ABSENTEEISM AT WORK PROJECT REPORT

(SHRAVYA SURESH)

CONTENTS…

Chapter 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

Our task is to build a regression model for the given data set which helps us to know the amount of loss faced by the company if the same trend of absenteeism continues till 2011.

A sample of the given dataset is given here,

Age	Work load Average da	Hit targe	Disciplinary failu	Educatio	Son	Social drinke	Social smoke
33	2,39,554	97	0	1	2	1	0
50	2,39,554	97	1	1	1	1	0
38	2,39,554	97	0	1	0	1	0
39	2,39,554	97	0	1	2	1	1
33	2,39,554	97	0	1	2	1	0
38	2,39,554	97	0	1	0	1	0

Pet	Weight	Height	Body mass index	Absenteeism time in hours
1	90	172	30	4
0	98	178	31	0
0	89	170	31	2
0	68	168	24	4
1	90	172	30	2
0	89	170	31	

The target variable of the given dataset is **Absenteeism time in hours** which is a continuous variable and hence it is a regression type of problem

The independent variables are,

1. Individual identification	2. Reason for absence
3. Month of absence	4. Day of the week
5. Seasons	6. Transportation expense

7. Distance from Residence to Work	8. Service time
9. Age	10. Work load Average/day
11. Hit target	12. Disciplinary failure
13. Education	14. Son
15. Social drinker	16. Social smoker
17. Pet	18. Weight
19. Height	20. Body mass index

Chapter 2

Methodology

The solution is divided into 3 parts.

1. Exploratory data analysis(EDA) was performed to explore the structure of data. Some of the basic assumptions were made about the data ie. Which variables are most likely causing churn. During exploration dataset was checked for missing values, multi collinearity and other model/algorithm specific assumptions.

2. After EDA, for learning two models were used, logistic regression and random forest. Some data pre-processing was done to prepare training data for learning model.

3. In the last part, performance tuning was done to increase the accuracy of models. Both the algorithms and EDA were implemented in R and python. Both implementations were similar with little difference due to difference in learning algorithm implementation.

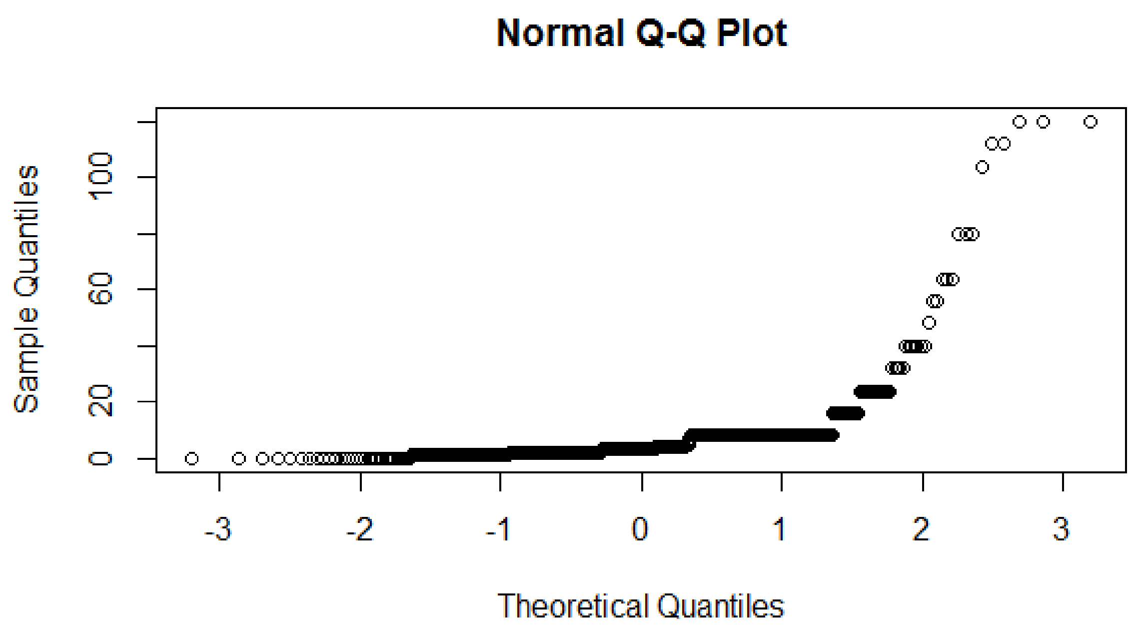
2.1 Exploratory Data Analysis

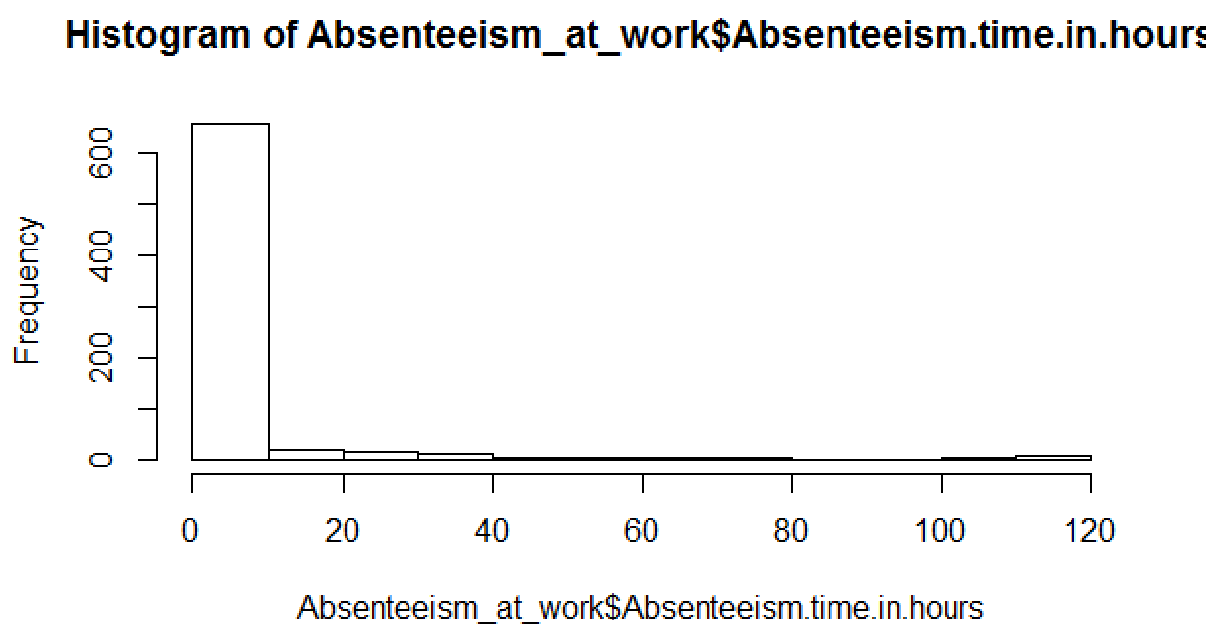
Exploratory data analysis a.k.a. EDA was performed on training data using R and python. We looked at the structure of training data and found 20 predictors, 1 target variable and 740 observations.

2.1.1 The Target Variable - **Absenteeism time in hours**

The target variable is a continuous variable. It shows us the average time of absenteeism in hours.

The normal distribution plot of the target variable is as shown,





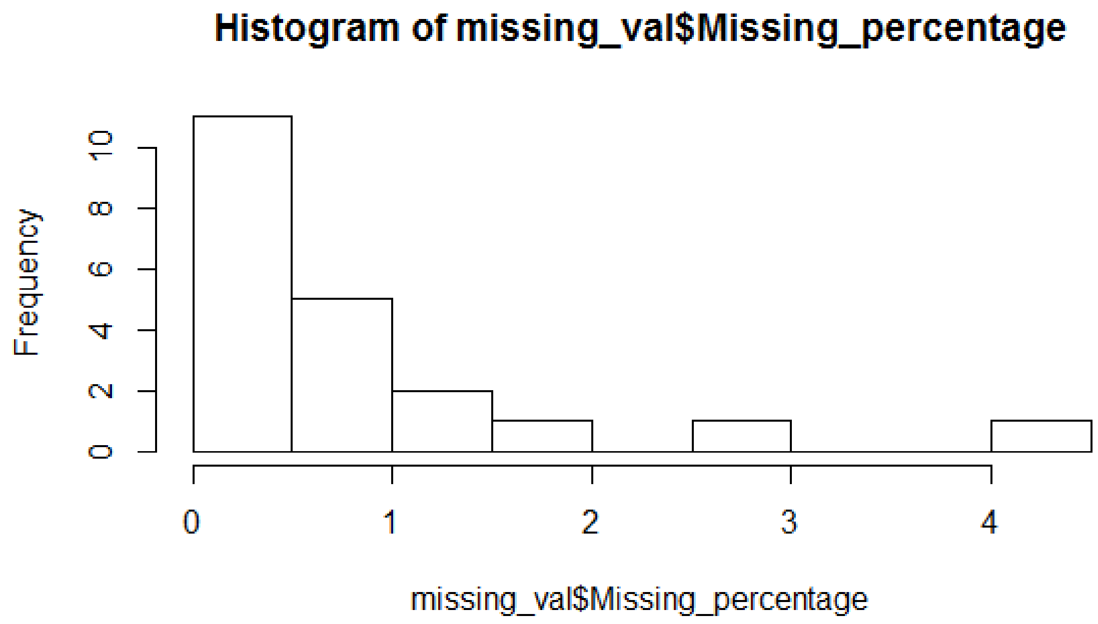
2.1.2 Missing Value Analysis

The missing values present in the data set as given in the table.

