Employee Absenteeism

SHANKAR SAMIDALA

24 June 2018

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

The objective of this project is to study employee details and behaviour. We are provided with a dataset that has employee work, absent reason and if the employee time of absence. Given below is the data we should analyse to answer company's question.

Table 1.1: Employee Absenteeism at work data (Columns: 1-6)

ID	Reason for Absence	Month of Absence	Day of the week	Seasons	Transportation expense
11	26	7	3	1	289
36	0	7	3	1	118
3	23	7	4	1	179
7	7	7	5	1	279
11	23	7	5	1	289
3	23	7	6	1	179

Table 1.2: Employee Absenteeism at work data (Columns: 7-12)

Distance from Residence to work	Service time	Age	Work load Average/day	Hit target	Disciplinary failure
36	13	33	2,39,554	97	0
13	18	50	2,39,554	97	1
51	18	38	2,39,554	97	0
5	14	39	2,39,554	97	0
36	13	33	2,39,554	97	0
51	18	38	2,39,554	97	0

Table 1.3: Employee Absenteeism at work data (Columns: 13-18)

Education	Son	Social drinker	Social smoker	Pet	Weight
1	2	1	0	1	90
1	1	1	0	0	98
1	0	1	0	0	89
1	2	1	1	0	68
1	2	1	0	1	90
1	0	1	0	0	89

Table 1.3: Employee Absenteeism at work data (Columns: 19-21)

Height	Body mass index	Absenteeism time in hours
172	30	4
178	31	0
170	31	2
168	24	4
172	30	2
170	31	

The data provided is done with exploratory analysis. The data of variables and are as follows.

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

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).

- Month of absence
- Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))
- Seasons (summer (1), autumn (2), winter (3), spring (4))
- Transportation expense
- Distance from Residence to Work (kilometers)
- Service time
- Age
- Work load Average/day
- Hit target
- Disciplinary failure (yes=1; no=0)
- Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))
- Son (number of children)
- Social drinker (yes=1; no=0)
- Social smoker (yes=1; no=0)
- Pet (number of pet)
- Weight
- Height
- Body mass index
- Absenteeism time in hours (target)

Our target variable is Absenteeism time in hours of which we have to answer the company loss and measures to be taken.

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. For our data we apply pre processing techniques that we necessary.
Our pre processing involves following steps:

1. Missing value analysis
2. Outlier Analysis
3. Feature selection
   3.1 Correlation Analysis
   3.2 ANOVA
4. Normalization

2.1.1 Missing value analysis

Missing values occur when no data value is stored for the variable in an observation. Missing values are a common occurrence, and you need to have a strategy for treating them. A missing value can signify a number of different things in your data. Perhaps the data was not available or not applicable or the event did not happen. It could be that the person who entered the data did not know the right value, or missed filling in. Typically, ignore the missing values, or exclude any records containing missing values, or replace missing values with the mean, or infer missing values from existing values. We check for missing values in our data. There are missing values in the data that was given. Of the 21 variables provides 18 variables had the missing values. We imputed the missing values using median method. The code for the missing value imputation is written in appendix

2.1.2 Outlier Analysis

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Outliers can drastically change the results of the data analysis and statistical modeling. There are numerous unfavorable impacts of outliers in the data set. It increases the error variance and reduces the power of statistical tests. If the outliers are non-randomly distributed, they can decrease normality. They can also impact the basic assumption of Regression, ANOVA and other statistical model assumptions. The boxplot for our data could be seen as follows

