Employee Absenteeism

*Abhay Sharma*

25th October 2018

Table of Contents

[1. Introduction 2](#_Toc529099143)

[1.1 Problem Statement 2](#_Toc529099144)

[1.2 Data Sets 3](#_Toc529099145)

[2. Methodology 4](#_Toc529099146)

[2.1 Data Preprocessing 4](#_Toc529099147)

[2.1.1 Missing Value Analysis 5](#_Toc529099148)

[2.1.2 Outlier Analysis 5](#_Toc529099149)

[2.1.3 Data Visualization 6](#_Toc529099150)

[2.1.4 Univariate Analysis 7](#_Toc529099151)

[2.2 Feature Engineering 8](#_Toc529099152)

[3. Modelling 10](#_Toc529099153)

[3.1 Decision Tree 10](#_Toc529099154)

[3.2 Random Forest 11](#_Toc529099155)

[4. Conclusion 13](#_Toc529099156)

[4.1 Model Evaluation 13](#_Toc529099157)

[Appendix A 14](#_Toc529099158)

[R Code 14](#_Toc529099159)

[Python Code 22](#_Toc529099160)

# Introduction

## 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?

## Data Sets

Data is described upon parameters such as the Reason for Absence, various things involved, health issue or work load would be the reason. The table represents a sample of various fields available in the data.

**Table 1.1** Absenteeism at Work (Column 1-7)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

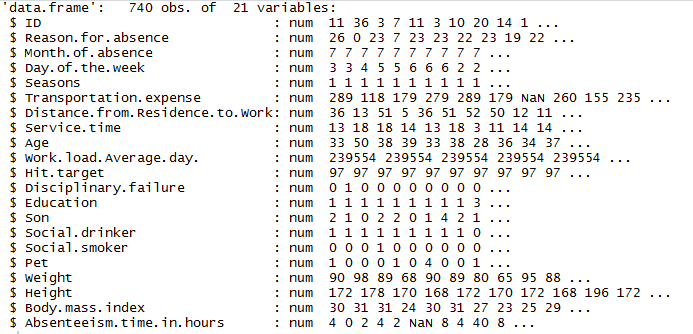
**Table 1.2** Absenteeism at Work (Column 8-14)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

As we can see in the table below we have the following 21 variables, using which we have to correctly predict the Employee Absenteeism time in hour for our target variable.

Summary of data is given below to know variables types and dimension of data.

**Fig 1.1** Summary of data

****

# Methodology

## 2.1 Data Preprocessing

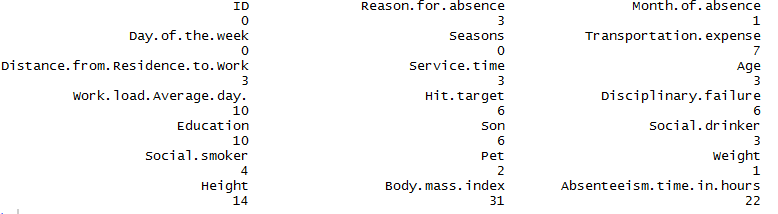
Data in real world is dirty it of no use until unless we apply data preprocessing on it. In other words, Pre- processing refers to the transformations applied to your data before feeding it to the algorithm. It’s a data mining technique which that involves transforming raw data into an understandable format or we can say that it prepares raw data to further processing. There are so many things that we do in data preprocessing like data cleaning, data integration, data transformation, or data reduction.

### 2.1.1 Missing Value Analysis

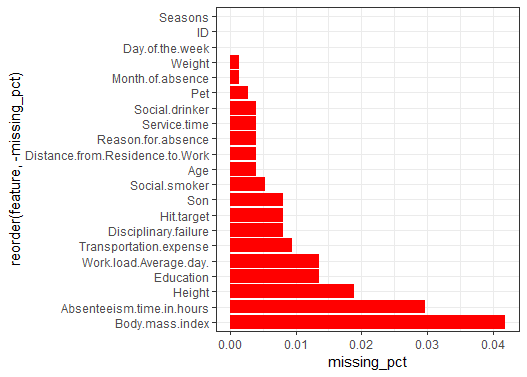
Missing Values Analysis is use to fill NULL values in data with some imputation techniques

But here in our Employee Absenteeism Data, we have null Values. By the way our data contain missing value. We will impute those values using KNN.

**Fig 2.1** Number of missing Values



**Fig 2.2** Visualization of missing Values



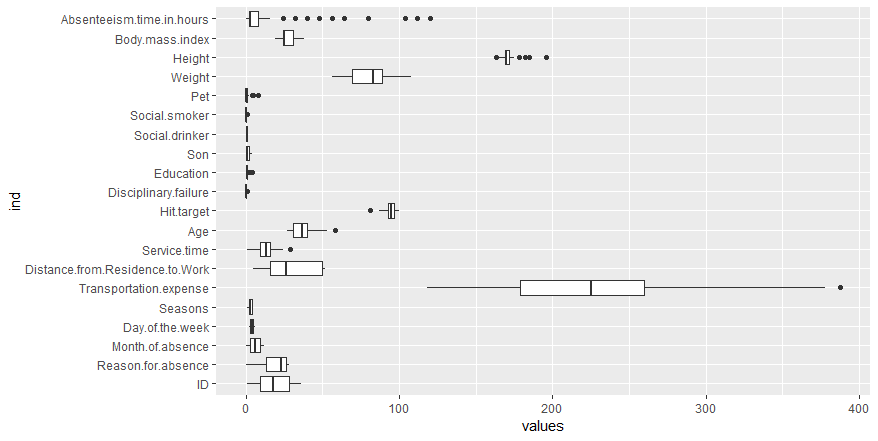
Ultimately, Figure 2.2 show bars which is sign of missing values, we are now 100% sure that our data need imputation of missing contain missing Values.

