EMPLOYEE ABSENTEEISM

DATA SCIENCE PROJECT

AARTI NIRMAL

Contents

1.Introduction

1.1 Project Description

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.

1.2 Problem statement

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?

1.3 Dataset

Load excel file to analysis the data and solve the problem.

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	26.0	7.0	3	1	289.0	36.0	13.0	33.0	239554.0	97.0
36	0.0	7.0	3	1	118.0	13.0	18.0	50.0	239554.0	97.0
3	23.0	7.0	4	1	179.0	51.0	18.0	38.0	239554.0	97.0
7	7.0	7.0	5	1	279.0	5.0	14.0	39.0	239554.0	97.0
11	23.0	7.0	5	1	289.0	36.0	13.0	33.0	239554.0	97.0

Disciplinary failure	Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
0.0	1.0	2.0	1.0	0.0	1.0	90.0	172.0	30.0	4.0
1.0	1.0	1.0	1.0	0.0	0.0	98.0	178.0	31.0	0.0
0.0	1.0	0.0	1.0	0.0	0.0	89.0	170.0	31.0	2.0
0.0	1.0	2.0	1.0	1.0	0.0	68.0	168.0	24.0	4.0
0.0	1.0	2.0	1.0	0.0	1.0	90.0	172.0	30.0	2.0

So we have 740 observations and 21 variables

Count of variables

Independent Variables : 20

Dependent Variables : 1(target variable)

Data Attributes :

1. Individual identification (ID)
2. Reason for absence (ICD). Absences attested by the International Code of Diseases (ICD) stratified into 21 categories (I to XXI) as follows:

i. Certain infectious and parasitic diseases
ii. Neoplasms
iii. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
iv. Endocrine, nutritional and metabolic diseases
v. Mental and behavioural 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)

So we have 20 independent variables and 1 is target variables i.e. Absenteeism time in hours. And the **target variable is continuous so it is a regression problem.**

1.4 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense of the data you have and then figure out what questions you want to ask and how to frame them, as well as how best to manipulate your available data sources to get the answers you need.

In our data set we have 21 variables which are either `int64` or `float64`

```
ID                                  int64
Reason for absence                float64
Month of absence                  float64
Day of the week                     int64
Seasons                             int64
Transportation expense            float64
Distance from Residence to Work   float64
Service time                      float64
Age                               float64
Work load Average/day             float64
Hit target                        float64
Disciplinary failure              float64
Education                         float64
Son                               float64
Social drinker                    float64
Social smoker                     float64
Pet                               float64
Weight                            float64
Height                            float64
Body mass index                   float64
Absenteeism time in hours         float64
dtype: object
```

Number of unique values of each variable :

```
1. ID                                  36
2. Reason for absence                  28
3. Month of absence                    13
4. Day of the week                      5
5. Seasons                              4
6. Transportation expense              24
7. Distance from Residence to Work     25
8. Service time                        18
9. Age                                 22
10.  Work load Average/day              38
11.  Hit target                         13
12.  Disciplinary failure                2
13.  Education                           4
14.  Son                                 5
15.  Social drinker                      2
16.  Social smoker                       2
17.  Pet                                 6
18.  Weight                             26
19.  Height                             14
```

```
   20.  Body mass index                    17
   21.  Absenteeism time in hours          19
dtype: int64
```

so we have first variables i.e. ID which is not carrying any significant information for analysis so we will remove that variable.
There are only 12 months but in our data set there is 13. 0 is no any month so we will replace 0 with nan value.

Now we have 20 variables. 10 are categorical variables and 10 are continuous variables.

Categorical Variables :

```
   1. Month of absence                   12
   2. Reason for absence                 28
   3. Day of the week                     5
   4. Seasons                             4
   5. Disciplinary failure                2
   6. Education                           4
   7. Son                                 5
   8. Social drinker                      2
   9. Social smoker                       2
   10.  Pet                                 6
```

Continuous variables :

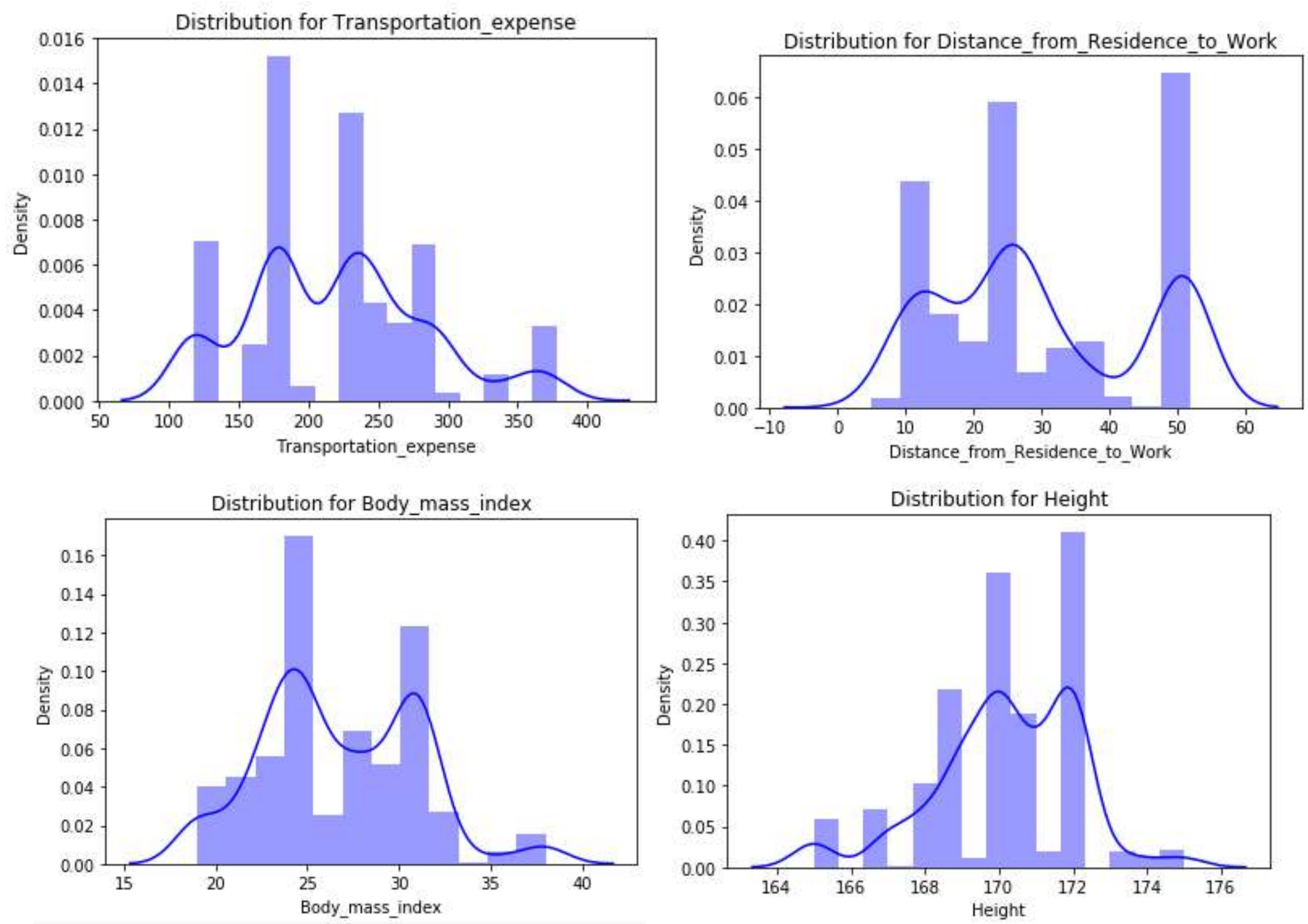
```
   1. Transportation expense             24
   2. Distance from Residence to Work    25
   3. Service time                       18
   4. Age                                22
   5. Work load Average/day              38
   6. Hit target                         13
   7. Weight                             26
   8. Height                             14
   9. Body mass index                    17
   10.  Absenteeism time in hours         19
```

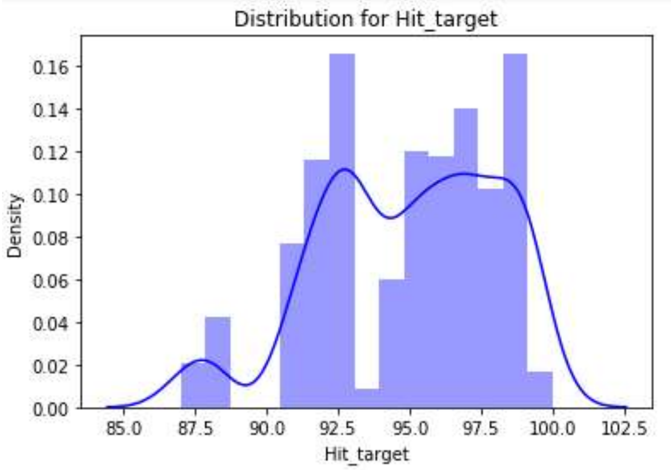
2.Methodology

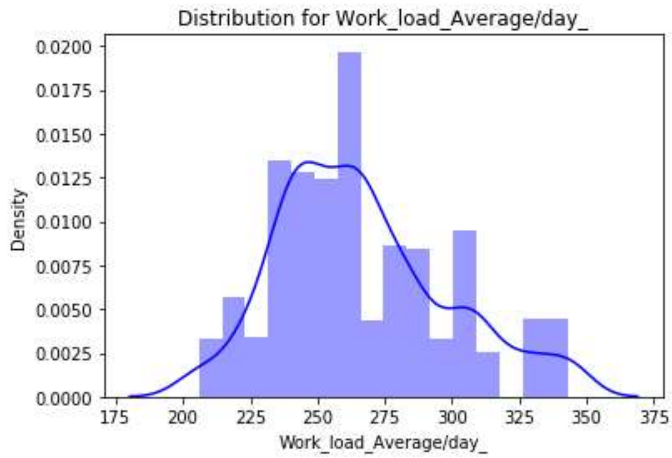
2.1 Data Pre Processing

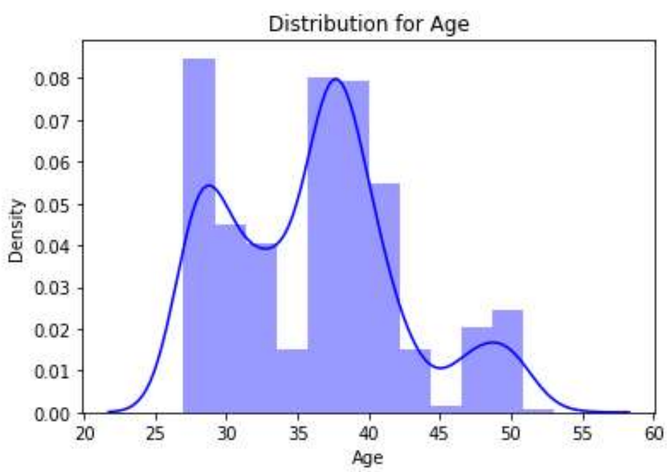
Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

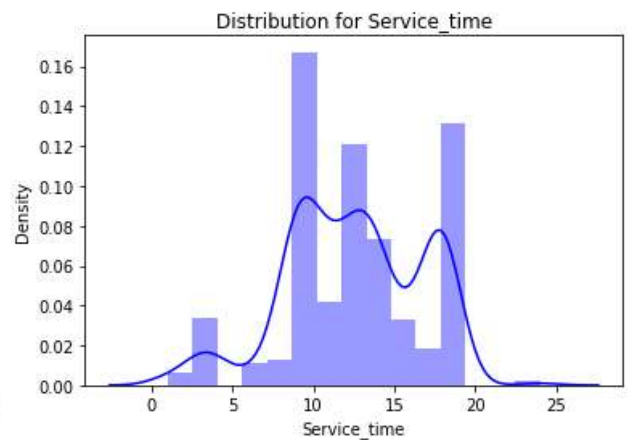
For further process we firstly need to find the distribution of variable because mostly regression analysis r equire normally distributed data.

