Project Report - Employee Absenteeism

Gourav Gangani

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

1.2.1. Dimension

740 Observation and 21 Variables. *Absenteeism time in hours* is the Target Variable.

1.2.2. Variables

1. Individual identification (ID)

2. Reason for absence (ICD). Absences attested by the International Code of Diseases (ICD) stratified into 21categories (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)

1.2.3. Data Snippet

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
11	26	7	3	1	289	36	13	33	239,554	97
36	0	7	3	1	118	13	18	50	239,554	97
3	23	7	4	1	179	51	18	38	239,554	97
7	7	7	5	1	279	5	14	39	239,554	97
11	23	7	5	1	289	36	13	33	239,554	97
3	23	7	6	1	179	51	18	38	239,554	97

Disciplinary.failure	Education	Son	Social.drinker	Social.smoker	Pet	Weight	Height	Body.mass.index	Absenteeism.time.in.hours
0	1	2	1	0	1	90	172	30	4
1	1	1	1	0	0	98	178	31	0
0	1	0	1	0	0	89	170	31	2
0	1	2	1	1	0	68	168	24	4
0	1	2	1	0	1	90	172	30	2
0	1	0	1	0	0	89	170	31	NA

1.2.4. Data Summary

Data Type Distribution

```
1.ID                              : int
2.Reason.for.absence              : int
3.Month.of.absence                : int
4.Day.of.the.week                 : int
5.Seasons                         : int
6.Transportation.expense          : int
7.Distance.from.Residence.to.Work: int
8.Service.time                    : int
9.Age                             : int
10.Work.load.Average.day          : Factor
11.Hit.target                     : int
12.Disciplinary.failure           : int
13.Education                      : int
14.Son                            : int
15.Social.drinker                 : int
16.Social.smoker                  : int
17.Pet                            : int
18.Weight                         : int
19.Height                         : int
20.Body.mass.index                : int
21.Absenteeism.time.in.hours      : int
```

Total Null Values:

	colSums(is.na(absentData))
ID	0
Reason.for.absence	46
Month.of.absence	4
Day.of.the.week	0
Seasons	0
Transportation.expense	7
Distance.from.Residence.to.Work	3
Service.time	3
Age	3
Work.load.Average.day	10
Hit.target	6
Disciplinary.failure	6
Education	10
Son	6
Social.drinker	3
Social.smoker	4
Pet	2
Weight	1
Height	14
Body.mass.index	31
Absenteeism.time.in.hours	22

Total Missing Values = 181

2. Methodology

2.1. Data Pre-Processing

In machine learning it is required to first pre-process the data before feeding the same into the algorithms. In order to conduct this, we need to go down through different methods such as Data Exploration, Data Visualization using graphs and plots, Data Cleaning to better understand the data.

2.2. Missing Value Analysis

Missing Values as the name suggest refers to unavailability of values in a variable. There can be various reasons behind missing values. For instance, if we talk about customer information on any social account then there would be lots of variables/observations where we can see missing values. How, few customers may not be comfortable to share their private information such as Date of birth, Phone Number etc and this is true too hence results to missing values in the dataset. There can be technical reasons behind missing values such the machine or software failed to record particular field while data entry.

As per our observation, we have found 181 missing values occurred among the various columns. Below is the table to depict the missing values in the Employee Absenteeism Dataset.

	colSums(is.na(absentData))
ID	0
Reason.for.absence	46
Month.of.absence	4
Day.of.the.week	0
Seasons	0
Transportation.expense	7
Distance.from.Residence.to.Work	3
Service.time	3
Age	3
Work.load.Average.day	10
Hit.target	6
Disciplinary.failure	6
Education	10
Son	6
Social.drinker	3
Social.smoker	4
Pet	2
Weight	1
Height	14
Body.mass.index	31
Absenteeism.time.in.hours	22

In order to infer the Missing Values we have used KNN method.

2.3. Outlier Analysis

Outlier analysis is a technique to find out the extreme values which are distant from the mean of the population. These values have the capability to inflate the Inter and Intra variance among the variables.

There can be multiple ways to treat outliers such as

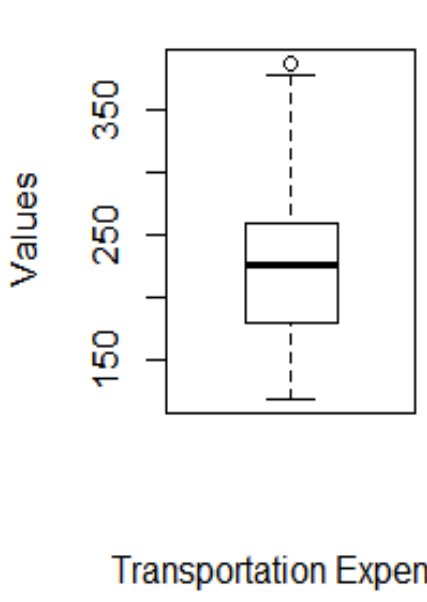
a) Replacing the Outliers by Null Values: In this technique we first finds out the outliers and then replace them by Null values. These null values can then be replaced by the Missing Value Imputation.

b) Remove the Outliers: In this technique we can first find out the outliers and then create a benchmark using Inter Quartile Range(IQR). This benchmark can be used to filter the outliers.

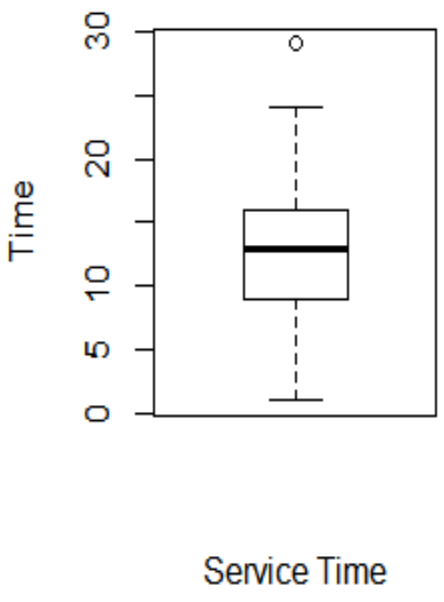
One thing which we need to understand is, whether the Outliers in the dataset are due to some error or real values. For instance, if we talk about Employee Dataset we will find majority of the salaries doesn't have much difference, but few of the salaries significantly way higher than the remaining ones. These salaries are of employees in the Top Management. Since it is real data we should not remove it. So in this case it is better to apply scaling method to normalize the data within 0 to 1 range.

In this project we have applied both the techniques for treating outliers.

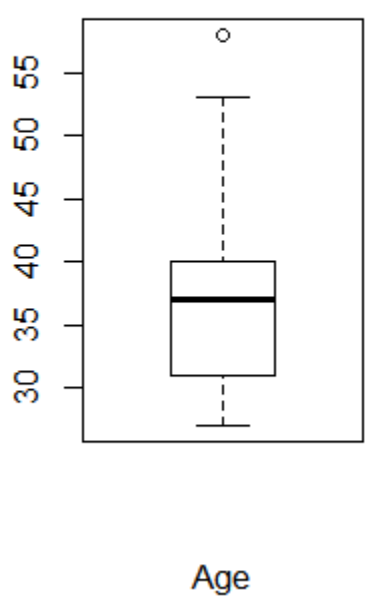
Outliers detected on few of the variables shown below:



