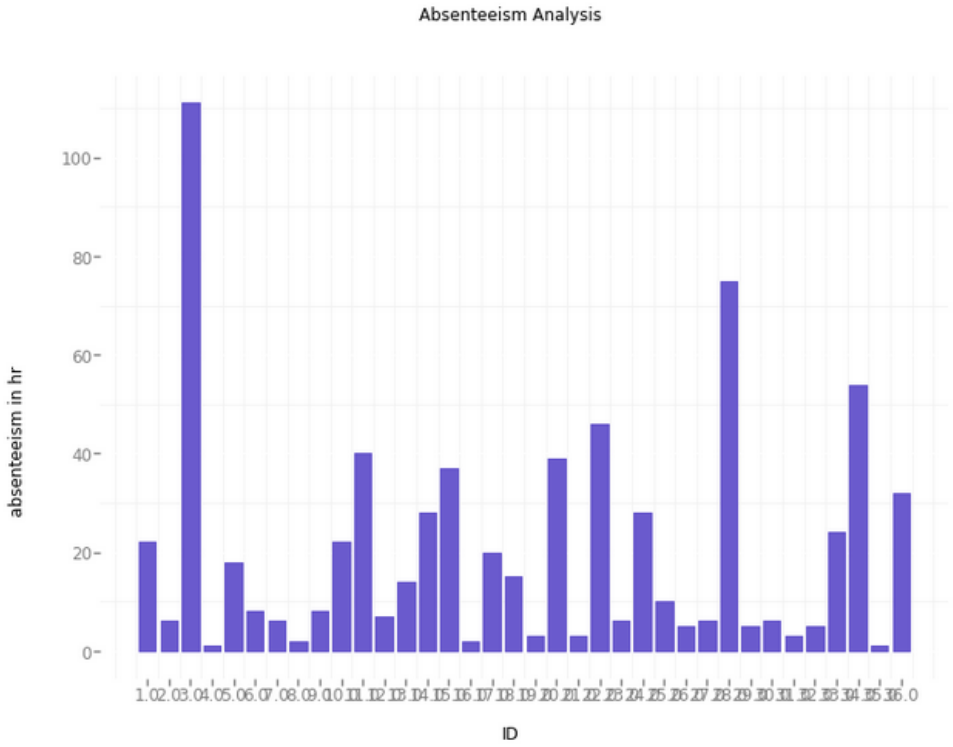
3.2 Answers of asked questions

The Changes which company should bring to reduce the number of absenteeism –

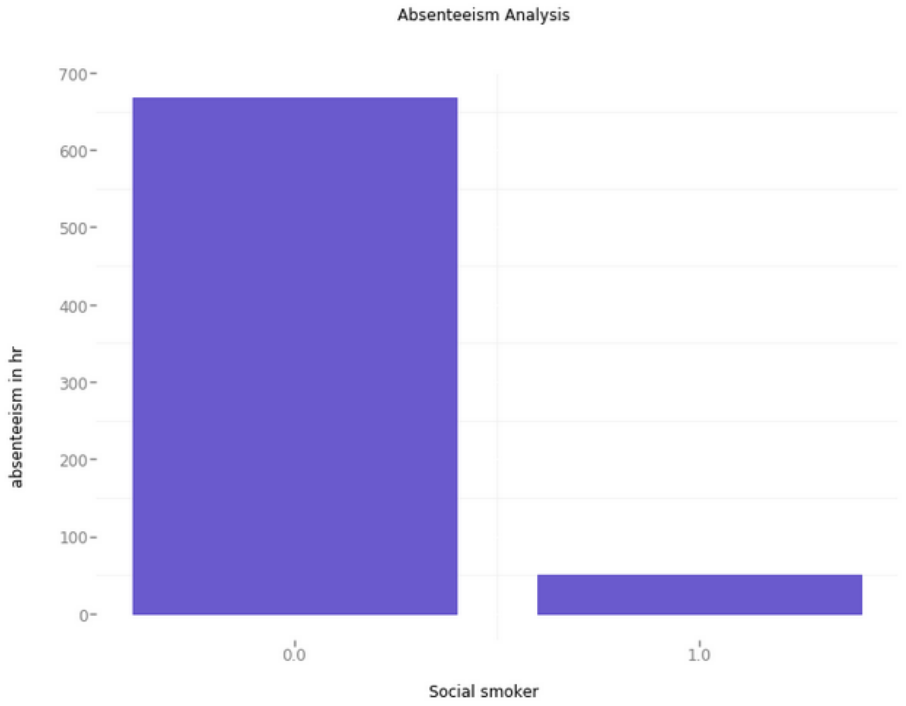
1. It is observed that employee with low education have maximum absentee time.



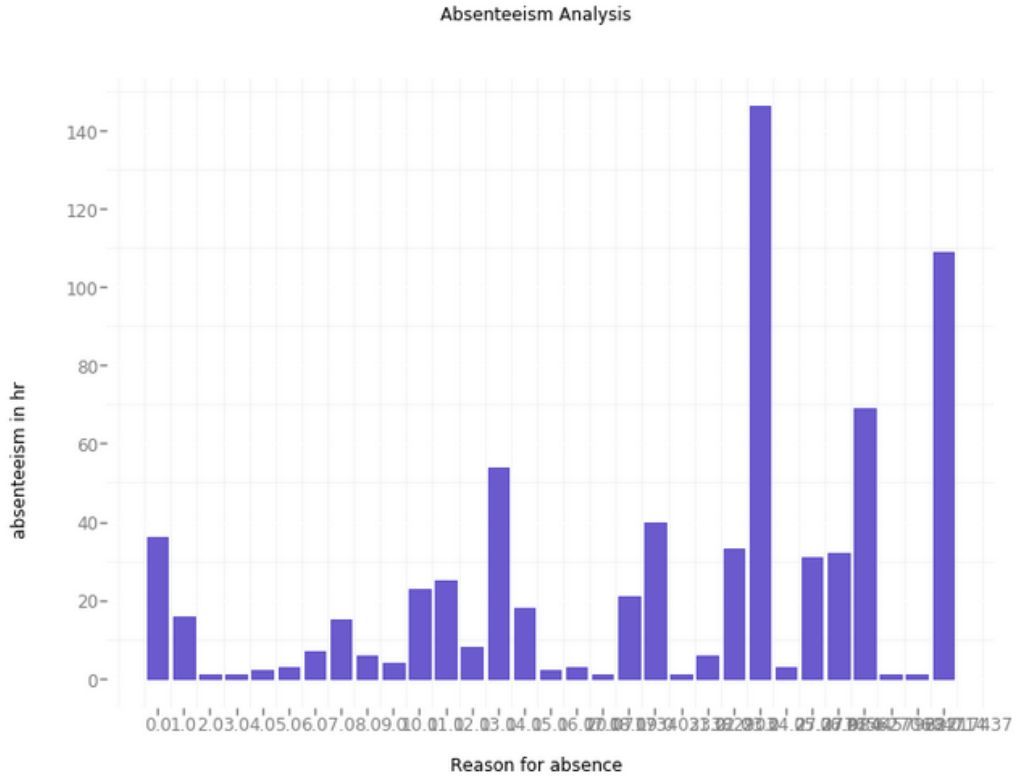
2. Some employee with **ID 3, 28, 34** are often absent from work, company should take action against them.



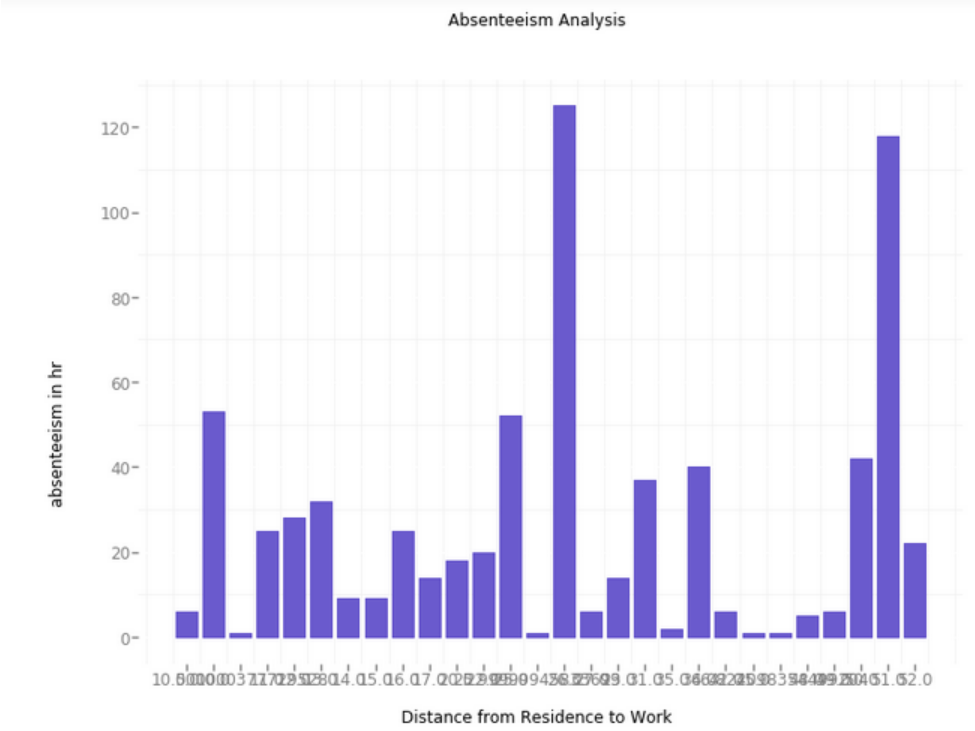
3. Employees who are social smoker have more absentee hour than who are not social smoker.