### 2.1.2 Outlier Analysis

The shown boxplot Fig: 2.3 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

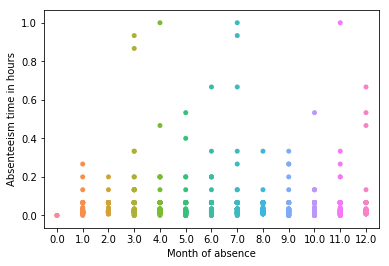
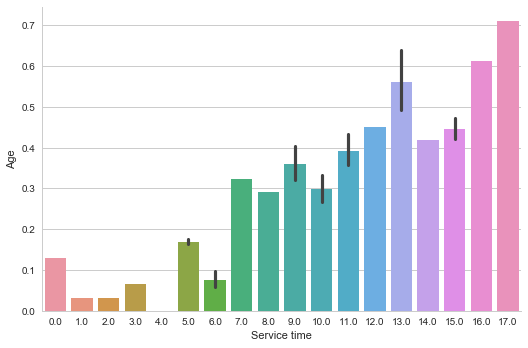
amount of outliers present in the target variable, which indicates a trend on Employee ' behavior, there can be pattern , we need to treat those outliers.

**Fig 2.3** Outlier Values

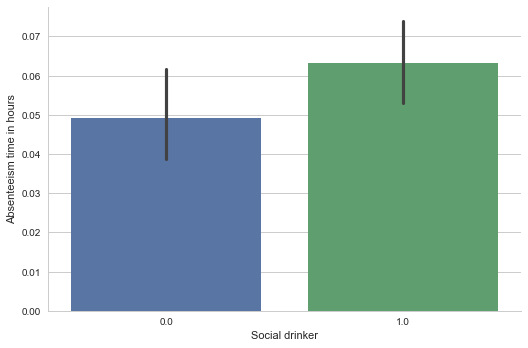
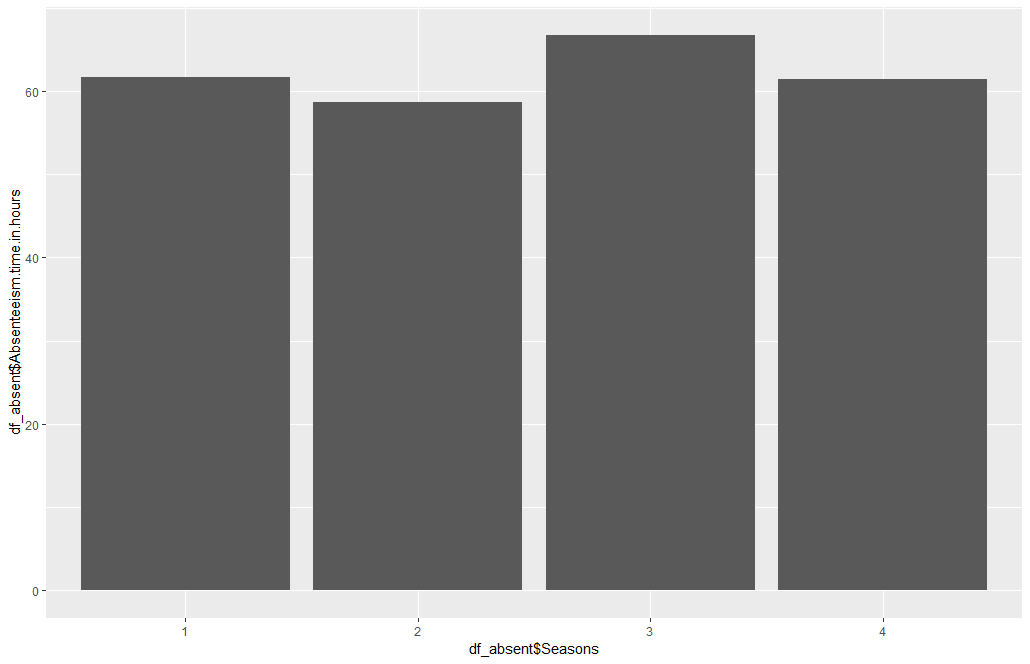


### 2.1.3 Data Visualization

Data Visualization is important concept it will help us to understand data, and will tell us answer of various questions also it will show relation between variables.Data visualization refers to the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization is an accessible way to see and understand trends, outliers, and patterns in data.

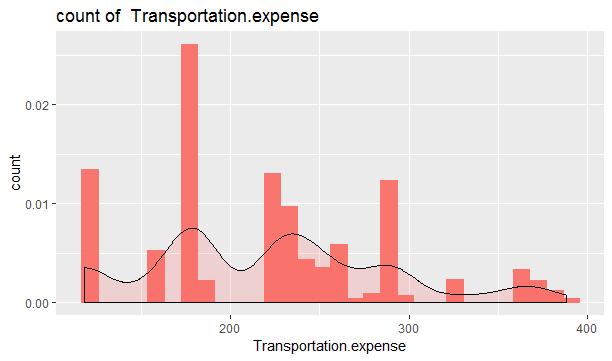
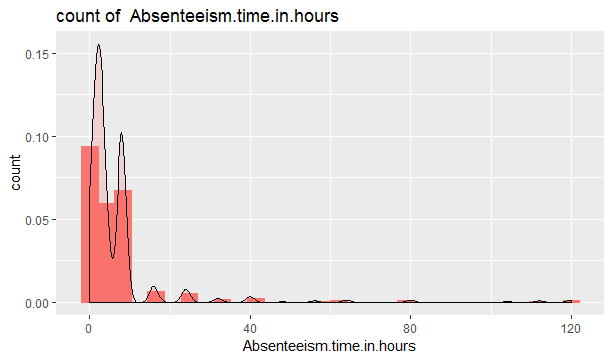
From above figure we can see that *Employees*  absent in several months contain pattern, Employees are of more age have relation with Service time.