Columns	Missing_percentage
Body.mass.index	4.189189189
Absenteeism.time.in.hours	2.972972973
Height	1.891891892
Work.load.Average.day	1.351351351

Education	1.351351351
Transportation.expense	0.945945946
Hit.target	0.810810811
Disciplinary.failure	0.810810811
Son	0.810810811
Social.smoker	0.540540541
Reason.for.absence	0.405405405
Distance.from.Residence.to.Work	0.405405405
Service.time	0.405405405
Age	0.405405405
Social.drinker	0.405405405
Pet	0.27027027
Month.of.absence	0.135135135
Weight	0.135135135
ID	0
Day.of.the.week	0
Seasons	0

As we can see, there are many missing values present in the dataset. A graph indicating the missing values is as follows,

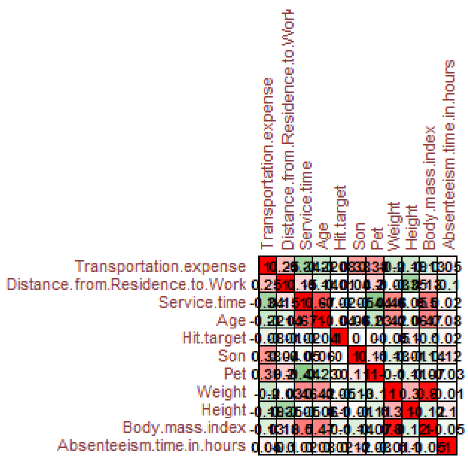


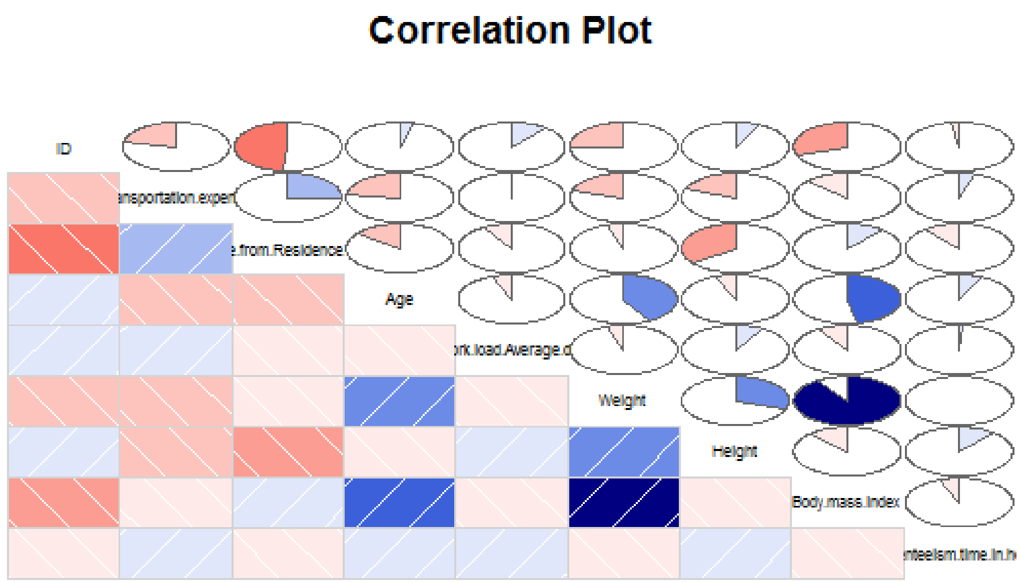
This situation is solved by the given R and Python code and hence the missing values of the whole data is eliminated using Knn imputation method.

2.1.3 Multicollinearity

Multicollinearity exists whenever two or more of the predictors in a regression model are moderately or highly correlated. Multicollinearity is the condition when one predictor can be used to predict other. The basic problem is multicollinearity results in unstable estimation of coefficients which makes it difficult to access the effect of independent variable on dependent variable. Correlation plot was used in R and python to detect highly collinear variables.

The correlation graphs obtained are as follows

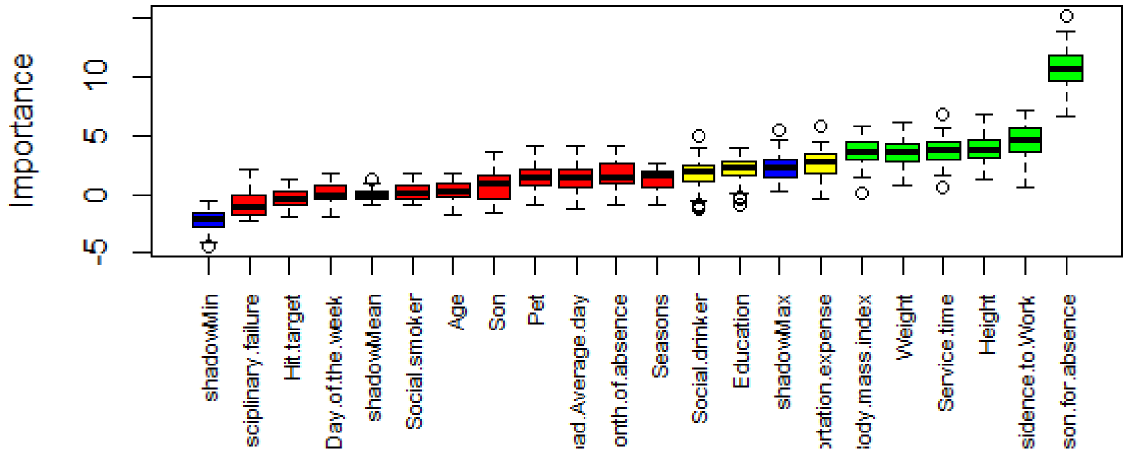


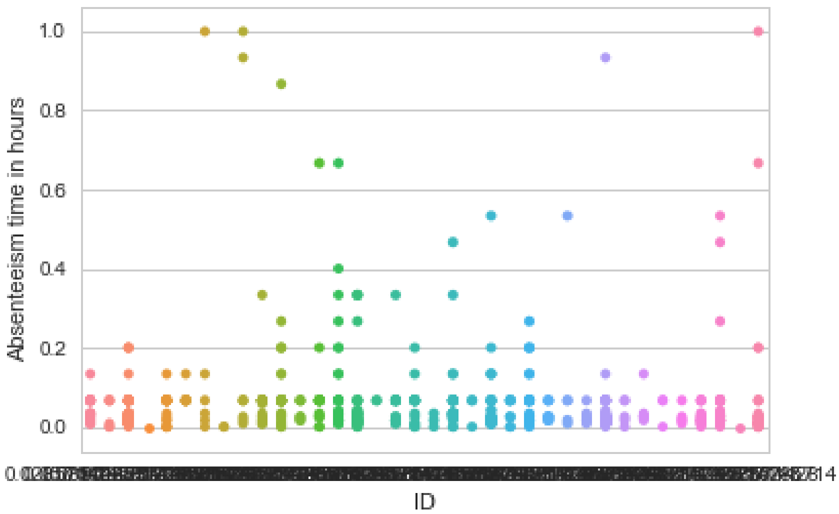


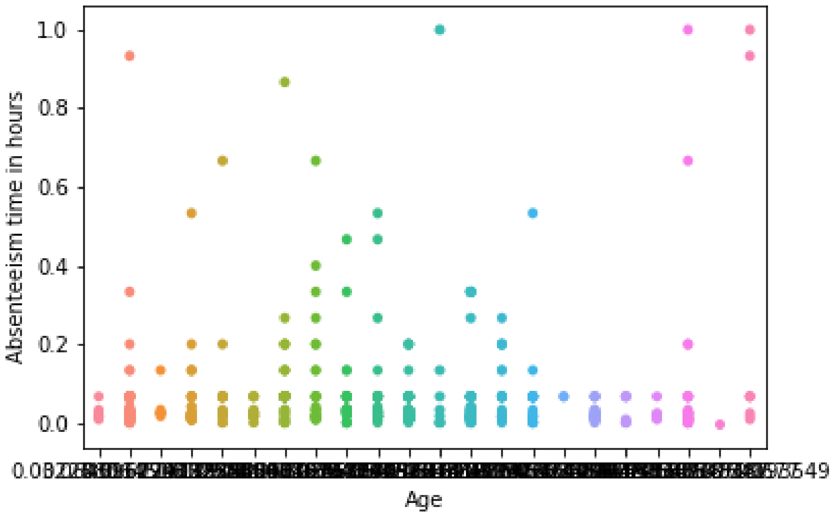
1. The weight predictor is highly correlated to body mass index
2. On applying the chi square test, the p values of the following variables are found to be greater than 0.05, Hit.target,Education,Social.smoker,Pet

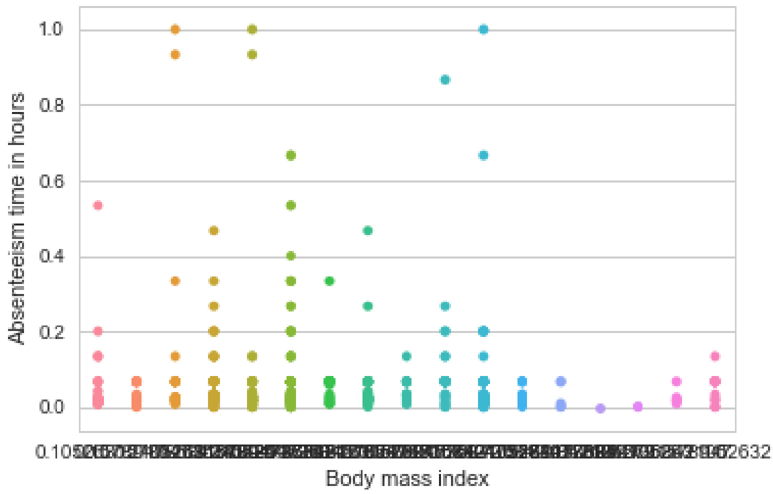
3. One of the assumptions of logistic regression is that logistic regression requires there to be little or no multicollinearity among the independent variables. This means that the independent variables should not be too highly correlated with each other. Due to this assumption, one the predictors from each set was removed when logistic learner was trained.