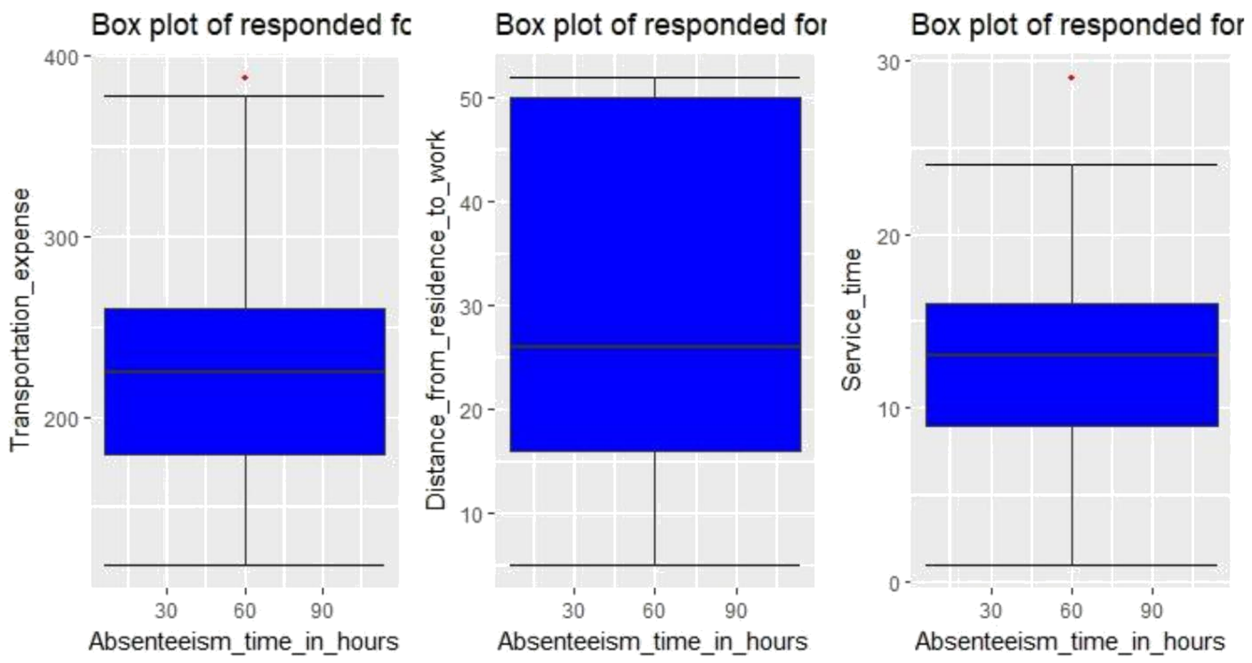

Box plot of responded fc
Transportation_expense
Absenteeism_time_in_hours
Box plot of responded for
Distance_from_residence_to_work
Absenteeism_time_in_hours
Box plot of responded for
Service_time
Absenteeism_time_in_hours