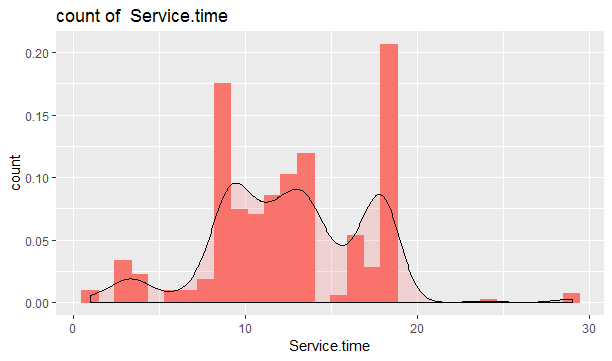
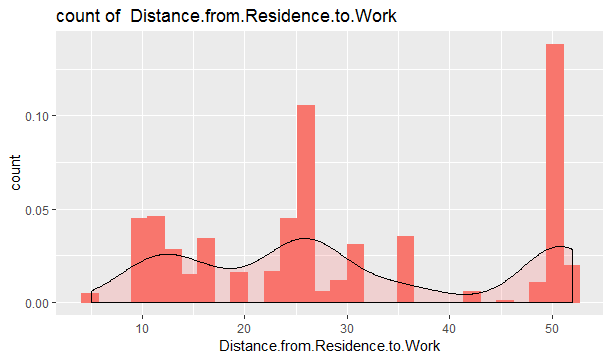
 

From above figure we can see that Drinkers took more leaves.

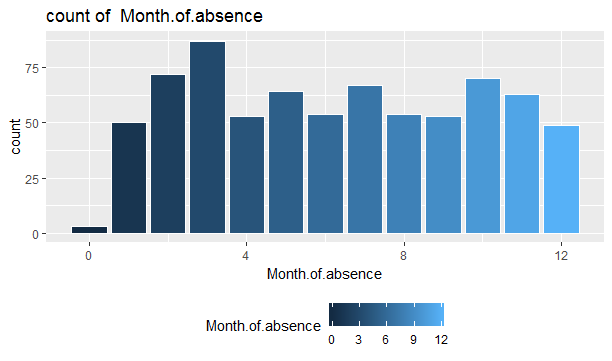
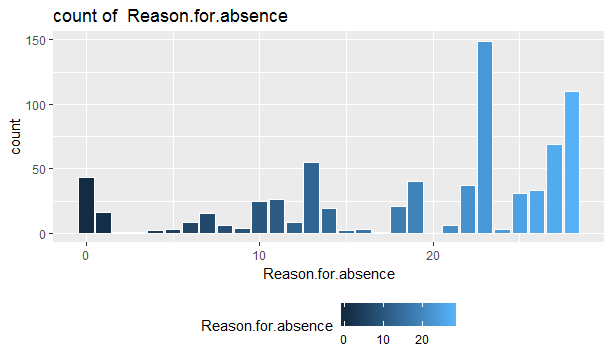
### 2.1.4 Univariate Analysis

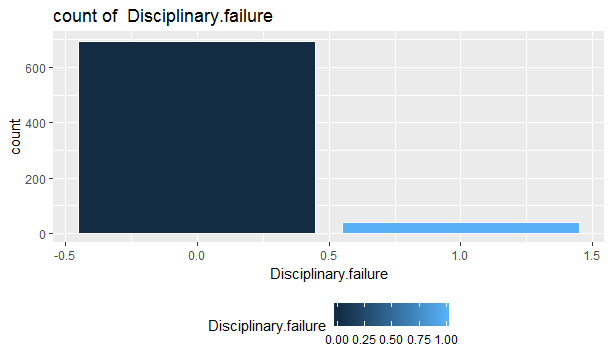
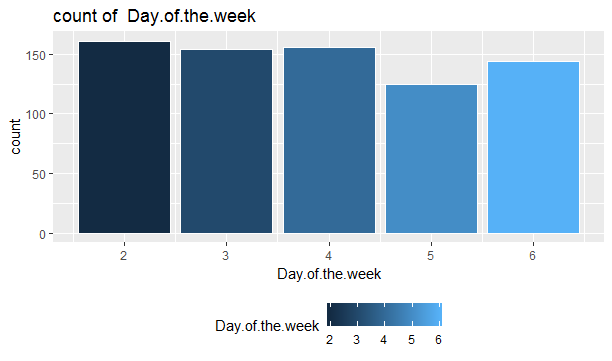
Univariate Analysis of Continuous Variable





Univariate Analysis of Categorical Variable



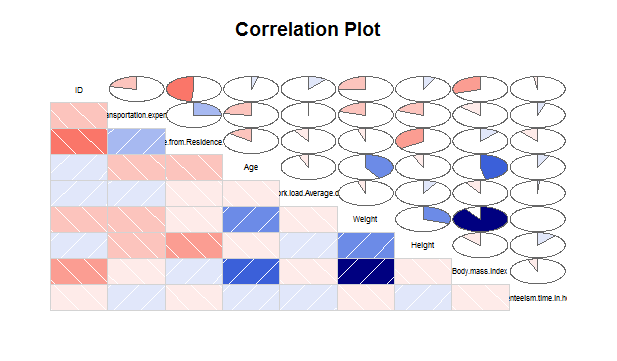


## 2.2 Feature Engineering

Feature Engineering 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. However, we

Considered taking correlation on the variables and make a heat map Fig: 2.5 to check relationships among the features and then dropping redundant variables.

