Employee Absenteeism

Abhay Sharma

25th October 2018

Table of 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 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

```
'data.frame':	740 obs. of  21 variables:
 $ ID                             : num  11 36 3 7 11 3 10 20 14 1 ...
 $ Reason.for.absence             : num  26 0 23 7 23 23 22 23 19 22 ...
 $ Month.of.absence               : num  7 7 7 7 7 7 7 7 7 7 ...
 $ Day.of.the.week                : num  3 3 4 5 5 6 6 6 2 2 ...
 $ Seasons                        : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Transportation.expense         : num  289 118 179 279 289 179 NaN 260 155 235 ...
 $ Distance.from.Residence.to.Work: num  36 13 51 5 36 51 52 50 12 11 ...
 $ Service.time                   : num  13 18 18 14 13 18 3 11 14 14 ...
 $ Age                            : num  33 50 38 39 33 38 28 36 34 37 ...
 $ Work.load.Average.day.         : num  239554 239554 239554 239554 239554 ...
 $ Hit.target                     : num  97 97 97 97 97 97 97 97 97 97 ...
 $ Disciplinary.failure           : num  0 1 0 0 0 0 0 0 0 0 ...
 $ Education                      : num  1 1 1 1 1 1 1 1 1 3 ...
 $ Son                            : num  2 1 0 2 2 0 1 4 2 1 ...
 $ Social.drinker                 : num  1 1 1 1 1 1 1 1 1 0 ...
 $ Social.smoker                  : num  0 0 0 1 0 0 0 0 0 0 ...
 $ Pet                            : num  1 0 0 0 1 0 4 0 0 1 ...
 $ Weight                         : num  90 98 89 68 90 89 80 65 95 88 ...
 $ Height                         : num  172 178 170 168 172 170 172 168 196 172 ...
 $ Body.mass.index                : num  30 31 31 24 30 31 27 23 25 29 ...
 $ Absenteeism.time.in.hours      : num  4 0 2 4 2 NaN 8 4 40 8 ...
```

2. 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.

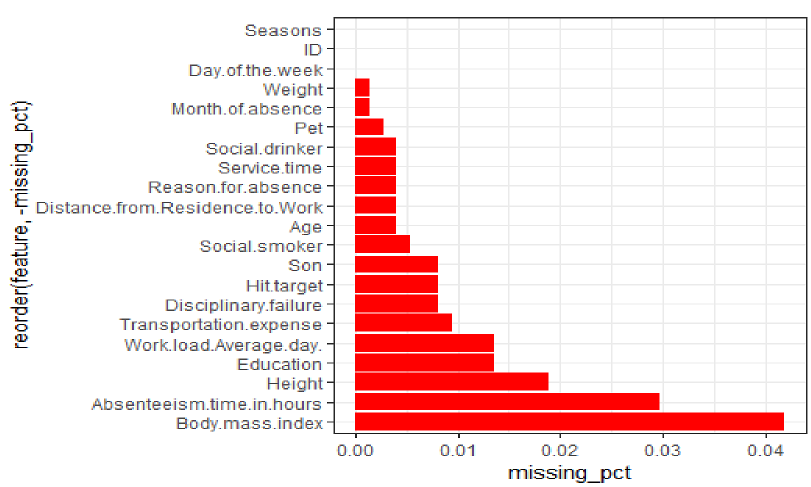
2.1.1 Missing Value Analysis

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

```
                           ID              Reason.for.absence              Month.of.absence
                            0                               3                             1
              Day.of.the.week                         Seasons        Transportation.expense
                            0                               0                             7
Distance.from.Residence.to.Work                  Service.time                           Age
                            3                               3                             3
           Work.load.Average.day.                  Hit.target            Disciplinary.failure
                           10                               6                             6
                    Education                             Son                Social.drinker
                           10                               6                             3
                 Social.smoker                            Pet                        Weight
                            4                               2                             1
                       Height                 Body.mass.index     Absenteeism.time.in.hours
                           14                              31                            22
```