Fig 1

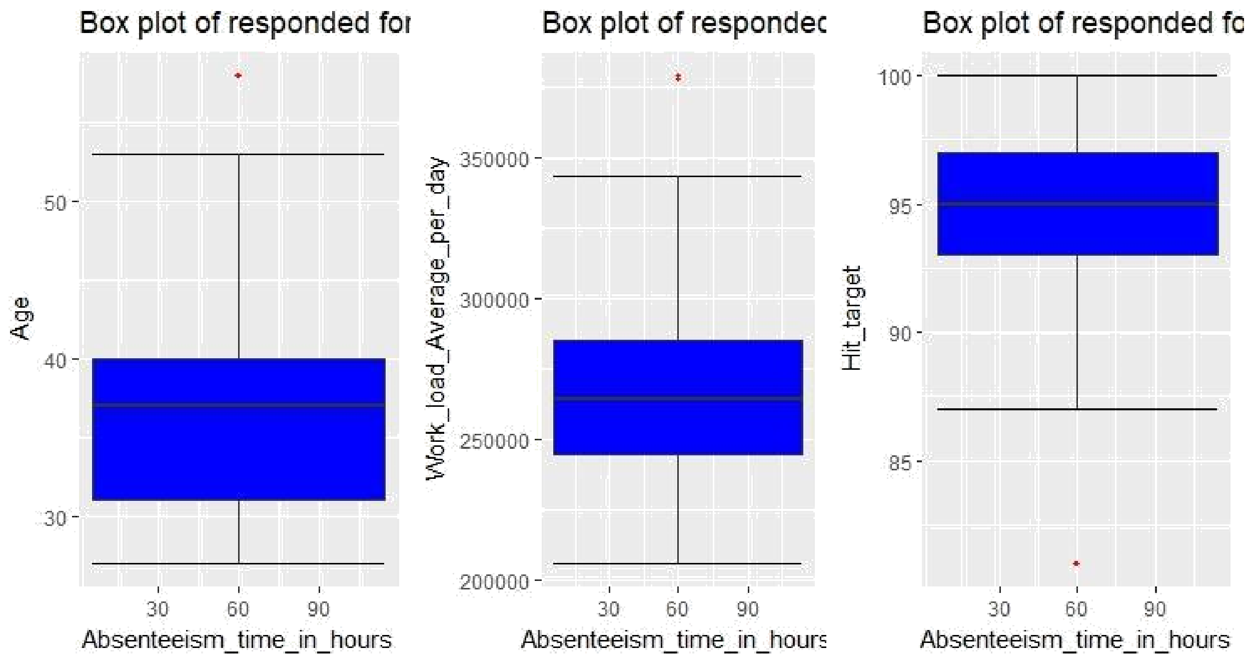

Box plot of responded for
Age
Absenteeism_time_in_hours
Box plot of respondec
Work_load_Average_per_day
Absenteeism_time_in_hours
Box plot of responded fc
Hit_target
Absenteeism_time_in_hours