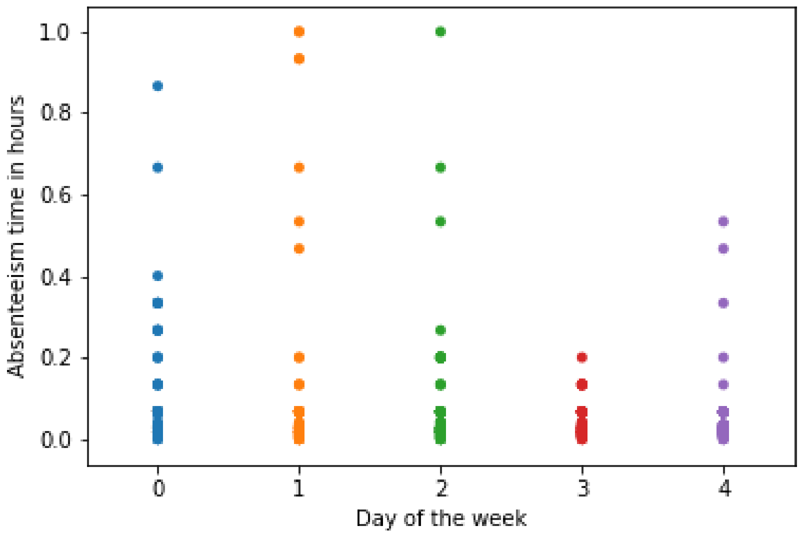
2.1.4 Analysis of Absenteeism time in hours with different predictors

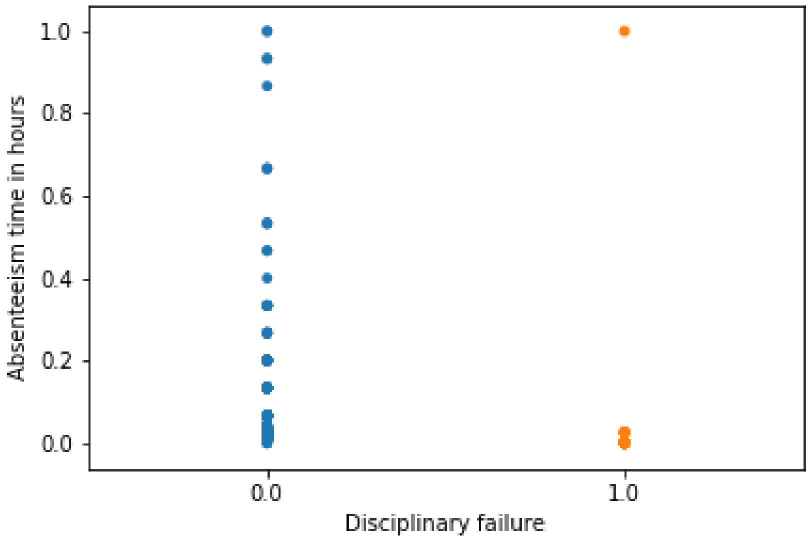


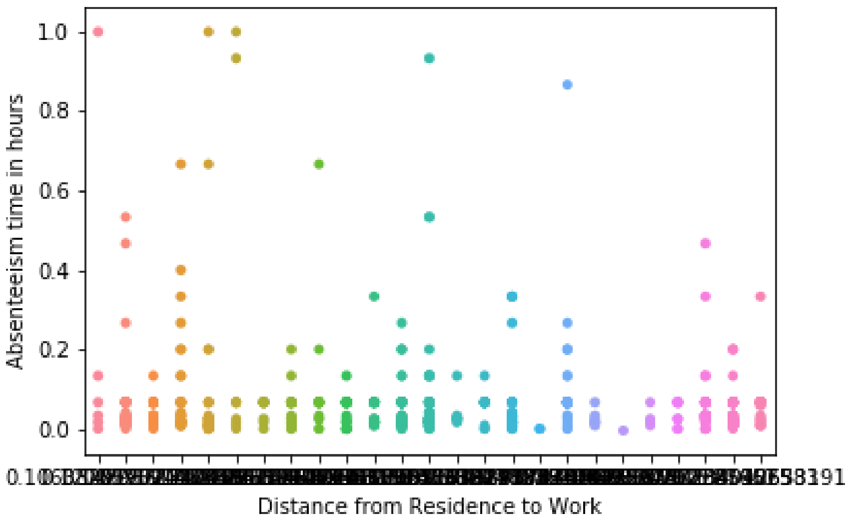


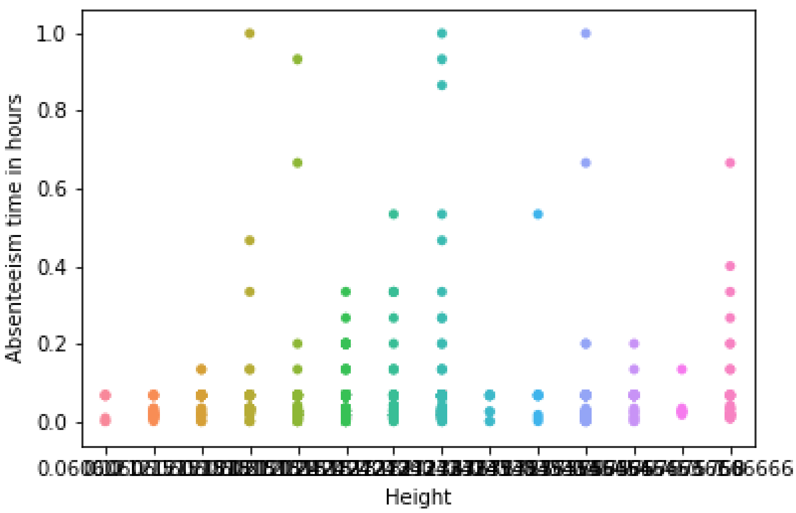

Absenteeism time in hours
Age

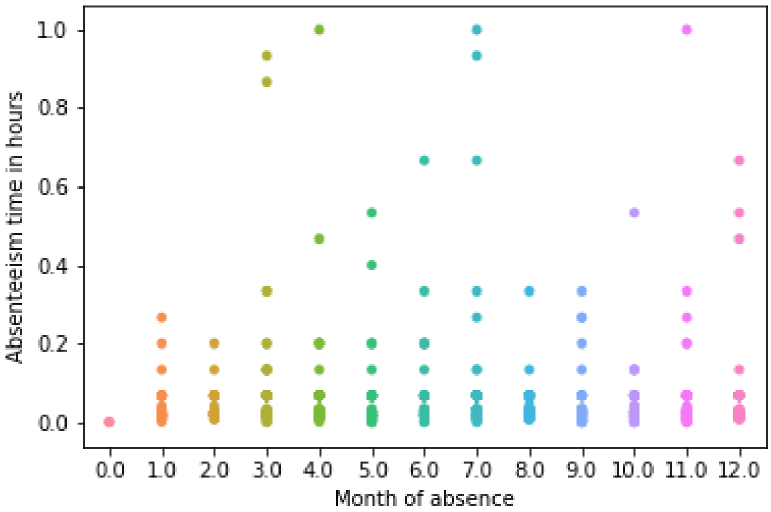

Absenteeism time in hours
Body mass index

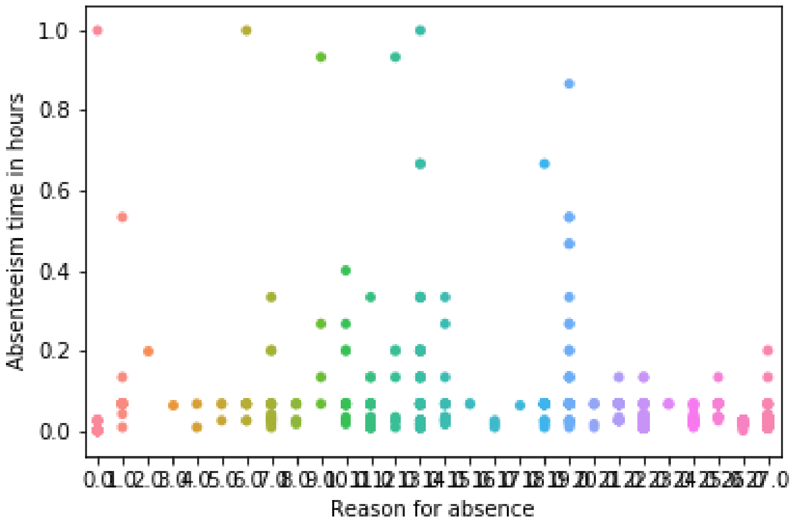

Absenteeism time in hours
Day of the week

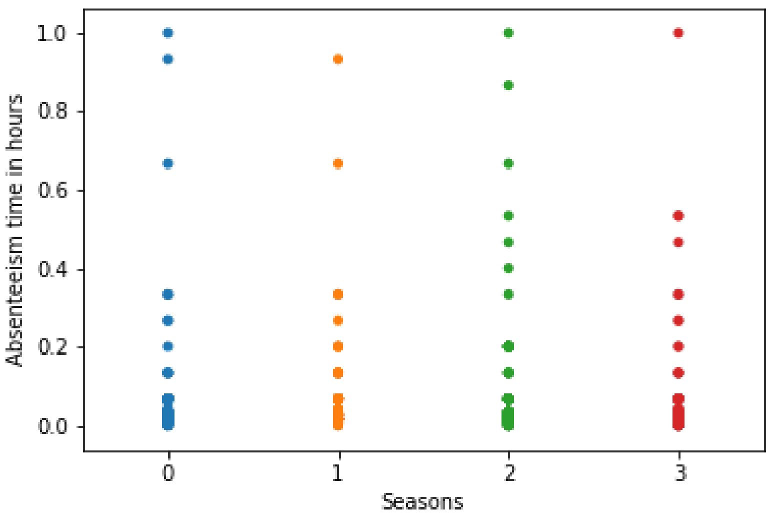

Absenteeism time in hours
Disciplinary failure
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0