4. Most often Reason for absence are medical consultation and dental consultation, company should take care of it.



5. Employees who has Distance from Residence to Work high more tends to absent more.

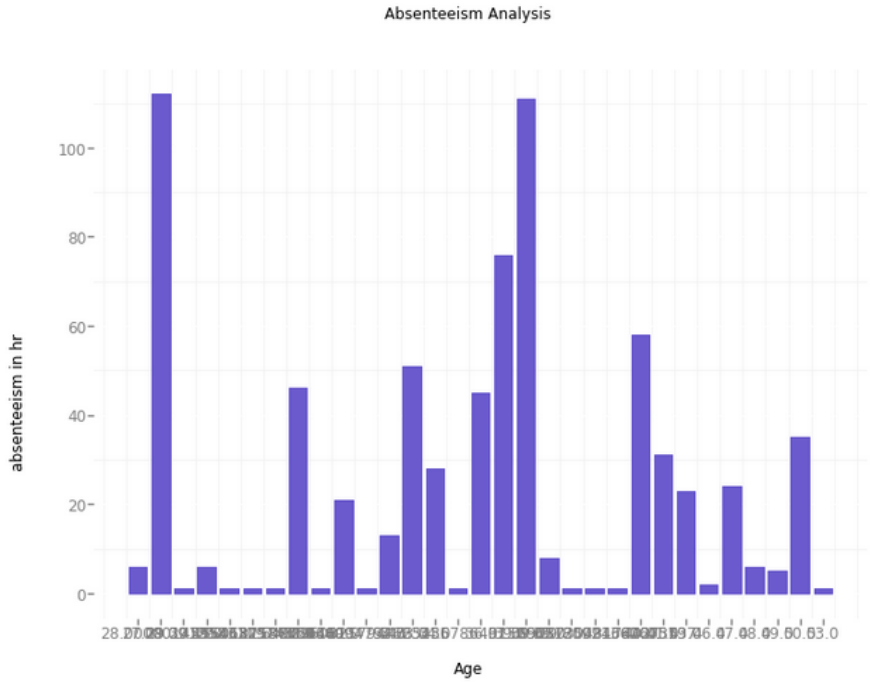


Appendix

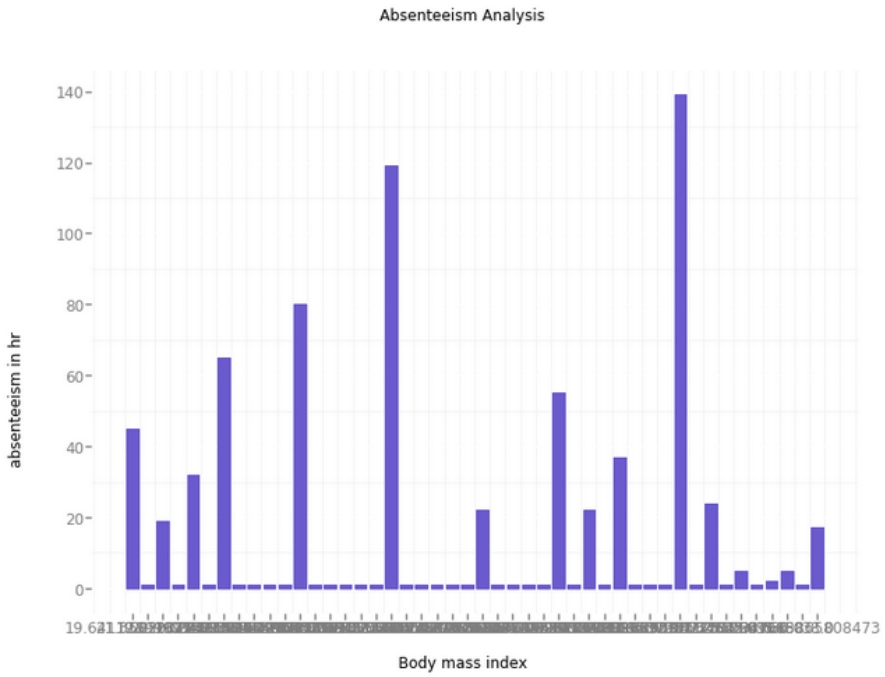
Extra Figures

Relationship of our target variable (Absentee time in hour) with other variables.