**Fig 2.5** Correlation plot of variables





From these graph we can see that there are some variables which have collinearity problems or they are highly correlated.

1. The weight predictor is highly correlated to body mass index
2. On applying the chi square test, the p values of the following variables are found to be greater than 0.05, Hit.target, Education, Social.smoker, Pet.
3. One of the assumptions of logistic regression is that logistic regression requires there to be little or no multicollinearity among the independent variables. This means that the independent variables should not be too highly correlated with each other. Due to this assumption, one the predictors from each set was removed when logistic learner was trained.

# Modelling

Absenteeism at work is a regression problem. 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.

## 3.1 Decision Tree

Train data was divided into train dataset and validation set.

• Logistic regression models were trained on train dataset.

• Validation set and AIC score was used to select the best models out of all trained models.

• Final test and prediction was performed on test data which was provided separately.

R implementation:

#decision tree analysis

#rpart for regression

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

#Predict for new test cases

predictions\_DT = predict(fit, test[,-16])

#MAPE

#calculate MAPE

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

MAPE(test[,16], predictions\_DT)

Python implementation:

*# Decision Tree*

*#Decision tree for regression*

fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:9], train.iloc[:,9])

*#checking for any missing valuses that has leeked in*

np.where(Absenteeism\_at\_work.values >= np.finfo(np.float64).max)

np.isnan(Absenteeism\_at\_work.values.any())

test = test.fillna(train.mean())

*#Decision tree for regression*

fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:15], train.iloc[:,15])

Absenteeism\_at\_work.shape

*#Apply model on test data*

predictions\_DT = fit\_DT.predict(test.iloc[:,0:15])

**def** rmse(predictions, targets):

**return** np.sqrt(((predictions - targets) \*\* 2).mean())

rmse(test.iloc[:,15], predictions\_DT)

## 3.2 Random Forest

After decision tree, random forest was trained. It was implemented in both R and python. In both implementations random forest was first trained with default setting and the hyper parameters tuning was used to find the best parameters.

R Implementation:

#Random Forest

library(randomForest)

RF\_model = randomForest(Absenteeism.time.in.hours ~ ., train, importance = TRUE, ntree = 1000)

#Extract rules fromn random forest

#transform rf object to an inTrees' format

library(RRF)

library(inTrees)

treeList <- RF2List(RF\_model)

#Extract rules

exec = extractRules(treeList, train[,-16]) # R-executable conditions

ruleExec <- extractRules(treeList,train[,-16],digits=4)

#Make rules more readable:

readableRules = presentRules(exec, colnames(train))

readableRules[1:2,]

#Get rule metrics

ruleMetric = getRuleMetric(exec, train[,-16], train$Absenteeism.time.in.hours) # get rule metrics

#Predict test data using random forest model

RF\_Predictions = predict(RF\_model, test[,-16])

Python implementation:

*#Divide data into train and test*

X = Absenteeism\_at\_work.values[:, 0:15]

Y = Absenteeism\_at\_work.values[:,15]

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

*#Random Forest*

**from** **sklearn.ensemble** **import** RandomForestClassifier

RF\_model = RandomForestClassifier(n\_estimators = 20).fit(X\_train, y\_train)

RF\_Predictions = RF\_model.predict(X\_test)

# Conclusion

## 4.1 Model Evaluation

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,

**Using R,**

**Rmse** value applying decision tree, **0.222542**

This means that our predictions vary from the actual value by about 0.222542

**Rmse** value using random forest, **0.2065729**

This means that our predictions vary from the actual value by about 0.2065729

**Using python,**

**Rmse** value applying decision tree, **0.22594499**

This means that our predictions vary from the actual value by about 0.22594499

**Rmse** value using random forest**, 0.2076225**

This means that our predictions vary from the actual value by about 0.20762259

Hence comparing R and python, since the error rate of R is comparatively better, we consider the code of R

AND on comparing the values of decision tree and random forest, since the error rate of random forest is comparatively better, we consider the value of random forest.

**Hence, finally, we are accepting the random forest model of R, which has an RMSE value of 0.2065729, which is negligible.**

# Appendix A

## R Code

#remove all the objects stored

rm(list=ls())

#set current working directory

setwd("F:/Absenteeism")

library(xlsx) # Super simple excel reader

library(mice) # missing values imputation

library(naniar) # visualize missing values

library(dplyr)

library(corrplot)

library(ggplot2)

library(tidyverse)

library(randomForest)

library(caret)

library(data.table)

library(Boruta)

library(rpart)

## Read the data

df\_absent <- read.xlsx2("Absenteeism\_at\_work\_Project.xls", sheetIndex = 1, header = TRUE, colClasses = NA)

################################# exploratory data analysis#######################################

# it is found that Month.of.absence , there are 13 months present in data, hence to replace the false data by NA

df\_absent = transform(df\_absent, Month.of.absence =

ifelse(Month.of.absence == 0, NA, Month.of.absence ))

str(df\_absent)

#changing the contious variables to categorical variables for the ease of performance

df\_absent$Reason.for.absence = as.factor(df\_absent$Reason.for.absence)

df\_absent$Month.of.absence = as.factor(df\_absent$Month.of.absence)

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

df\_absent$Seasons = as.factor(df\_absent$Seasons)

df\_absent$Service.time = as.factor(df\_absent$Service.time)

df\_absent$Hit.target = as.factor(df\_absent$Hit.target)

df\_absent$Disciplinary.failure = as.factor(df\_absent$Disciplinary.failure)

df\_absent$Education = as.factor(df\_absent$Education)

df\_absent$Son = as.factor(df\_absent$Son)

df\_absent$Social.drinker = as.factor(df\_absent$Social.drinker)

df\_absent$Social.smoker = as.factor(df\_absent$Social.smoker)

df\_absent$Pet = as.factor(df\_absent$Pet)

df\_absent$Work.load.Average.day = as.numeric(df\_absent$Work.load.Average.day )