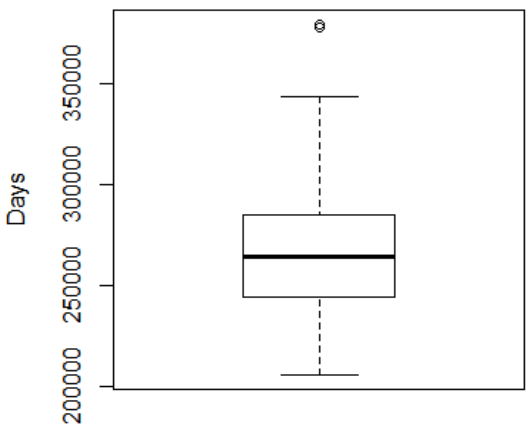
Transportation Expense



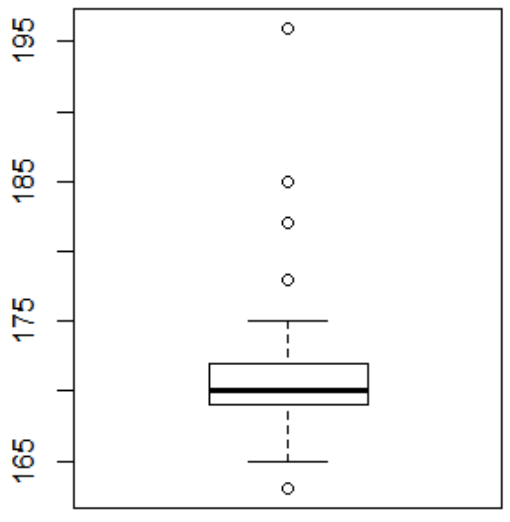
Service Time



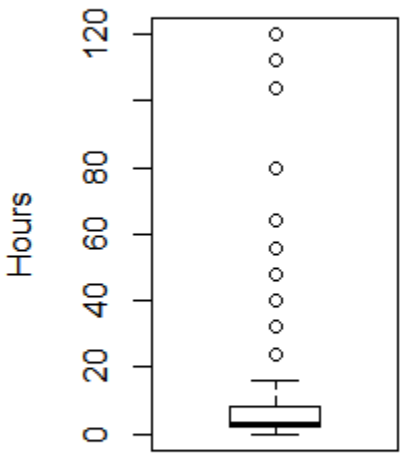
Age



Work Load



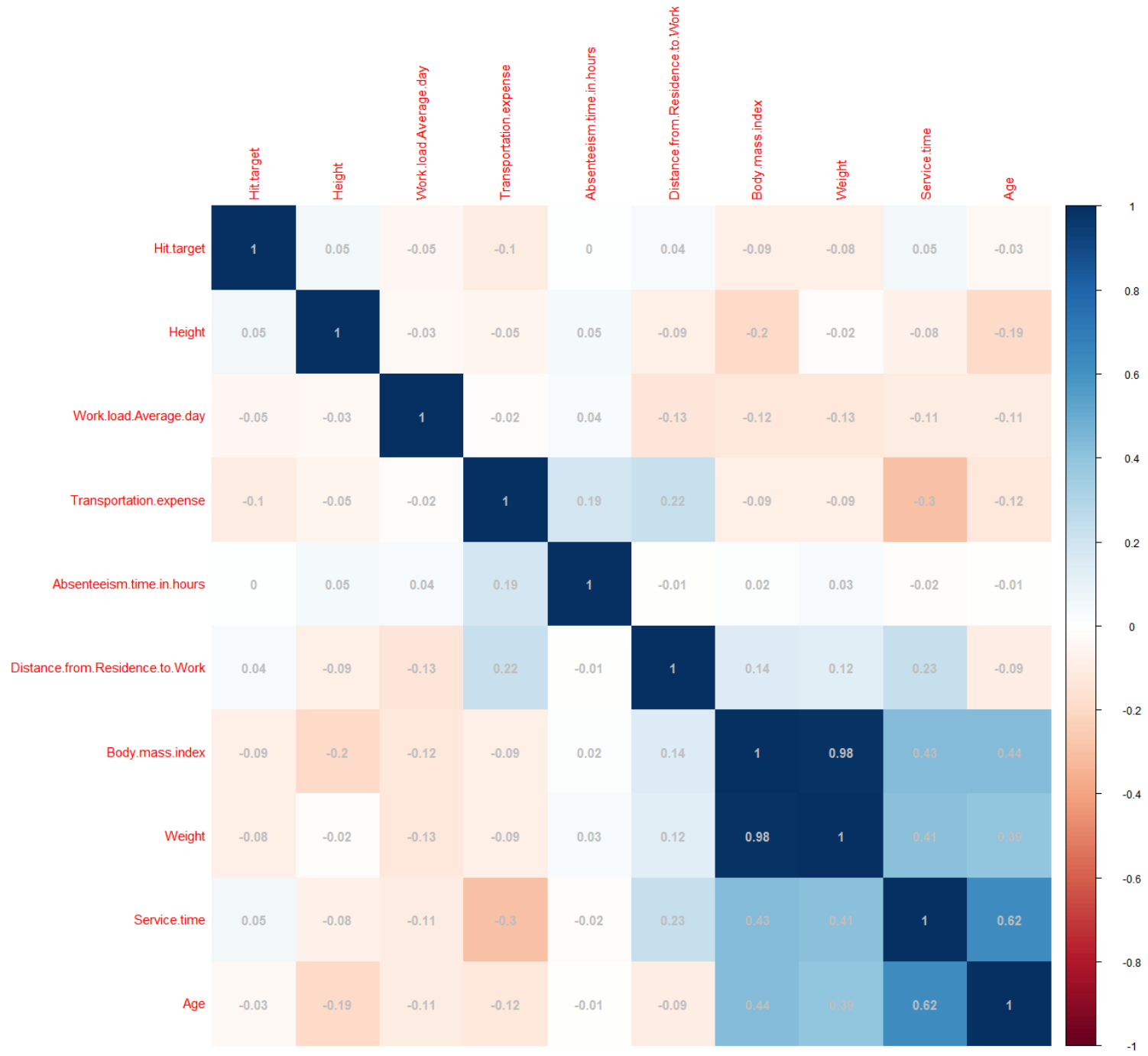
Height



Absenteeism

2.4. Feature Selection

Feature selection plays an important role in composing a subset of data with the most significant variables. It also helps us to understand the relationship among the variables. Correlation and Multi-Collinearity are the main techniques which we generally performs on the data. Below is the figure to show the correlation score between the variables.



With the help of correlation plot we can see that Body Mass Index and Weight has high correlation near to 1. Which determines that we can drop one of the variable from the dataset. Similarly Service time and Age has a significant correlation too.

2.5. Feature Scaling

While exploring the data we observed that there is a significant variability among the variables which can impact the models. For instance if we compare the mean of Work Load Average Day and Service Time, we will find a huge difference. This difference can impact the modeling, while mapping them in mathematical algorithms.

In order to treat the variability we use some standardization methods which brings normality in the data. After scaling, the data turns up into 0 to 1 range. We have used the normalization technique.

2.6. Principal Component Analysis

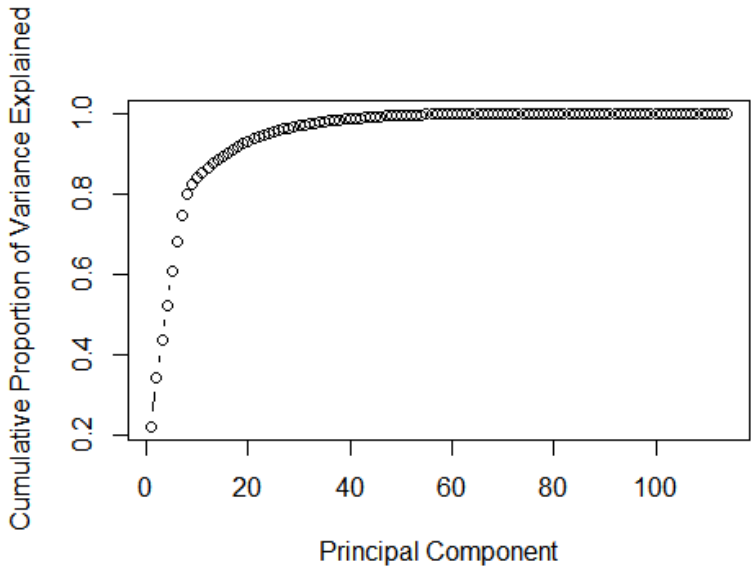
Principal Component Analysis is a technique which extracts the most significant variables from a high dimensional dataset and helps in achieving better predictions. Since it only works on the numeric data, so the categorical variables needs to be converted into continuous variables.

Motivation for PCA:

1. We observed that there is significant variability among the variables.

2. To treat the multicollinearity.

3. To get the better accuracy in the model.



With the help of graph we can observe that 95% of variability can be explained by 45 variables among the 116 variables. Hence we can only select 45 variables.

3. Modelling

Once we are done with Data-Preprocessing, we feed the processed data into the models. Since our target variable is continuous in nature we will be using regression models such as

a) Linear Regression

b) Decision Tree

c) Random Forest

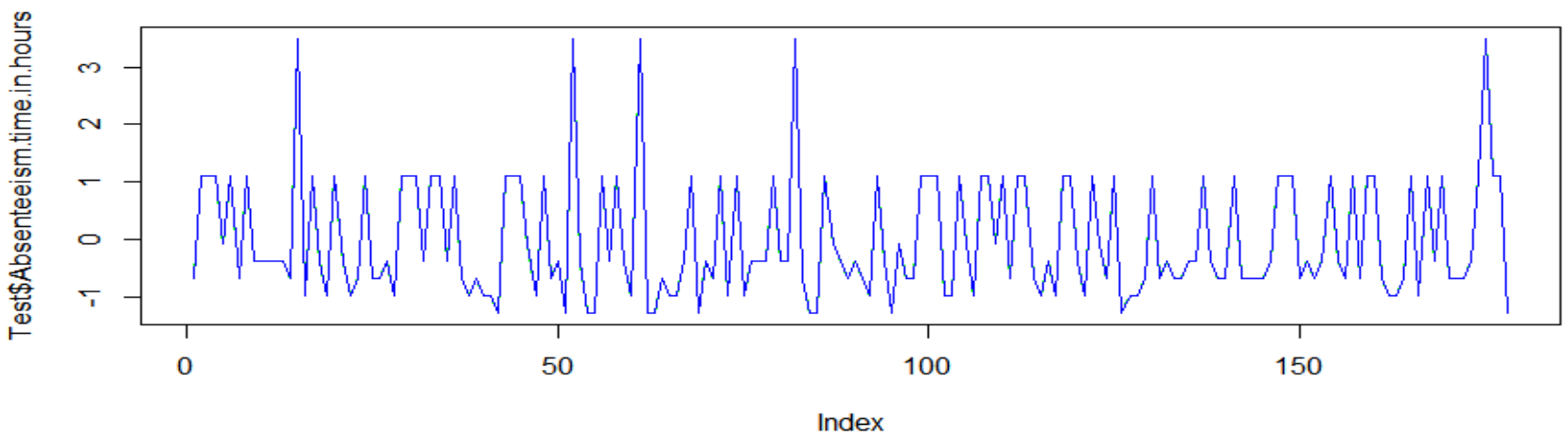
3.1. Linear Regression

Multiple linear regression, which is probably the best known statistical model. In simple terms, the dependent variable is assumed to be a linear function of several independent variables (predictors), where each of them has a weight (regression coefficient) that is expected to be statistically significant in the final model. The results are usually highly interpretable and, provided some conditions are met, have good accuracy.

Results:

Platform	Before PCA - RMSE	Before PCA – R-Squared	After PCA - RMSE	After PCA – R Squared
R	0.45	0.51	0.28	0.91
Python	2.89	0.27	8.633839188446864e-12	1.0

Graph between Actual and Predicted Values (obtained from R):



Blue line shows the predicted values whereas the green line shows the actual values. In our case blue line has completely overridden the green line which mean excellent accuracy.

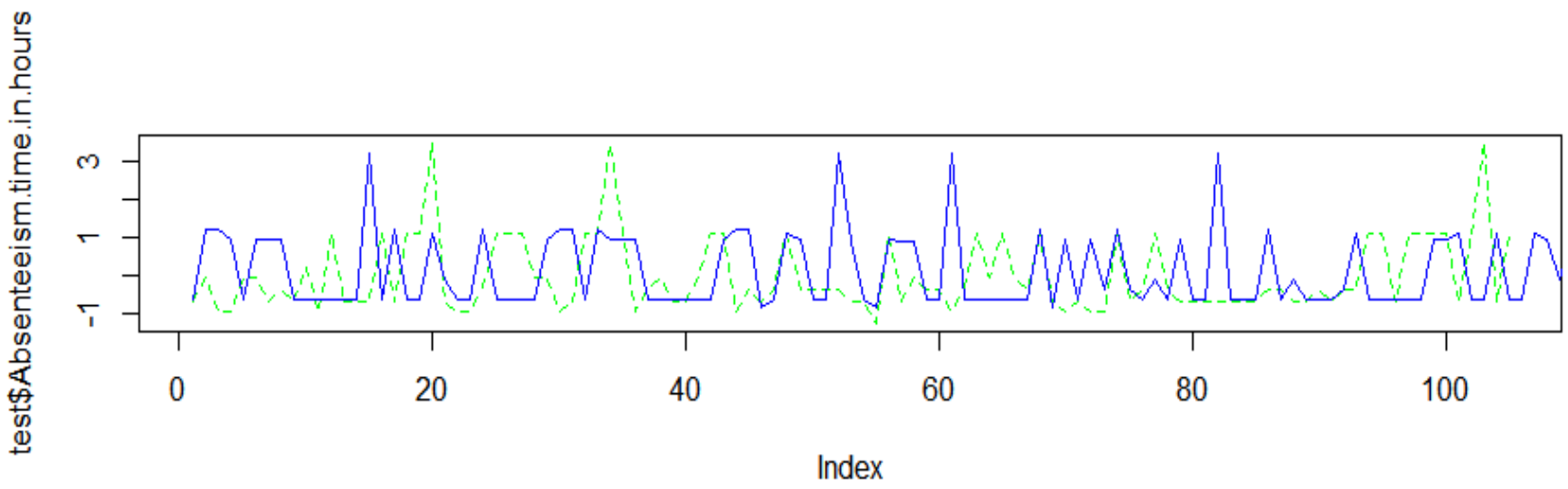
3.2. Decision Trees

Decision trees are used for both classification and regression problems. A decision tree is a tree where each node represents a feature (attribute), each link (branch) represents a decision (rule) and each leaf represents an outcome (categorical or continues value). The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from the training dataset.