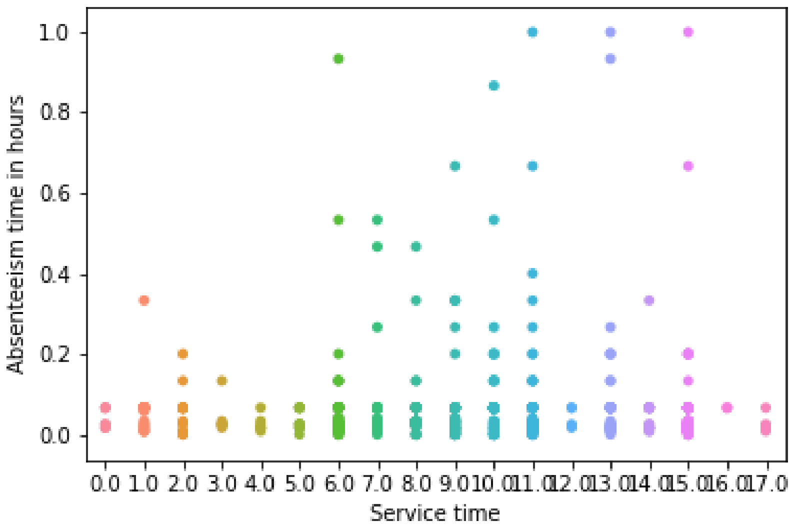

Absenteeism time in hours
Distance from Residence to Work
1.0
0.8
0.6
0.4
0.2
0.0


Absenteeism time in hours
Height
1.0
0.8
0.6
0.4
0.2
0.0

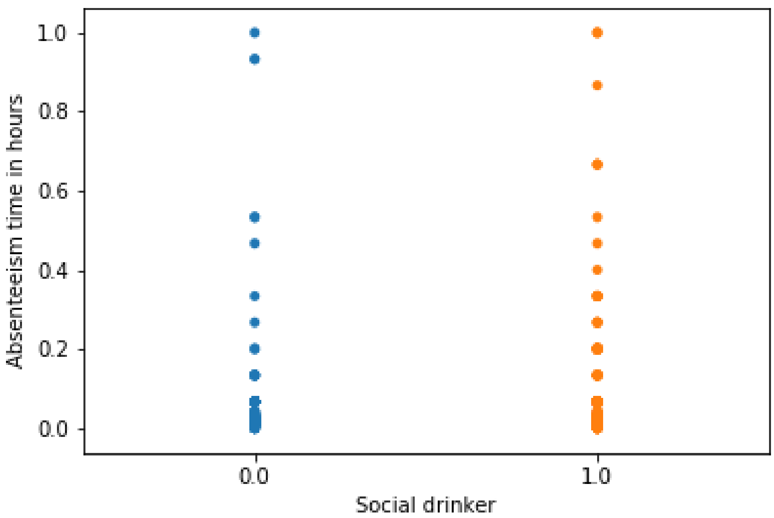

Absenteeism time in hours
Month of absence


Absenteeism time in hours
Reason for absence

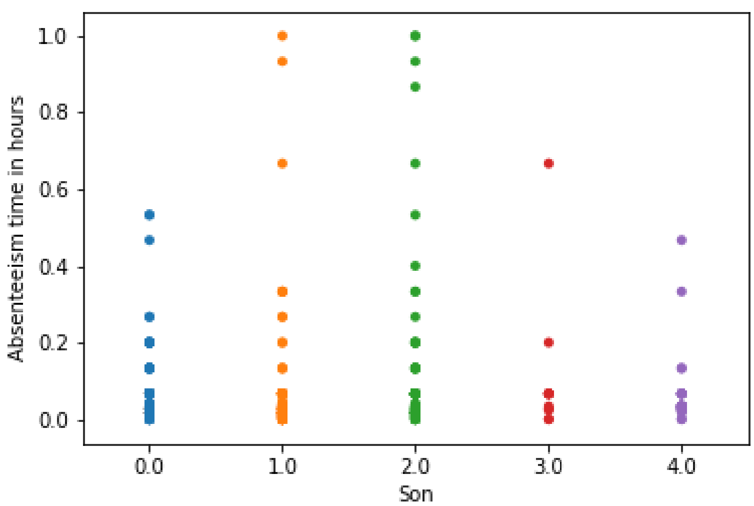

Absenteeism time in hours
Seasons



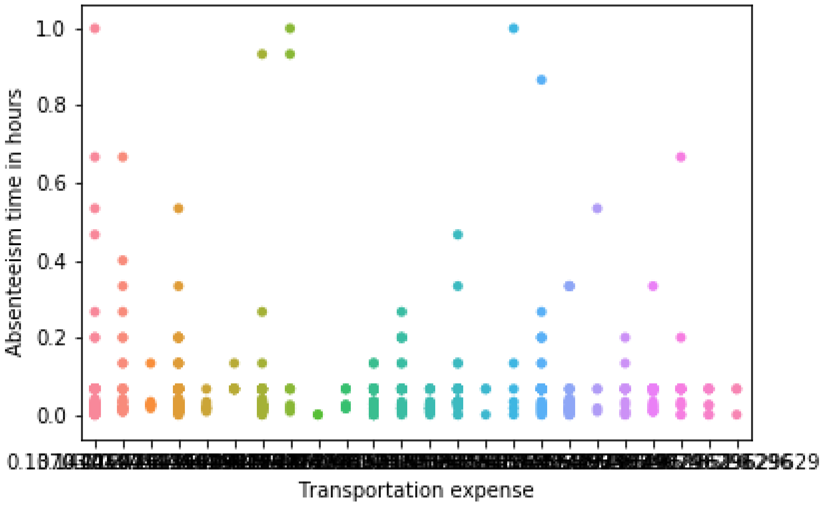
Absenteeism time in hours
Service time
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0
0.0 0.2 0.4 0.6 0.8 1.0



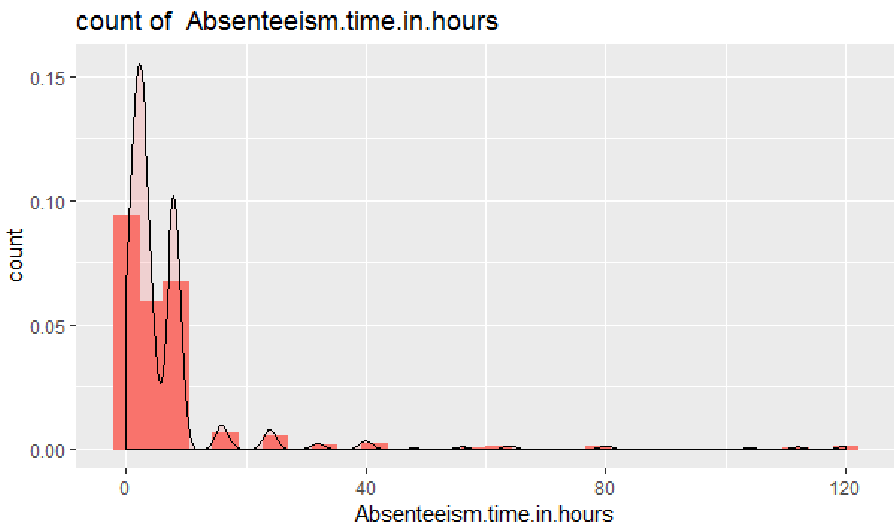
Absenteeism time in hours
Social drinker
0.0 1.0
0.0 0.2 0.4 0.6 0.8 1.0