################################## Outlier analysis ###################################

outlierKD <- function(dt, var) {

var\_name <- eval(substitute(var),eval(dt))

na1 <- sum(is.na(var\_name))

m1 <- mean(var\_name, na.rm = T)

par(mfrow=c(2, 2), oma=c(0,0,3,0))

boxplot(var\_name, main="With outliers")

hist(var\_name, main="With outliers", xlab=NA, ylab=NA)

outlier <- boxplot.stats(var\_name)$out

mo <- mean(outlier)

var\_name <- ifelse(var\_name %in% outlier, NA, var\_name)

boxplot(var\_name, main="Without outliers")

hist(var\_name, main="Without outliers", xlab=NA, ylab=NA)

title("Outlier Check", outer=TRUE)

na2 <- sum(is.na(var\_name))

cat("Outliers identified:", na2 - na1, "n")

cat("Propotion (%) of outliers:", round((na2 - na1) / sum(!is.na(var\_name))\*100, 1), "n")

cat("Mean of the outliers:", round(mo, 2), "n")

m2 <- mean(var\_name, na.rm = T)

cat("Mean without removing outliers:", round(m1, 2), "n")

cat("Mean if we remove outliers:", round(m2, 2), "n")

response <- readline(prompt="Do you want to remove outliers and to replace with NA? [yes/no]: ")

if(response == "y" | response == "yes"){

dt[as.character(substitute(var))] <- invisible(var\_name)

assign(as.character(as.list(match.call())$dt), dt, envir = .GlobalEnv)

cat("Outliers successfully removed", "n")

return(invisible(dt))

} else{

cat("Nothing changed", "n")

return(invisible(var\_name))

}

}

outlierKD(df\_absent,Absenteeism.time.in.hours)# outliers detected and replaced by NA

outlierKD(df\_absent,Transportation.expense) #no outliers

outlierKD(df\_absent,Distance.from.Residence.to.Work) #no outliers

outlierKD(df\_absent,Service.time) #no outliers

outlierKD(df\_absent,Age) #no outliers

outlierKD(df\_absent,Work.load.Average.day.) # 1 found and replaced with NA

outlierKD(df\_absent,Hit.target) # 1 found and replaced with NA

#outlierKD(df\_absent,Son) # no outliers

#outlierKD(df\_absent,Pet) # no outliers

outlierKD(df\_absent,Weight) # no outliers

outlierKD(df\_absent,Height) # no outliers

outlierKD(df\_absent,Body.mass.index) #no outliers

################################## Missing value analysis ##################################

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

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

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

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

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

row.names(missing\_val) = NULL

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

write.csv(missing\_val, "Miising\_perc.csv", row.names = F)

#ggplot analysis

ggplot(data = missing\_val[1:3,], aes(x=reorder(Columns, -Missing\_percentage),y = Missing\_percentage))+

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

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

library(ggplot2)

#actual value =30

#df\_absent[1,20]

#df\_absent[1,20]= NA

# kNN Imputation=29.84314

#after various calculations, it is found that knn imputation method suits the best for the data. hence here we are applying knn imputation

library(DMwR)

df\_absent = knnImputation(df\_absent, k = 3)

sum(is.na(df\_absent))

################################### BoxPlots - Distribution and Outlier Check ##################################

numeric\_index = sapply(df\_absent,is.numeric) #selecting only numeric

numeric\_data = df\_absent[,numeric\_index]

cnames = colnames(numeric\_data)

library(ggplot2)

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = "responded"), data = subset(df\_absent))+

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=cnames[i],x="responded")+

ggtitle(paste("Box plot of responded for",cnames[i])))

}

################################## feature selection ##################################

library(corrgram)

## Correlation Plot - to check multicolinearity between continous variables

corrgram(df\_absent[,numeric\_index], order = F,

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

df\_absent$Absenteeism.time.in.hours = as.factor(df\_absent$Absenteeism.time.in.hours)

## Chi-squared Test of Independence-to check the multicolinearity between categorical variables

factor\_index = sapply(df\_absent,is.factor)

factor\_data = df\_absent[,factor\_index]

for (i in 1:12)

{

print(names(factor\_data)[i])

print(chisq.test(table(factor\_data$Absenteeism.time.in.hours,factor\_data[,i])))

}

df\_absent$Absenteeism.time.in.hours = as.numeric(df\_absent$Absenteeism.time.in.hours)

################################### Feature reduction ##################################

## Dimension Reduction

df\_absent = subset(df\_absent,

select = -c(Weight,Hit.target,Education,Social.smoker,Pet))

#Feature Scaling

#Normality check

qqnorm(df\_absent$Absenteeism.time.in.hours )

hist(df\_absent$Absenteeism.time.in.hours )

str(df\_absent)

#Normalisation

cnames = c("ID","Transportation.expense","Distance.from.Residence.to.Work","Height","Age","Work.load.Average.day","Body.mass.index",

"Absenteeism.time.in.hours")

for(i in cnames){

print(i)

df\_absent[,i] = (df\_absent[,i] - min(df\_absent[,i]))/

(max(df\_absent[,i] - min(df\_absent[,i])))

}

########################## Univariate Distribution and Analysis ###############################

# function for univariate analysis for continous variables

# function inpus:

# 1. dataset - input dataset

# 2. variable - variable for univariate analysis

# 3. variableName - variable title in string

# example. univariate\_analysis(df\_absent,Absenteeism.time.in.hours,

# "Absenteeism.time.in.hours")

univariate\_analysis <- function(dataset, variable,variableName){

var\_name = eval(substitute(variable), eval(dataset))

if(is.numeric(var\_name)){

print(summary(var\_name))

ggplot(df\_absent, aes(var\_name)) +

geom\_histogram(aes(y=..density..,binwidth=.5,colour="black", fill="white"))+

geom\_density(alpha=.2, fill="#FF6666")+

labs(x = variableName, y = "count") +

ggtitle(paste("count of ",variableName)) +

theme(legend.position = "null")

}else{

print("This is categorical variable.")