Results:

Platform	Before PCA - RMSE	Before PCA – R-Squared	After PCA - RMSE	After PCA – R Squared
R	0.85	0.25	0.47	0.79
Python	4.00	-0.38	0.0	1.0

Graph between Actual and Predicted Values (obtained from R):



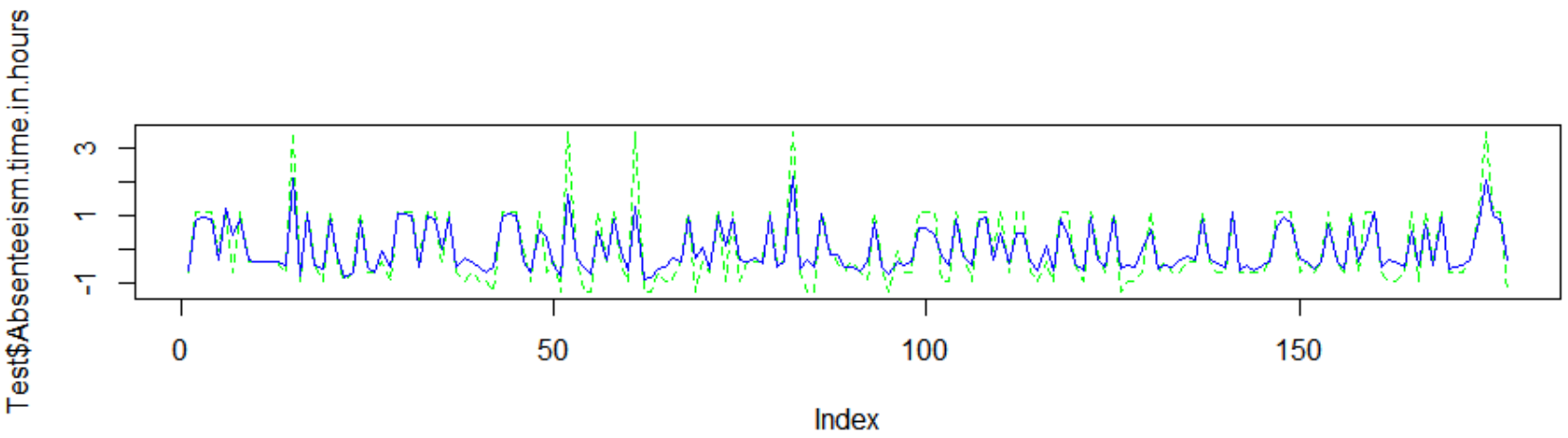
3.2. Random Forest

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression problems. We have used 500 decision trees to run down Random Forest Algorithm.

Results:

Platform	Before PCA - RMSE	Before PCA – R-Squared	After PCA - RMSE	After PCA – R Squared
R	0.83	0.29	0.25	0.94
Python	2.7	0.35	0.03	0.99

Graph between Actual and Predicted Values (obtained from R):



4. Conclusion

4.1. Model Evaluation

To evaluate the model we have used two metrics such RMSE and R-Squared.

RMSE determines how the residuals are spread across the regression line. So lower the RMSE score better the concentration of residuals around the regression line which determines good accuracy.

R-Squared score determines how good the model is to explain the variability among the variables. Higher the R- Square better the model accuracy.

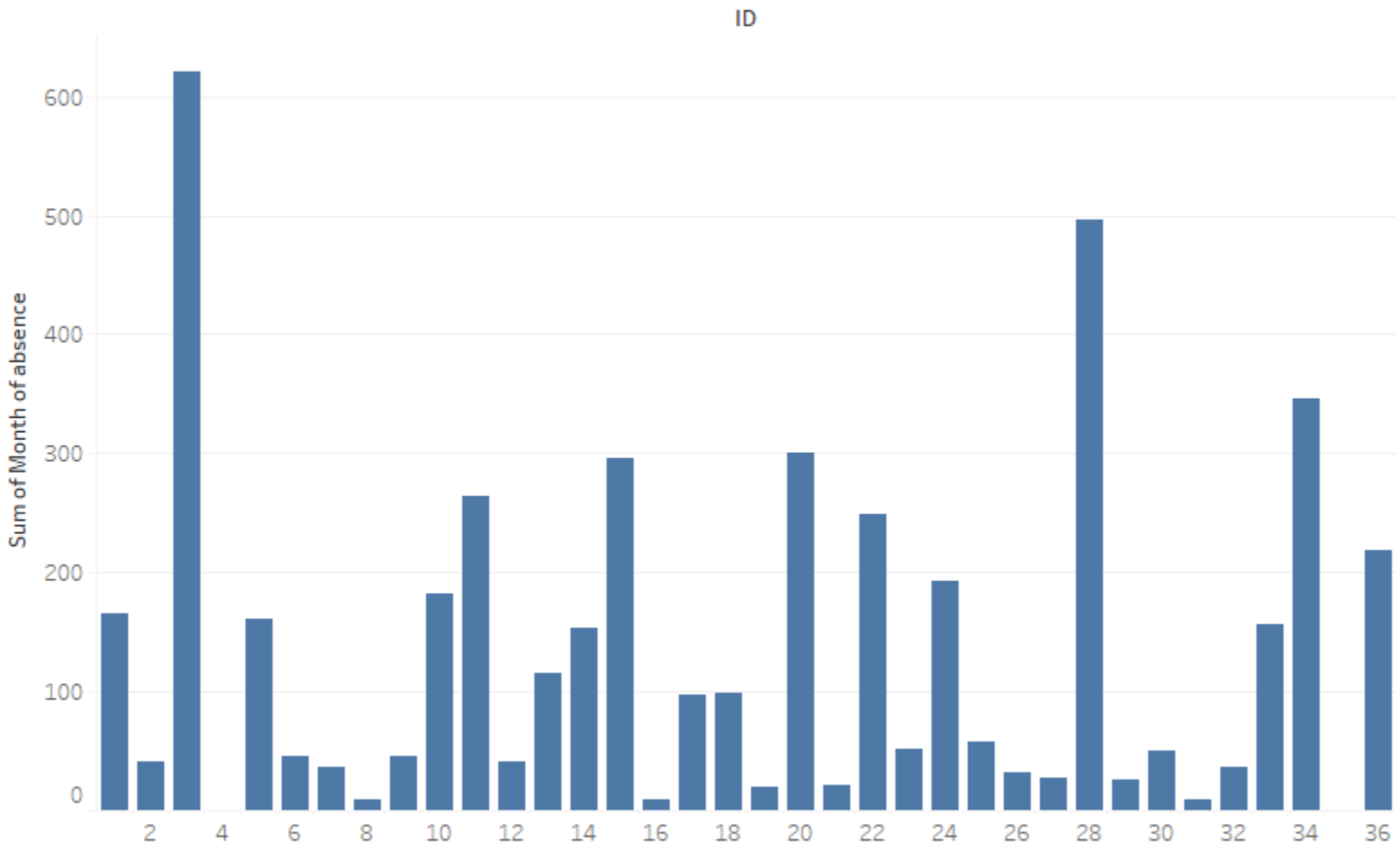
4.2. Model Selection

After evaluating the models based on the metrics we got two suitable models which are Decision Tree and Linear Regression. After applying PCA, Decision Trees provided comparatively better results than other Model In Python whereas in R, Linear Regression is the finest model.

4.3. Business's Problem Solutions

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

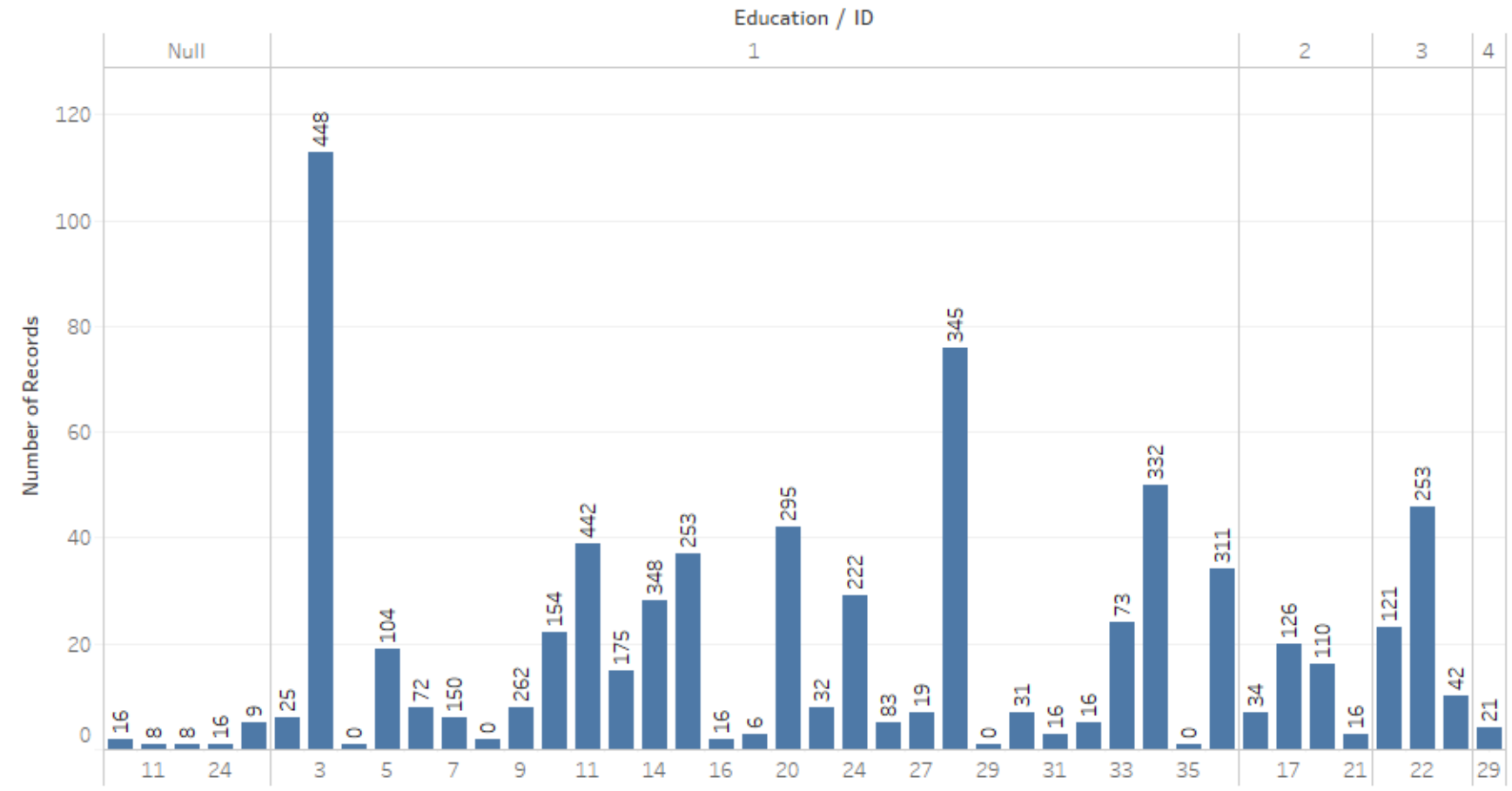
4.3.1. Month of Absence for Each ID



Sum of Month of absence for each ID.