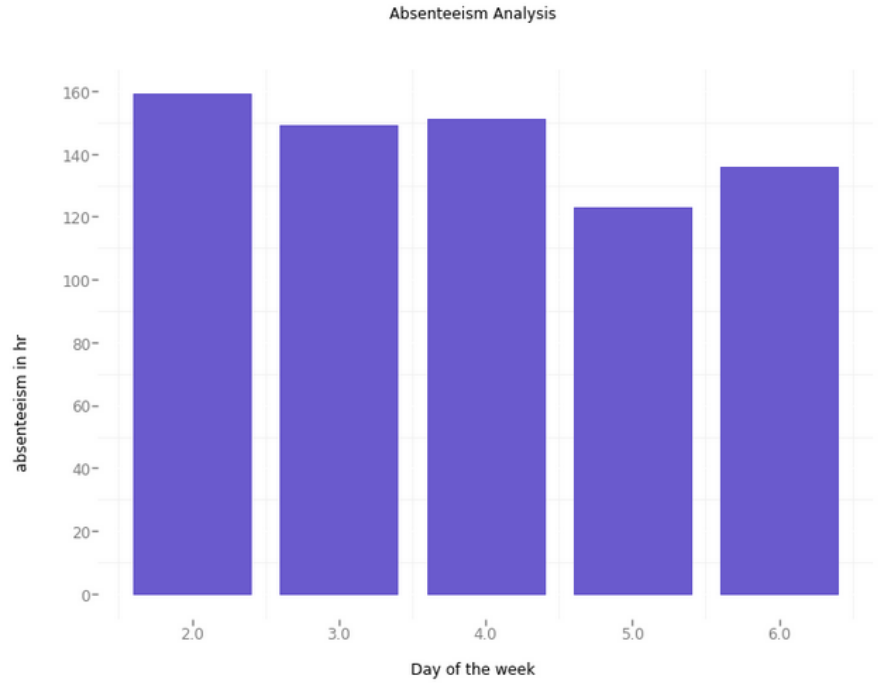
1. **With "Age"**



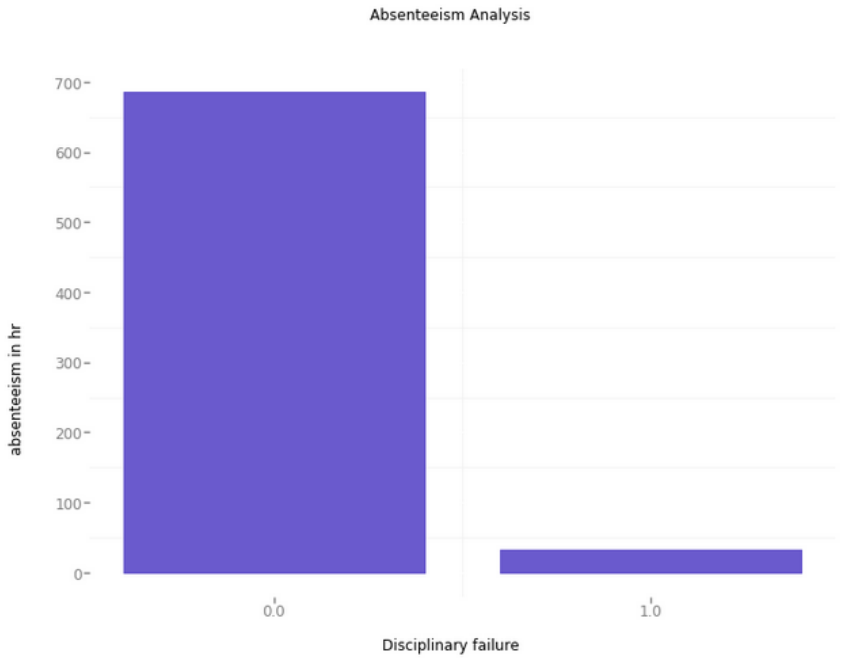
2. **With "Body mass index"**



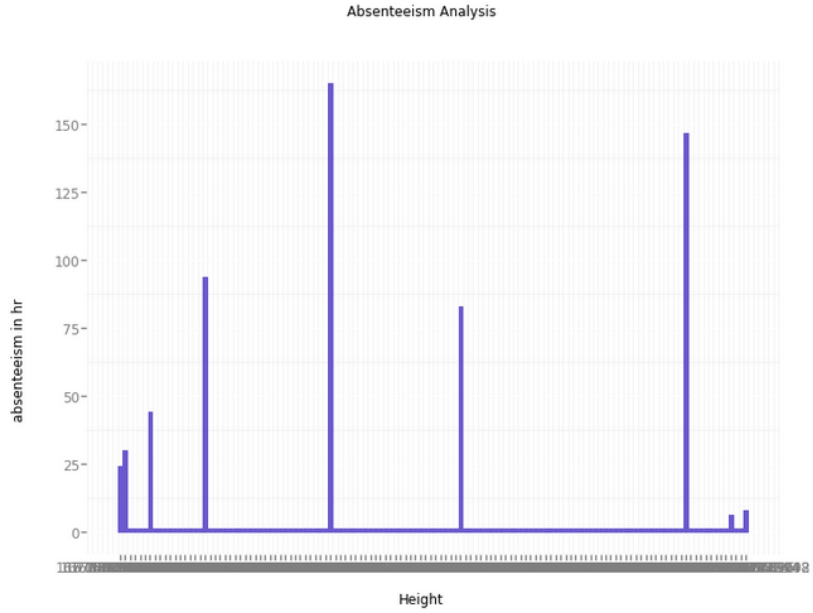
3. With “Day of week”



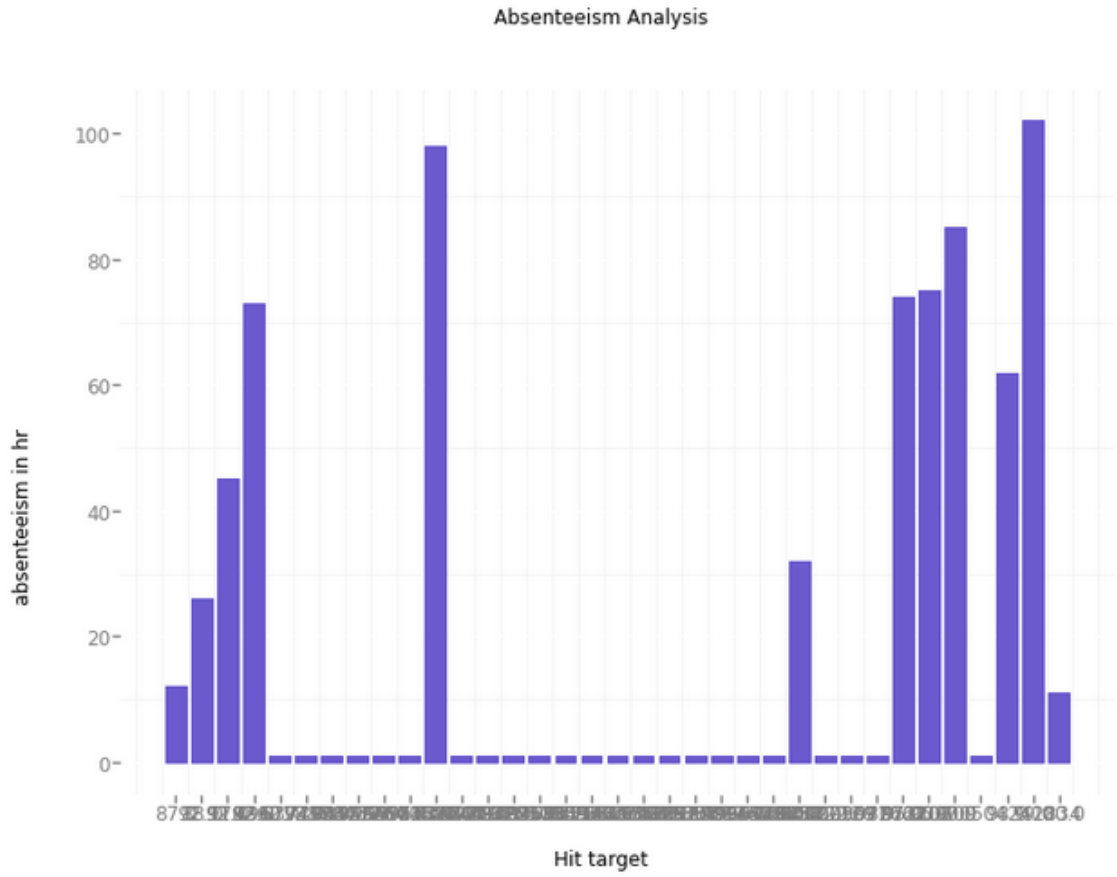
4. With “Disciplinary failure”