}

}

# function for univariate analysis for categorical variables

# function inpus:

# 1. dataset - input dataset

# 2. variable - variable for univariate analysis

# 3. variableName - variable title in string

# example. univariate\_analysis(df\_absent,ID,

# "ID")

univariate\_catogrical <- function(dataset,variable, variableName){

variable <- enquo(variable)

percentage <- dataset %>%

select(!!variable) %>%

group\_by(!!variable) %>%

summarise(n = n()) %>%

mutate(percantage = (n / sum(n)) \* 100)

print(percentage)

dataset %>%

count(!!variable) %>%

ggplot(mapping = aes\_(x = rlang::quo\_expr(variable),

y = quote(n), fill = rlang::quo\_expr(variable))) +

geom\_bar(stat = 'identity',

colour = 'white') +

labs(x = variableName, y = "count") +

ggtitle(paste("count of ",variableName)) +

theme(legend.position = "bottom") -> p

plot(p)

}

################################### Univariate analysis of continous variables ################################

univariate\_analysis(df\_absent,Absenteeism.time.in.hours,"Absenteeism.time.in.hours")

univariate\_analysis(df\_absent,Transportation.expense,"Transportation.expense")

univariate\_analysis(df\_absent,Distance.from.Residence.to.Work,

"Distance.from.Residence.to.Work")

univariate\_analysis(df\_absent,Service.time,"Service.time")

univariate\_analysis(df\_absent,Age,"Age")

univariate\_analysis(df\_absent,Work.load.Average.day ,"Work.load.Average.day ")

#univariate\_analysis(df\_absent,Hit.target ,"Hit.target")

univariate\_analysis(df\_absent,Son ,"Son")

#univariate\_analysis(df\_absent,Pet ,"Pet")

#univariate\_analysis(df\_absent,Weight ,"Weight")

univariate\_analysis(df\_absent,Height ,"Height")

univariate\_analysis(df\_absent,Body.mass.index ,"Body.mass.index")

################################### univariate analysis of categorical variables ##################################

univariate\_catogrical(df\_absent,ID,"Id")

univariate\_catogrical(df\_absent,Reason.for.absence,"Reason.for.absence")

univariate\_catogrical(df\_absent,Month.of.absence,"Month.of.absence")

univariate\_catogrical(df\_absent,Day.of.the.week,"Day.of.the.week")

univariate\_catogrical(df\_absent,Seasons,"Seasons")

univariate\_catogrical(df\_absent,Disciplinary.failure,"Disciplinary.failure")

univariate\_catogrical(df\_absent,Education,"Education")

univariate\_catogrical(df\_absent,Social.drinker,"Social.drinker")

univariate\_catogrical(df\_absent,Social.smoker,"Social.smoker")

################################### Sampling ##################################

##Systematic sampling

#Function to generate Kth index

sys.sample = function(N,n)

{

k = ceiling(N/n)

r = sample(1:k, 1)

sys.samp = seq(r, r + k\*(n-1), k)

}

lis = sys.sample(740, 300) #select the repective rows

# #Create index variable in the data

df\_absent$index = 1:740

# #Extract subset from whole data

systematic\_data = df\_absent[which(df\_absent$index %in% lis),]

################################### Model Development ##################################

#Clean the environment

library(DataCombine)

rmExcept("df\_absent")

#Divide data into train and test using stratified sampling method

set.seed(1234)

df\_absent$description = NULL

library(caret)

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

train = df\_absent[ train.index,]

test = df\_absent[-train.index,]

#load libraries

library(rpart)

#decision tree analysis

#rpart for regression

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

#Predict for new test cases

predictions\_DT = predict(fit, test[,-16])

#MAPE

#calculate MAPE

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

MAPE(test[,16], predictions\_DT)

#Random Forest

library(randomForest)

RF\_model = randomForest(Absenteeism.time.in.hours ~ ., train, importance = TRUE, ntree = 1000)

#Extract rules fromn random forest

#transform rf object to an inTrees' format

library(RRF)

library(inTrees)

treeList <- RF2List(RF\_model)

#Extract rules

exec = extractRules(treeList, train[,-16]) # R-executable conditions

ruleExec <- extractRules(treeList,train[,-16],digits=4)

#Make rules more readable:

readableRules = presentRules(exec, colnames(train))

readableRules[1:2,]

#Get rule metrics

ruleMetric = getRuleMetric(exec, train[,-16], train$Absenteeism.time.in.hours) # get rule metrics

#Predict test data using random forest model

RF\_Predictions = predict(RF\_model, test[,-16])

#rmse calculation

#install.packages("Metrics")

library(Metrics)

rmse(test$Absenteeism.time.in.hours, RF\_Predictions)

#rmse value for random forest is 0.2065729

rmse(test$Absenteeism.time.in.hours, predictions\_DT)

#rmse value for decision tree is 0.222542

## Python Code

*#Load libraries*

**import** **os**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**from** **scipy.stats** **import** chi2\_contingency