As per the graph, The Employees with ID: 3,28,34,20 are the employees who have the maximum number of absence. Business needs to probe on these employees.

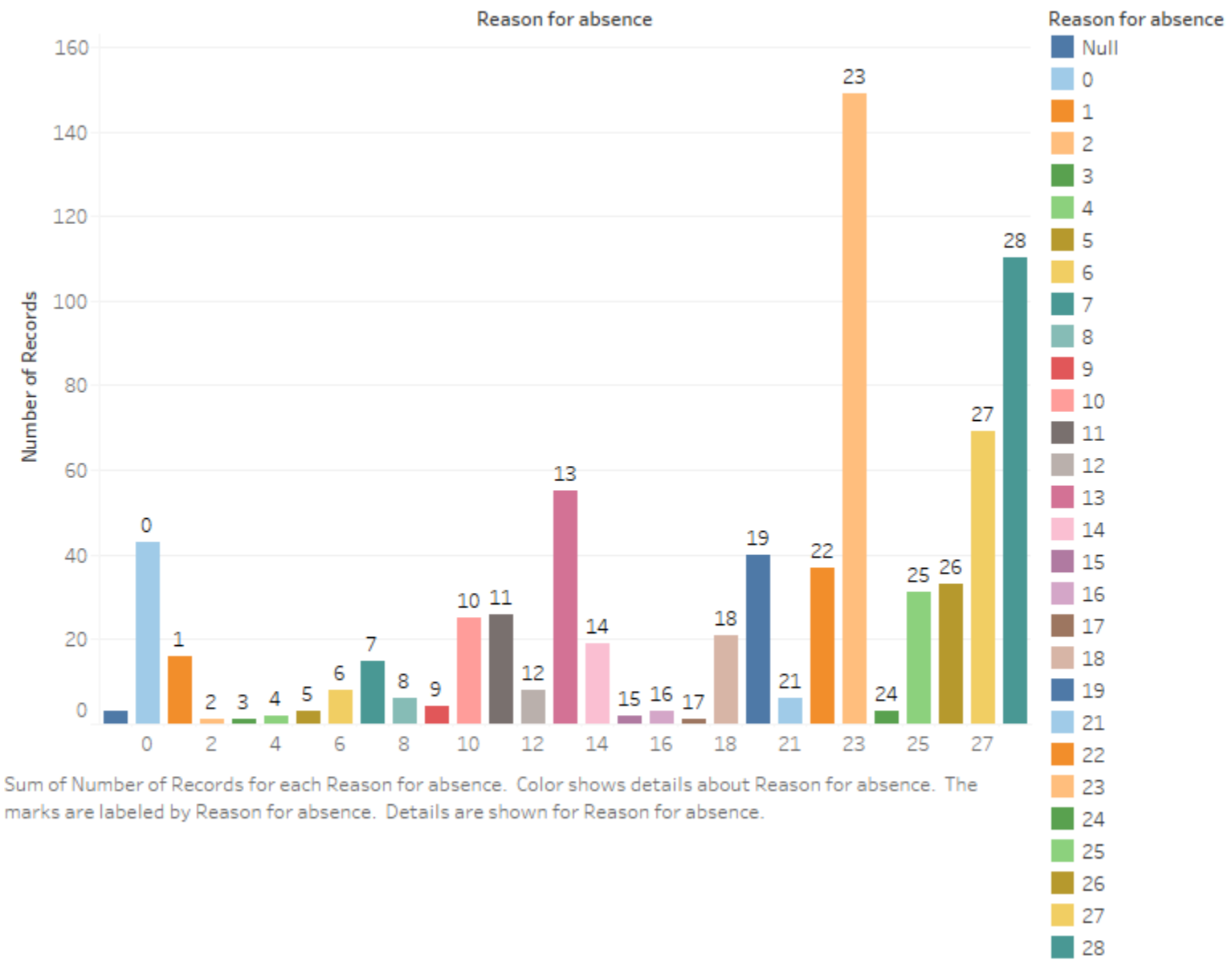
4.3.2. Absent Hours Vs Education



Sum of Number of Records for each ID broken down by Education. The marks are labeled by sum of Absenteeism time in hours.

In the above given graph we can observe that Employee's having education level Graduation are tend to absent more in comparison to the employees of other education levels. Specially Employee ID's: 3, 28 and 34 needs to be tracked.

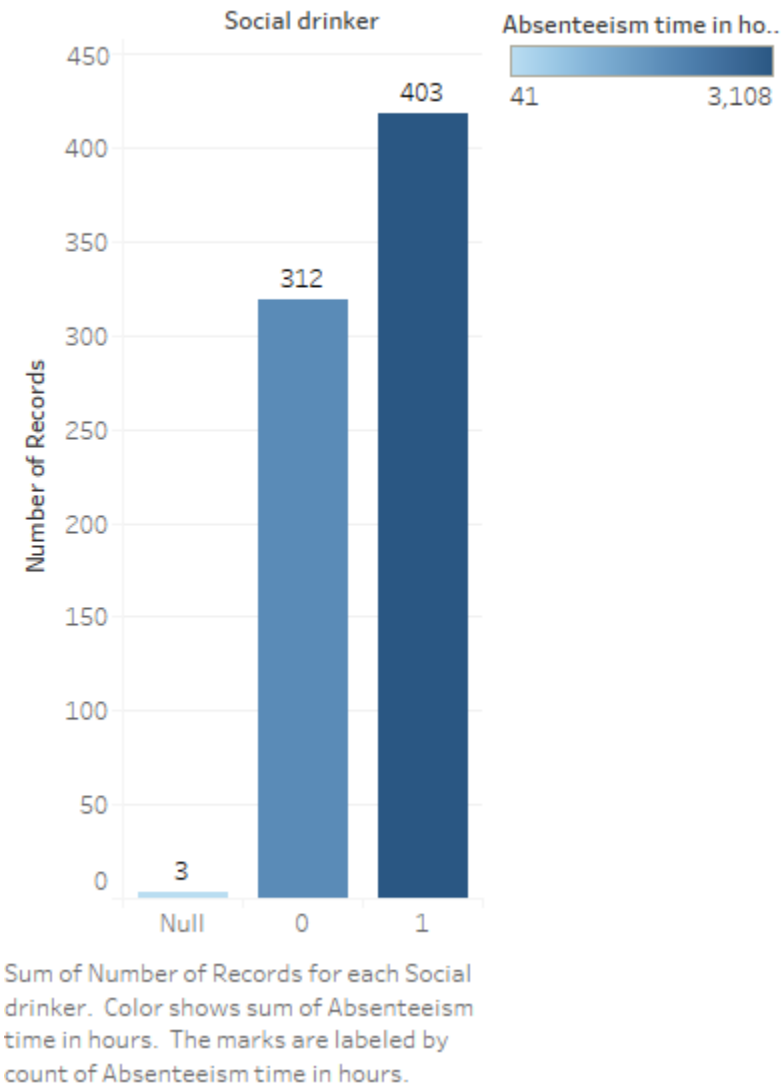
4.3.3. Reason of Absence Vs Absent Hours



Sum of Number of Records for each Reason for absence. Color shows details about Reason for absence. The marks are labeled by Reason for absence. Details are shown for Reason for absence.

In the graph we can observe that Reason 23, 27 and 28 are the major reason of employee absence. Business can provide medical checkups Such as Dental Checkups, regularly to its employees so that they would not take leaves to do so.

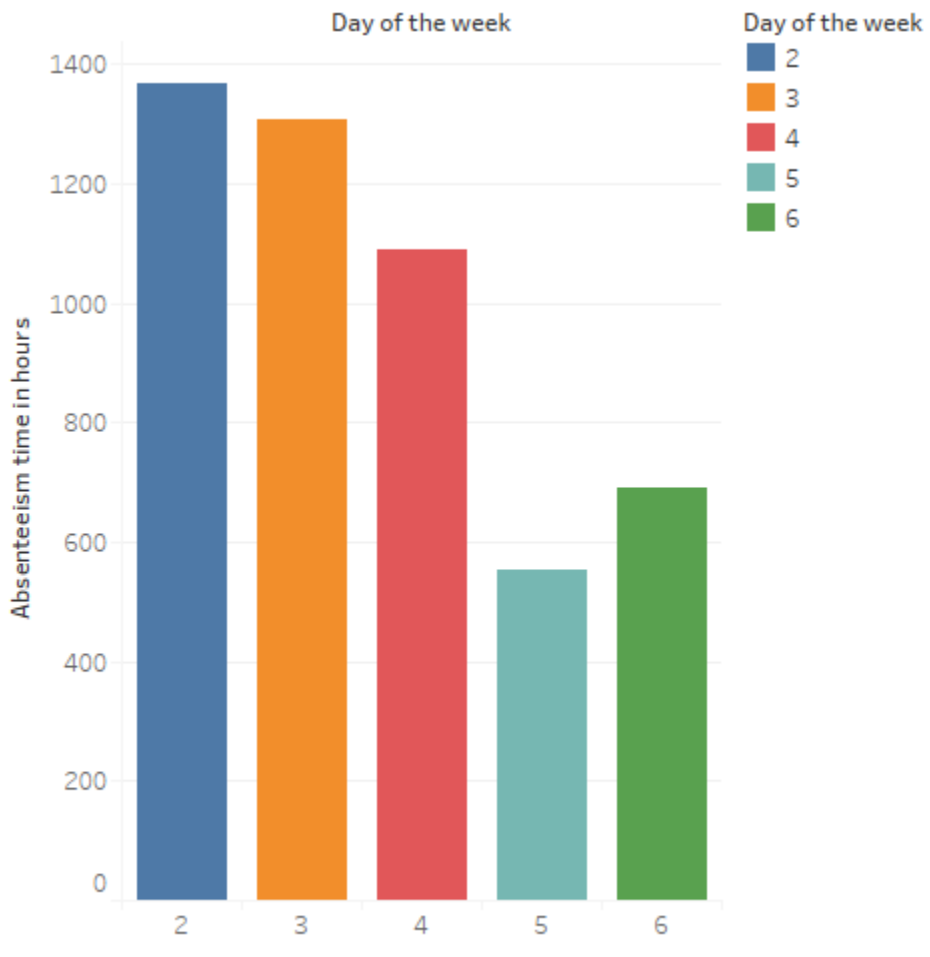
4.3.4. Social Drinker Vs Absent Hours



Sum of Number of Records for each Social drinker. Color shows sum of Absenteeism time in hours. The marks are labeled by count of Absenteeism time in hours.