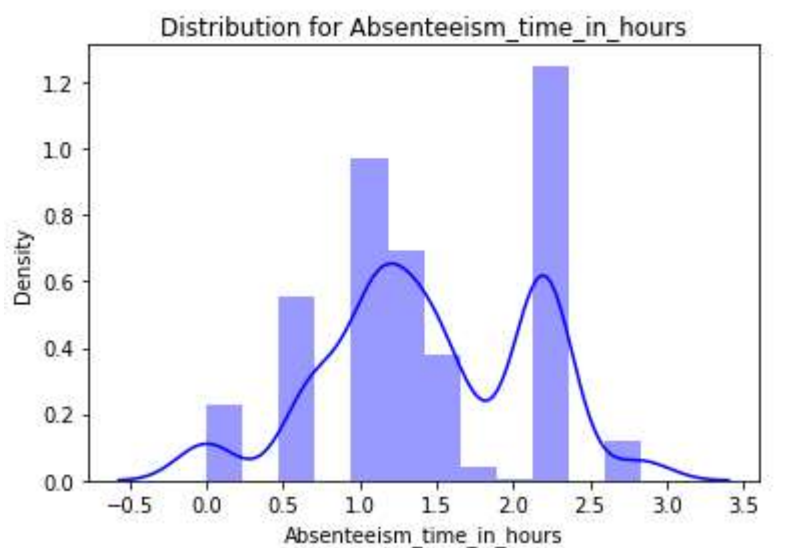



Distribution for Hit_target
Density
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16
85.0 87.5 90.0 92.5 95.0 97.5 100.0 102.5
Hit_target


Distribution for Work_load_Average/day_
Density
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200
175 200 225 250 275 300 325 350 375
Work_load_Average/day_


Distribution for Age
Density
0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08
20 25 30 35 40 45 50 55 60
Age

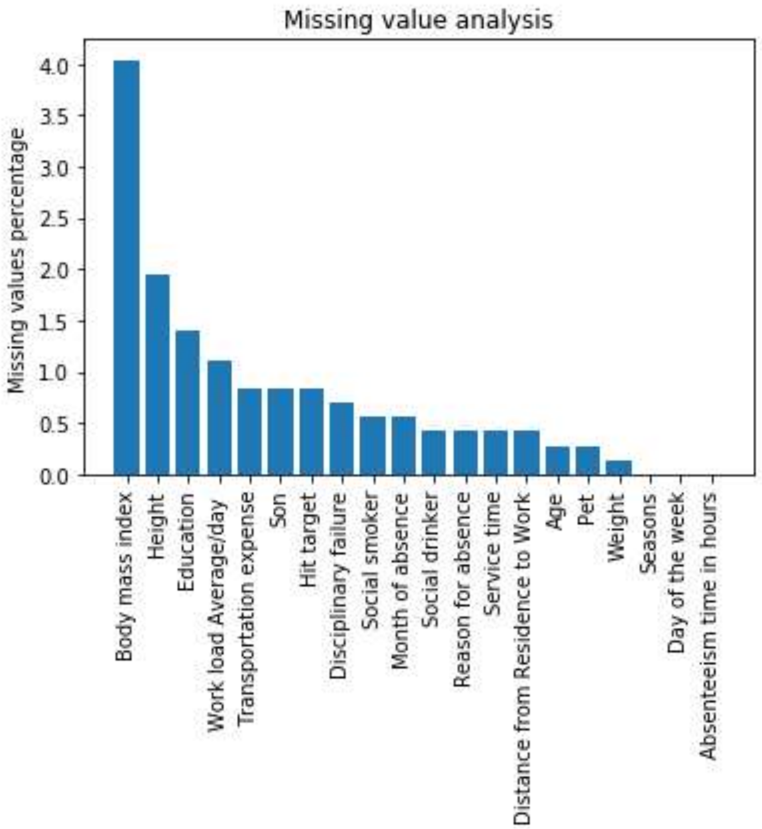

Distribution for Service_time
Density
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16
0 5 10 15 20 25
Service_time


Distribution for Absenteeism_time_in_hours
Density
0.0 0.2 0.4 0.6 0.8 1.0 1.2
−0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Absenteeism_time_in_hours

2.1.1 Missing Value Analysis

When we get the data for analysis, there is chances that values of some observation are blank, that blank value called missing value. These missing values affect in result, that may be in high or low. If a variable contains below 30% of missing values than we can consider that variable in our analysis otherwise we need to reject that.

So in our data set there are missing values exist



So in above figure we can get to know that there are missing values but below 30%, so we will consider the variables and impute missing values.

After applying different methods we conclude that KNN imputation is best for impute missing value in this data set. So we will apply KNN imputation for missing value imputation.

2.1.2 Outlier Analysis

Observation which lies an abnormal distance from other values from data set is known as outlier. These type of values major effects on our analysis and the result. For this we need to do outlier analysis.

In outlier analysis we need to first detect the outlier with some technique like graphical or statistical and then replace it with NA. In our analysis we use **Box Plot** method to detect the outlier, because it is graphical technique so it is easy to find out outlier. The box plot technique represents distribution of values in 25th, 50th and 75th percentile. And values which are not lies in between 25th to 75th percentile that is outlier.

From below figures 2.1.2.1, 2.1.2.2 and 2.1.2.3 we apply box plot for find outliers and we can observed that **Age, work.load.Average.day, Hit.target, Transportation expense,Service time, Height** have outliers.
Now we will replace these outliers with NA values and then apply KNN imputation to impute missing values.