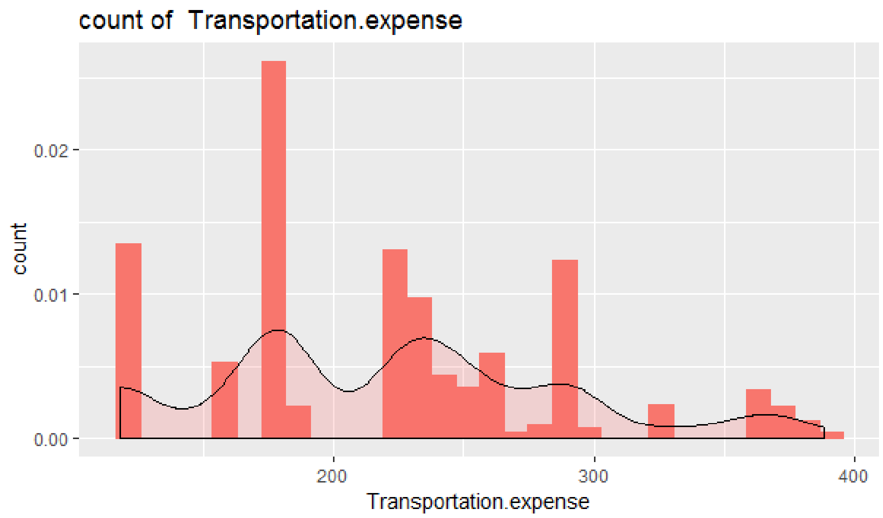


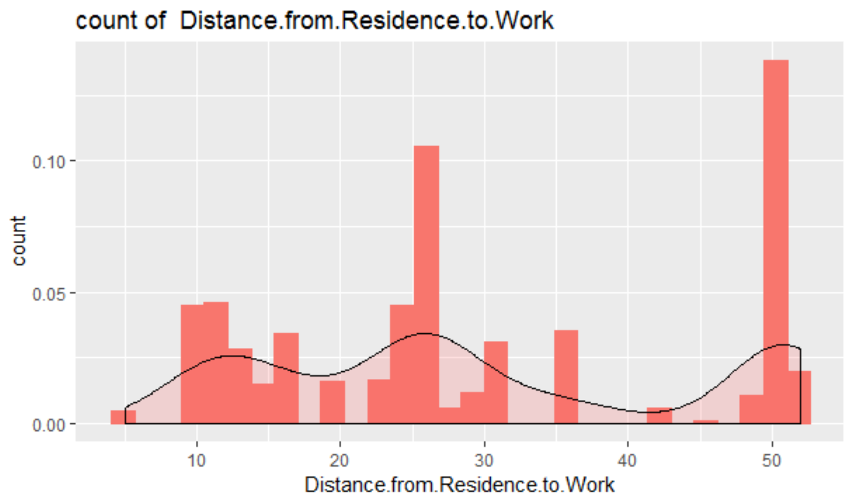
Absenteeism time in hours
Son
0.0 1.0 2.0 3.0 4.0
0.0 0.2 0.4 0.6 0.8 1.0

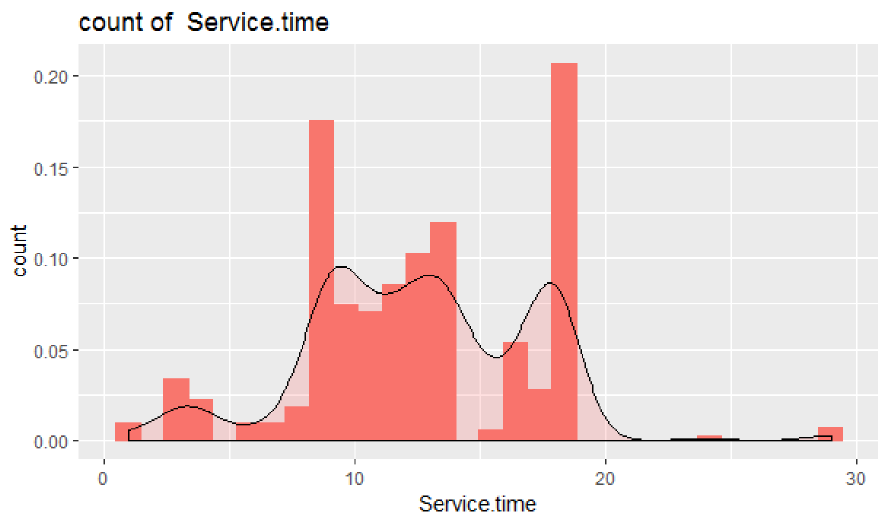


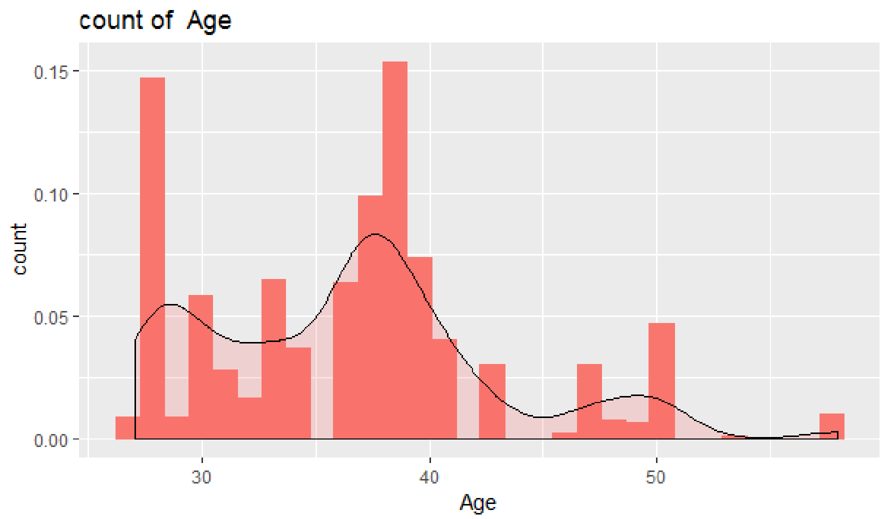
2.1.5 Univariate analysis of continuous variables

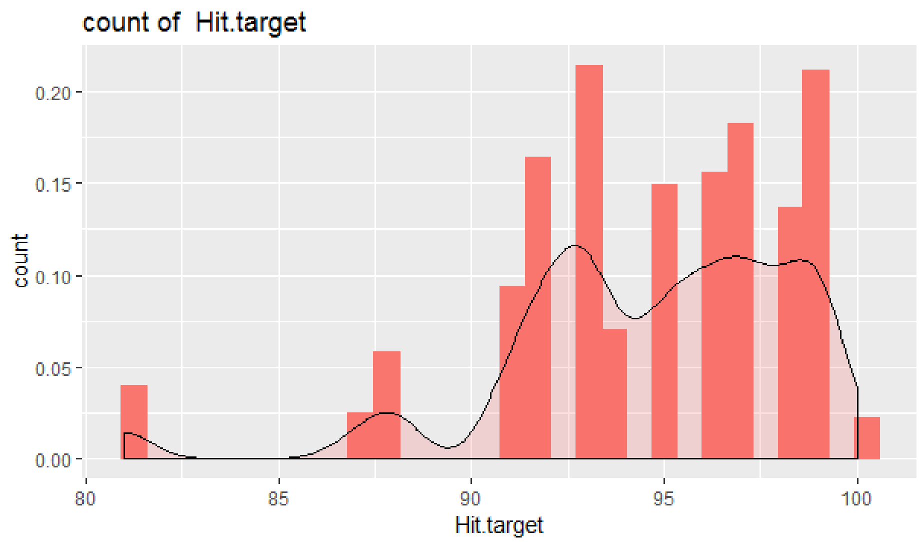


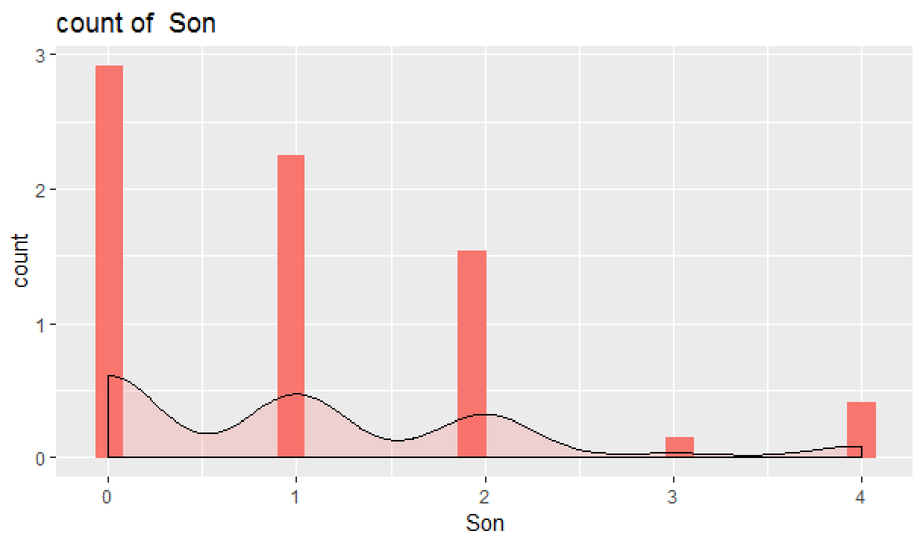

count of Transportation.expense
count
0.02
0.01
0.00
200
300
400
Transportation.expense

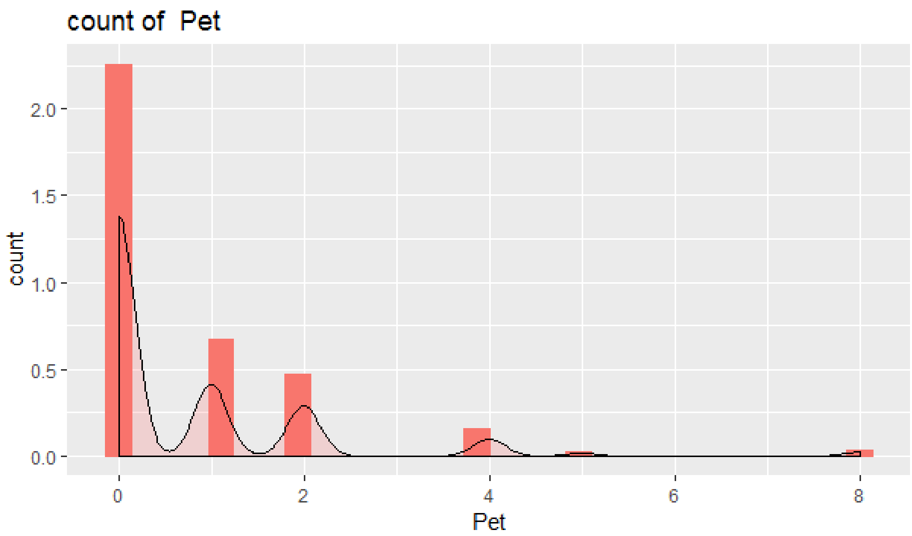

count of Distance.from.Residence.to.Work
count
0.10
0.05
0.00
10
20
30
40
50
Distance.from.Residence.to.Work