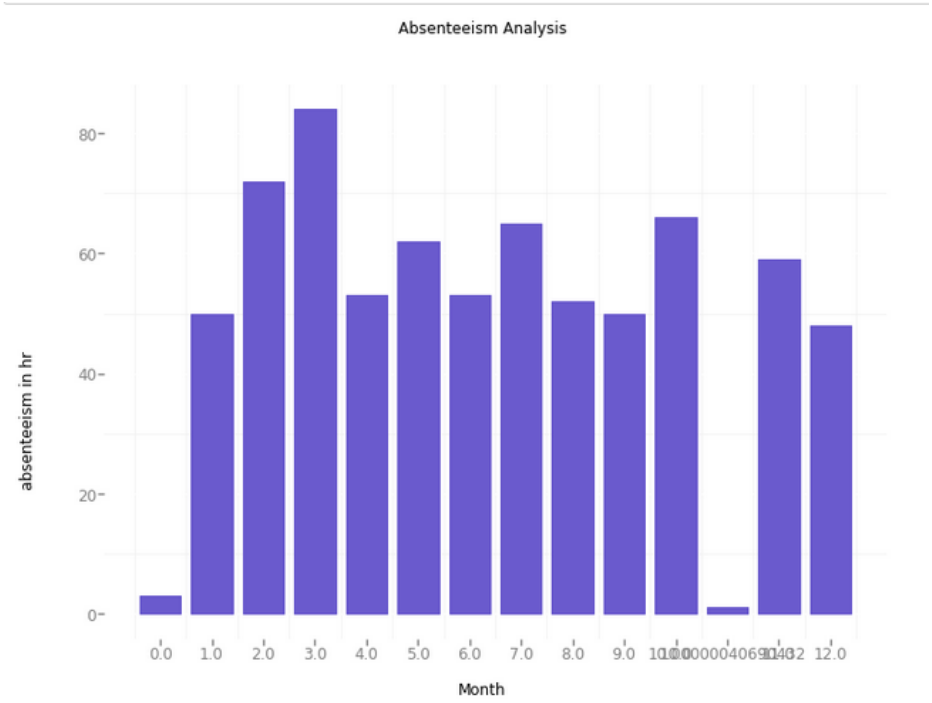
5. **With “Height”**



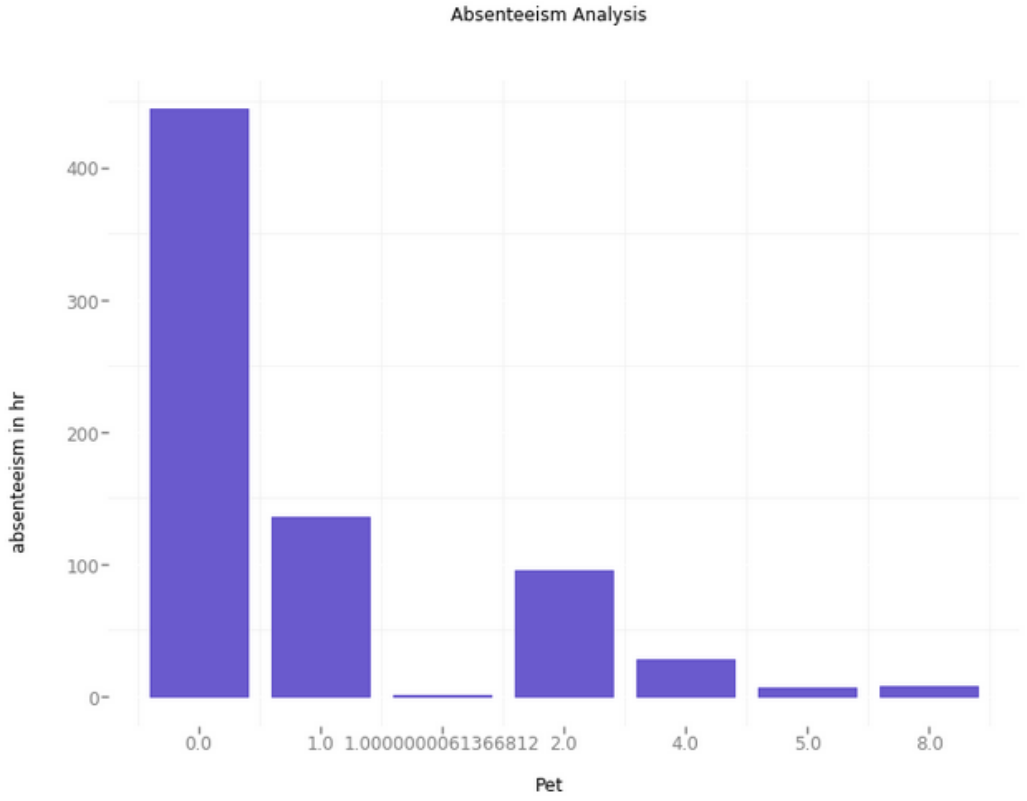
6. **With “Hit target”**



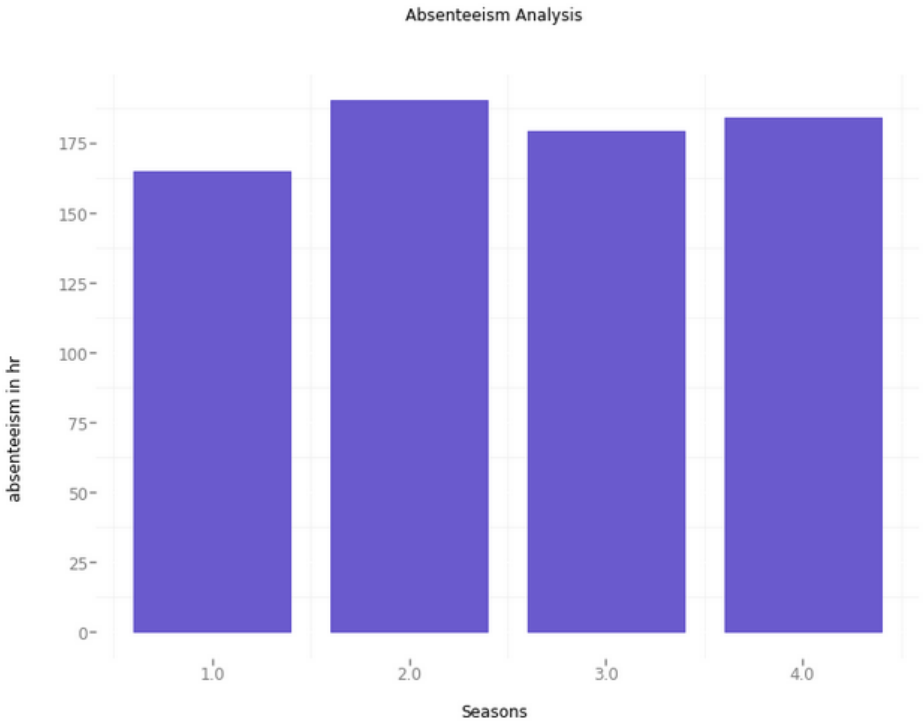
7. **With “Month of absence”**



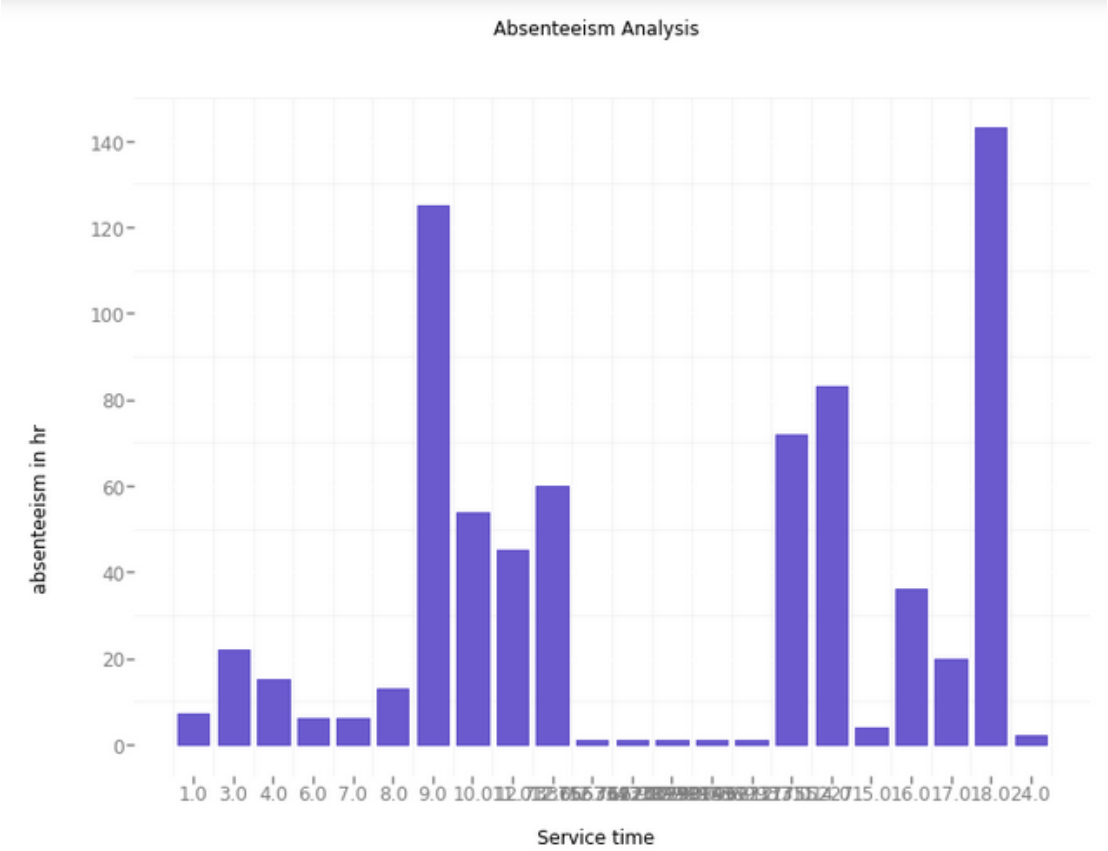
8. **With "Pet"**



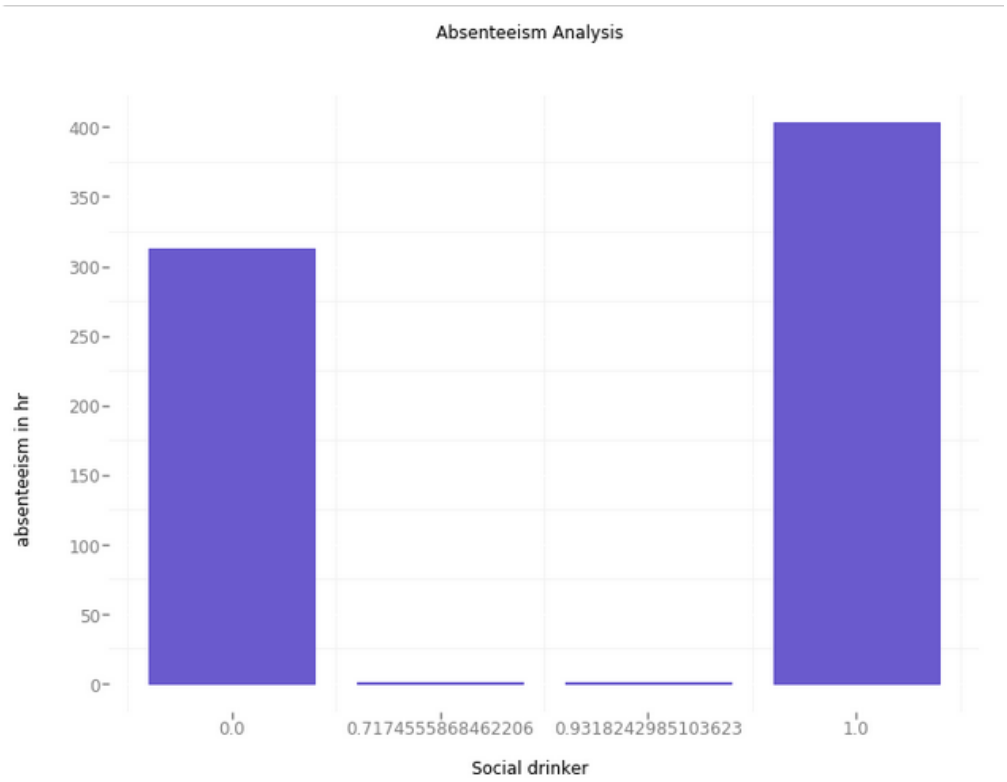
9. **With "Seasons"**



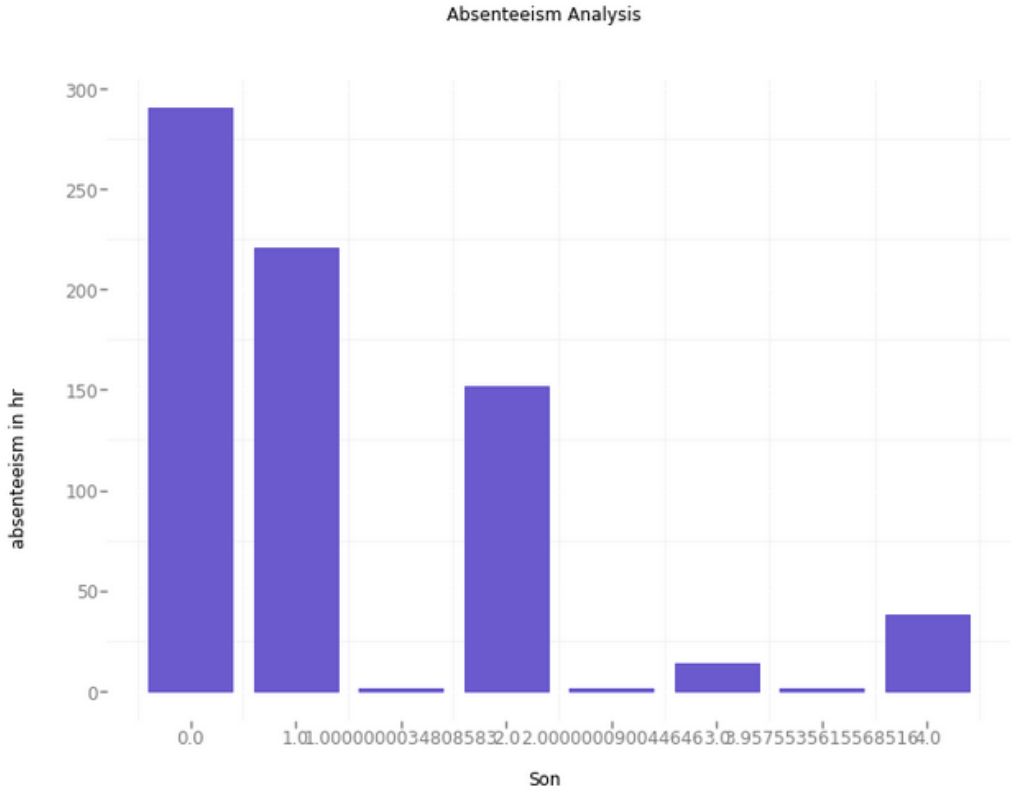
10. With “Service time”



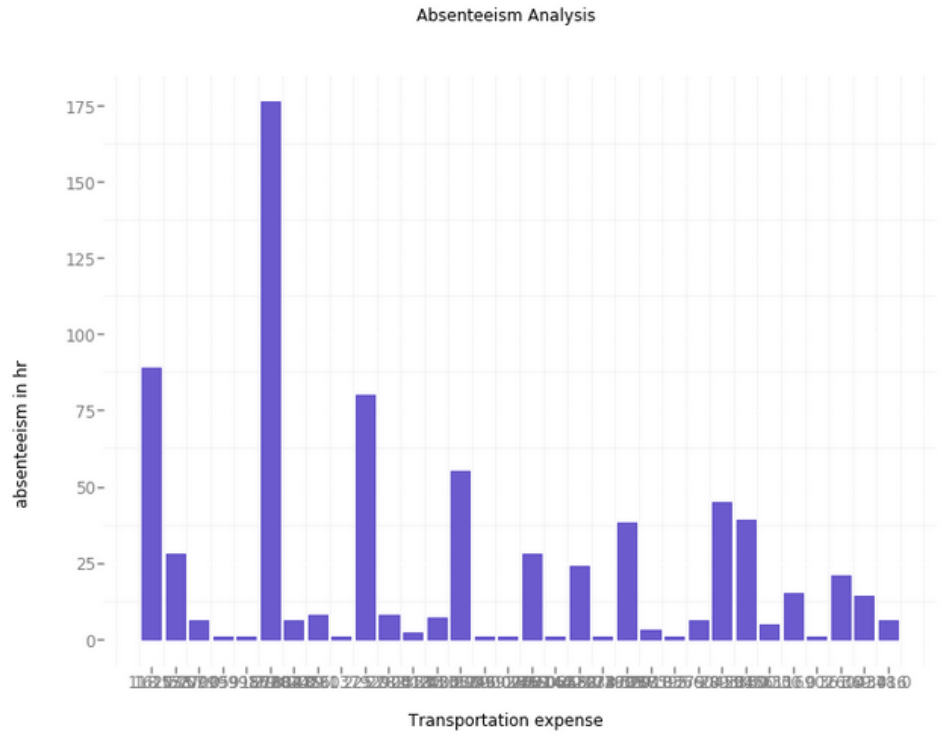
11. With “Social drinker”



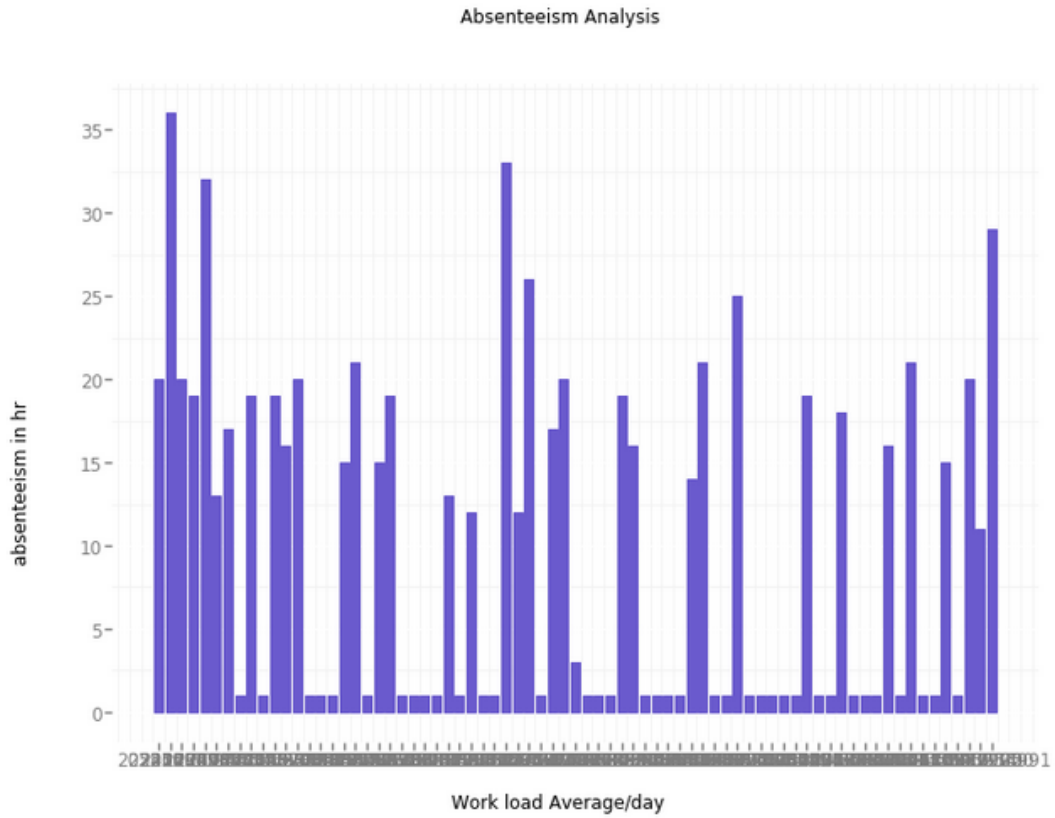
12. With “Son”



13. With “Transportation Expense”



14. With “Work load/day”



Codes

R Code

```
#Remove all objects stored
rm(list=ls(all=T))

# Setting working directory
setwd("F:/Data Science/Edwisor Workspace/Project/Employee Absentism")

#Load Required Libraries
library(ggplot2)
library(corrgram)
library(DMwR)
library(caret)
library(randomForest)
library(unbalanced)
library(dummies)
library(e1071)
library(Information)
library(MASS)
library(rpart)
library(gbm)
library(ROSE)
library(xlsx)
library(DataCombine)
library(rpart)

#Load data
```

```
absent_data = read.xlsx("Absenteeism_at_work_Project.xls", sheetIndex = 1, header = TRUE, colClasses = NA)



#****************EXPLORATORY DATA ANALYSIS******************

#Dimension of the data

dim(absent_data)


#Structure of the data

str(absent_data)


# Variable namesof the data

colnames(absent_data)


# Separating Continuous and Categorical Variables

continuous_vars = 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')


catagorical_vars = c('ID','Reason.for.absence','Month.of.absence','Day.of.the.week',

                'Seasons','Disciplinary.failure', 'Education', 'Social.drinker',

                'Social.smoker', 'Son', 'Pet')




#*************MISSING VALUE ANALYSIS**************

#Creating dataframe with missing values present in each variable

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

```
#Convert row into column

missing_val$Columns = row.names(missing_val)

row.names(missing_val) = NULL

#Rename variable

names(missing_val)[1] =  "Missing_percentage"


#Calculate missing value percentage

missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(absent_data)) * 100


#Sort missing value proportion in descending order

missing_val = missing_val[order(-missing_val$Missing_percentage),]