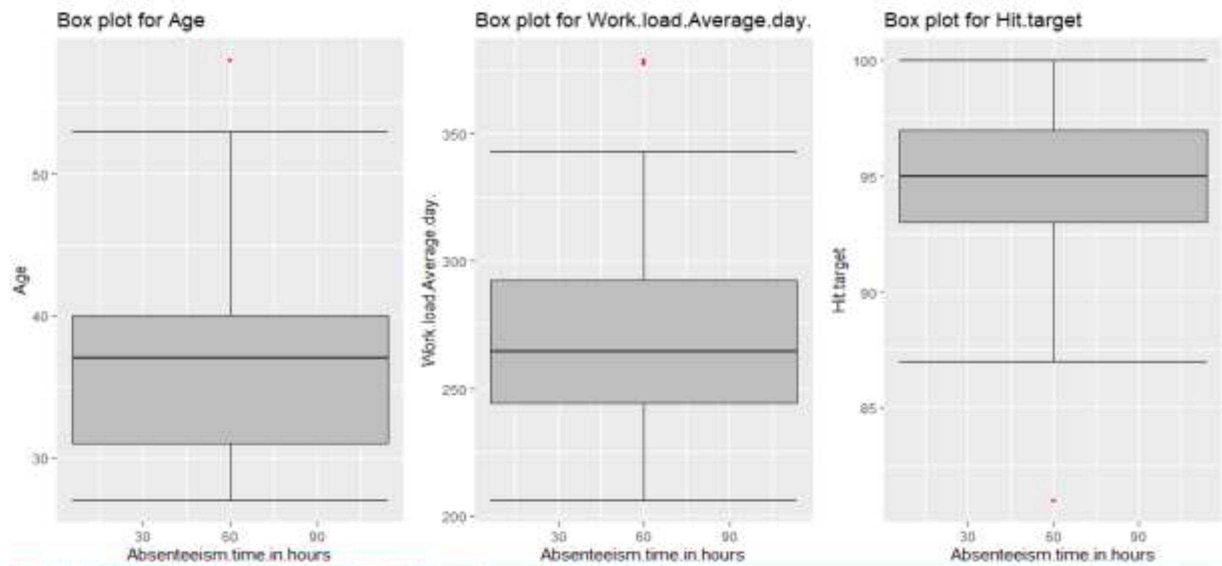


Figure : 2.1.2.1

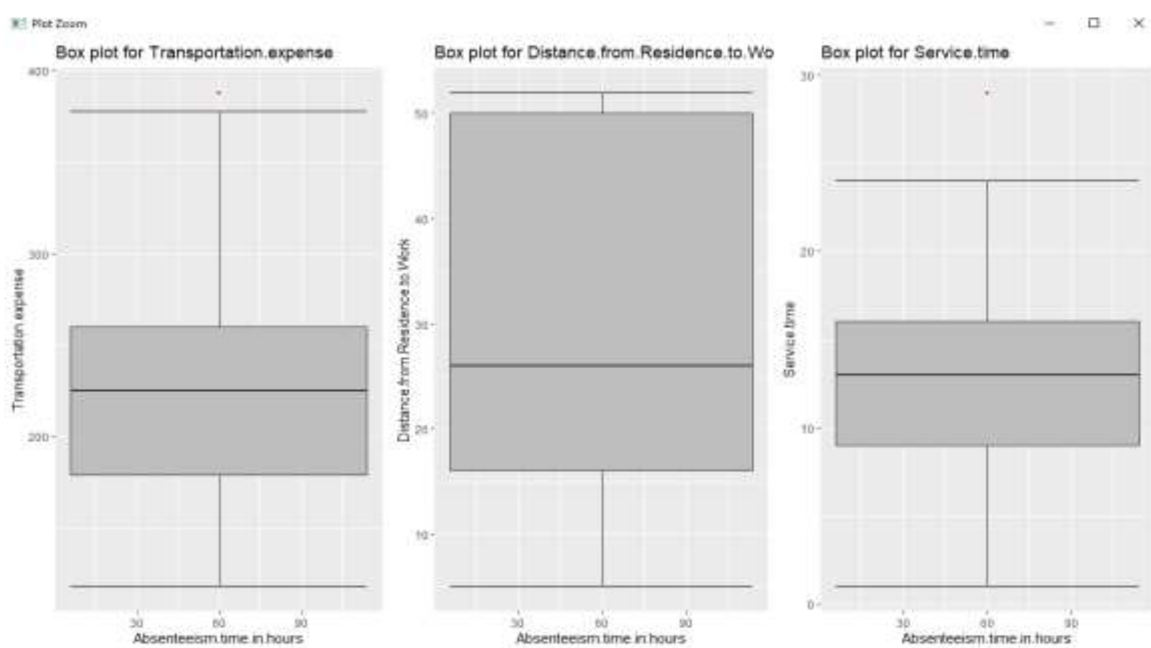


Figure : 2.1.2.2

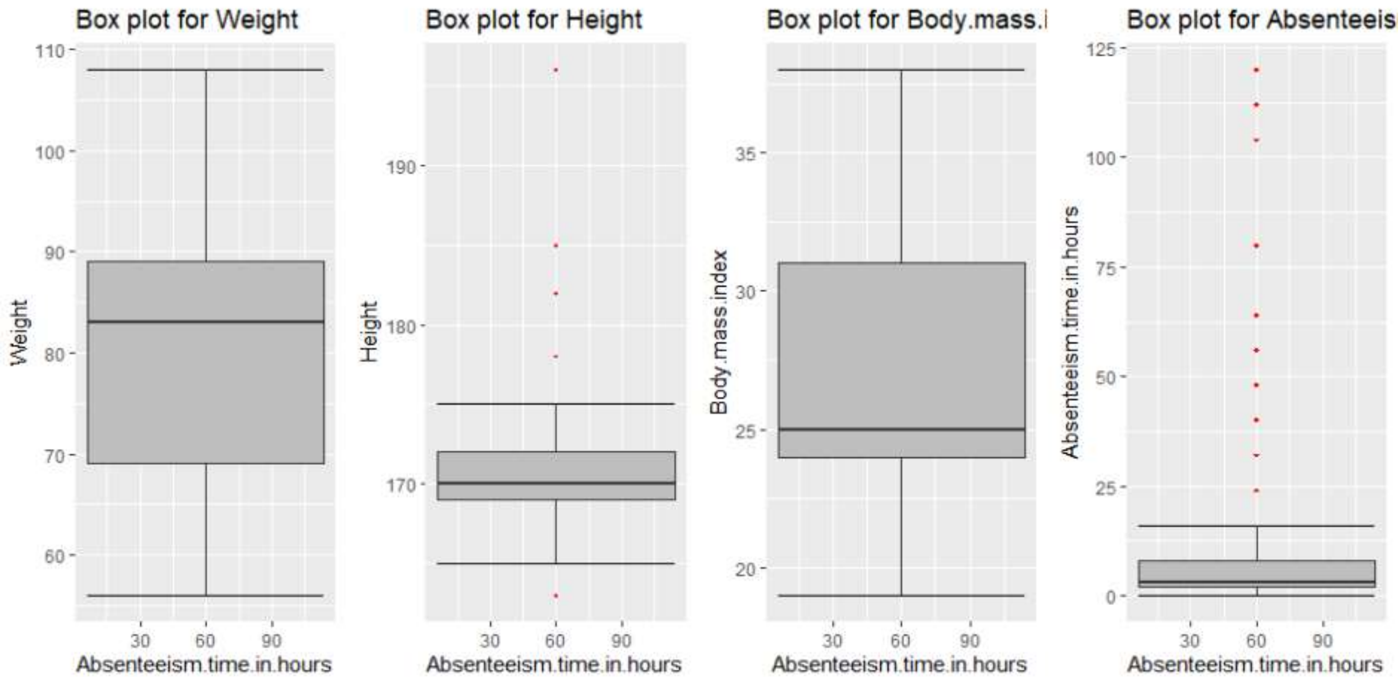
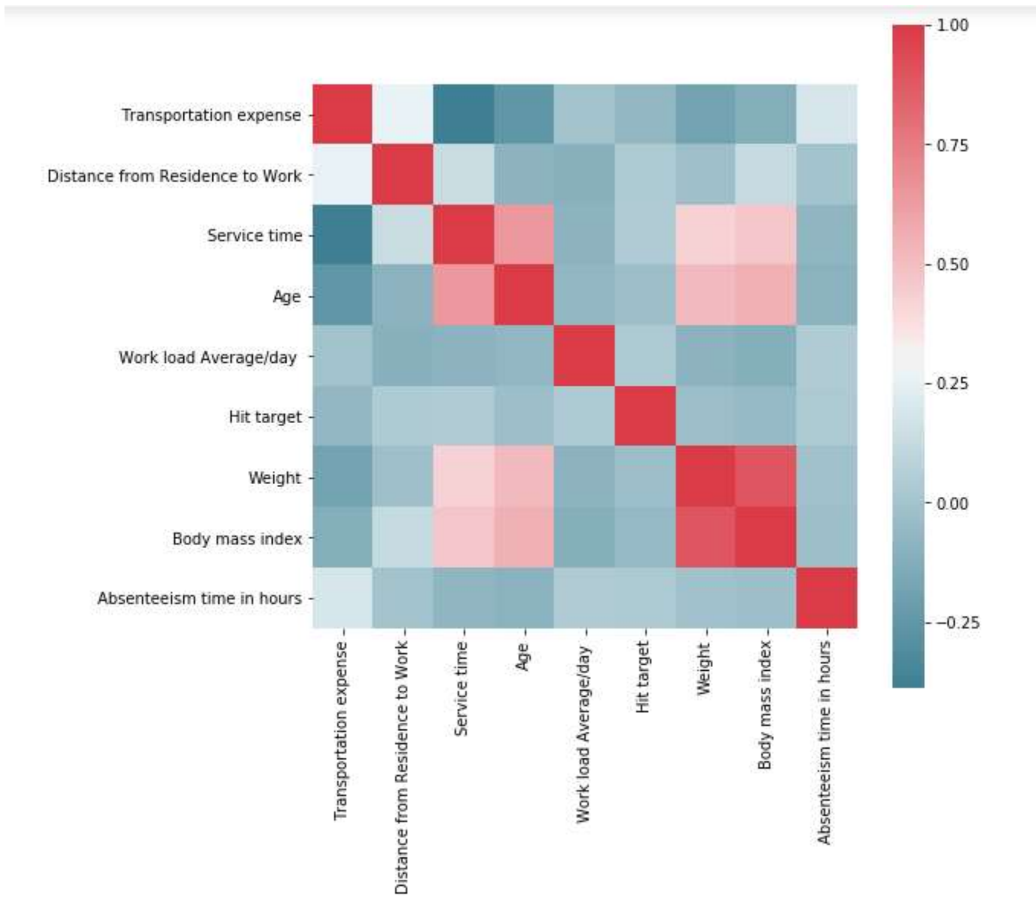


Figure : 2.1.2.3

2.1.3 Feature Selection

Feature selection is another technique of pre processing of data. It is clear from its name that select the features from data. It is basically stands for extracting the relevant and meaningful feature out of the data. Its main objective is to remove unrelated attributes from data and reduce the complexity. Because not all the features are carrying significant information or some of them are carrying same information so for that we need to apply feature selection technique. For this we use **Correlation analysis** for numeric variables and **ANOVA test** for categorical variables.

Correlation Analysis for continuous variables



From the above plot we can conclude that **weight** and **body mass index** are highly correlated with each other so we will remove one of them. We will go with **body mass index** and remove **weight**.

Now we will apply ANOVA test for categorical variables

```
                         sum_sq     df          F        PR(>F)
Reason_for_absence  3526.306799   26.0  19.77275  1.355448e-65
Residual            4444.825053  648.0       NaN           NaN
                       sum_sq     df         F    PR(>F)
Month_of_absence     0.118210    1.0  0.009981  0.920451
Residual          7971.013641  673.0       NaN       NaN
                      sum_sq     df         F    PR(>F)
Day_of_the_week    54.732404    1.0  4.652988  0.031353
Residual         7916.399448  673.0       NaN       NaN
               sum_sq     df         F    PR(>F)
Seasons     21.215651    1.0  1.796011  0.180648
Residual  7949.916200  673.0       NaN       NaN
                           sum_sq     df          F        PR(>F)
Disciplinary_failure   622.624853    1.0  57.021995  1.408164e-13
Residual              7348.506998  673.0        NaN           NaN
                sum_sq     df         F    PR(>F)
Education     7.770082    1.0  0.656666  0.418026
Residual   7963.361770  673.0       NaN       NaN
               sum_sq     df          F    PR(>F)
Son        204.153341    1.0  17.689659  0.00003
```

```
Residual  7766.978511  673.0        NaN       NaN
                    sum_sq     df          F    PR(>F)
Social_drinker    45.528306    1.0  3.866021  0.049684
Residual        7925.603546  673.0        NaN       NaN
                   sum_sq     df          F    PR(>F)
Social_smoker    23.473259    1.0  1.987693  0.159044
Residual       7947.658593  673.0        NaN       NaN
              sum_sq     df          F    PR(>F)
Pet         7.368513    1.0  0.622697  0.430325
Residual  7963.763338  673.0        NaN       NaN
```

2.1.4 Feature Scaling

Feature Scaling or Standardization: It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm[1].
Real world dataset contains features that highly vary in magnitudes, units, and range. Normalisation should be performed when the scale of a feature is irrelevant or misleading and not should Normalise when the scale is meaningful.

The algorithms which use Euclidean Distance measure are sensitive to Magnitudes. Here feature scaling helps to weigh all the features equally.

2.2 Model Development

Now we have pre processed data. With this pre processed data we develop model to predict result. For this firstly we divide the data into train and test. Perform the algorithm on train data to develop model and get prediction and then apply that model on test data. Chose the algorithm which provide more accurate result.

2.2.1 Decisio n Tree

It is supervised machine learning algorithm. A predictive model based on a branching series of Boolean tests. It can be used for classification and regression. There are number of different types of decision trees that can be used in Machine learning algorithms.

Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. Extremely easy to understand by the business users. Build some intuitions about your customer base. E.g. “are customers with different family sizes truly different?”

2.2.2 Random Forest

To build n number of trees to have more accuracy on dataset.

The forest why because we build n number of decision trees. Random because to build any decision tree we are going to select randomly n number of observations.

For each decision tree, we use different-different observation from same dataset. RF called as an ensemble that consists of many decision trees. It can be used for classification and regression.

It will give you the estimate of what variable are important in classification which help us to classify or predict any value for the new test case.

2.2.3 Linear Regression

Linear regression only use for regression data not for classification data.

Prediction Model

– Simple linear regression

– Multiple linear regression

• Describe relationship among variables

• The one simple case is where a dependent variable may be related to independent or explanatory variable

3. Conclusion

From model develop we apply different machine algorithms and predict the result. Now we conclude which model is more accurate for Employee Absenteeism.

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. For this we calculate RMSE and Rsquared for each model.

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.
The formula is:

$$RMSE = \sqrt{\overline{(f - o)^2}}$$

Where:

- f = forecasts (expected values or unknown results),
- o = observed values (known results).

R-squared is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.
R-square is a comparison of residual sum of squares *(SS_{res})* with total sum of squares*(SS_{tot})*. Total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line.
Here the result of each model in python and R.