Fig 2.2 Visualization of missing Values

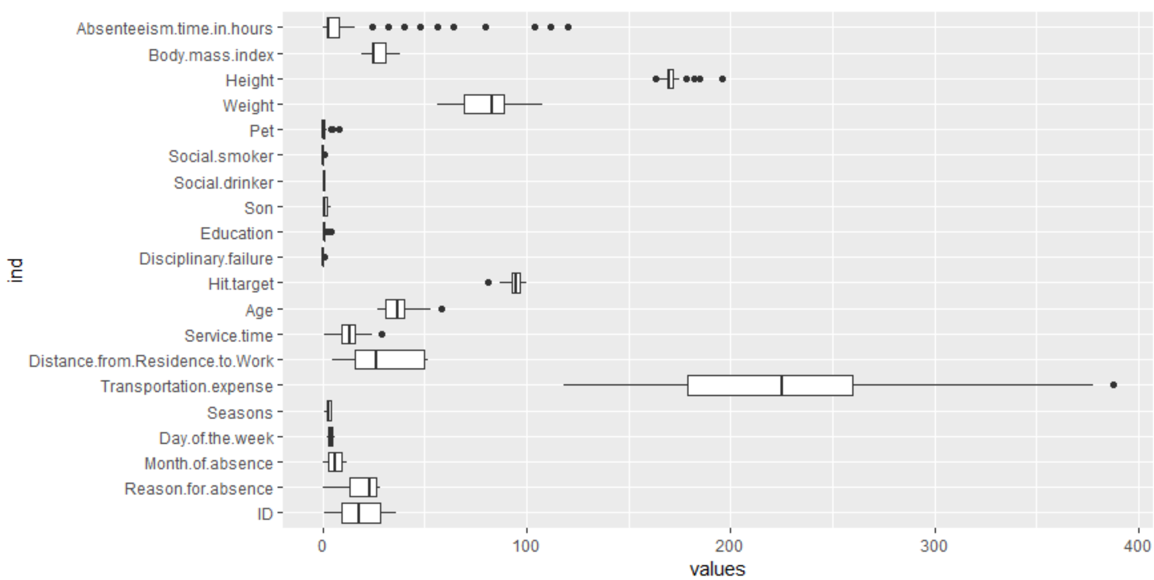


Ultimately, Figure 2.2 show bars which is sign of missing values, we are now 100% sure that our data need imputation of missing contain missing Values.

2.1.2 Outlier Analysis

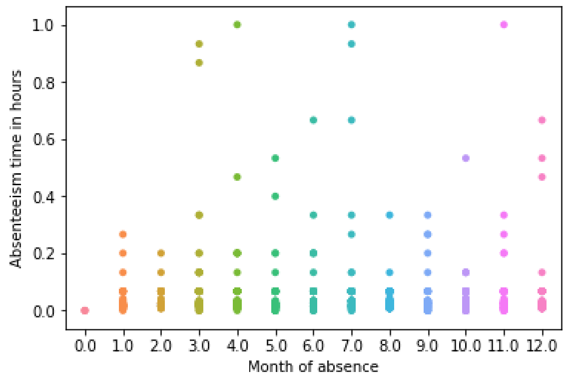
The shown boxplot Fig: 2.3 refers outliers on the predictors variables, we can see various outliers associated with the features. Even though, the data has considerable amount of outliers, the approach is to retain every outlier and grab respective behavior of all employees. As shown there are significant 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.