The employees who takes alcohol are more likely to absent but if we compare the stats with the employees who don't drink is not so much. Still the Business can spread awareness that employee should not drink in the office hours and at times before joining the office.

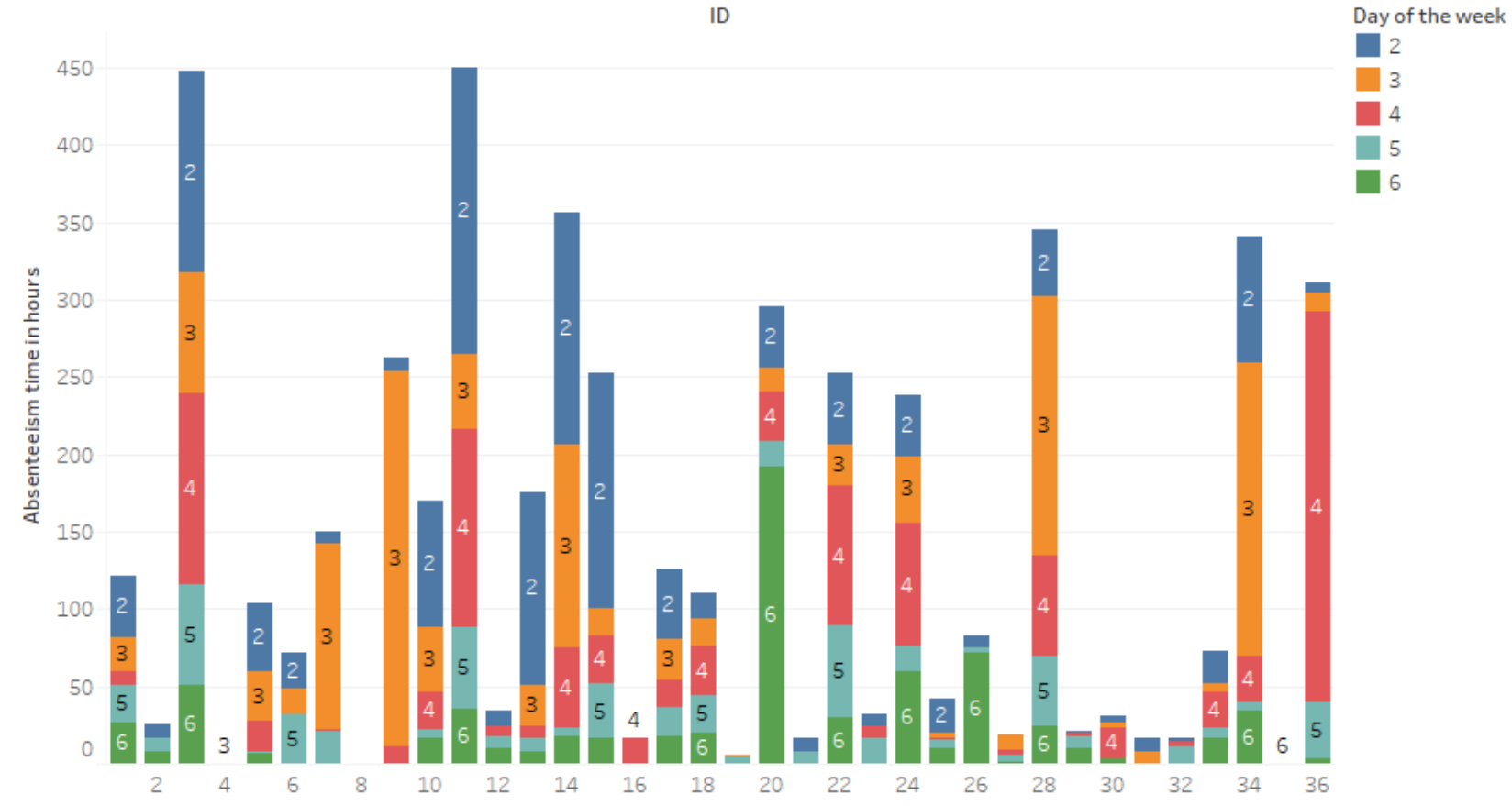
4.3.5. Day of Week Vs Absent Hours



Sum of Absenteeism time in hours for each Day of the week. Color shows details about Day of the week.

Employees are more likely to absent on Mondays or Tuesdays comparatively to other days in the week. The reason could be combining Monday makes a long weekend.

4.3.6. Day of Week Vs Absent Hours Vs Employees



Sum of Absenteeism time in hours for each ID. Color shows details about Day of the week. The marks are labeled by Day of the week.

This graph is depicting the employee wise absence against the days. If we observe

a) Employee's with ID: 3, 11 and 15 are more likely to absent on Monday.

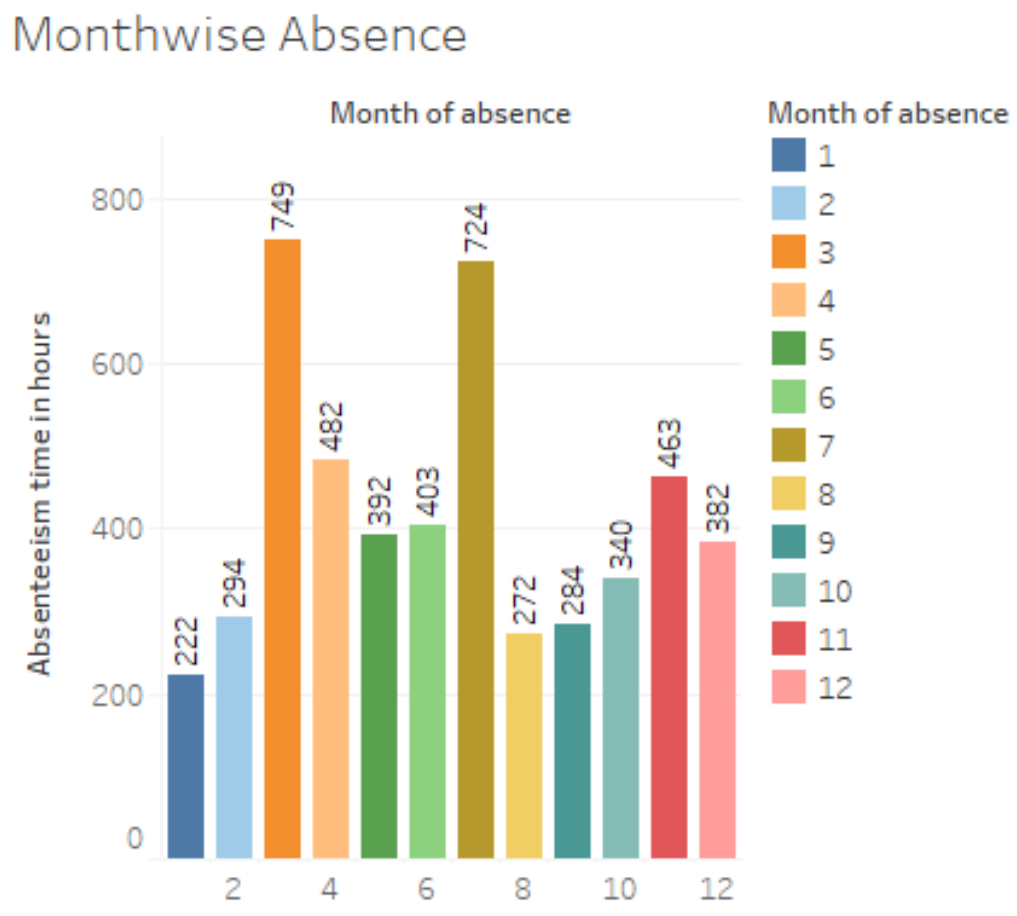
b) Employee's with ID: 9, 28 and 34 are more likely to absent on Tuesday.

c) Employee's with ID: 20 are more likely to absent on Friday.

d) Employee's with ID: 3, 11 and 36 are more likely to absent on Wednesday.

An interesting fact is Employee's with ID: 3 and 11 are tend to take leaves on Monday and Wednesday in almost same proportion. Business needs to investigate with these employees what is the reason.

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



Sum of Absenteeism time in hours for each Month of absence. Color shows details about Month of absence. The marks are labeled by sum of Absenteeism time in hours. The view is filtered on Month of absence, which excludes Null and 0.

From the given data we can observe that the data is of 3 years from July to June. The aggregated absenteeism hours has been divided by 3 to get average absent hours for every month.

Month	Absent Hours - 2011
Jan	74
Feb	73.5
Mar	249.6
Apr	160.6
May	130.6
Jun	134.33
Jul	241.33
Aug	90.66
Sep	94.66
Oct	113.33
Nov	154.33
Dec	127.33

March and July will be the months in 2011 in which maximum absenteeism can be reported.