In python

	Model Name	RMSE train	RMSE Test	rsquare train	rsquare test
0	Decision Tree	0.538543	0.525706	0.319479	0.397021
1	Linear Regression	0.419173	0.420268	0.587726	0.614638
2	Random Forest	0.180568	0.396333	0.923497	0.657283

In R

	Model	MAPE.Train	MAPE.Test	RSquare.Train	RSquare.Test
1	Decision Tree	0.4296919	0.4626810	0.5732321	0.4682043
2	Ramdon Forest	0.2439306	0.2704545	0.8830654	0.8290696
3	Linear Regression	0.4168058	0.4336782	0.5984452	0.5307052

3.2 Model Selection

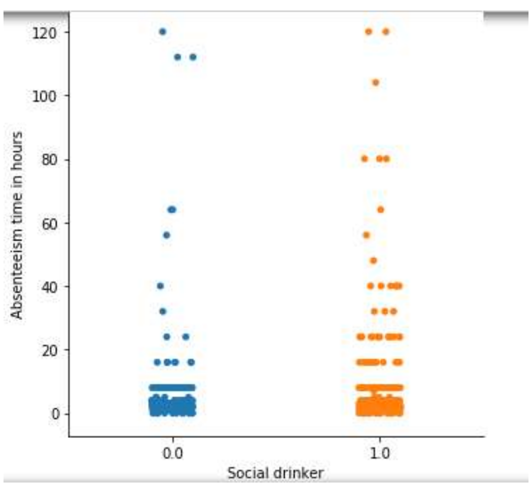
From above model evaluation we should go with **Random Forest** because from both python and R we get the lowest RMSE for both train and test data and RSquared values which is nearest to 1.

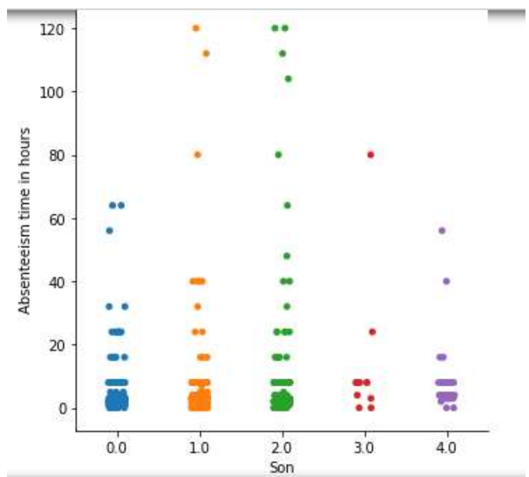
3.3Answers of Asked Questions

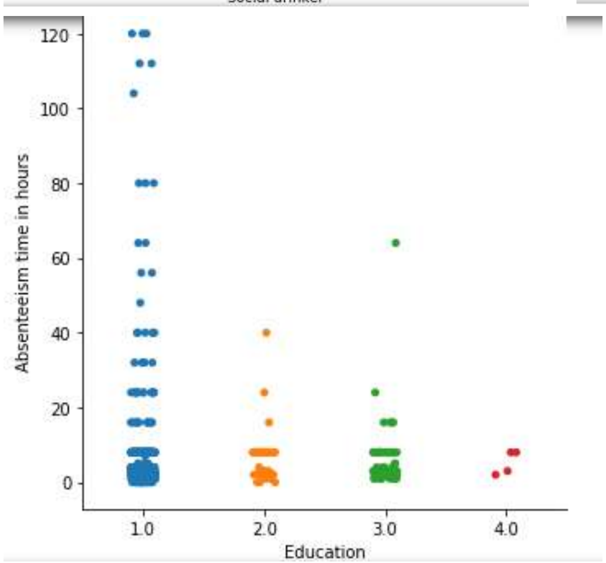
Q.1 What changes company should bring to reduce the number of absenteeism?

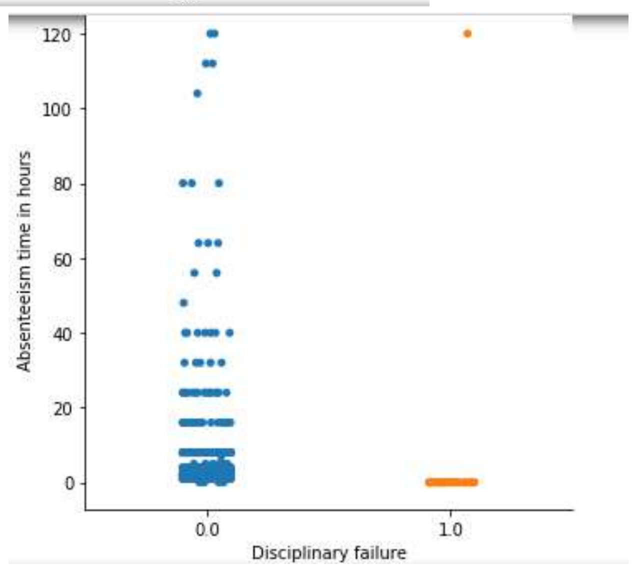
Ans. From the below visualizations we can observed that people which don't have pet and have 1-2 children have maximum Absenteeism. And also people who are socila drinker have more Absenteeism than non drinker and people who are non social smoker have more Absenteeism. People with high school eduction have maximum Absenteeism with reason of Code of Diseases 1. Absenteeism for all months and season are constant. And yes Disciplinary failure with 0 have maximum Absenteeism.

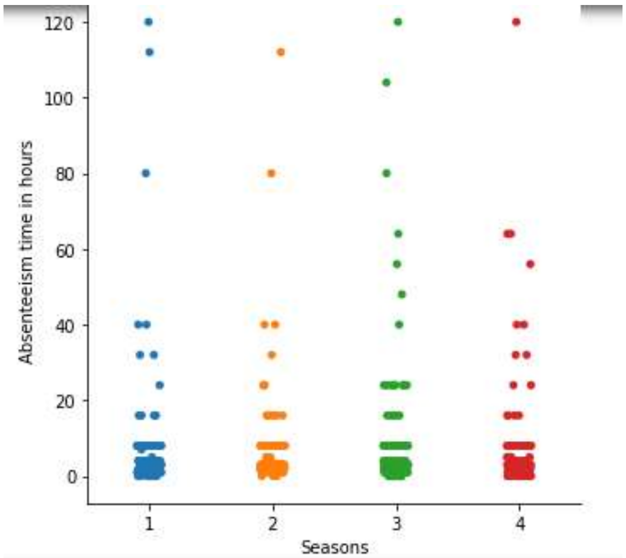
So company should more focus on the above points to reduce the number of absenteeism.

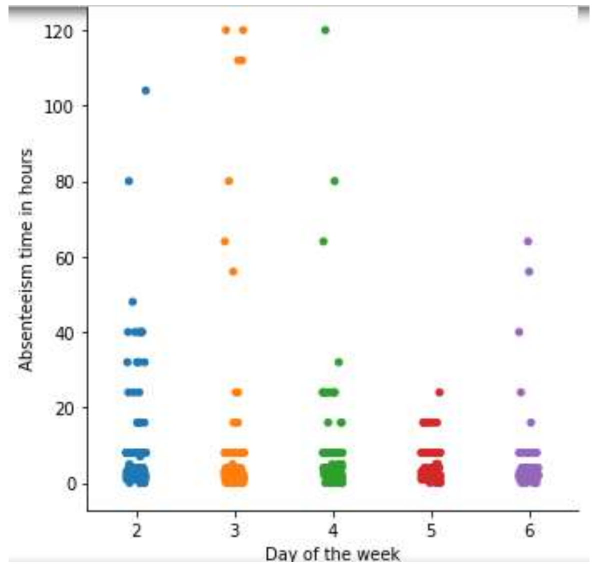


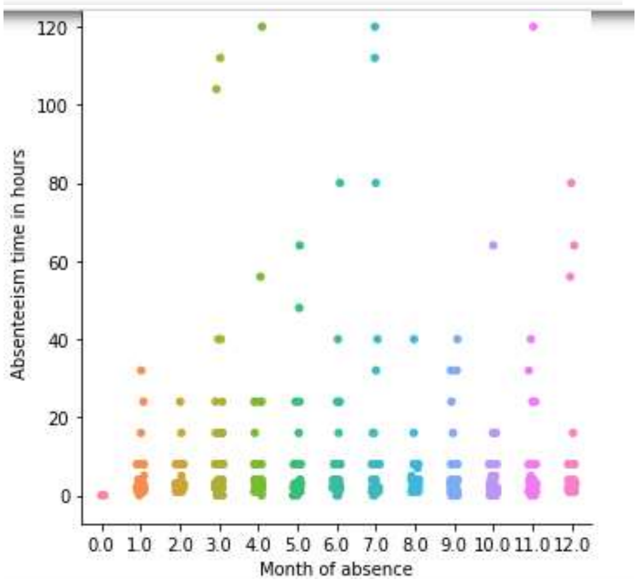


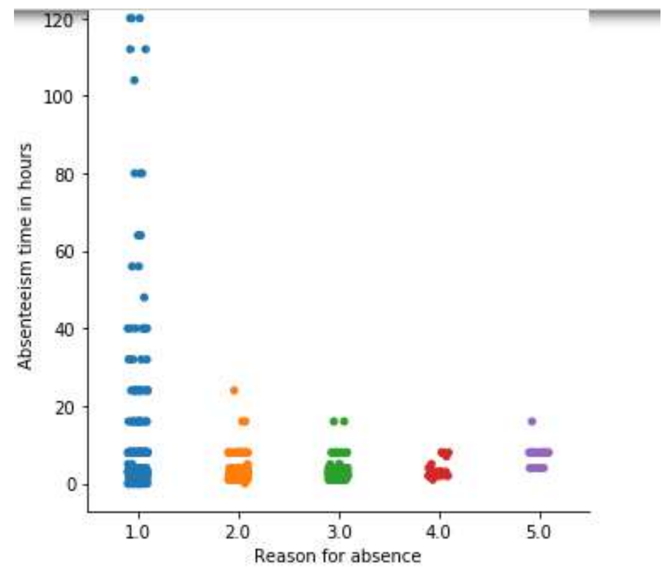


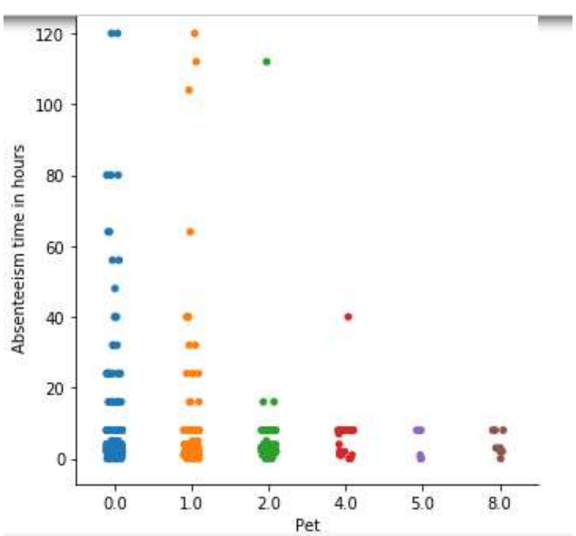



Absenteeism time in hours
0
20
40
60
80
100
120
1
2
3
4
Seasons


Absenteeism time in hours
0
20
40
60
80
100
120
2
3
4
5
6
Day of the week


Absenteeism time in hours
0
20
40
60
80
100
120
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
Month of absence


Absenteeism time in hours
0
20
40
60
80
100
120
1.0
2.0
3.0
4.0
5.0
Reason for absence


Absenteeism time in hours
0
20
40
60
80
100
120
0.0
1.0
2.0
4.0
5.0
8.0
Pet

Q.2 How much losses every month can we project in 2011 if same trend of absenteeism continues?