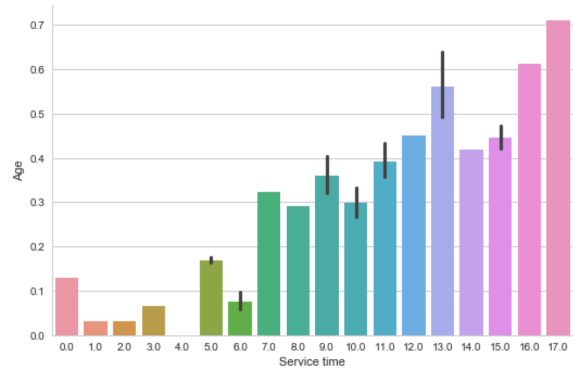
Fig 2.3 Outlier Values



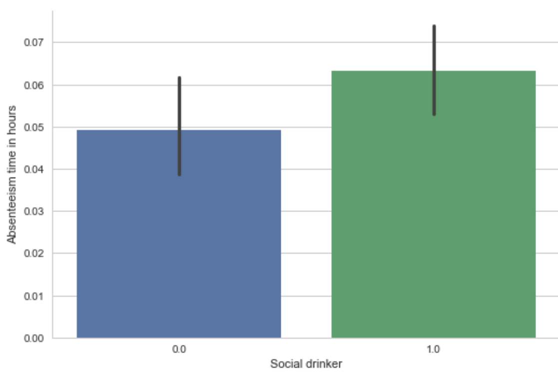
2.1.3 Data Visualization

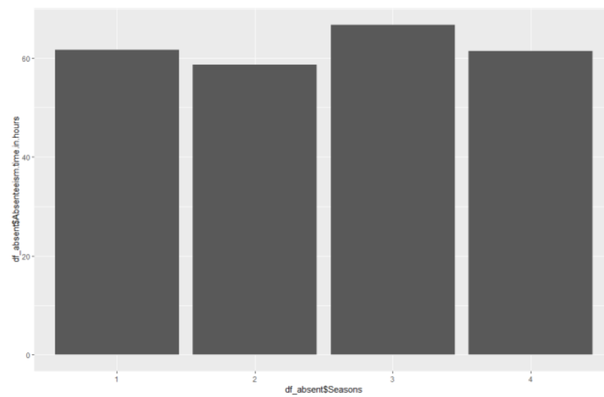
Data Visualization is important concept it will help us to understand data, and will tell us answer of various questions also it will show relation between variables. Data visualization refers to the graphical representation of information and data. By using visual elements like charts, graphs, and maps,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

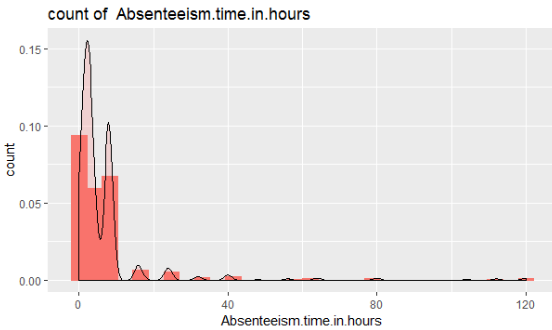


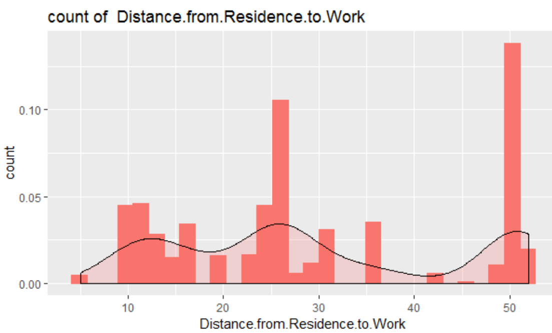


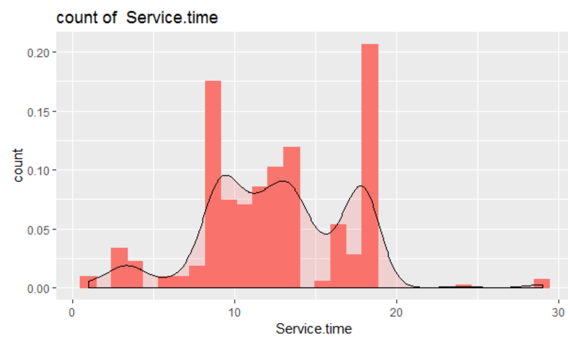
From above figure we can see that Drinkers took more leaves.

2.1.4 Univariate Analysis

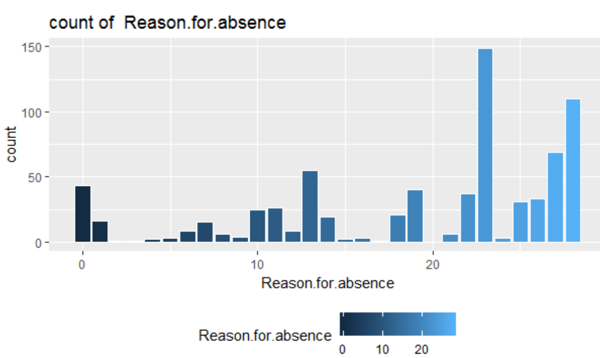
Univariate Analysis of Continuous Variable