**import** **seaborn** **as** **sns**

**from** **random** **import** randrange, uniform

**from** **sklearn.cross\_validation** **import** train\_test\_split

**from** **sklearn.tree** **import** DecisionTreeRegressor

**from** **sklearn** **import** linear\_model

**from** **sklearn.cross\_validation** **import** train\_test\_split

*#Set working directory*

os.chdir("C:/Users/SHRAVYA/Desktop/edwisor/project 1")

*#Load data*

Absenteeism\_at\_work = pd.read\_csv("Absenteeism\_at\_work\_Project.csv")

*#----------------PRE PROCESSING-EXPLORATORY DATA ANALYSIS--------------#*

*#Exploratory Data Analysis*

Absenteeism\_at\_work['Reason for absence']=Absenteeism\_at\_work['Reason for absence'].astype(object)

Absenteeism\_at\_work['Month of absence']=Absenteeism\_at\_work['Month of absence'].astype(object)

Absenteeism\_at\_work['Day of the week']=Absenteeism\_at\_work['Day of the week'].astype(object)

Absenteeism\_at\_work['Seasons']=Absenteeism\_at\_work['Seasons'].astype(object)

Absenteeism\_at\_work['Service time']=Absenteeism\_at\_work['Service time'].astype(object)

Absenteeism\_at\_work['Hit target']=Absenteeism\_at\_work['Hit target'].astype(object)

Absenteeism\_at\_work['Disciplinary failure']=Absenteeism\_at\_work['Disciplinary failure'].astype(object)

Absenteeism\_at\_work['Education']=Absenteeism\_at\_work['Education'].astype(object)

Absenteeism\_at\_work['Son']=Absenteeism\_at\_work['Son'].astype(object)

Absenteeism\_at\_work['Social drinker']=Absenteeism\_at\_work['Social drinker'].astype(object)

Absenteeism\_at\_work['Social smoker']=Absenteeism\_at\_work['Social smoker'].astype(object)

Absenteeism\_at\_work['Pet']=Absenteeism\_at\_work['Pet'].astype(object)

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

*#Create dataframe with missing percentage*

missing\_val = pd.DataFrame(Absenteeism\_at\_work.isnull().sum())

*#Reset index*

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

*#Rename variable*

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

*#Calculate percentage*

missing\_val['Missing\_percentage'] = (missing\_val['Missing\_percentage']/len(Absenteeism\_at\_work))\*100

*#descending order*

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

*#save output results*

missing\_val.to\_csv("Missing\_perc.csv", index = **False**)

*#KNN imputation*

*#Assigning levels to the categories*

lis = []

**for** i **in** range(0, Absenteeism\_at\_work.shape[1]):

*#print(i)*

**if**(Absenteeism\_at\_work.iloc[:,i].dtypes == 'object'):

Absenteeism\_at\_work.iloc[:,i] = pd.Categorical(Absenteeism\_at\_work.iloc[:,i])

*#print(marketing\_train[[i]])*

Absenteeism\_at\_work.iloc[:,i] = Absenteeism\_at\_work.iloc[:,i].cat.codes

Absenteeism\_at\_work.iloc[:,i] = Absenteeism\_at\_work.iloc[:,i].astype('object')

lis.append(Absenteeism\_at\_work.columns[i])

*#replace -1 with NA to impute*

**for** i **in** range(0, Absenteeism\_at\_work.shape[1]):

Absenteeism\_at\_work.iloc[:,i] = Absenteeism\_at\_work.iloc[:,i].replace(-1, np.nan)

*#Impute with median*

Absenteeism\_at\_work['Absenteeism time in hours'] = Absenteeism\_at\_work['Absenteeism time in hours'].fillna(Absenteeism\_at\_work['Absenteeism time in hours'].median())

Absenteeism\_at\_work['Body mass index'] = Absenteeism\_at\_work['Body mass index'].fillna(Absenteeism\_at\_work['Body mass index'].median())

Absenteeism\_at\_work['Height'] = Absenteeism\_at\_work['Height'].fillna(Absenteeism\_at\_work['Height'].median())

Absenteeism\_at\_work['Weight'] = Absenteeism\_at\_work['Weight'].fillna(Absenteeism\_at\_work['Weight'].median())

Absenteeism\_at\_work['Pet'] = Absenteeism\_at\_work['Pet'].fillna(Absenteeism\_at\_work['Pet'].median())

Absenteeism\_at\_work['Social smoker'] = Absenteeism\_at\_work['Social smoker'].fillna(Absenteeism\_at\_work['Social smoker'].median())

Absenteeism\_at\_work['Social drinker'] = Absenteeism\_at\_work['Social drinker'].fillna(Absenteeism\_at\_work['Social drinker'].median())

Absenteeism\_at\_work['Son'] = Absenteeism\_at\_work['Son'].fillna(Absenteeism\_at\_work['Son'].median())

Absenteeism\_at\_work['Education'] = Absenteeism\_at\_work['Education'].fillna(Absenteeism\_at\_work['Education'].median())

Absenteeism\_at\_work['Disciplinary failure'] = Absenteeism\_at\_work['Disciplinary failure'].fillna(Absenteeism\_at\_work['Disciplinary failure'].median())

Absenteeism\_at\_work['Hit target'] = Absenteeism\_at\_work['Hit target'].fillna(Absenteeism\_at\_work['Hit target'].median())

Absenteeism\_at\_work['Age'] = Absenteeism\_at\_work['Age'].fillna(Absenteeism\_at\_work['Age'].median())

Absenteeism\_at\_work['Service time'] = Absenteeism\_at\_work['Service time'].fillna(Absenteeism\_at\_work['Service time'].median())

Absenteeism\_at\_work['Distance from Residence to Work'] = Absenteeism\_at\_work['Distance from Residence to Work'].fillna(Absenteeism\_at\_work['Distance from Residence to Work'].median())