Ans. As we don't have data for 2011 so we are consider the given data to predict the loss. If the same trend follow in 2011 then the loss will also same. We have a formula to find loss

Work loss = (workload per day* Absenteeism_in_hours)/ 24

We use the above formula and get the below table that shows loss of each months. We can conclude from the below data that in 3rd month i.e. March have highest absenteeism time and highest loss.

Month_of_absence	Absenteeism time/month(hrs.)	work_loss_average/day_
1.0	222.0	2887.355750
2.0	294.0	3321.114667
3.0	749.0	8454.375917
4.0	482.0	5544.403583
5.0	392.0	4396.434417
6.0	403.0	5293.046917
7.0	724.0	7627.470250
8.0	272.0	2631.874000
9.0	284.0	3250.310292
10.0	340.0	3719.012000
11.0	463.0	5451.547417
12.0	382.0	3500.401792

4.Code

4.1 R code

```
#remove previous data if any

rm(list = ls())


#set working directory

setwd("I:/DATA Scientist Assignments/Employee Absenteeism project")
```

```
#check current workinh directory
getwd()

#load some directories which will use in analysis
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

lapply(x, require, character.only = TRUE)
rm(x)

#load data
library("xlsx")
data_employee = read.xlsx("Absenteeism_at_work_Project.xls", sheetIndex = 1)

View(data_employee)
head(data_employee,10)
str(data_employee)
unique(data_employee$Month.of.absence)
length(colnames(data_employee))
names(data_employee)
#drop ID variables as it is not containing any significant information
data_employee = subset(data_employee, select = -(ID))

#Count the unique value of each variable
unique_val = data.frame(sapply(data_employee, function(x) length(unique(x))))


#replace 0 with NA in "Month.of.absence" variable because there is no month 0
```

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

#divide Work.load.Average.day by 1000(Got to know from support)
data_employee$Work.load.Average.day. = data_employee$Work.load.Average.day./1000

#convert categorcal variable type
data_employee$Reason.for.absence = as.factor(data_employee$Reason.for.absence)
data_employee$Month.of.absence = as.factor(data_employee$Month.of.absence)
data_employee$Day.of.the.week = as.factor(data_employee$Day.of.the.week)
data_employee$Seasons = as.factor(data_employee$Seasons)
data_employee$Disciplinary.failure = as.factor(data_employee$Disciplinary.failure)
data_employee$Education = as.factor(data_employee$Education)
data_employee$Son = as.factor(data_employee$Son)
data_employee$Social.drinker = as.factor(data_employee$Social.drinker)
data_employee$Social.smoker = as.factor(data_employee$Social.smoker)
data_employee$Pet = as.factor(data_employee$Pet)

#divide continous and categorical variables
cnames = c("Transportation expense", "Distance from Residence to Work", "Service time", "Age",
        "Work load Average/day", "Hit target", "Weight", "Body mass index", "Absenteeism time in hours")
cat_names = c("Reason for absence", "Month of absence", "Day of the week", "Seasons", "Disciplinary failure", "Education",
          "Son", "Social drinker", "Social smoker", "Pet")

unique(data_employee$Reason.for.absence)
ggplot(data_employee, aes(x = Transportation.expense)) +
  geom_histogram(aes(y =..density..),
            breaks = seq(0, 3, by = 0.5),
```

```
            fill="blue",
            alpha = .4,position="dodge") +
  geom_density(col=4) +
  labs(title="Distribution of Transportation expense") +
  labs(x="Transportation expense", y="Density of Transportation expense") +
  theme(legend.position="top")



#----------------------------------------Data Pre Processing-----------------------------------#
#Missing value analysis
#check weather target variables have missing value or not
sum(is.na(data_employee$Absenteeism.time.in.hours))
#remove those observations which "Absenteeism time in hours" has missing values
data_employee = data_employee[(!data_employee$Absenteeism.time.in.hours %in% NA),]

missingValue = data.frame(apply(data_employee,2, function(x){sum(is.na(x))}))

missingValue$Columns = row.names(missingValue)
names(missingValue)[1] =  "Missing_percentage"
missingValue$Missing_percentage = (missingValue$Missing_percentage/nrow(data_employee)) * 100
missingValue = missingValue[order(-missingValue$Missing_percentage),]
row.names(missingValue) = NULL
missingValue = missingValue[,c(2,1)]
write.csv(missingValue, "Miising_perc.csv", row.names = F)

#visualise missing values
 ggplot(data = missingValue[1:20,], aes(x=reorder(Columns, -Missing_percentage),y = 
Missing_percentage))+
   geom_bar(stat = "identity",fill = "blue")+xlab("Parameter")+
```

```
  ggtitle("Missing data percentage") + theme_bw()

#applying mode method for categorical variables
 mode = function(x){
  uni = unique(x)
  uni[which.max(tabulate(match(x, uni)))]
}

for(i in cat_names){
  print(i)
  data_employee[,i][is.na(data_employee[,i])] = mode(data_employee[i])
}

data_employee$Transportation.expense[40]
#Actual value = 179
#Mean  = 220.4613
#median = 225
#KNN = 179

#mean Imputation
data_employee$Transportation.expense[40] = NA
data_employee$Transportation.expense[is.na(data_employee$Transportation.expense)] =
mean(data_employee$Transportation.expense, na.rm = T)
data_employee$Transportation.expense[40]

#Median imputation
data_employee$Transportation.expense[40] = NA
data_employee$Transportation.expense[is.na(data_employee$Transportation.expense)] =
median(data_employee$Transportation.expense, na.rm = T)
```

```
data_employee$Transportation.expense[40]

#knn imputation
data_employee$Transportation.expense[40] = NA
data_employee = knnImputation(data_employee, k = 3)
data_employee$Transportation.expense[40]

#so we observed that with knn imputation we get acurate value so we will go with knn imputation
sum(is.na(data_employee))

#-------------Outlier analysis-------------------#
#data manupulation: convert string categories into factor numeric
for(i in 1:ncol(data_employee)){
  if(class(data_employee[,i]) == 'factor'){
    data_employee[,i] = factor(data_employee[,i], labels = (1:length(levels(factor(data_employee[,i])))))
  }
}
rm(i)

#Boxplot to find outlier
library(ggplot2)
number_index = sapply(data_employee, is.numeric)
numeric_data = data_employee[, number_index]
cnames = colnames(numeric_data)
for(i in 1:length(cnames)){
  assign(paste0("DB", i), ggplot(aes_string(y = (cnames[i]), x = "Absenteeism.time.in.hours"), data = subset(data_employee))+
          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="Absenteeism.time.in.hours")+
            ggtitle(paste("Box plot for",cnames[i])))
}
rm(i)

## Plotting plots together
gridExtra::grid.arrange(DB1, DB2,DB3, ncol=3)
gridExtra::grid.arrange(DB4,DB5,DB6, ncol=3)
gridExtra::grid.arrange(DB7,DB8,DB9,DB10, ncol=4)

#Remove outlier using boxplot
temp = data_employee
data_employee = temp

for (i in cnames){
  val = data_employee[,i][data_employee[,i] %in% boxplot.stats(data_employee[,i])$out]
  print(val)
  data_employee[,i][data_employee[,i] %in% val] = NA
}
sum(is.na(data_employee$Height))

#replace NA with knn imputaion
data_employee = knnImputation(data_employee, k = 3)

#----------------feature selection-----------------#
#Correltion analysis for continous variales
```

```
corrgram(data_employee[,number_index], order = F ,upper.panel = panel.pie,
    text.panel = panel.txt, main ="correlation plot for numeric variables")

#we can observed that body mass and weight are highly correlated with each other


#ANOVA test for categorical variables
factorVal = sapply(data_employee, is.factor)
factorVariables = data_employee[, factorVal]
cat_variables = names(factorVariables)
for(i in cat_variables){
  print(i)
  anovaresult = summary(aov(formula = Absenteeism.time.in.hours~data_employee[,i],data_employee))
  print(anovaresult)
}

#Now we will remove the values which are highly correlated to eanch other and have >0.05 p value
data_employee = subset(data_employee, select = -
c(Weight,Social.smoker,Education,Seasons,Day.of.the.week))
dim(data_employee)


#-------------feature scaling----------------------#
cat_del_ind = sapply(data_employee, is.numeric)
cat_del = data_employee[, cat_del_ind]
cnames_del = names(cat_del)
#skewness test
library(propagate)
for(i in cnames_del){
  print(i)
  skew = skewness(data_employee[,i])
```

```
  print(skew)
}
hist(data_employee$Transportation.expense, col = "blue", xlab = "Transportation.expense", ylab =
"Frequency",
     main = "histogram of Transportation.expense")
hist(data_employee$Distance.from.Residence.to.Work, col = "blue", xlab =
"Distance.from.Residence.to.Work", ylab = "Frequency",
     main = "histogram of Distance.from.Residence.to.Work")
hist(data_employee$Service.time, col = "blue", xlab = "Service.time", ylab = "Frequency",
     main = "histogram of Service.time")
hist(data_employee$Absenteeism.time.in.hours, col = "blue", xlab = "Absenteeism.time.in.hours", ylab =
"Frequency",
     main = "histogram of Absenteeism.time.in.hours")


#logtransform
data_employee$Absenteeism.time.in.hours = log1p(data_employee$Absenteeism.time.in.hours)


#from above histograms we can say that data is not normally distributed so for that normalisation is best
way
#Normalization
for(i in cnames_del){
  if(i != "Absenteeism.time.in.hours"){
    print(i)
    data_employee[,i] = (data_employee[,i] - min(data_employee[,i]))/(max(data_employee[,i]) -
min(data_employee[,i]))
    print(data_employee[,i])
  }
}
rm(i)
```

```
#summary
for(i in cnames_del){
  print(summary(data_employee[,i]))
}
#as summary the data is in now normalised form
#write the pre processed data to drive
write.csv(data_employee, "data_employee.csv", row.names = FALSE)




#------------------------Model Development---------------------#
#Clean the environment
rmExcept("data_employee")

#Divide data into train and test using stratified sampling method
set.seed(6789)
train.index = sample(1:nrow(data_employee), .80 * nrow(data_employee))
train = data_employee[ train.index,]
test  = data_employee[-train.index,]

#RMSE
rmse = function(y,y1){
  sqrt(mean(abs(y-y1)^2))
}

#R square
rsquare = function(y,y1){
  cor(y,y1)^2
```

```
}

#--------------------Decision tree--------------#
#Load Libraries
library(rpart)
library(MASS)

#rpart for regression
DT_model = rpart(Absenteeism.time.in.hours ~ ., data = train, method = "anova")

#Predict for train cases
train_DT = predict(DT_model, train[-15])

#predict for test cases
test_DT = predict(DT_model, test[-15])

#rmse
RMSE_DT_train = (rmse(train[,15], train_DT))
#RMSE_DT_train = 0.4296919
RMSE_DT_test = (rmse(test[,15], test_DT))
#MAPE_DT_test = 0.462681

#Rsquare
rquare_train_DT = rsquare(train[,15], train_DT)
#rquare_train = 0.5732321
rquare_test_DT = rsquare(test[,15], test_DT)
#rquare_test =  0.4682043
```