5. Appendix

5.1. Figures

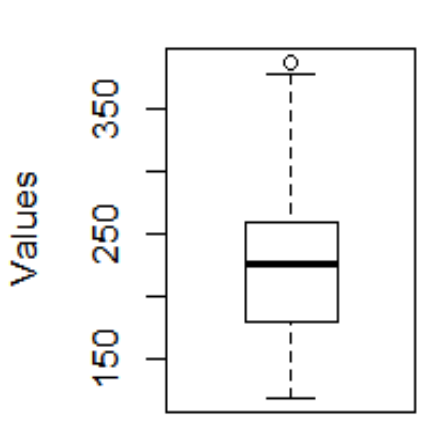
1.2.3. Data Snippet

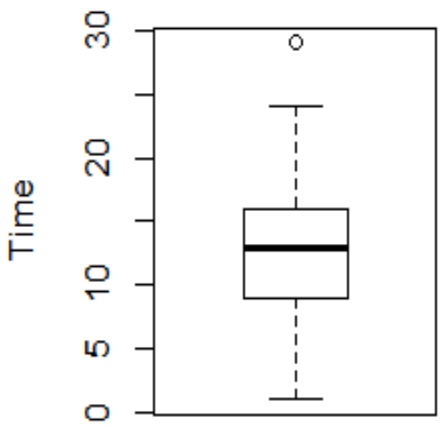
ID	Reason.for.absence	Month.of.absence	Day.of.the.week	Seasons	Transportation.expense	Distance.from.Residence.to.Work	Service.time	Age	Work.load.Average.day	Hit.target
11	26	7	3	1	289	36	13	33	239,554	97
36	0	7	3	1	118	13	18	50	239,554	97
3	23	7	4	1	179	51	18	38	239,554	97
7	7	7	5	1	279	5	14	39	239,554	97
11	23	7	5	1	289	36	13	33	239,554	97
3	23	7	6	1	179	51	18	38	239,554	97

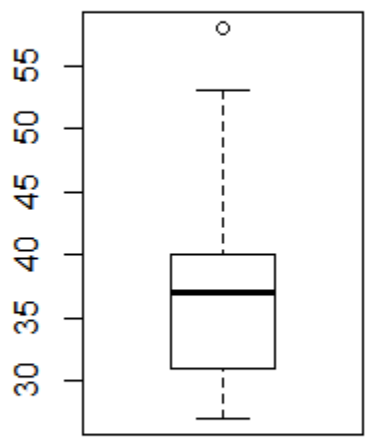
Disciplinary.failure	Education	Son	Social.drinker	Social.smoker	Pet	Weight	Height	Body.mass.index	Absenteeism.time.in.hours
0	1	2	1	0	1	90	172	30	4
1	1	1	1	0	0	98	178	31	0
0	1	0	1	0	0	89	170	31	2
0	1	2	1	1	0	68	168	24	4
0	1	2	1	0	1	90	172	30	2
0	1	0	1	0	0	89	170	31	NA

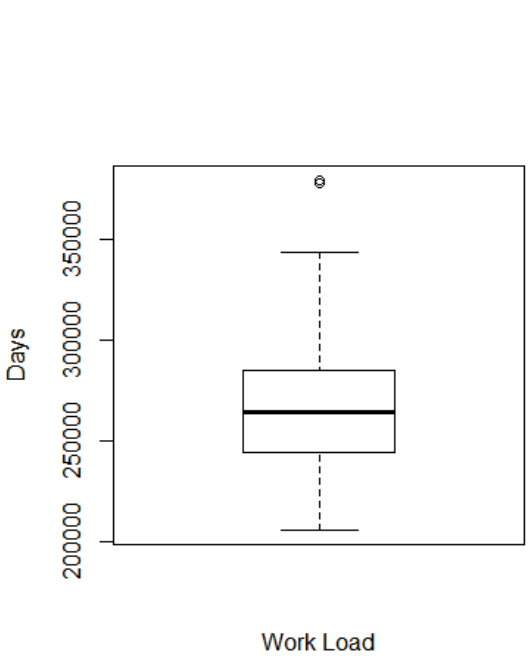
Outlier Detection:

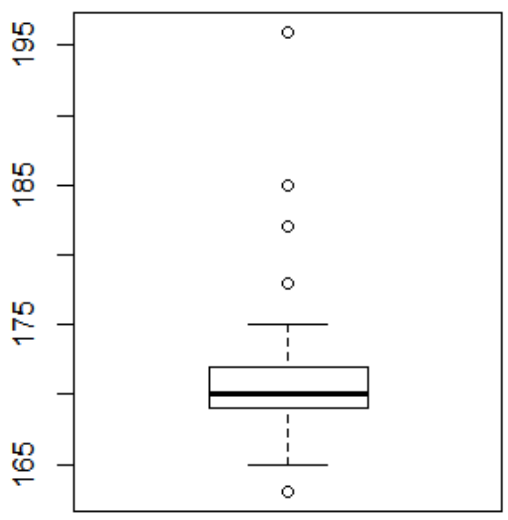
Outliers detected on few of the variables shown below:

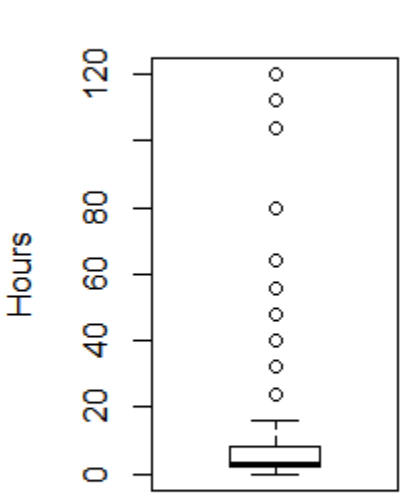




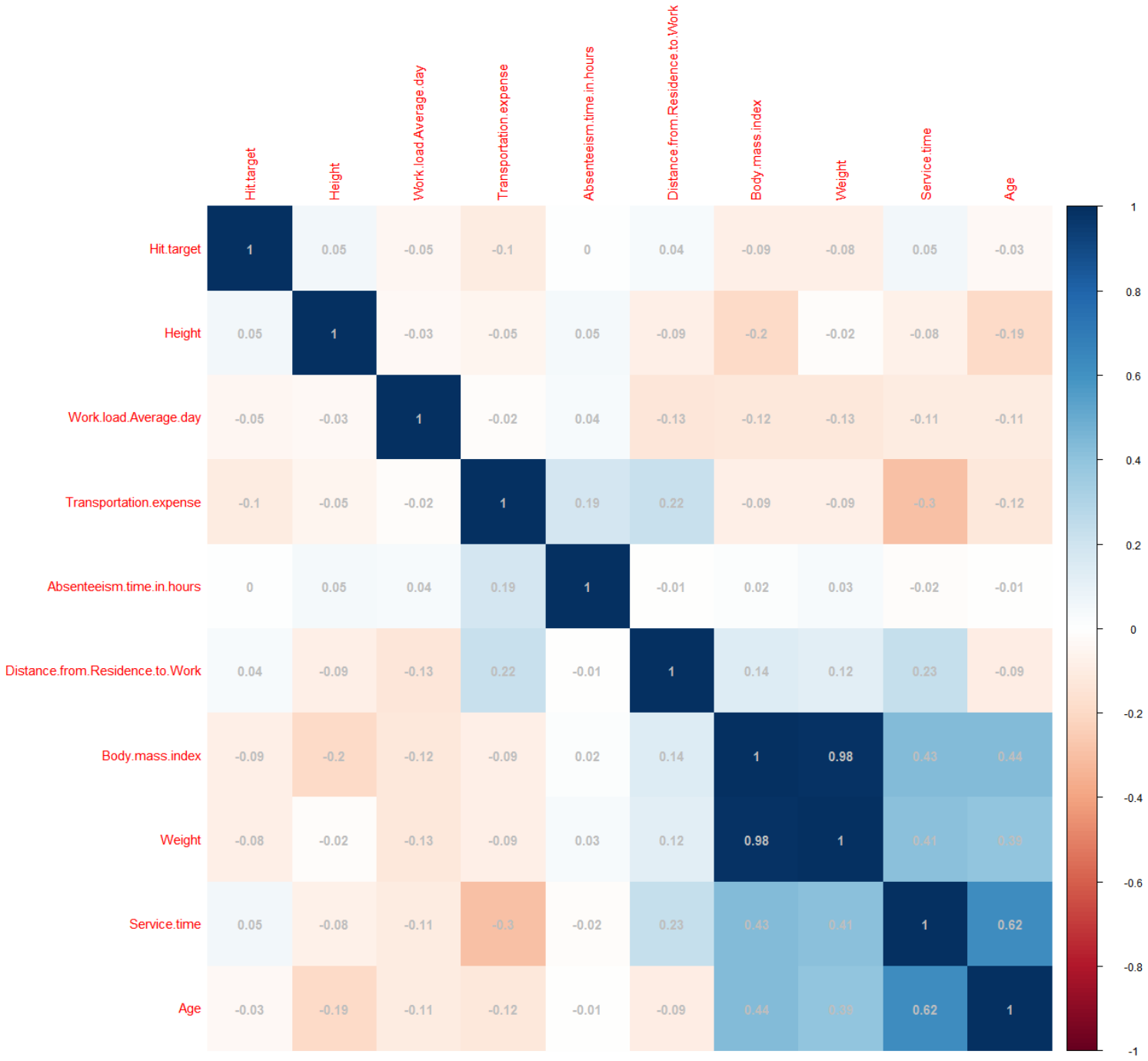




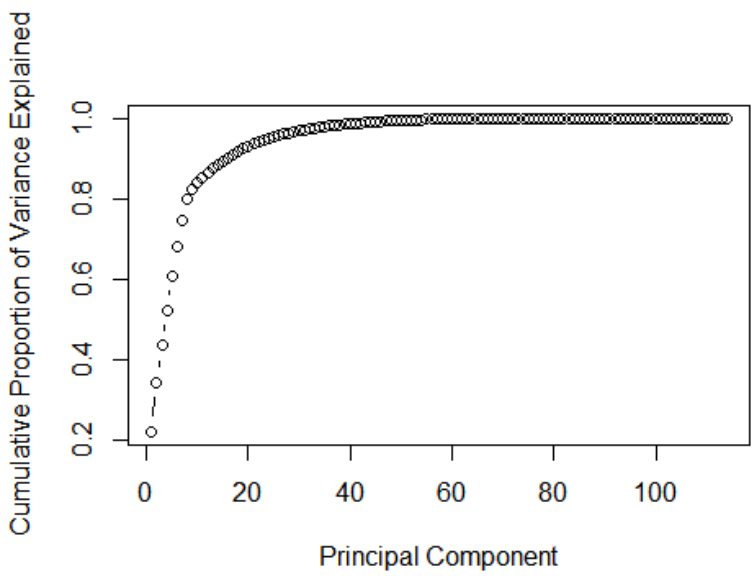




Multicollinearity:



PCA:



6. Code – Files

6.1. R Code

```
#Employee Absenteeism


rm(list=ls())


#---------------------------- Data Load -------------------------#

absentData <- read.csv("D:\\Edwisor\\Absentism\\absent.csv")

head(absentData)


#Work.load.average.day has comma in the number

absentData$Work.load.Average.day <- gsub(",","", absentData$Work.load.Average.day)



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

#Exploratory Data Analysis

str(absentData)

summary(absentData)


head(absentData)




#---------------------- Data Type Conversion ---------------------#

absentData$ID = as.factor(as.character(absentData$ID))


absentData$Reason.for.absence[absentData$Reason.for.absence %in% 0] = NA

absentData$Reason.for.absence = as.factor(as.character(absentData$Reason.for.absence))


absentData$Month.of.absence[absentData$Month.of.absence %in% 0] = NA

absentData$Month.of.absence = as.factor(as.character(absentData$Month.of.absence))


absentData$Day.of.the.week = as.factor(as.character(absentData$Day.of.the.week))

absentData$Seasons = as.factor(as.character(absentData$Seasons))

absentData$Disciplinary.failure = as.factor(as.character(absentData$Disciplinary.failure))

absentData$Education = as.factor(as.character(absentData$Education))

absentData$Son = as.factor(as.character(absentData$Son))

absentData$Social.drinker = as.factor(as.character(absentData$Social.drinker))

absentData$Social.smoker = as.factor(as.character(absentData$Social.smoker))

absentData$Pet = as.factor(as.character(absentData$Pet))

absentData$Work.load.Average.day = as.numeric(absentData$Work.load.Average.day)

absentData$Body.mass.index = as.numeric(absentData$Body.mass.index)



#------------------------ Missing Value Analysis -----------------#
```

```
#Total percentage of missing values in the dataset

library(VIM)

missingValue <- as.data.frame(colSums(is.na(absentData)))

sum(missingValue)

sum(missingValue*100)/nrow(absentData) # 23.10 % of the values are missing from the dataset and we cannot delete such a big amount of missing values


missingValueData <- kNN(absentData, k = 5)

summary(missingValueData)

newData <- subset(missingValueData, select = ID:Absenteeism.time.in.hours)


#Generating BoxPlots to see outliers

boxplot(newData$Transportation.expense, xlab = "Transportation Expense", ylab = "Values") #Has Outliers

boxplot(newData$Distance.from.Residence.to.Work, xlab = "Distance between Office and Home", ylab = "Miles") #No Outliers

boxplot(newData$Service.time, xlab = "Service Time", ylab = "Time")#Has Outliers

boxplot(newData$Age, xlab = 'Age')#Has Outliers

boxplot(newData$Work.load.Average.day, xlab = "Work Load", ylab = "Days") #Has outliers

boxplot(newData$Hit.target, xlab ="Target") #no outliers

boxplot(newData$Weight, xlab = "Weight") #no outliers

boxplot(newData$Height, xlab = 'Height') #Has outliers

boxplot(newData$Body.mass.index, xlab = "Body Mass Index") # no outliers

boxplot(newData$Absenteeism.time.in.hours, xlab = "Absenteeism", ylab = "Hours")#Has outliers


#Creating Benchmarks to remove outliers

t_Expense    <- 260 + 1.5*IQR(newData$Transportation.expense)

service_Time <- 16 + 1.5*IQR(newData$Service.time)

age          <- 40 + 1.5*IQR(newData$Age)

work_Load    <- 284853 + 1.5*IQR(newData$Work.load.Average.day)

height       <- 172 + 1.5*IQR(newData$Height)

absent_Hours <- 8 + 1.5*IQR(newData$Absenteeism.time.in.hours)


#newData < - newData[newData$Transportation.expense < t_Expense]


cleanData <- subset(newData, Transportation.expense <= t_Expense)

cleanData <- subset(cleanData, Service.time <= service_Time)

cleanData <- subset(cleanData, Work.load.Average.day <= work_Load)

cleanData <- subset(cleanData, Height <= height)

cleanData <- subset(cleanData, Absenteeism.time.in.hours <= absent_Hours)


cleanData <- cleanData[complete.cases(cleanData),]


boxplot(cleanData$Height, xlab = 'Height')

boxplot(cleanData$Transportation.expense)


#bScaling <- sample(2, nrow(cleanData), replace = TRUE, prob = c(0.8,0.2))

#bScaleTrain <- cleanData[bScaling == 1, ]
```

```
#bScaleTest  <- cleanData[bScaling == 2, ]

#bScaleTest$Reason.for.absence[bScaleTest$Reason.for.absence == 3 ] <- NA
#bScaleTest <- bScaleTest[complete.cases(bScaleTest),]



#------------------------------- Feature Selection ----------------------#
library(usdm)
library(corrplot)

#Correlation Analysis
numeric_index = sapply(cleanData, is.numeric)
numeric_data = cleanData[,numeric_index]
vifcor(numeric_data)

corrplot(cor(numeric_data), method = "number", addCoef.col="grey", order = "AOE")
pairs(numeric_data)

corPlot <- cor(numeric_data)
corrplot(corPlot, method = "shade")

#------------------------------- Feature Scaling ----------------------#
#Variability is among the variables. So we need to normalize the data

library("dplyr")
library(faraway)
numericData <- select_if(cleanData, is.numeric)
factorData  <- select_if(cleanData, is.factor)
numericData <- as.data.frame(scale(numericData))

#This will help us to get the predictions on the original scale
#scaleList   <- list(scale = attr(numericData,"scaled:scale"),
#                    center = attr(numericData,"scaled:center"))

finalData   <- data.frame(factorData,numericData)
finalData   <- data.frame(finalData)

#To check the correlations
corTable <- round(cor(numericData),2)
corTable[corTable <= 0.5] <- 0

#--------------------------- Data Partitioning ----------------------#
ind   <- sample(2, nrow(finalData), replace = T, prob = c(0.8,0.2))
train <- finalData[ind == 1,]
test  <- finalData[ind == 2,]

test$Reason.for.absence[test$Reason.for.absence == 17 | test$Reason.for.absence == 3
```

```
                | test$Reason.for.absence == 5 | test$Reason.for.absence == 4
                | test$Reason.for.absence == 24] <- NA
test$Pet[test$Pet == 8] <- NA
test <- test[complete.cases(test),]



#--------------------------- Linear Regression ----------------------#
set.seed(1234)

library(caret)
linearModel <- lm(Absenteeism.time.in.hours~., data=train[,!colnames(train) %in% c("ID")])

summary(linearModel)
predictLinear <- predict(linearModel, test, type="response")

#omod <- lm(Absenteeism.time.in.hours~., data=bScaleTrain[,!colnames(bScaleTrain) %in% c("ID")])
#summary(omod)
#op <- predict(omod, bScaleTest)
#usp <- predictLinear * scaleList$scale['Absenteeism.time.in.hour']
#+ scaleList$center["Absenteeism.time.in.hour"]
#all.equal(op, usp)

#Calcuate MAE, RMSE, R-sqaured for testing data
print(postResample(pred = linearModel, obs = test$Absenteeism.time.in.hours))



predictedData <- data.frame(test$ID,predictLinear)
#lrResult <- postResample(pred = predictLinear, obs = test$Absenteeism.time.in.hours)

#Model Performance
plot(test$Absenteeism.time.in.hours, type = 'l', lty = 1.8, col = 'green')
lines(predictLinear, type = 'l', col = 'blue')

hist(test$Absenteeism.time.in.hours)
hist(predictLinear)

#New Data Frame containing the Predicted value against the employee ID
write.csv(predictedData,"D:\\Edwisor\\\\Absentism\\test.csv", row.names = TRUE)

#------------------------------ Random Forest ----------------------------#
library(randomForest)
rfModel <- randomForest(Absenteeism.time.in.hours~., data = train, ntree = 500)
summary(rfModel$predicted)
predictRandom <- predict(rfModel, test)
```

```
rfResult <- postResample(pred = predictRandom, obs = test$Absenteeism.time.in.hours)


hist(predictRandom)

#------------------------------ Decision Tree ----------------------------#
library(rpart)
decisionModel   <- rpart(Absenteeism.time.in.hours~., data = train, method = "anova")
predictDecision <- predict(decisionModel,test)

#Create data frame for actual and predicted values
dtData  <- data.frame(test$ID,predictDecision)

#Calcuate MAE, RMSE, R-sqaured for testing data
library(caret)
dResult <- postResample(pred = predictDecision, obs = test$Absenteeism.time.in.hours)

#------------------------- Model Improvement -------------------#
#1. Found Multicolinearity
#2. More number of independent variables.
#3. Poor accuracy in the models.



#--- Dimensionality Reduction using PCA - Test -------------------#
#Out of 21 Variables we have 11 variables with Factor Datatype, hence we need to convert them to numeric
library(dummies)

newData <- dummy.data.frame(finalData, names = c("ID","Reason.for.absence","Month.of.absence",
                                        "Disciplinary.failure","Education","Son",
                                        "Social.drinker","Social.smoker","Pet","Seasons",
                                        "Day.of.the.week"))
#Added numeric data in the dataframe. Before it was missing the predictor variable.
"Day.of.the.week"))
newData <- data.frame(numericData,newData)


#Data Partinioning
ind <- sample(2,nrow(newData), replace = T, prob = c(0.7,0.3))
Train <- newData[ind == 1,]
Test  <- newData[ind == 2,]

#Principal component analysis
pComp <- prcomp(Train)

#Standard deviation of each principal component
pStdev <- pComp$sdev
```

```
#Variance
pVar <- pStdev^2

#Proportion of variance explained
proVar <- pVar/sum(pVar)

plot(cumsum(proVar), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     type = "b")


#Add a training set with principal components
#train.data = data.frame(Absenteeism.time.in.hours = Train$Absenteeism.time.in.hours, pComp$x)
pcaTrain <- data.frame(Absenteeism.time.in.hours = Train$Absenteeism.time.in.hours, pComp$x)

# From the above plot selecting 45 components since it explains almost 95+ % data variance
#train.data =train.data[,1:9]
pcaTrain <- pcaTrain[,1:50]


#Transform test data into PCA
#test.data = predict(prin_comp, newdata = Test)
pcaTest <- predict(pComp, newdata = Test)
pcaTest <- as.data.frame(pcaTest)
#test.data = as.data.frame(test.data)

#Select the first 9 components
pcaTest <- pcaTest[,1:50]

#------------------------- Decision Tree with PCA  ----------------------------------#
#RMSE: 0.2814679
#Rsquared: 0.915544
#MAE: 3 0.1724320

dtModelImproved <- rpart(Absenteeism.time.in.hours ~., data = pcaTrain, method = "anova")

#Predict the test cases
dtPredictionImproved <- predict(dtModelImproved,pcaTest)

#Calcuate MAE, RMSE, R-sqaured for testing data
print(postResample(pred <- dtPredictionImproved, obs = Test$Absenteeism.time.in.hours))

#Plot a graph for actual vs predicted values
plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")
lines(dtPredictionImproved,col="blue")

#Create data frame for actual and predicted values
```

```
df_pred <- data.frame("actual"=Test$Absenteeism.time.in.hours.1, "dt_pred"= dtPredictionImproved)

#Unscaling the data to get Actual Predictions
Test * attr(Test, 'scaled:scale') + attr(Test, 'scaled:center')

#------------------------------ Random Forest with PCA ----------------------------------------------------#
#RMSE: 0.2546469
#R squared: 0.9445420
#MAE: 0.1460709

#Train the Model
rfModelImproved <- randomForest(Absenteeism.time.in.hours~., data = pcaTrain, ntrees = 500)

#Prediction on test data
rfPredictionImproved <- predict(rfModelImproved,pcaTest)

#Calcuate MAE, RMSE, R-sqaured for testing data
print(postResample(pred = rfPredictionImproved, obs = Test$Absenteeism.time.in.hours))


#Plot a graph for actual vs predicted values
plot(Test$Absenteeism.time.in.hours,type="l",lty=2,col="green")
lines(rfPredictionImproved,col="blue")


#------------------------------ Linear Regression with PCA ------------------------#
#RMSE: 0.02477026
#R squared: 0.99936171
#MAE: 0.01694319

#Train the Model
lrModelImproved <- lm(Absenteeism.time.in.hours ~ ., data = pcaTrain)
summary(lrModelImproved)

#Predict the test cases
lrPredictionsImproved <- predict(lrModelImproved,pcaTest)

#Calcuate MAE, RMSE, R-sqaured for testing data
print(postResample(pred = lrPredictionsImproved, obs =Test$Absenteeism.time.in.hours))

#Plot a graph for actual vs predicted values
plot(Test$Absenteeism.time.in.hours,type="l",lty=2,col="green")
lines(lrPredictionsImproved,col="blue")
```