#Rearrange columns

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


# Save output result as csv file

write.csv(missing_val, "Missing_percentage_R.csv", row.names = F)


#Plot top 3 variables with missing values

ggplot(data = missing_val[1:3,], aes(x=reorder(Columns, -Missing_percentage),y =
Missing_percentage))+

geom_bar(stat = "identity",fill = "blue")+xlab("Variables")+

ggtitle("Percentage of missing values") + theme_bw()


#Create missing value

#Check value

absent_data[95,20]
```

```
#Create missing value

absent_data[95,20] = NA


# Actual Value = 32

# Mean = 26.67

# Median = 25

# KNN = 32



#Mean Method

#absent_data$Body.mass.index[is.na(absent_data$Body.mass.index)] =
mean(absent_data$Body.mass.index, na.rm = T)

#Check value

#absent_data[95,20]


#Create missing value

#absent_data[95,20] = NA


#Median Method

#absent_data$Body.mass.index[is.na(absent_data$Body.mass.index)] =
median(absent_data$Body.mass.index, na.rm = T)

#Check value

#absent_data[95,20]


#Create missing value

absent_data[95,20] = NA


#KNN Imputation

absent_data = knnImputation(absent_data, k = 3)
```

```
#Check value
absent_data[95,20]

# Checking for missing value
sum(is.na(absent_data))



#*************OULIER ANALYSIS**************
# BoxPlots - Distribution and Outlier Check

# Boxplot for continuous variables
for (i in 1:length(continuous_vars))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (continuous_vars[i]), x = "Absenteeism.time.in.hours"), data
= subset(absent_data))+
        stat_boxplot(geom = "errorbar", width = 0.5) +
        geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                     outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=continuous_vars[i],x="Absenteeism.time.in.hours")+
        ggtitle(paste("Box plot for outliers in absenteeism data variables",continuous_vars[i])))
}

# Plotting plots together
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
gridExtra::grid.arrange(gn5,gn6,ncol=2)
gridExtra::grid.arrange(gn7,gn8,ncol=2)
```

```
gridExtra::grid.arrange(gn9,gn10,ncol=2)



#Outlier removal using boxplot method
#loop to remove outliers from all variables
for(i in continuous_vars)
{
  print(i)
  #Extract outliers and store in val
  val = absent_data[,i][absent_data[,i] %in% boxplot.stats(absent_data[,i])$out]
  #Remove outliers and store cleaned data back in data
  absent_data = absent_data[which(!absent_data[,i] %in% val),]
}

#Replace all outliers with NA
#for(i in continuous_vars)
{
  #Extract outliers and store in val
  #val = absent_data[,i][absent_data[,i] %in% boxplot.stats(absent_data[,i])$out]
  #Replace outliers with NA
  #absent_data[,i][absent_data[,i] %in% val] = NA
}

# Imputing missing values using KNN
#absent_data = knnImputation(absent_data, k=3)



#**********FEATURE SELECTION********
```

```
#Correlation Plot for continuous variables
corrgram(absent_data[,continuous_vars], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#ANOVA test for Categprical variable
summary(aov(formula = Absenteeism.time.in.hours~ID,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Reason.for.absence,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Month.of.absence,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Day.of.the.week,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Seasons,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Disciplinary.failure,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Education,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Social.drinker,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Social.smoker,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Son,data = absent_data))
summary(aov(formula = Absenteeism.time.in.hours~Pet,data = absent_data))

# Dimension Reduction
absent_data = subset(absent_data, select = -c(Weight))

#---------------------------------Feature Scaling---------------------------------#
#Check Normality of target variable
qqnorm(absent_data$Absenteeism.time.in.hours)
hist(absent_data$Absenteeism.time.in.hours)

#Check range of target variable
range(absent_data$Absenteeism.time.in.hours)

#Updating the continuous variable for further processing
```

```
continuous_vars = 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')

# Normalization
for(i in continuous_vars)
{
  print(i)
  absent_data[,i] = (absent_data[,i] - min(absent_data[,i]))/(max(absent_data[,i])-min(absent_data[,i]))
}

#Check range of target variable
range(absent_data$Absenteeism.time.in.hours)

#Create back up of data
absent_backup = absent_data

#Creating dummy data for categorical variables
library(mlr)
absent_data = dummy.data.frame(absent_data, catagorical_vars)



#**************Sampling******************
#Stratified sampling method
#Divide data into train and test
set.seed(123)
train.index = createDataPartition(absent_data$Absenteeism.time.in.hours, p = .80, list = FALSE)
absent_train = absent_data[ train.index,]
```

```
absent_test  = absent_data[-train.index,]

#*************Decision tree for classification**************
#Develop Model on training data
fit_DT = rpart(Absenteeism.time.in.hours ~., data = absent_train, method = "anova")

#Summary of DT model
summary(fit_DT)

#write rules into disk
write(capture.output(summary(fit_DT)), "absent_data_rules.txt")

#Lets predict for train data
pred_DT_train = predict(fit_DT, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"])

#Lets predict for test data
pred_DT_test = predict(fit_DT,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"])

#Error metrics
#For training data
print(postResample(pred = pred_DT_train, obs = absent_train[,107]))

#For testing data
print(postResample(pred = pred_DT_test, obs = absent_test[,107]))



#*************LINEAR REGRESSION*************
set.seed(123)
```

```
#Develop Model on training data
fit_LR = lm(Absenteeism.time.in.hours ~ ., data = absent_train)

#Lets predict for train data
pred_LR_train = predict(fit_LR, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"])

#Lets predict for test data
pred_LR_test = predict(fit_LR,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"])

#Error Metrics
# For training data
print(postResample(pred = pred_LR_train, obs = absent_train[,107]))

# For testing data
print(postResample(pred = pred_LR_test, obs = absent_test[,107]))



#***********Random Forest***********

set.seed(123)

#Develop Model on train data
fit_RF = randomForest(Absenteeism.time.in.hours~., data = absent_train)

#Lets predict for training data
pred_RF_train = predict(fit_RF, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"])

#Lets predict for test data
pred_RF_test = predict(fit_RF,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"])
```

```
# For training data

print(postResample(pred = pred_RF_train, obs = absent_train[,107]))


# For testing data

print(postResample(pred = pred_RF_test, obs = absent_test[,107]))



#************************XGBoost***********************


set.seed(123)


#Develop Model on training data

fit_XGB = gbm(Absenteeism.time.in.hours~., data = absent_train, n.trees = 500, interaction.depth = 2)


#Lets predict for train data

pred_XGB_train = predict(fit_XGB, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"],
n.trees = 500)


#Lets predict for test data

pred_XGB_test = predict(fit_XGB,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"],
n.trees = 500)


# For training data

print(postResample(pred = pred_XGB_train, obs = absent_train[,107]))


# For testing data

print(postResample(pred = pred_XGB_test, obs = absent_test[,107]))
```

```
#****************Dimensionality Reduction using PCA***************



#Principal Component Analysis

absent_pca = prcomp(absent_train)


#compute standard deviation of each principal component

absent_stddev = absent_pca$sdev


#compute variance

absent_var = absent_stddev^2


#proportion of variance explained

prop_var = absent_var/sum(absent_var)


#Plot

plot(cumsum(prop_var), xlab = "Principal Component",

    ylab = "Proportion of Variance Explained",

    type = "b")


#add a training set with principal components

train.data = data.frame(Absenteeism.time.in.hours = absent_train$Absenteeism.time.in.hours,
absent_pca$x)


# From the above plot selecting 45 components since it explains almost 95+ % data variance

train.data =train.data[,1:45]
```

```
#transform test into PCA
test.data = predict(absent_pca, newdata = absent_test)
test.data = as.data.frame(test.data)

#select the first 45 components
test.data=test.data[,1:45]



#************************Model Development after Dimensionality
Reduction************************

##Decision tree for classification
#Develop Model on training data
fit_DT = rpart(Absenteeism.time.in.hours ~., data = absent_train, method = "anova")

#Summary of DT model
summary(fit_DT)

#write rules into disk
write(capture.output(summary(fit_DT)), "absent_data_rules.txt")

#Lets predict for train data
pred_DT_train = predict(fit_DT, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"])

#Lets predict for test data
pred_DT_test = predict(fit_DT,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"])

#Error metrics
#For training data
```

```
print(postResample(pred = pred_DT_train, obs = absent_train[,107]))

#For testing data
print(postResample(pred = pred_DT_test, obs = absent_test[,107]))



#*************LINEAR REGRESSION*************
set.seed(123)

#Develop Model on training data
fit_LR = lm(Absenteeism.time.in.hours ~ ., data = absent_train)

#Lets predict for train data
pred_LR_train = predict(fit_LR, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"])

#Lets predict for test data
pred_LR_test = predict(fit_LR,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"])

#Error Metrics
# For training data
print(postResample(pred = pred_LR_train, obs = absent_train[,107]))