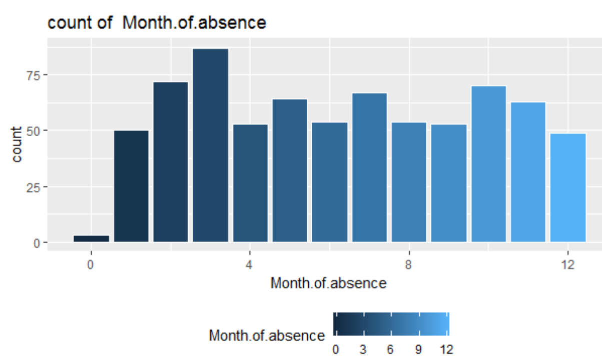


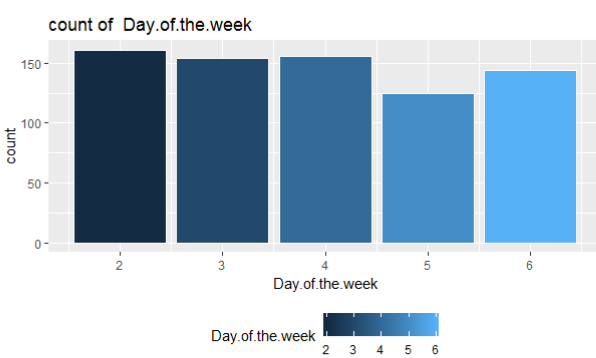


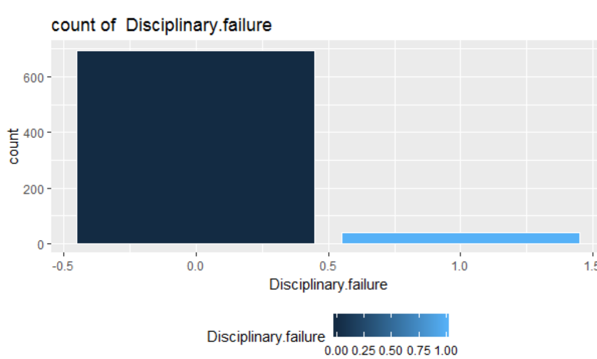


Univariate Analysis of Categorical Variable





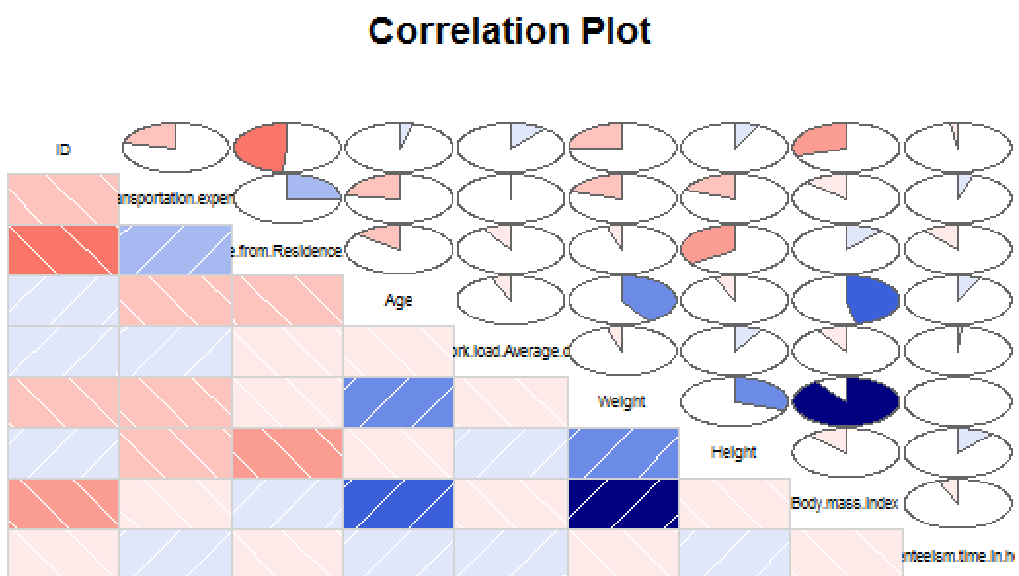




2.2 Feature Engineering

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.

3. 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
```

```python
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:

```r
#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)
```

4. 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.Averag
e.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

```python
#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 smoke
r'].median())
Absenteeism_at_work['Social drinker'] = Absenteeism_at_work['Social drinker'].fillna(Absenteeism_at_work['Social drinke
r'].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['Dis
ciplinary 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'].fill
na(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['Mo
nth of absence'].median())
Absenteeism_at_work['Reason for absence'] = Absenteeism_at_work['Reason for absence'].fillna(Absenteeism_at_work['R
eason for absence'].median())
Absenteeism_at_work['Work load Average/day '] = Absenteeism_at_work['Work load Average/day '].fillna(Absenteeism_a
t_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", "Abse
nteeism 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", "Disci
plinary 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)
```

```python
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')
```