```
#------------------Random Forest-----------------#

library(randomForest)

#delevelop model using random forest

RF_model = randomForest(Absenteeism.time.in.hours~., data_employee, nTree = 500, importance =
TRUE)


#apply on train data

RF_train_predict = predict(RF_model, train[,-15])


#apply on test data

RF_test_predict = predict(RF_model, test[,-15])


#RMSE for train

RF_RMSE_train = (rmse(train[,15], RF_train_predict))

#RMSE  0.2439306


#RMSE for test

RF_RMSE_test = (rmse(test[,15], RF_test_predict))

#RMSE 0.2704545


#RSquare for train

RSquare_train_RF= rsquare(train[,15], RF_train_predict)

#Rsquare 0.8830654


#RSquare for test

RSquare_test_RF = rsquare(test[,15], RF_test_predict)

#Rsquare  0.8290696
```

```
#----------------------Linear Regression----------------#

library(usdm)

colnames(data_employee)

cnames = c("Transportation.expense", "Distance.from.Residence.to.Work" ,"Service.time", "Age",
           "Work.load.Average.day.", "Hit.target", "Height","Body.mass.index",
"Absenteeism.time.in.hours")

vif(data_employee[,cnames])

vifcor(data_employee[,cnames], th = 0.7)


#develop linear regression model

LR_model = lm(Absenteeism.time.in.hours~., data = train)

summary(LR_model)


#apply on train data

LR_train = predict(LR_model, train[,-15])


#apply on test

LR_test = predict(LR_model, test[,-15])


#RMSE for train

LR_RMSE_train = (rmse(train[,15],LR_train))

#RMSE  0.4168058


#RMSE for test

LR_RMSE_test = (RMSE(test[,15], LR_test))

#RMSE 0.4336782


#Rsquare for train

rsquare_train_LR = rsquare(train[,15],LR_train)
```

#0.5984452

#Rsquare for test

rsquare_test_LR = rsquare(test[,15], LR_test)

#0.5307052

#------------------Result------------------#

result = data.frame(Model = c('Decision Tree', 'Ramdon Forest', 'Linear Regression'),

'MAPE Train' = c(RMSE_DT_train,RF_RMSE_train,LR_RMSE_train),

'MAPE Test' = c(RMSE_DT_test,RF_RMSE_test,LR_RMSE_test),

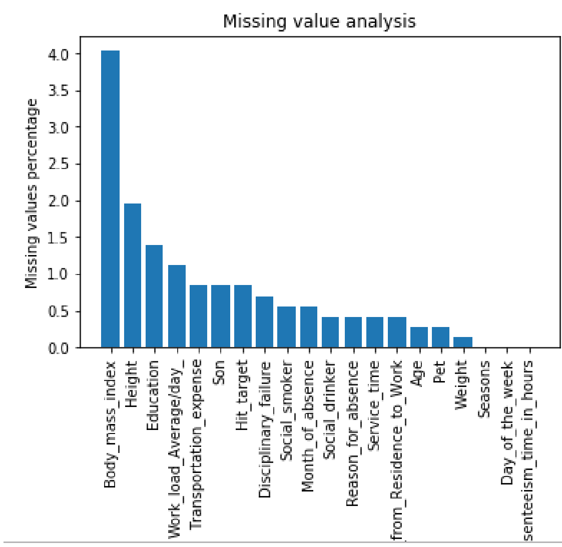
'RSquare Train' = c(rquare_train_DT,RSquare_train_RF,rsquare_train_LR),

'RSquare Test' = c(rquare_test_DT,RSquare_test_RF,rsquare_test_LR))

write.csv(result, "Result,csv", row.names = FALSE)

Python Code :

```
#plot missing values on bar graph
plt.bar(missingvalue["Variables"], missingvalue["Missing_percentage"] )
plt.xticks(rotation=90)
plt.ylabel("Missing values percentage")
plt.title('Missing value analysis')

plt.show()
```



```python
#drop the observation which "Absenteeism time in hours" has missing value
data_employee = data_employee.drop(data_employee[data_employee['Absenteeism_time_in_hours'].isnull()].index, axis = 0)
print(data_employee.shape)
```

```
(718, 20)
```

```python
data_employee['Absenteeism_time_in_hours'].isnull().sum()
```

```
0
```

```python
missingvalue = pd.DataFrame(data_employee.isnull().sum()).reset_index()
missingvalue = missingvalue.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})
missingvalue['Missing_percentage'] = (missingvalue['Missing_percentage']/len(data_employee))*100
missingvalue = missingvalue.sort_values('Missing_percentage', ascending = False).reset_index(drop = True)

missingvalue.to_csv("Miising_perc.csv", index = False)
```

```python
missingvalue
```

	Variables	Missing_percentage
0	Body_mass_index	4.038997
1	Height	1.949861
2	Education	1.392758
3	Work_load_Average/day_	1.114206
4	Transportation_expense	0.835655
5	Son	0.835655

**Missing Value analysis

```python
#Create dataframe with missing percentage
missing_val = pd.DataFrame(data_employee.isnull().sum())
```

```python
missing_val
```

	0
Reason_for_absence	3
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

```
data_employee.columns = data_employee.columns.str.replace(' ', '_')
```

```
data_employee.columns
```

```
Index(['ID', 'Reason_for_absence', 'Month_of_absence', 'Day_of_the_week',
       'Seasons', 'Transportation_expense', 'Distance_from_Residence_to_Work',
       'Service_time', 'Age', 'Work_load_Average/day_', 'Hit_target',
       'Disciplinary_failure', 'Education', 'Son', 'Social_drinker',
       'Social_smoker', 'Pet', 'Weight', 'Height', 'Body_mass_index',
       'Absenteeism_time_in_hours'],
      dtype='object')
```

```
#Continuous variables
cnames = ["ID","Transportation_expense", "Distance_from_Residence_to_Work", "Service_time", "Age",
          "Work_load_Average/day_", "Hit_target", "Weight", "Height", "Body_mass_index", "Absenteeism_time_in_hours"]

#Categorical variables
cat_names = ['Reason_for_absence', 'Month_of_absence', 'Day_of_the_week', 'Seasons', 'Disciplinary_failure', 'Education',
             'Son', 'Social_drinker', 'Social_smoker', 'Pet']
```

```
#divide "Work load Average/day " by 1000(as get to know from support team)
data_employee["Work_load_Average/day_"] = data_employee["Work_load_Average/day_"]/1000
```

**Data Pre Processing

**Missing Value analysis

```
data_employee['Month of absence'].unique()
```

```
array([ 7.,  8.,  9., 10., nan, 11., 12.,  1.,  2.,  3.,  4.,  5.,  6.,
        0.])
```

```
#Replace 0 values of month with nan as it is not any month
data_employee['Month of absence'] = data_employee['Month of absence'].replace(0,np.NaN)
```

```
data_employee['Month of absence'].unique()
```

```
array([ 7.,  8.,  9., 10., nan, 11., 12.,  1.,  2.,  3.,  4.,  5.,  6.])
```

```
data_employee.nunique()
```

```
ID                                 36
Reason for absence                 28
Month of absence                   12
Day of the week                     5
Seasons                             4
Transportation expense             24
Distance from Residence to Work    25
Service time                       18
Age                                22
Work load Average/day              38
Hit target                         13
Disciplinary failure                2
Education                           4
Son                                 5
Social drinker                      2
Social smoker                       2
```

```python
#drop ID column as it not contain any significant information
data_employee = data_employee.drop(['ID'], axis = 1)
```

```python
data_employee.shape
```

```
(740, 20)
```

```python
data_employee.columns
```

```
Index(['ID', 'Reason for absence', 'Month of absence', 'Day of the week',
       'Seasons', 'Transportation expense', 'Distance from Residence to Work',
       'Service time', 'Age', 'Work load Average/day ', 'Hit target',
       'Disciplinary failure', 'Education', 'Son', 'Social drinker',
       'Social smoker', 'Pet', 'Weight', 'Height', 'Body mass index',
       'Absenteeism time in hours'],
      dtype='object')
```

```python
data_employee['Month of absence']
```

```
0      7.0
1      7.0
2      7.0
3      7.0
4      7.0
      ...
735    7.0
736    7.0
737    0.0
738    0.0
739    0.0
Name: Month of absence, Length: 740, dtype: float64
```

```python
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 fancyimpute import KNN
```

```
Using TensorFlow backend.
```

```python
os.chdir("I:\DATA Scientist Assignments\Employee Absenteeism project")
```

```python
data_employee = pd.read_excel("Absenteeism_at_work_Project.xls")
```

```python
data_employee.shape
```

```
(740, 21)
```

```python
type(data_employee)
```

```
pandas.core.frame.DataFrame
```

```python
data_employee.nunique()
```

```
ID                                 36
Reason for absence                 28
Month of absence                   13
Day of the week                     5
Seasons                             4
```

```python
Monthly_loss= Absenteeism_hours_monthly.rename(columns={'Absenteeism_time_in_hours': 'Absenteeism time/month(hr
                                'work_loss_average/day': 'Work loss per month'})
```

```python
Monthly_loss
```

	Absenteeism time/month(hrs.)	work_loss_average/day_
Month_of_absence		
1.0	222.0	2887.355750
2.0	294.0	3321.114667
3.0	749.0	8454.375917
4.0	482.0	5544.403583
5.0	392.0	4396.434417
6.0	403.0	5293.046917
7.0	724.0	7627.470250
8.0	272.0	2631.874000
9.0	284.0	3250.310292
10.0	340.0	3719.012000
11.0	463.0	5451.547417
12.0	382.0	3500.401792

```
>>>>>>>>Thank You<<<<<<<<
```

>>>2. How much losses every month can we project in 2011 if same trend of absenteeism continues?<<<