6.2. Python Code

```
#------------------------------- Employee Absenteeism Project ----------------------------------#
#Import Libraries
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score
from fancyimpute import KNN
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
#% matplotlib inline

#------------------------- Data Load ----------------------------------#
absentData = pd.read_excel("D:\\Edwisor\\Absentism\\absent.xls ")

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

absentData.shape
absentData.head()
absentData.dtypes

#---------------------- Data Type Conversion --------------------------#
absentData['ID'] = absentData['ID'].astype('category')
absentData['Reason for absence'] = absentData['Reason for absence'].replace(0,np.nan)
absentData['Reason for absence'] = absentData['Reason for absence'].astype('category')

absentData['Month of absence']   = absentData['Month of absence'].replace(0,np.nan)
absentData['Month of absence']   = absentData['Month of absence'].astype('category')

absentData['Day of the week']    = absentData['Day of the week'].astype('category')
absentData['Seasons']            = absentData['Seasons'].astype('category')
absentData['Disciplinary failure']    = absentData['Disciplinary failure'].astype('category')

absentData['Education']    = absentData['Education'].astype('category')
absentData['Son']          = absentData['Son'].astype('category')
absentData['Social drinker']    = absentData['Social drinker'].astype('category')
absentData['Social smoker']     = absentData['Social smoker'].astype('category')
absentData['Pet']               = absentData['Pet'].astype('category')

absentData.dtypes


#------------------------------- Missing Value Analysis ---------------------------------#
missingVal  = pd.DataFrame(absentData.isnull().sum()).sum()
missingValPercent = missingVal/len(absentData.index)*100
missingValPercent.round()

#Approx 24% values are null in the dataset. So we need to impute them by suitable method.

#Missing Value Imputation
```

```
absentData.isnull().sum()

absentData = pd.DataFrame(KNN(k = 3).fit_transform(absentData), columns = absentData.columns)

absentData = absentData.round()


#------------------------------ Outlier Analysis ----------------------------------------#

sns.boxplot(data=absentData[['Absenteeism time in hours','Service time','Height','Weight','Transportation expense','Age']])

fig=plt.gcf()

fig.set_size_inches(8,12)

sns.boxplot(data=absentData[['Work load Average/day']])



#Computing the benchmark for the numeric values

numericValues = ['Work load Average/day','Distance from Residence to Work', 'Service time', 'Age','Transportation expense','Hit target', 'Weight', 'Height', 'Body mass index', 'Absenteeism time in hours']

for i in numericValues:
    q75, q25 = np.percentile(absentData[i], [75,25])

    # Computing IQR
    iqr = q75 - q25

    # Computing upper and lower threshold/benchmark
    minThres = q25 - (iqr*1.5)
    maxThres = q75 + (iqr*1.5)
    print (maxThres)
    # Replacing all the outliers with NA
    absentData.loc[absentData[i]< minThres,i] = np.nan
    absentData.loc[absentData[i]> maxThres,i] = np.nan



#Impute missing values with KNN
cleanData = pd.DataFrame(KNN(k = 3).fit_transform(absentData), columns = absentData.columns)

# Checking if there is any missing value
cleanData.isnull().sum()
cleanData = cleanData.round()
dummy = cleanData
#------------------------------------ Feature Selection -------------------------------------#
#Get dataframe with all continuous variables
dfNumeric = cleanData.loc[:,numericValues]

#Check for multicollinearity using corelation graph
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(10, 10))

#Generate correlation matrix
corr = dfNumeric.corr()

#------------------------------ Feature Scaling ----------------------------------------#
# Normalization of continuous variables
```

```
for i in numericValues:
    if i == 'Absenteeism time in hours':
        continue
    cleanData[i] = (cleanData[i] - cleanData[i].min())/(cleanData[i].max()-cleanData[i].min())


#------------------------------ Machine Learning Models ------------------------------------#
#Splitting data into train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(cleanData.iloc[:, cleanData.columns != 'Absenteeism time in hours'], cleanData.iloc[:,
20], test_size = 0.30, random_state = 1)



#---------------------------------- Linear Regression Model ---------------------------------#
# Root Mean Squared Error: 2.898405340060082
# R^2 Score(coefficient of determination) = 0.2772050386036977
from sklearn.linear_model import LinearRegression

#Build Linear regression model
lrModel = LinearRegression().fit(X_train , y_train)

#Perdict for test records
lrModelPred = lrModel.predict(X_test)

#Storing results in a data frame for Actual and Predicted values
lrResult = pd.DataFrame({'Actual': y_test, 'Predicted': lrModelPred})
print(lrResult.head())

#Calculate RMSE and R-squared value
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

print("Root Mean Squared Error: "+str(RMSE(y_test, lrModelPred)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, lrModelPred)))



#---------------------------------- Decision Tree Model ---------------------------------------#
# Root Mean Squared Error: 4.008437047973632
# R^2 Score(coefficient of determination) = -0.38244228432563765

from sklearn.tree import DecisionTreeRegressor
dtModel = DecisionTreeRegressor(random_state = 1).fit(X_train,y_train)

#Perdict for test records
dtModelPred = dtModel.predict(X_test)

#Storing results in a data frame for Actual and Predicted values
dtResult = pd.DataFrame({'Actual': y_test, 'Predicted': dtModelPred})
print(dtResult.head())
```

```
#Define function to calculate RMSE
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, dtModelPred)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, dtModelPred)))



#---------------------------------- Random Forest Model ----------------------------------------#
# Root Mean Squared Error: 2.737971200171947
# R^2 Score(coefficient of determination) = 0.3550075584440454

from sklearn.ensemble import RandomForestRegressor

#Build random forest
rfModel = RandomForestRegressor(n_estimators = 500, random_state = 1).fit(X_train,y_train)

#Perdict for test records
rfModelPred = rfModel.predict(X_test)

#Storing results in a data frame for Actual and Predicted values
rfResult = pd.DataFrame({'Actual': y_test, 'Pred': rfModelPred})
print(rfResult.head())

#Calculate RMSE and R-squared value
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

print("Root Mean Squared Error: "+str(RMSE(y_test, rfModelPred )))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, rfModelPred )))


#------------------------------ Dimensionality Reduction Using PCA --------------------------#
#Store the Dependent Variable
depVar = cleanData['Absenteeism time in hours']
factorVariable = ['ID','Reason for absence','Month of absence','Day of the week','Seasons','Disciplinary failure', 'Education', 'Social
drinker','Social smoker', 'Pet', 'Son']

#PCA works on numeric variables hence converting factor into numeric
pcaData = pd.get_dummies(data = cleanData, columns = factorVariable)

#Import library for PCA
from sklearn.decomposition import PCA
#Converting data to numpy array
nArray = pcaData.values

#Dataset has 116 variables
```

```
pca = PCA(n_components=116)
pca.fit(nArray)

#Proportion of variance explained
var = pca.explained_variance_ratio_

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

#Draw the plot
plt.plot(var1)
plt.show()

pca = PCA(n_components=45)

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

#Splitting data into train and test data
X_trainImp, X_testImp, y_trainImp, y_testImp = train_test_split(nArray,depVar, test_size=0.3, random_state = 1)

#---------------------- Linear Model Improved Using PCA ----------------------------------------#
#Root Mean Squared Error: 8.633839188446864e-12
#R^2 Score(coefficient of determination) = 1.0

lrModelImproved = LinearRegression().fit(X_trainImp , y_trainImp)

#Perdict for test records
lrModelPredImproved = lrModelImproved.predict(X_testImp)

#Storing results in a data frame for Actual and Predicted values
lrResultImproved = pd.DataFrame({'Actual': y_test, 'Predicted': lrModelPredImproved})
print(lrResultImproved.head())

def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

print("Root Mean Squared Error: "+str(RMSE(y_test, lrModelPredImproved)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, lrModelPredImproved)))

#---------------------- Decision Tree Model Improved Using PCA ----------------------------------#
#Root Mean Squared Error: 0.0
#R^2 Score(coefficient of determination) = 1.0

dtModelImproved = DecisionTreeRegressor(random_state = 1).fit(X_trainImp,y_trainImp)

#Perdict for test records
dtModelPredImproved = dtModelImproved.predict(X_testImp)

#Storing results in a data frame for Actual and Predicted values
```

```
dtResultImproved = pd.DataFrame({'Actual': y_test, 'Predicted': dtModelPredImproved})
print(dtResultImproved.head())

#Define function to calculate RMSE
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, dtModelPredImproved)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, dtModelPredImproved)))

#---------------------- Random Forest Model Improved Using PCA ---------------------------------------#
#Root Mean Squared Error: 0.03548759458415528
#R^2 Score(coefficient of determination) = 0.9998916447395986

#Build random forest
rfModelImproved = RandomForestRegressor(n_estimators = 500, random_state = 1).fit(X_trainImp,y_trainImp)

#Perdict for test records
rfModelPredImproved = rfModelImproved.predict(X_testImp)

#Storing results in a data frame for Actual and Predicted values
rfResultImproved = pd.DataFrame({'Actual': y_test, 'Pred': rfModelPredImproved})
print(rfResult.head())

#Calculate RMSE and R-squared value
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

print("Root Mean Squared Error: "+str(RMSE(y_test, rfModelPredImproved)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, rfModelPredImproved)))
```