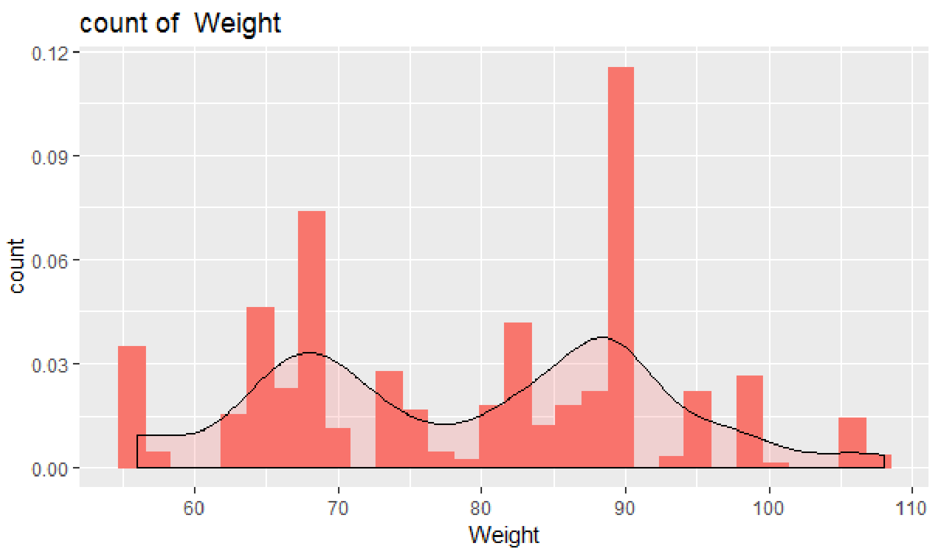

count of Service.time
count
0.20
0.15
0.10
0.05
0.00
0
10
20
30
Service.time

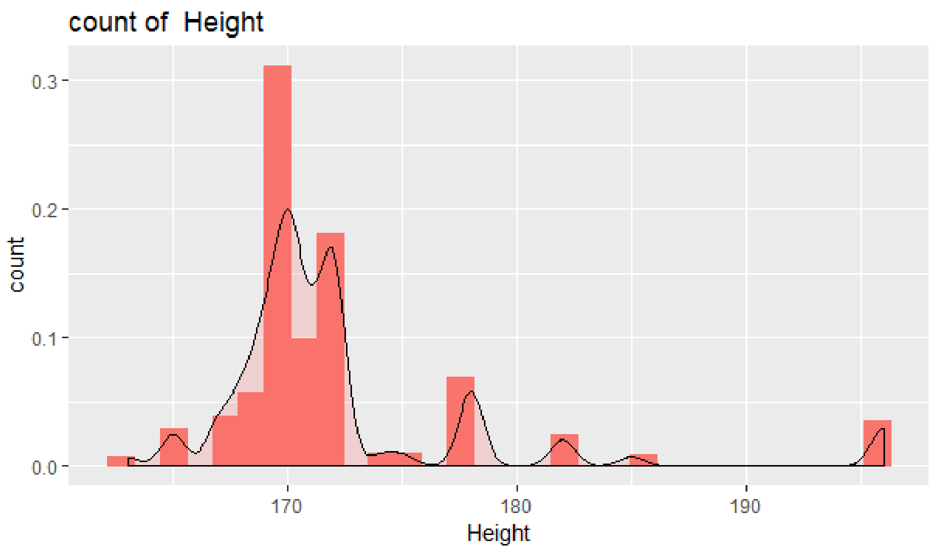

count of Age
0.15
0.10
0.05
0.00
count
30
40
50
Age

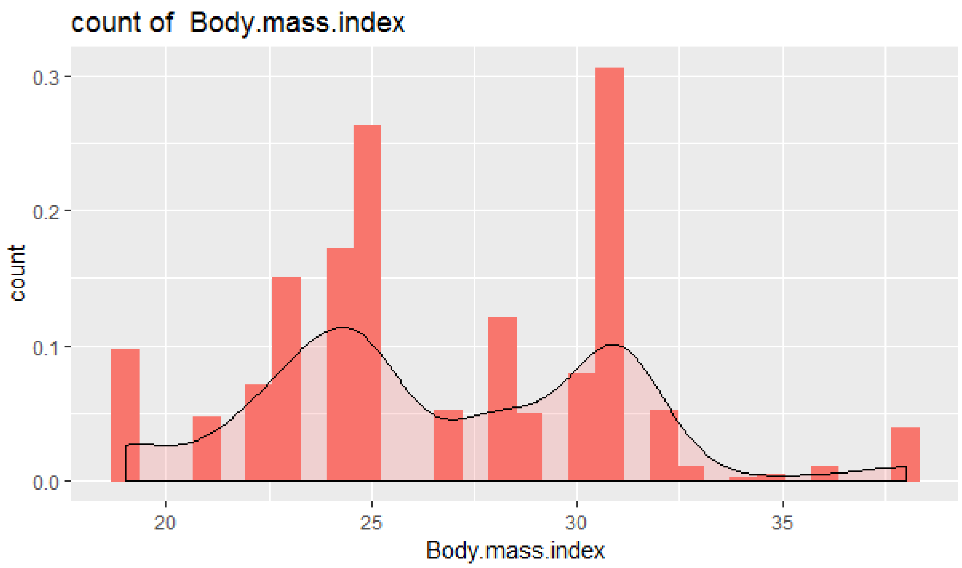

count of Hit.target
0.20
0.15
0.10
0.05
0.00
count
80
85
90
95
100
Hit.target


count of Son
3
2
1
0
count
0
1
2
3
4
Son

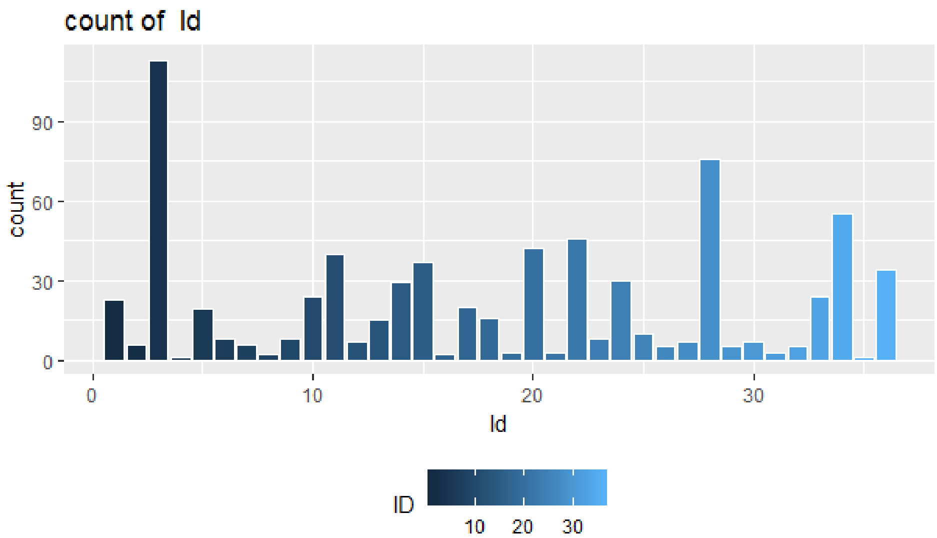

count of Pet
count
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
Pet

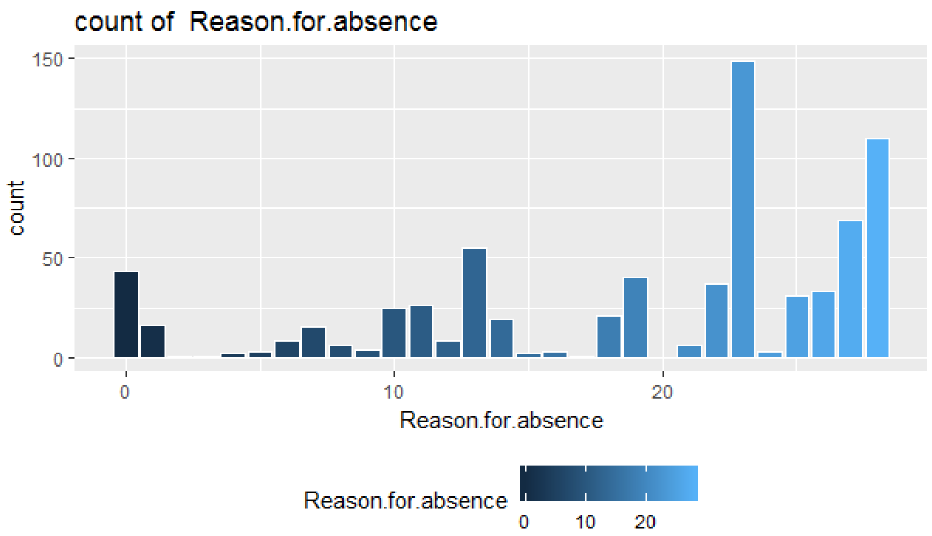

count of Weight
count
0.12
0.09
0.06
0.03
0.00
60
70
80
90
100
110
Weight

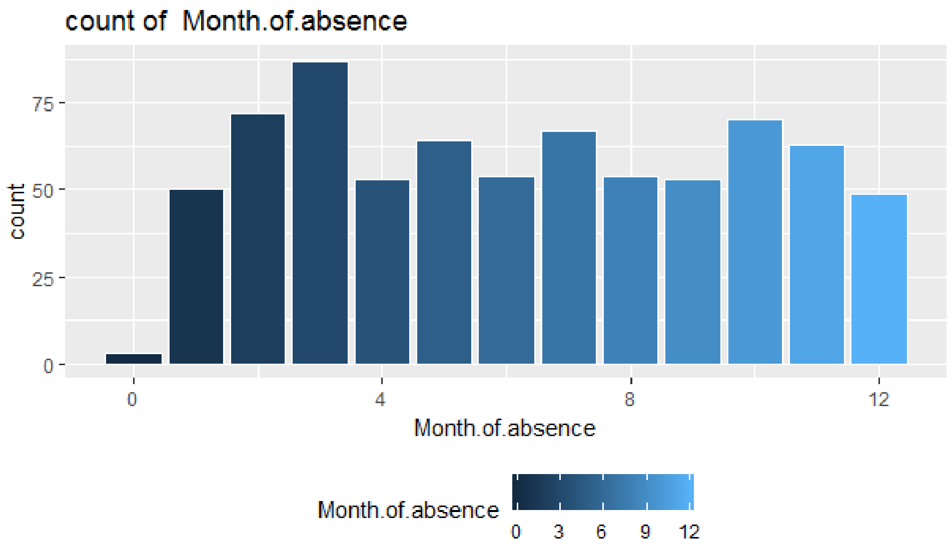

count of Height
count
0.3
0.2
0.1
0.0
170
180
190
Height

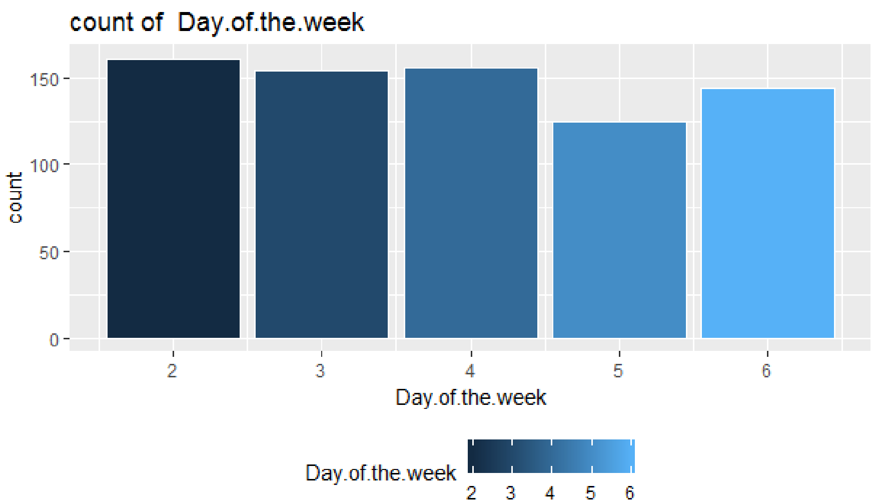


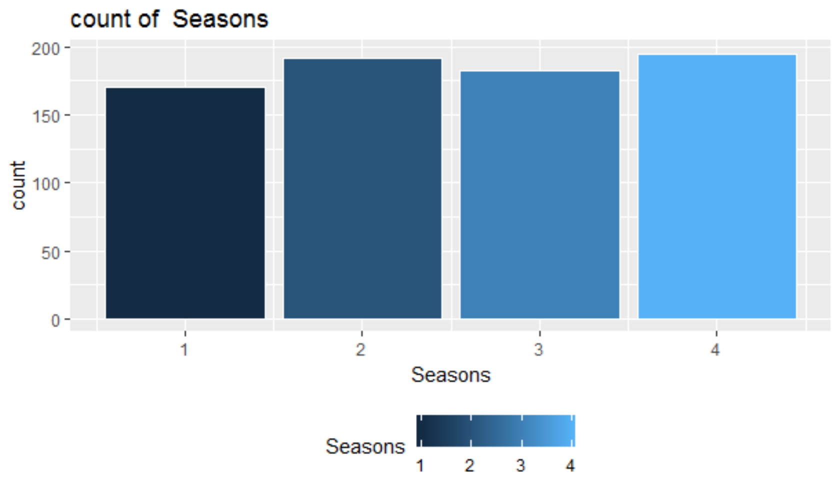
2.1.6 univariate analysis of categorical variables