```
predict_loss = data_employee
```

```
predict_loss.head()
```

	ID	Reason_for_absence	Month_of_absence	Day_of_the_week	Seasons	Transportation_expense	Distance_from_Re
0	11	5.0	7.0	3	1	289.0	
1	36	1.0	7.0	3	1	118.0	
2	3	3.0	7.0	4	1	179.0	
3	7	1.0	7.0	5	1	279.0	
4	11	3.0	7.0	5	1	289.0	

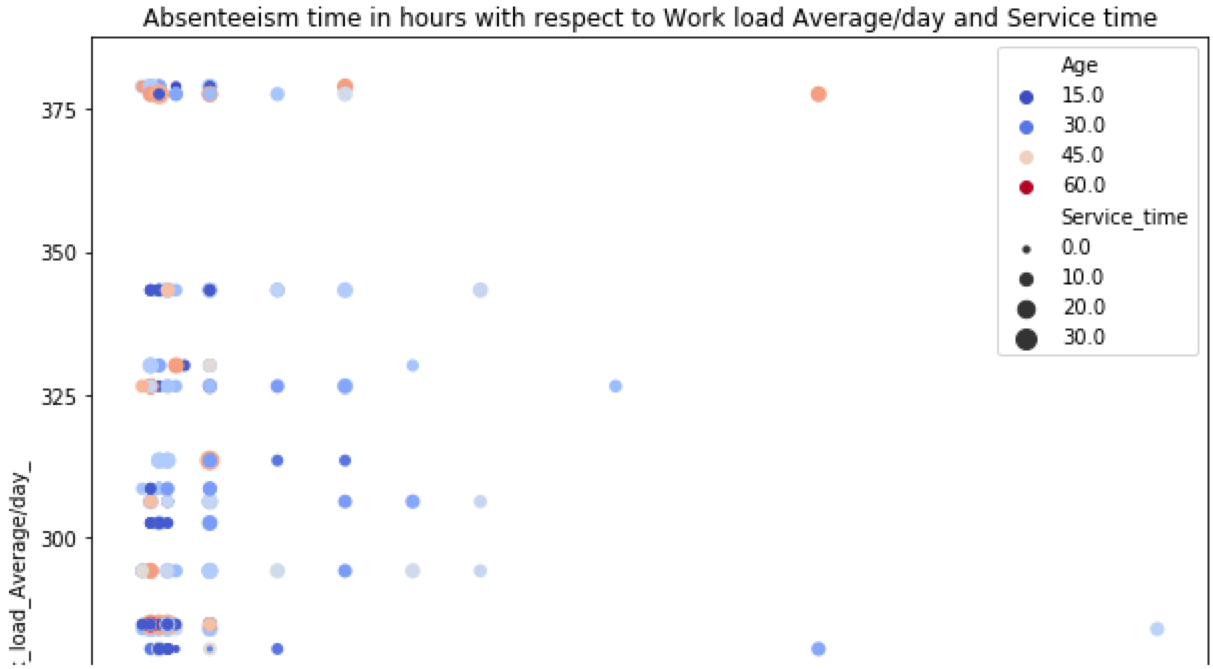
5 rows × 21 columns

```
#work loss per month-
predict_loss['work_loss_average/day_'] = 0
for i  in range(len(predict_loss)):
    predict_loss['work_loss_average/day_'].loc[i] = ((predict_loss['Work_load_Average/day_'].loc
```

```
C:\Users\dcdc\Anaconda3\lib\site-packages\pandas\core\indexing.py:205: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/ind
```

```python
#Absenteeism Visualization
f, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(x="Absenteeism_time_in_hours", y = "Work_load_Average/day_",
                hue = "Age", size = "Service_time",
                palette = "coolwarm", sizes = (10,100), linewidth=0,
                data = data_employee, ax=ax)
plt.title("Absenteeism time in hours with respect to Work load Average/day and Service time")
plt.xlabel("Absenteeism_time_in_hours")
plt.ylabel("Work_load_Average/day_")
```

```
Text(0, 0.5, 'Work_load_Average/day_')
```



**1. What changes company should bring to reduce the number of absenteeism?<<<

```
temp = data_employee
data_employee = temp
```

```
data_employee.columns
```

```
Index(['ID', 'Reason_for_absence', 'Month_of_absence', 'Day_of_the_week',
       'Seasons', 'Transportation_expense', 'Distance_from_Residence_to_Work',
       'Service_time', 'Age', 'Work_load_Average/day_', 'Hit_target',
       'Disciplinary_failure', 'Education', 'Son', 'Social_drinker',
       'Social_smoker', 'Pet', 'Weight', 'Height', 'Body_mass_index',
       'Absenteeism_time_in_hours'],
      dtype='object')
```

```
data_employee['Reason_for_absence'] = data_employee['Reason_for_absence'].replace({0:1,1:1,2:1,3
```

```
data_employee.head()
```

ID	Reason_for_absence	Month_of_absence	Day_of_the_week	Seasons	Transportation_expense	Distance_from_Re

Random Forest

```
#import library for Random Forest
from sklearn.ensemble import RandomForestRegressor
```

```
RF_model = RandomForestRegressor(n_estimators = 100).fit(xtrain, ytrain)
```

```
# make the predictions by the model on train data
RF_train = RF_model.predict(xtrain)

# make the predictions by the model on test data
RF_test = RF_model.predict(xtest)
```

```
#RMSE
train_RMSE_FR = RMSE(ytrain,RF_train)

#RMSE
test_RMSE_FR = RMSE(ytest,RF_test)

#rsquare
rsquare_train_FR = r2_score(ytrain,RF_train)

#rsquare
rsquare_test_FR = r2_score(ytest,RF_test)
```

```
print("RMSE for train : " + str(train_RMSE_FR))
```

Linear Regression

```
LR_model = sm.OLS(ytrain,xtrain).fit()
LR_model.summary()
```

OLS Regression Results

Dep. Variable:	Absenteeism_time_in_hours	**R-squared:**	0.588
Model:	OLS	**Adj. R-squared:**	0.553
Method:	Least Squares	**F-statistic:**	16.73
Date:	Thu, 12 Sep 2019	**Prob (F-statistic):**	3.52e-75
Time:	22:49:19	**Log-Likelihood:**	-315.39
No. Observations:	574	**AIC:**	722.8
Df Residuals:	528	**BIC:**	923.0
Df Model:	45		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Transportation_expense	0.0013	0.000	2.822	0.005	0.000	0.002

```
# make the predictions by the model on train data
LR_train = LR_model.predict(xtrain)
```

Decision Tree

```
#import library
from sklearn.tree import DecisionTreeRegressor
```

```
#Decision tree for regression
DT_model = DecisionTreeRegressor(max_depth=2).fit(xtrain, ytrain)
```

```
#prediction for train data
DT_train = DT_model.predict(xtrain)

#prediction for test data
DT_test = DT_model.predict(xtest)
```

```
#MAPE
train_RMSE_DT = RMSE(ytrain,DT_train)

#MAPE
test_RMSE_DT = RMSE(ytest,DT_test)

#rsquare
rsquare_train_DT = r2_score(ytrain,DT_train)

#rsquare
rsquare_test_DT = r2_score(ytest,DT_test)
```

```
print("MAPE for train : " + str(train_RMSE_DT))
```

Model Development

> Divide data into train and test

```
#import libraries
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

```
temp = data_employee
data_employee = temp
```

```
#create categorical variables to dummy variables
data_employee = pd.get_dummies(data_employee,columns = cat_names)
```

```
data_employee.shape
```

```
(718, 51)
```

```
#divide data for predictor and target
x = data_employee.drop(['Absenteeism_time_in_hours'], axis = 1)
y = data_employee['Absenteeism_time_in_hours']
```

```
#Calculate RMSE
from sklearn.metrics import mean_squared_error
```

```
#Normalization
for i in cnames :
    if(i == "Absenteeism_time_in_hours") :
        continue
        print(i)
        data_employee[i] = (data_employee[i] - min(data_employee[i]))/(max(data_employee[i]) - m
        print(data_employee[i])
```

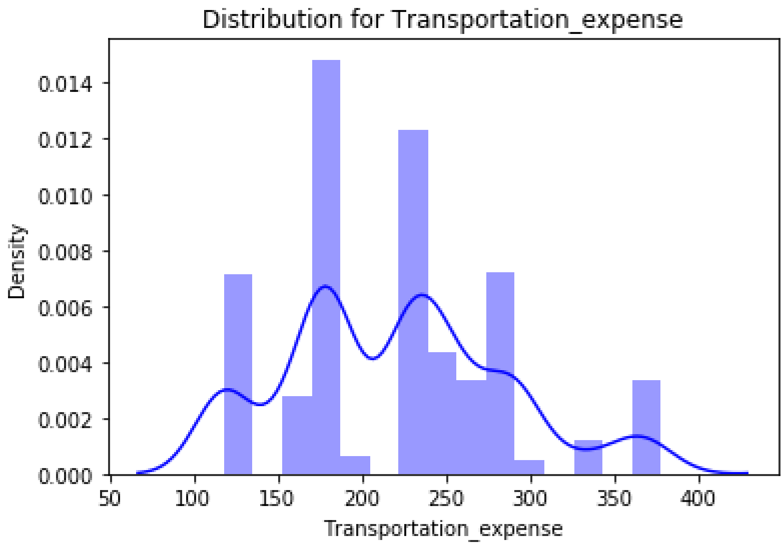
```
data_employee.describe()
```

	Reason_for_absence	Day_of_the_week	Transportation_expense	Distance_from_Residence_to_Work	Service_time
count	718.000000	718.000000	718.000000	718.000000	718.000000
mean	19.409471	3.899721	219.968061	29.548747	12.473538
std	8.279768	1.419519	65.325153	14.774540	4.149406
min	0.000000	2.000000	118.000000	5.000000	1.000000
25%	13.000000	3.000000	179.000000	16.000000	9.000000
50%	23.000000	4.000000	225.000000	26.000000	13.000000
75%	26.000000	5.000000	260.000000	50.000000	16.000000
max	28.000000	6.000000	378.000000	52.000000	24.000000

```
#write to hard disk
data_employee.to_csv("Employee Absenteeism pre processed.csv", index = False)
```

```
#Checking weather the data is normaly distributed or not

for i in cnames :
    print(i)
    sns.distplot(data_employee[i], bins ='auto', color = 'blue')
    plt.title("Distribution for "+i)
    plt.ylabel("Density")
    plt.show()
```

```
Transportation_expense
```




```
Distance from Residence to Work
```

```
#Normalization
for i in cnames :
    if(i == "Absenteeism_time_in_hours") :
```

```python
from scipy import stats
```

```python
#checking skewness of continous variables
for i in cnames :
    skewness = stats.describe(data_employee.loc[:,i])
    print(str(i))
    print(skewness)
    print("-----------------------------------------------------------------------------
```

```
Transportation_expense
DescribeResult(nobs=718, minmax=(118.0, 378.0), mean=219.9680611246248, variance=4267.375558496
75, kurtosis=-0.3336017718231994)
-----------------------------------------------------------------------------------------
Distance_from_Residence_to_Work
DescribeResult(nobs=718, minmax=(5.0, 52.0), mean=29.54874651596777, variance=218.2870207524922
kurtosis=-1.2387084853913544)
-----------------------------------------------------------------------------------------
Service_time
DescribeResult(nobs=718, minmax=(1.0, 24.0), mean=12.473537605854853, variance=17.217569340089
95, kurtosis=-0.17000453330171572)
-----------------------------------------------------------------------------------------
Age
DescribeResult(nobs=718, minmax=(27.0, 53.0), mean=36.1593055120994, variance=37.20720263700773
kurtosis=-0.2554544607658835)
-----------------------------------------------------------------------------------------
Work_load_Average/day_
DescribeResult(nobs=718, minmax=(205.917, 343.253), mean=267.27147338407667, variance=1044.4996
91637545, kurtosis=-0.22192536316821787)
```

```python
#since skewness of target variable is high so we will apply log to transform it
data_employee['Absenteeism_time_in_hours'] = np.log1p(data_employee['Absenteeism_time_in_hours'
```

```python
#drop variables
data_employee = data_employee.drop(['Weight', 'Pet', 'Social_smoker', 'Education', 'Seasons', 'M
```

```python
data_employee.shape
```

```
(718, 14)
```