# For testing data
print(postResample(pred = pred_LR_test, obs = absent_test[,107]))



#***********Random Forest***********

set.seed(123)
```

```
#Develop Model on train data

fit_RF = randomForest(Absenteeism.time.in.hours~., data = absent_train)


#Lets predict for training data

pred_RF_train = predict(fit_RF, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"])


#Lets predict for test data

pred_RF_test = predict(fit_RF,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"])


# For training data

print(postResample(pred = pred_RF_train, obs = absent_train[,107]))


# For testing data

print(postResample(pred = pred_RF_test, obs = absent_test[,107]))



#**********XGBoost**********


set.seed(123)


#Develop Model on training data

fit_XGB = gbm(Absenteeism.time.in.hours~., data = absent_train, n.trees = 500, interaction.depth = 2)


#Lets predict for train data

pred_XGB_train = predict(fit_XGB, absent_train[,names(absent_test) != "Absenteeism.time.in.hours"],
n.trees = 500)


#Lets predict for test data
```

```
pred_XGB_test = predict(fit_XGB,absent_test[,names(absent_test) != "Absenteeism.time.in.hours"],
n.trees = 500)


# For training data

print(postResample(pred = pred_XGB_train, obs = absent_train[,107]))


# For testing data

print(postResample(pred = pred_XGB_test, obs = absent_test[,107]))


#*******************Visualisations*******************

library(ggplot2)

#barplot


#Education

ggplot(absent_backup, aes_string(x = absent_backup$Education,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Education") + ylab('absenteeism')


#Id

ggplot(absent_backup, aes_string(x = absent_backup$ID,y = absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("ID") + ylab('absenteeism')


#Social Smoker

ggplot(absent_backup, aes_string(x = absent_backup$Social.smoker,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Social.smoker") + ylab('absenteeism')


#Reason for absence

ggplot(absent_backup, aes_string(x = absent_backup$Reason.for.absence,y =
absent_data$Absenteeism.time.in.hours)) +
```

```
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Reason.for.absence") +
ylab('absenteeism')


#Distance from Residence to Work

ggplot(absent_backup, aes_string(x = absent_backup$Distance.from.Residence.to.Work,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Distance.from.Residence.to.Work") +
ylab('absenteeism')


#Age

ggplot(absent_backup, aes_string(x = absent_backup$Age,y = absent_data$Absenteeism.time.in.hours))
+

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Age") + ylab('absenteeism')


#Day of the week

ggplot(absent_backup, aes_string(x = absent_backup$Day.of.the.week,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Day.of.the.week") + ylab('absenteeism')


#Disciplinary failure

ggplot(absent_backup, aes_string(x = absent_backup$Disciplinary.failure,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Disciplinary.failure") +
ylab('absenteeism')


#Hit target

ggplot(absent_backup, aes_string(x = absent_backup$Hit.target,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Hit.target") + ylab('absenteeism')


#Month of absence
```

```
ggplot(absent_backup, aes_string(x = absent_backup$Month.of.absence,y =
absent_data$Absenteeism.time.in.hours)) +
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Month of absence") +
ylab('absenteeism')


#Pet
ggplot(absent_backup, aes_string(x = absent_backup$Pet,y = absent_data$Absenteeism.time.in.hours))
+
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Pet") + ylab('absenteeism')


#Seasons
ggplot(absent_backup, aes_string(x = absent_backup$Seasons,y =
absent_data$Absenteeism.time.in.hours)) +
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Seasons") + ylab('absenteeism')


#Service time
ggplot(absent_backup, aes_string(x = absent_backup$Service.time,y =
absent_data$Absenteeism.time.in.hours)) +
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Service.time") + ylab('absenteeism')


#Social drinker
ggplot(absent_backup, aes_string(x = absent_backup$Social.drinker,y =
absent_data$Absenteeism.time.in.hours)) +
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Social.drinker") + ylab('absenteeism')


#Transportation expense
ggplot(absent_backup, aes_string(x = absent_backup$Transportation.expense,y =
absent_data$Absenteeism.time.in.hours)) +
  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Transportation.expense") +
ylab('absenteeism')
```

```
#Work load Average/day

ggplot(absent_backup, aes_string(x = absent_backup$Work.load.Average.day.,y =
absent_data$Absenteeism.time.in.hours)) +

  geom_bar(stat="identity",fill =  "blue") + theme_bw() +  xlab("Work load Average/day") +
ylab('absenteeism')
```

Python Code

```
#Import required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import chi2_contingency

from fancyimpute import KNN

import os

from sklearn.metrics import r2_score

from scipy import stats

%matplotlib inline


#Set working directory

os.chdir("F:/Data Science/Edwisor Workspace/Project/Employee Absentism")

#Load data

absent_data = pd.read_excel("Absenteeism_at_work_Project.xls")

#Data Type of all the variables

absent_data.dtypes

#Check Number of Unique values in each variable

absent_data.nunique()

#Check dimension of data

absent_data.shape
```

```
# Separating Continuous and Categorical Variables

continuous_vars = ['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']


categorical_vars = ['ID','Reason for absence','Month of absence','Day of the week',

            'Seasons','Disciplinary failure', 'Education', 'Social drinker',

            'Social smoker', 'Pet', 'Son']

#Create dataframe with missing values present in each variable

missing_val = pd.DataFrame(absent_data.isnull().sum())

#View missing values

missing_val

#Reset Index

missing_val = missing_val.reset_index()

#Rename variable

missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})

#Calculate missing value percentage

missing_val['Missing_percentage'] = (missing_val['Missing_percentage']/len(absent_data))*100

#Sort missing values in descending order

missing_val = missing_val.sort_values('Missing_percentage', ascending = False).reset_index(drop = True)

missing_val

# Save output result as csv file

missing_val.to_csv("Missing_percentage_Python.csv", index = False)

# Droping observation in which "Absenteeism time in hours" has missing value

absent_data = absent_data.drop(absent_data[absent_data['Absenteeism time in hours'].isnull()].index,
axis=0)

# Checking for "Body mass index" column

# Actual value = 24

# Mean = 26.68
```

```
# Median = 25
# KNN = 24
#Check data
absent_data['Body mass index'].iloc[95]
#create missing value
absent_data['Body mass index'].iloc[95] = np.nan
#Apply KNN imputation algorithm
absent_data = pd.DataFrame(KNN(k = 3).fit_transform(absent_data), columns = absent_data.columns)
absent_data['Body mass index'].iloc[95]
#Check if all the missing values are imputed
absent_data.isnull().sum()
# Ploting BoxPlot of continuous variable Transportation expense
plt.boxplot(absent_data['Transportation expense'])
plt.xlabel("Transportation expense")
plt.title("BoxPlot of Transportation expense")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Service time
plt.boxplot(absent_data['Service time'])
plt.xlabel("Service time")
plt.title("BoxPlot of Service time")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Age
plt.boxplot(absent_data['Age'])
plt.xlabel("Age")
plt.title("BoxPlot of Age")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Work load Average/day
plt.boxplot(absent_data['Work load Average/day '])
plt.xlabel("Work load Average/day")
```

```
plt.title("BoxPlot of Work load Average/day")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Transportation expense
plt.boxplot(absent_data['Transportation expense'])
plt.xlabel("Transportation expense")
plt.title("BoxPlot of Transportation expense")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Transportation expense
plt.boxplot(absent_data['Hit target'])
plt.xlabel("Hit target")
plt.title("BoxPlot of Hit target")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Weight
plt.boxplot(absent_data['Weight'])
plt.xlabel("Weight")
plt.title("BoxPlot of Weight")
plt.ylabel('Values')
#Ploting BoxPlot of continuous variable Height
plt.boxplot(absent_data['Height'])
plt.xlabel("Height")
plt.title("BoxPlot of Height")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Body mass index
plt.boxplot(absent_data['Body mass index'])
plt.xlabel("Body mass index")
plt.title("BoxPlot of Body mass index")
plt.ylabel('Values')
# Ploting BoxPlot of continuous variable Absenteeism time in hours
plt.boxplot(absent_data['Absenteeism time in hours'])
```

```
plt.xlabel("Absenteeism time in hours")
plt.title("BoxPlot of Absenteeism time in hours")
plt.ylabel('Values')
#Storing variables with no outlier
no_outliers = ['Distance from Residence to Work', 'Weight', 'Body mass index']
#Loop to detect and remove outliers
for i in continuous_vars:
    if i in no_outliers:
        continue
    # Getting 75 and 25 percentile
    q75, q25 = np.percentile(absent_data[i], [75,25])
    #Calculating Interquartile range
    iqr = q75 - q25
    # Calculating upper and lower fence
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    #Replacing all the outliers value with NA
    absent_data.loc[absent_data[i]< min,i] = np.nan
    absent_data.loc[absent_data[i]> max,i] = np.nan

# Impute missing values created with KNN
absent_data = pd.DataFrame(KNN(k = 3).fit_transform(absent_data), columns = absent_data.columns)
# Check for missing values
absent_data.isnull().sum()
#Correlation plot for continuous variables
absent_data_corr = absent_data.loc[:,continuous_vars]