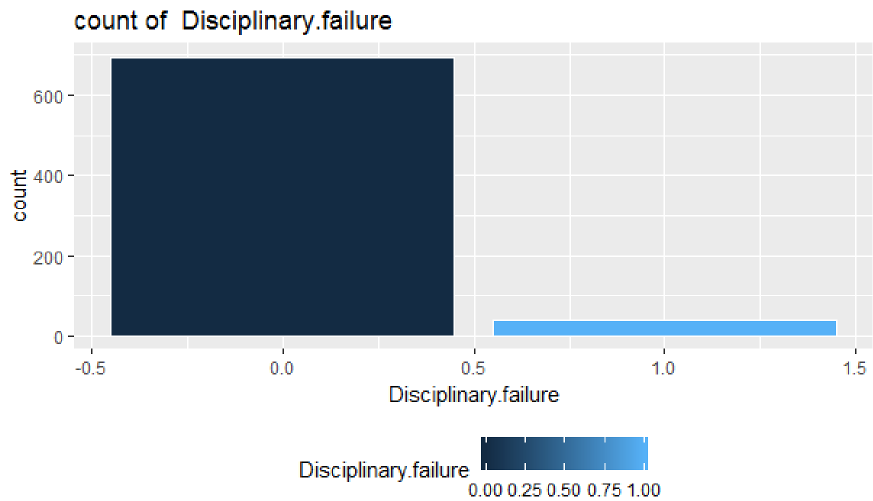


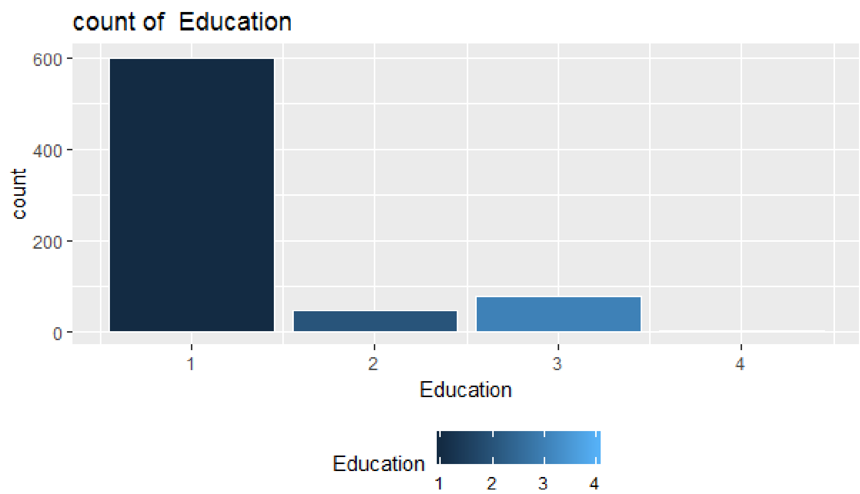

count of Reason.for.absence
count
Reason.for.absence
Reason.for.absence
0
10
20

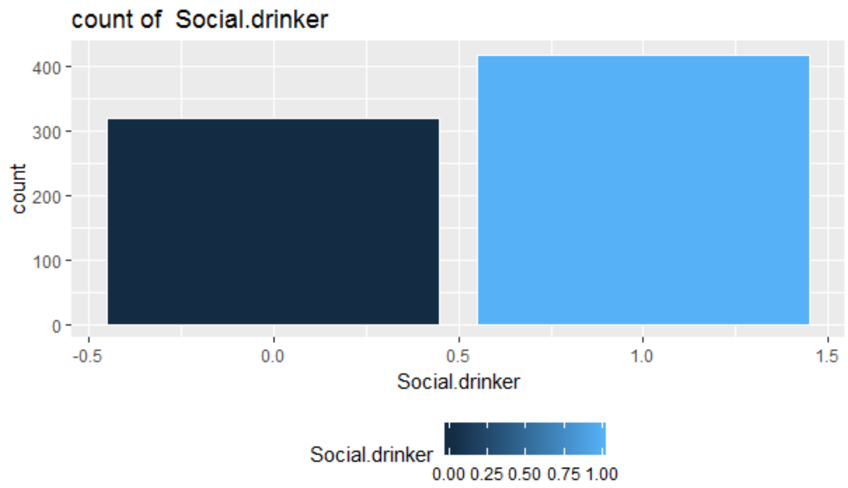

count of Month.of.absence
count
Month.of.absence
Month.of.absence
0
3
6
9
12

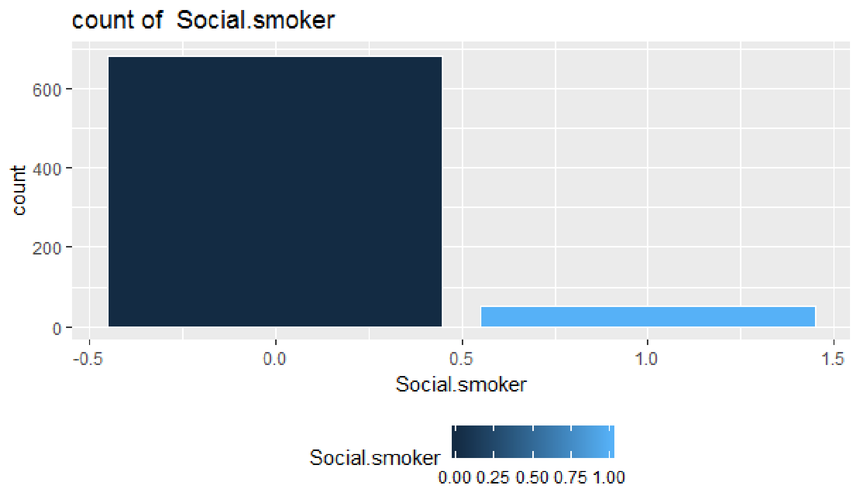

count of Day.of.the.week
count
Day.of.the.week
Day.of.the.week
2
3
4
5
6


count of Seasons
200
150
100
50
0
count
1
2
3
4
Seasons
Seasons
1
2
3
4


count of Disciplinary.failure
600
400
200
0
count
-0.5
0.0
0.5
1.0
1.5
Disciplinary.failure
Disciplinary.failure
0.00 0.25 0.50 0.75 1.00


count of Education
600
400
200
0
count
1
2
3
4
Education
Education
1
2
3
4





2.2 Modelling

Absenteeism at work is a regression problem. Here according to the problem statement, we are supposed to predict the loss encurred 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 optimised parameters.

2.2.1 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:

```
#Clean the environment

library(DataCombine)


rmExcept("Absenteeism_at_work")

#Divide data into train and test using stratified sampling method

set.seed(1234)

Absenteeism_at_work$description = NULL

library(caret)

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

train = Absenteeism_at_work[ train.index,]

test  = Absenteeism_at_work[-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])
```

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)
```

2.2.2 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

#Presdict 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)
```

Python implementation:

```python
#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)
```

Chapter 3

RESULT

As we can see, we have appied all the possible preprocessing analysis to our dataset to make it suitabl e 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 cod e of R

AND on comparing the values of decision tree and random forest, since the error rate of random fore st 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.

COMPLETE R CODE

```
#remove all the objects stored

rm(list=ls())

#set current working directory

setwd("C:/Users/SHRAVYA/Desktop/edwisor/project 1")

#install packages

install.packages(c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "In
formation", "MASS", "rpart", "gbm", "ROSE", "sampling", "DataCombine", "inTrees'"))


library(readxl)     # 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

Absenteeism_at_work = read.csv("Absenteeism_at_work_Project.csv", header = T, na.strings = c(" ", "", "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

Absenteeism_at_work = transform(Absenteeism_at_work, Month.of.absence =

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


str(Absenteeism_at_work)

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

Absenteeism_at_work$Reason.for.absence = as.factor(Absenteeism_at_work$Reason.for.absence)

Absenteeism_at_work$Month.of.absence = as.factor(Absenteeism_at_work$Month.of.absence)

Absenteeism_at_work$Day.of.the.week = as.factor(Absenteeism_at_work$Day.of.the.week)

Absenteeism_at_work$Seasons = as.factor(Absenteeism_at_work$Seasons)

Absenteeism_at_work$Service.time = as.factor(Absenteeism_at_work$Service.time)
```

```
Absenteeism_at_work$Hit.target = as.factor(Absenteeism_at_work$Hit.target)
Absenteeism_at_work$Disciplinary.failure = as.factor(Absenteeism_at_work$Disciplinary.failure)
Absenteeism_at_work$Education = as.factor(Absenteeism_at_work$Education)
Absenteeism_at_work$Son = as.factor(Absenteeism_at_work$Son)
Absenteeism_at_work$Social.drinker = as.factor(Absenteeism_at_work$Social.drinker)
Absenteeism_at_work$Social.smoker = as.factor(Absenteeism_at_work$Social.smoker)
Absenteeism_at_work$Pet = as.factor(Absenteeism_at_work$Pet)
Absenteeism_at_work$Work.load.Average.day  = as.numeric(Absenteeism_at_work$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(Absenteeism_at_work,Absenteeism.time.in.hours)
# outliers detected and replaced by NA

outlierKD(Absenteeism_at_work,Transportation.expense) #no outliers
outlierKD(Absenteeism_at_work,Distance.from.Residence.to.Work) #no outliers
outlierKD(Absenteeism_at_work,Service.time) #no outliers
outlierKD(Absenteeism_at_work,Age) #no outliers
outlierKD(Absenteeism_at_work,Work.load.Average.day) # 1 found and replaced with NA
outlierKD(Absenteeism_at_work,Hit.target) # 1 found and replaced with NA
outlierKD(Absenteeism_at_work,Son) # no outliers
outlierKD(Absenteeism_at_work,Pet) # no outliers
outlierKD(Absenteeism_at_work,Weight) # no outliers
outlierKD(Absenteeism_at_work,Height) # no outliers
outlierKD(Absenteeism_at_work,Body.mass.index) #no outliers
#................................missing value analysis..............................................