> > **Feature Scaling

```python
#Updating cantinuse and categorical variables
#Continuous variables
cnames = ["Transportation_expense", "Distance_from_Residence_to_Work", "Service_time", "Age",
          "Work_load_Average/day_", "Hit_target", "Height", "Body_mass_index", "Absenteeism_time_

#Categorical variables
cat_names = ['Reason_for_absence', 'Day_of_the_week', 'Disciplinary_failure',
             'Son', 'Social_drinker']
```

```python
from scipy import stats
```

```python
#checking skewness of continous variables
for i in cnames :
    skewness = stats.describe(data_employee.loc[:,i])
    print(str(i))
```

```python
cat_names = ['Reason_for_absence', 'Month_of_absence', 'Day_of_the_week', 'Seasons', 'Disciplina
             'Son', 'Social_drinker', 'Social_smoker', 'Pet']
```

```python
data_employee.columns
```

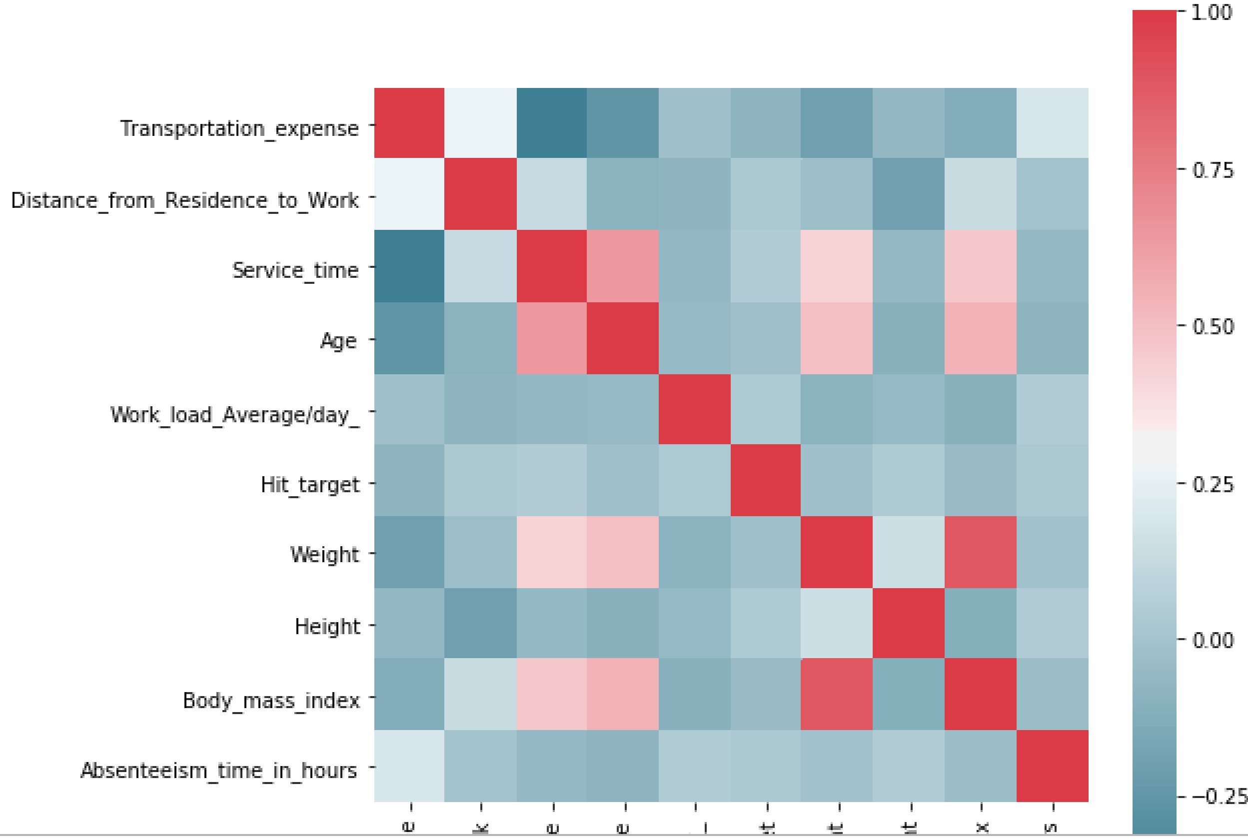
```
Index(['Reason_for_absence', 'Month_of_absence', 'Day_of_the_week', 'Seasons',
       'Transportation_expense', 'Distance_from_Residence_to_Work',
       'Service_time', 'Age', 'Work_load_Average/day_', 'Hit_target',
       'Disciplinary_failure', 'Education', 'Son', 'Social_drinker',
       'Social_smoker', 'Pet', 'Weight', 'Height', 'Body_mass_index',
       'Absenteeism_time_in_hours'],
      dtype='object')
```

```python
#ANOVA test for categorical variable and target numeric variable
import statsmodels.api as sm
from statsmodels.formula.api import ols

label = 'Absenteeism_time_in_hours'
for i in cat_names :
    frame = label + '~' + i
    model = ols(frame, data = data_employee).fit()
    anova = sm.stats.anova_lm(model, typ=2)
    print(anova)
```

```
                         sum_sq     df          F    PR(>F)
Reason_for_absence   204.595963    1.0  17.937466  0.000026
Residual            8166.745042  716.0        NaN       NaN
                       sum_sq     df         F    PR(>F)
Month_of_absence     0.280067    1.0  0.023955  0.877043
Residual          8371.060938  716.0       NaN       NaN
                     sum_sq     df         F    PR(>F)
```

```python
#Plot using seaborn library
f, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(corrAna, mask=np.zeros_like(corrAna, dtype=np.bool), cmap=sns.diverging_palette(220,
            square=True, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ddc9554748>
```



**Feature selection

```
df = data_employee.copy()
data_employee = df.copy()
```

```
#correlation analysis for continuse variables
#extract only numerical variable
data_num = data_employee.loc[:,cnames]

corrAna = data_num.corr()
```

```
corrAna
```

	Transportation_expense	Distance_from_Residence_to_Work	Service_time	Age
Transportation_expense	1.000000	0.266953	-0.385904	-0.261337
Distance_from_Residence_to_Work	0.266953	1.000000	0.121563	-0.095107
Service_time	-0.385904	0.121563	1.000000	0.646498
Age	-0.261337	-0.095107	0.646498	1.000000
Work_load_Average/day_	-0.014743	-0.081671	-0.063297	-0.054406
Hit_target	-0.083629	0.026998	0.050159	-0.018698
Weight	-0.198483	-0.031946	0.423626	0.498222

```
missing_val = pd.DataFrame(data_employee.isnull().sum())
```

```
missing_val
```

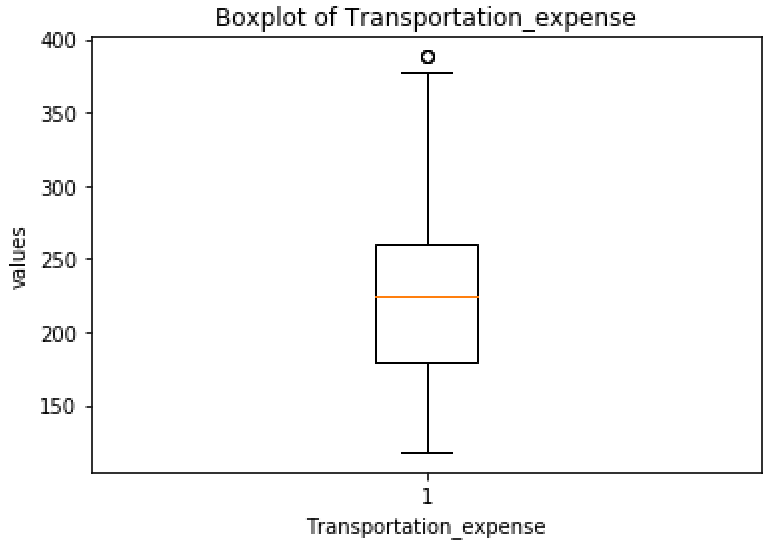
	0
Reason_for_absence	0
Month_of_absence	0
Day_of_the_week	0
Seasons	0
Transportation_expense	3
Distance_from_Residence_to_Work	0
Service_time	5
Age	8
Work_load_Average/day_	29
Hit_target	19
Disciplinary_failure	0
Education	0
Son	0
Social_drinker	0
Social_smoker	0
Pet	0
Weight	0

```python
#calulate iqr, min and max
for i in cnames:
    print(i)
    q75, q25 = np.percentile(data_employee.loc[:,i], [75 ,25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print("Minimum : "+ str(min))
    print("Maximum : "+ str(max))
    print("IQR : "+ str(iqr))

    #Replace with nan
    data_employee.loc[data_employee[i]< min,i] = np.nan
    data_employee.loc[data_employee[i]> max,i] = np.nan
```

```
Transportation_expense
Minimum : 57.5
Maximum : 381.5
IQR : 81.0
Distance_from_Residence_to_Work
Minimum : -35.0
Maximum : 101.0
IQR : 34.0
Service_time
Minimum : -1.5
Maximum : 26.5
IQR : 7.0
Age
Minimum : 17.5
Maximum : 53.5
IQR : 9.0
Work load Average/day
```

```
df = data_employee.copy()
data_employee = df.copy()

#Outlier analysis
for i in cnames :
    print(i)
    plt.boxplot(data_employee[i])
    plt.xlabel(i)
    plt.ylabel("values")
    plt.title("Boxplot of "+i)
    plt.show()
```

```
Transportation_expense
```



```
#KNN imputation method
data_employee = pd.DataFrame(KNN(k = 3).fit_transform(data_employee), columns = data_employee.co
```

```
Imputing row 1/718 with 0 missing, elapsed time: 0.633
Imputing row 101/718 with 0 missing, elapsed time: 0.659
Imputing row 201/718 with 0 missing, elapsed time: 0.661
Imputing row 301/718 with 0 missing, elapsed time: 0.662
Imputing row 401/718 with 0 missing, elapsed time: 0.662
Imputing row 501/718 with 1 missing, elapsed time: 0.663
Imputing row 601/718 with 0 missing, elapsed time: 0.664
Imputing row 701/718 with 0 missing, elapsed time: 0.664
```

```
data_employee["Body_mass_index"][30]
```

```
33.0
```

> > **outlier analysis

```
df = data_employee.copy()
data_employee = df.copy()

#Outlier analysis
for i in cnames :
    print(i)
```

```
#Missing value imputation for numeric values
data_employee["Body_mass_index"][30]
#actual value = 31
#Mean = 26.700581395348838
#median = 25.0
#KNN = 33.0
```

```
31.0
```

```
data_employee["Body_mass_index"][30] = np.NaN
```

```
C:\Users\dcdc\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/ind
sus-a-copy
  """Entry point for launching an IPython kernel.
```

```
#Mean method
data_employee["Body_mass_index"] = data_employee["Body_mass_index"].fillna(data_employee["Body_m
```

```
data_employee["Body_mass_index"][30]
```

```
26.70083378372634
```

```
data_employee["Body_mass_index"][30]= np.NaN
```

```
C:\Users\dcdc\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

5. References

For data clean and modeling

https://edwisor.com/career-data-science

https://www.techopedia.com/definition/14650/data-preprocessing

https://www.statisticshowto.datasciencecentral.com/rmse/