#Set width and height of the plot
f, ax = plt.subplots(figsize=(10, 10))
```

```
#correlation matrix
corr = absent_data_corr.corr()


#Plot heatmap
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
        cmap=sns.diverging_palette(220, 50, as_cmap=True),
        square=True, ax=ax, annot = True)
plt.plot()
#ANOVA test for categorical variables
for i in categorical_vars:
   f, p = stats.f_oneway(absent_data[i], absent_data["Absenteeism time in hours"])
   print("P value for "+str(i)+" is "+str(p))
# Droping the variables which has redundant information
absent_data = absent_data.drop(['Weight'], axis = 1)
absent_data.shape
# Updating the Continuous Variables and Categorical Variables after droping some variables
drop = ['Weight']
continuous_vars = [i for i in continuous_vars if i not in drop]
#Keeping a back up data
backup_data = absent_data.copy()
#Checking for normally distributed variable in data
for i in continuous_vars:
   if i == 'Absenteeism time in hours':
      continue
   sns.distplot(absent_data[i],bins = 'auto')
   plt.title("Checking data distribution for  "+str(i))
   plt.ylabel("Density")
   plt.show()
```

```
#Normalization
for i in continuous_vars:
    if i == 'Absenteeism time in hours':
        continue
    absent_data[i] = (absent_data[i] - absent_data[i].min())/(absent_data[i].max()-absent_data[i].min())
#Creating dummy data for categorical variables
absent_data = pd.get_dummies(data = absent_data, columns = categorical_vars)
#Take backup of data
absent_data_copy = absent_data.copy()
absent_data.shape
absent_data_copy.shape
#Divide data into train
#X denotes independent variables, y denotes dependent variable
# Use train_test_split sampling function
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( absent_data.iloc[:, absent_data.columns != 'Absenteeism time in hours'], absent_data.iloc[:, 8], test_size = 0.20)
#Import Decision Tree libraries
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

#Build model on top of training dataset
fit_DT = DecisionTreeRegressor(max_depth = 2).fit(X_train,y_train)

#Calculate RMSE for training data to check model overfit
pred_train = fit_DT.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculating RMSE for test data to check accuracy
```

```
pred_test = fit_DT.predict(X_test)
rmse_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_train))
print("Root Mean Squared Error For Test data = "+str(rmse_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
#Import Random Forest libraries
from sklearn.ensemble import RandomForestRegressor

#Build model on top of training dataset
fit_RF = RandomForestRegressor(n_estimators = 500).fit(X_train,y_train)

#Calculate RMSE for training data to model overfit
pred_train = fit_RF.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculate RMSE for test data to check accuracy
pred_test = fit_RF.predict(X_test)
rmse_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_train))
print("Root Mean Squared Error For Test data = "+str(rmse_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
# Importing Linear Regression libraries
from sklearn.linear_model import LinearRegression

# Build model on top of training dataset
fit_LR = LinearRegression().fit(X_train , y_train)
```

```
# Calculating RMSE for training data to check for over fitting
pred_train = fit_LR.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculating RMSE for test data to check accuracy
pred_test = fit_LR.predict(X_test)
rmse_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_train))
print("Root Mean Squared Error For Test data = "+str(rmse_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
#Import GradientBoosting library
from sklearn.ensemble import GradientBoostingRegressor

#Build model on top of train dataset
fit_GB = GradientBoostingRegressor().fit(X_train, y_train)

# Calculate RMSE for training data to check model over fitting
pred_train = fit_GB.predict(X_train)
rmse_for_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculating RMSE for test data to check accuracy
pred_test = fit_GB.predict(X_test)
rmse_for_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_for_train))
print("Root Mean Squared Error For Test data = "+str(rmse_for_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
#Store target variable in absent_target
```

```
absent_target = absent_data['Absenteeism time in hours']
#Drop target variable
absent_data.drop(['Absenteeism time in hours'], inplace = True, axis=1)
#Check structure
absent_data.shape
from sklearn.decomposition import PCA

#Convert data to numpy array
X = absent_data_copy.values

# Data has 129 variables so no of components of PCA = 129
pca = PCA(n_components=129)
pca.fit(X)

#The amount of variance each PCA explains
var= pca.explained_variance_ratio_

#Cumulative Variance
var1=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)

plt.plot(var1)
plt.show()
X.shape
#From the above plot selecting 45 components as it explains < 95%  variance
pca = PCA(n_components=45)

# Fitting the selected components to the data
pca.fit(X)
```

```
# Using train_test_split sampling function for test and train data split
X_train, X_test, y_train, y_test = train_test_split(X,absent_target, test_size=0.2)
X_test.shape
#Import Decision Tree libraries
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

#Build model on top of training dataset
fit_DT = DecisionTreeRegressor(max_depth = 2).fit(X_train,y_train)

#Calculate RMSE for training data to check model overfit
pred_train = fit_DT.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculating RMSE for test data to check accuracy
pred_test = fit_DT.predict(X_test)
rmse_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_train))
print("Root Mean Squared Error For Test data = "+str(rmse_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
#Import Random Forest libraries
from sklearn.ensemble import RandomForestRegressor

#Build model on top of training dataset
fit_RF = RandomForestRegressor(n_estimators = 500).fit(X_train,y_train)

#Calculate RMSE for training data to model overfit
pred_train = fit_RF.predict(X_train)
```

```
rmse_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculate RMSE for test data to check accuracy
pred_test = fit_RF.predict(X_test)
rmse_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_train))
print("Root Mean Squared Error For Test data = "+str(rmse_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
# Importing Linear Regression libraries
from sklearn.linear_model import LinearRegression

# Build model on top of training dataset
fit_LR = LinearRegression().fit(X_train , y_train)

# Calculating RMSE for training data to check for over fitting
pred_train = fit_LR.predict(X_train)
rmse_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculating RMSE for test data to check accuracy
pred_test = fit_LR.predict(X_test)
rmse_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_train))
print("Root Mean Squared Error For Test data = "+str(rmse_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
#Import GradientBoosting library
from sklearn.ensemble import GradientBoostingRegressor
```

```
#Build model on top of train dataset
fit_GB = GradientBoostingRegressor().fit(X_train, y_train)

# Calculate RMSE for training data to check model over fitting
pred_train = fit_GB.predict(X_train)
rmse_for_train = np.sqrt(mean_squared_error(y_train,pred_train))

# Calculating RMSE for test data to check accuracy
pred_test = fit_GB.predict(X_test)
rmse_for_test =np.sqrt(mean_squared_error(y_test,pred_test))

print("Root Mean Squared Error For Training data = "+str(rmse_for_train))
print("Root Mean Squared Error For Test data = "+str(rmse_for_test))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test,pred_test)))
#import libraries
from ggplot import *
#Education
ggplot(backup_data, aes(x='Education', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Education") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#ID
ggplot(backup_data, aes(x='ID', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("ID") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Social smoker
ggplot(backup_data, aes(x='Social smoker', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
```

```
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Social smoker") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Reason for absence
ggplot(backup_data, aes(x='Reason for absence', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Reason for absence") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") +
theme_bw()
#Distance from Residence to Work
ggplot(backup_data, aes(x='Distance from Residence to Work', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Distance from Residence to Work") + ylab("absenteeism in hr") + ggtitle("Absenteeism
Analysis") + theme_bw()
#Age
ggplot(backup_data, aes(x='Age', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Age") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Body mass index
ggplot(backup_data, aes(x='Body mass index', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Body mass index") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Day of the week
ggplot(backup_data, aes(x='Day of the week', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Day of the week") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
```

```
#Disciplinary failure
ggplot(backup_data, aes(x='Disciplinary failure', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Disciplinary failure") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Height
ggplot(backup_data, aes(x='Height', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Height") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Hit target
ggplot(backup_data, aes(x='Hit target', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Hit target") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Month of absence
ggplot(backup_data, aes(x='Month of absence', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Month") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Pet
ggplot(backup_data, aes(x='Pet', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Pet") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Seasons
ggplot(backup_data, aes(x='Seasons', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
```

```
   xlab("Seasons") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Service time
ggplot(backup_data, aes(x='Service time', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Service time") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Social drinker
ggplot(backup_data, aes(x='Social drinker', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Social drinker") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Son
ggplot(backup_data, aes(x='Son', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Son") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") + theme_bw()
#Transportation expense
ggplot(backup_data, aes(x='Transportation expense', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Transportation expense") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") +
theme_bw()
#Work load Average/day
ggplot(backup_data, aes(x='Work load Average/day ', y='Absenteeism time in hours')) +\
   geom_bar(fill= "SlateBlue") +\
   scale_color_brewer(type='diverging', palette=4) +\
   xlab("Work load Average/day ") + ylab("absenteeism in hr") + ggtitle("Absenteeism Analysis") +
theme_bw()
```

References

1. For Data Cleaning and Model Development - https://edwisor.com/career-data-scientist
2. For PCA - https://www.analyticsvidhya.com
3. For Visualization – Youtube.com