missing_val = data.frame(apply(Absenteeism_at_work,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$Missing_percentage/nrow(Absenteeism_at_work)) * 100
missing_val = missing_val[order(-missing_val$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
```

```
#Absenteeism_at_work[1,20]

#Absenteeism_at_work[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 k
nn imputation

library(DMwR)

Absenteeism_at_work = knnImputation(Absenteeism_at_work, k = 3)

sum(is.na(Absenteeism_at_work))



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


numeric_index = sapply(Absenteeism_at_work,is.numeric) #selecting only numeric


numeric_data = Absenteeism_at_work[,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(Absenteeism_at_work))+

          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(Absenteeism_at_work[,numeric_index], order = F,

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

Absenteeism_at_work$Absenteeism.time.in.hours = as.factor(Absenteeism_at_work$Absenteeism.time.in.hours)


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

```
factor_index = sapply(Absenteeism_at_work,is.factor)

factor_data = Absenteeism_at_work[,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])))

}


Absenteeism_at_work$Absenteeism.time.in.hours = as.numeric(Absenteeism_at_work$Absenteeism.time.in.hours)

#.................................feature reduction...................................

## Dimension Reduction

Absenteeism_at_work = subset(Absenteeism_at_work,

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

#Feature Scaling

#Normality check

qqnorm(Absenteeism_at_work$Absenteeism.time.in.hours )

hist(Absenteeism_at_work$Absenteeism.time.in.hours )

str(Absenteeism_at_work)

#Normalisation

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

         "Absenteeism.time.in.hours")


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])))

}

# ............................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(Absenteeism_at_work,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(Absenteeism_at_work, 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(Absenteeism_at_work,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(Absenteeism_at_work,Absenteeism.time.in.hours,"Absenteeism.time.in.hours")

univariate_analysis(Absenteeism_at_work,Transportation.expense,"Transportation.expense")

univariate_analysis(Absenteeism_at_work,Distance.from.Residence.to.Work,
            "Distance.from.Residence.to.Work")

univariate_analysis(Absenteeism_at_work,Service.time,"Service.time")

univariate_analysis(Absenteeism_at_work,Age,"Age")

univariate_analysis(Absenteeism_at_work,Work.load.Average.day ,"Work.load.Average.day ")

univariate_analysis(Absenteeism_at_work,Hit.target ,"Hit.target")

univariate_analysis(Absenteeism_at_work,Son ,"Son")

univariate_analysis(Absenteeism_at_work,Pet ,"Pet")
```

```
univariate_analysis(Absenteeism_at_work,Weight ,"Weight")

univariate_analysis(Absenteeism_at_work,Height ,"Height")

univariate_analysis(Absenteeism_at_work,Body.mass.index ,"Body.mass.index")



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

univariate_catogrical(Absenteeism_at_work,ID,"Id")

univariate_catogrical(Absenteeism_at_work,Reason.for.absence,"Reason.for.absence")

univariate_catogrical(Absenteeism_at_work,Month.of.absence,"Month.of.absence")

univariate_catogrical(Absenteeism_at_work,Day.of.the.week,"Day.of.the.week")

univariate_catogrical(Absenteeism_at_work,Seasons,"Seasons")

univariate_catogrical(Absenteeism_at_work,Disciplinary.failure,"Disciplinary.failure")

univariate_catogrical(Absenteeism_at_work,Education,"Education")

univariate_catogrical(Absenteeism_at_work,Social.drinker,"Social.drinker")

univariate_catogrical(Absenteeism_at_work,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
Absenteeism_at_work$index = 1:740
# #Extract subset from whole data
systematic_data = Absenteeism_at_work[which(Absenteeism_at_work$index %in% lis),]
#..............................Model Development..........................................................#
#Clean the environment
library(DataCombine)

rmExcept("Absenteeism_at_work")
#Divide data into train and test using stratified sampling method
set.seed(1234)
Absenteeism_at_work$description = NULL
library(caret)
train.index = createDataPartition(Absenteeism_at_work$Absenteeism.time.in.hours, p = .80, list = FALSE)
train = Absenteeism_at_work[ train.index,]
test  = Absenteeism_at_work[-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

#Presdict 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
```

COMPLETE 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);
```

```python
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')
```