Absenteeism\_at\_work['Transportation expense'] = Absenteeism\_at\_work['Transportation expense'].fillna(Absenteeism\_at\_work['Transportation expense'].median())

Absenteeism\_at\_work['Month of absence'] = Absenteeism\_at\_work['Month of absence'].fillna(Absenteeism\_at\_work['Month of absence'].median())

Absenteeism\_at\_work['Reason for absence'] = Absenteeism\_at\_work['Reason for absence'].fillna(Absenteeism\_at\_work['Reason for absence'].median())

Absenteeism\_at\_work['Work load Average/day '] = Absenteeism\_at\_work['Work load Average/day '].fillna(Absenteeism\_at\_work['Work load Average/day '].median())

Absenteeism\_at\_work.isnull().sum()

Absenteeism\_at\_work = Absenteeism\_at\_work.dropna(how='all')

Absenteeism\_at\_work.isnull().sum()

cnames = ["ID", "Transportation expense", "Distance from Residence to Work", "Age", "Height", "Body mass index", "Absenteeism time in hours"]

*#-------------------FEATURE SELECTION-------------------------#*

*##Correlation analysis*

*#Correlation plot*

df\_corr = Absenteeism\_at\_work.loc[:,cnames]

*#Set the width and hieght of the plot*

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

*#Generate correlation matrix*

corr = df\_corr.corr()

*#Plot using seaborn library*

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

square=**True**, ax=ax)

plt.savefig('correlation.png')

*#Chisquare test of independence*

*#Save categorical variables*

cat\_names = ["Reason for absence", "Month of absence", "Day of the week", "Seasons", "Service time", "Hit target", "Disciplinary failure", "Education", "Son", "Social drinker","Social smoker","Pet"]

*#loop for chi square values*

**for** i **in** cat\_names:

print(i)

chi2, p, dof, ex = chi2\_contingency(pd.crosstab(Absenteeism\_at\_work['Absenteeism time in hours'], Absenteeism\_at\_work[i]))

print(p)

Reason for absence

7.262525646531397e-126

Month of absence

2.5138924624334413e-08

Day of the week

0.003021081110471532

Seasons

1.0699164671285167e-06

Service time

0.0005117811788141375

Hit target

0.0011492200973353258

Disciplinary failure

2.811327292697691e-103

Education

0.966890372726654

Son

1.548005892620854e-08

Social drinker

0.0023832329972678858

Social smoker

0.5104529781136267

Pet

0.12306376012607578

*#--------------------FEATURE SCALING--------------------------#*

*#feature reduction*

Absenteeism\_at\_work = Absenteeism\_at\_work.drop(['Weight', 'Hit target', 'Education', 'Social smoker', 'Pet'], axis=1)

*#Nomalisation*

**for** i **in** cnames:

print(i)

Absenteeism\_at\_work[i] = (Absenteeism\_at\_work[i] - min(Absenteeism\_at\_work[i]))/(max(Absenteeism\_at\_work[i]) - min(Absenteeism\_at\_work[i]))

ID

Transportation expense

Distance from Residence to Work

Age

Height

Body mass index

Absenteeism time in hours

*#--------------------------DATA SAMPLING------------------------------#*

*#Divide data into train and test*

train, test = train\_test\_split(Absenteeism\_at\_work, test\_size=0.25, random\_state=42)

*#------------------------------MODELLING----------------------------------#*

*# Decision Tree*

*#Decision tree for regression*

fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:9], train.iloc[:,9])

*#checking for any missing valuses that has leeked in*

np.where(Absenteeism\_at\_work.values >= np.finfo(np.float64).max)

np.isnan(Absenteeism\_at\_work.values.any())

test = test.fillna(train.mean())

*#Decision tree for regression*

fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:15], train.iloc[:,15])

Absenteeism\_at\_work.shape

*#Apply model on test data*

predictions\_DT = fit\_DT.predict(test.iloc[:,0:15])

**def** rmse(predictions, targets):

**return** np.sqrt(((predictions - targets) \*\* 2).mean())

rmse(test.iloc[:,15], predictions\_DT)

*#rmse value using decision tree is 0.225944999314018*

*#Divide data into train and test*

X = Absenteeism\_at\_work.values[:, 0:15]

Y = Absenteeism\_at\_work.values[:,15]

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

*#Random Forest*

**from** **sklearn.ensemble** **import** RandomForestClassifier

RF\_model = RandomForestClassifier(n\_estimators = 20).fit(X\_train, y\_train)

RF\_Predictions = RF\_model.predict(X\_test)

*#---------------------PLOTS OF VARIABLES-------------------------------#*

*#plots*

**import** **matplotlib** **as** **mpl**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

sns.set(style="whitegrid", color\_codes=**True**)

np.random.seed(sum(map(ord, "categorical")))

Absenteeism\_at\_work.columns

sns.stripplot(x="Body mass index", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Body mass index.png')

sns.stripplot(x="Reason for absence", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Reason for absence.png')

sns.stripplot(x="Month of absence", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Month of absence.png')

sns.stripplot(x="Day of the week", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Day of the week.png')

sns.stripplot(x="Seasons", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Seasons.png')

sns.stripplot(x="Transportation expense", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Transportation expense.png')

sns.stripplot(x="Distance from Residence to Work", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Distance from Residence to Work.png')

sns.stripplot(x="Service time", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Service time.png')

sns.stripplot(x="Age", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Age.png')

sns.stripplot(x="Disciplinary failure", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Disciplinary failure.png')

sns.stripplot(x="Son", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Son.png')

sns.stripplot(x="Social drinker", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Social drinker.png')

sns.stripplot(x="Height", y="Absenteeism time in hours", data=Absenteeism\_at\_work);

plt.savefig('Height.png')