Fig 2

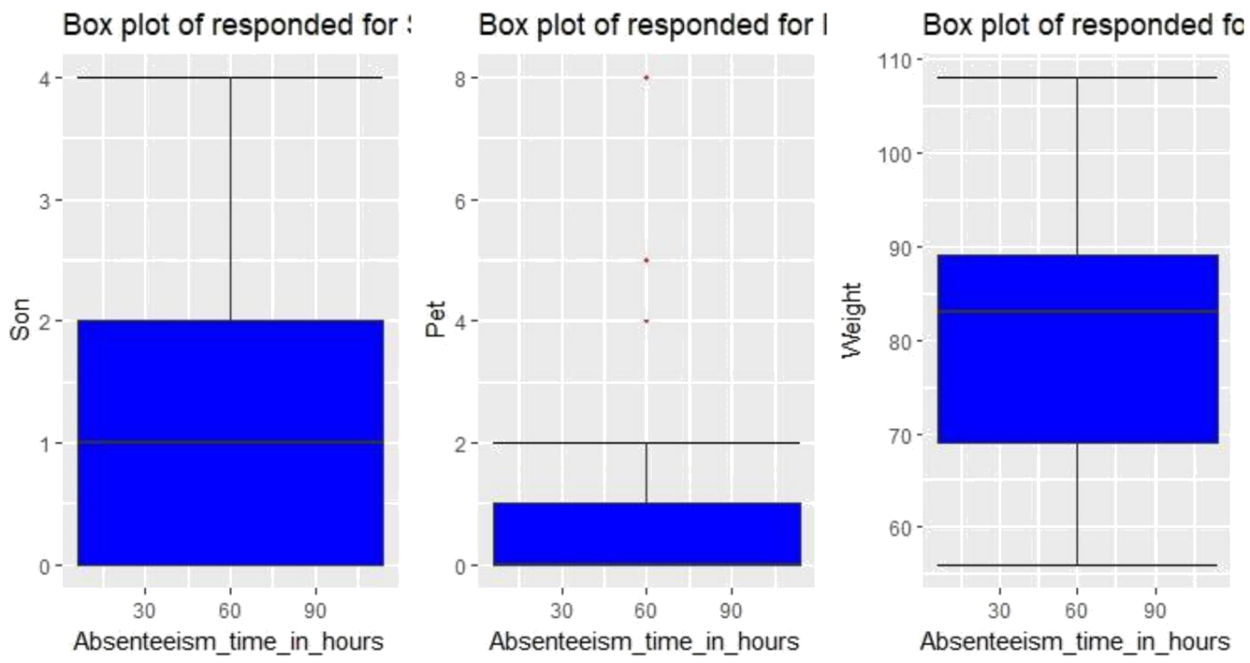


Fig 3

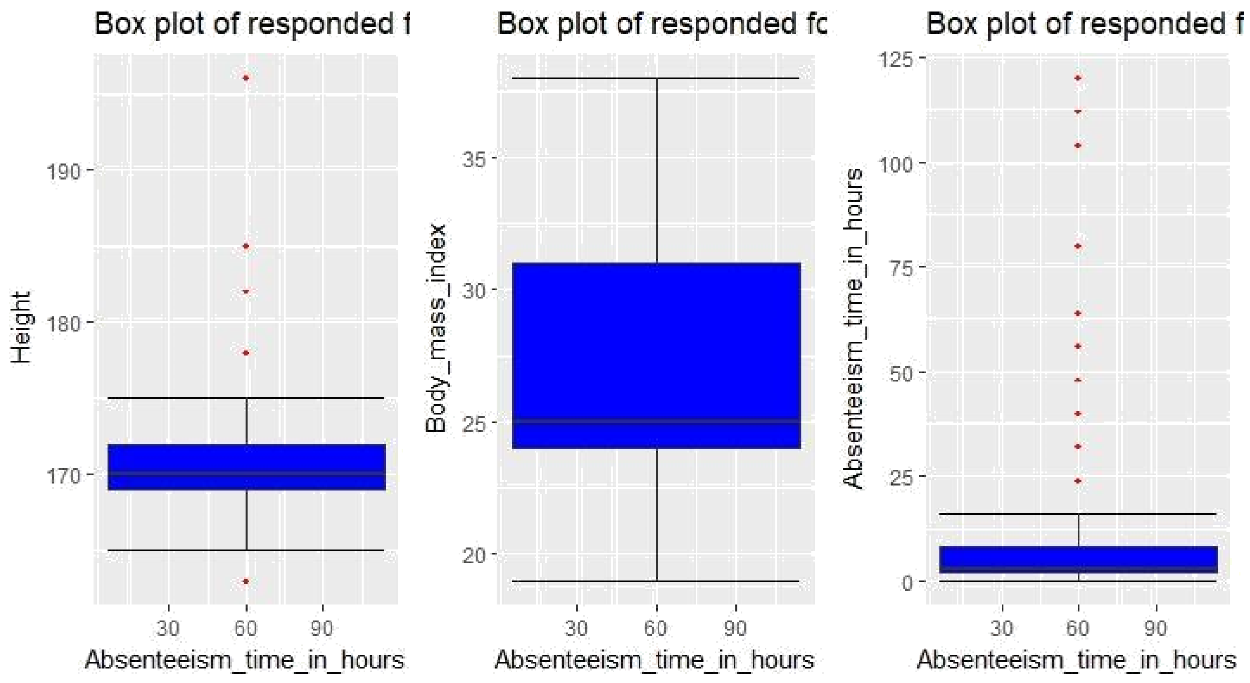
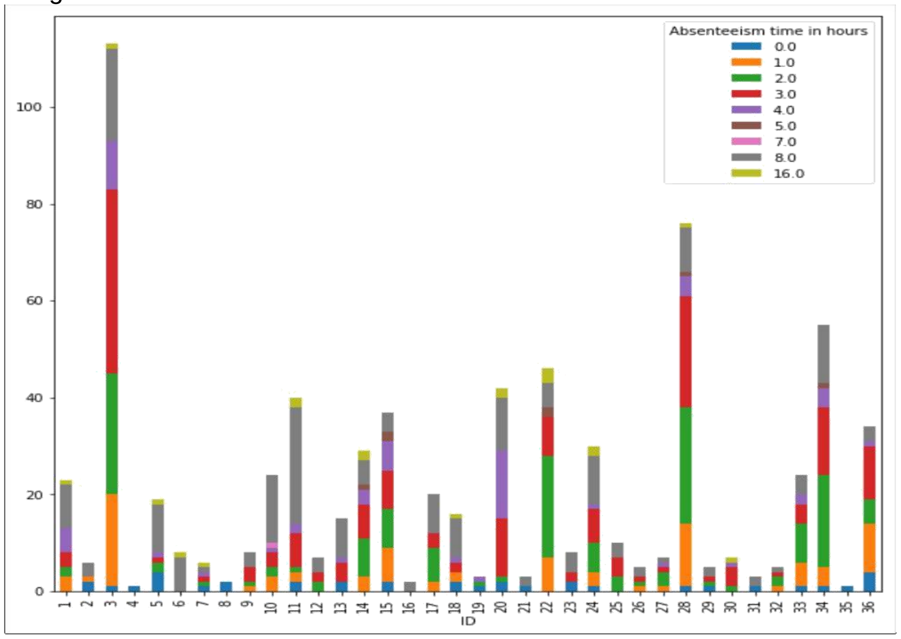


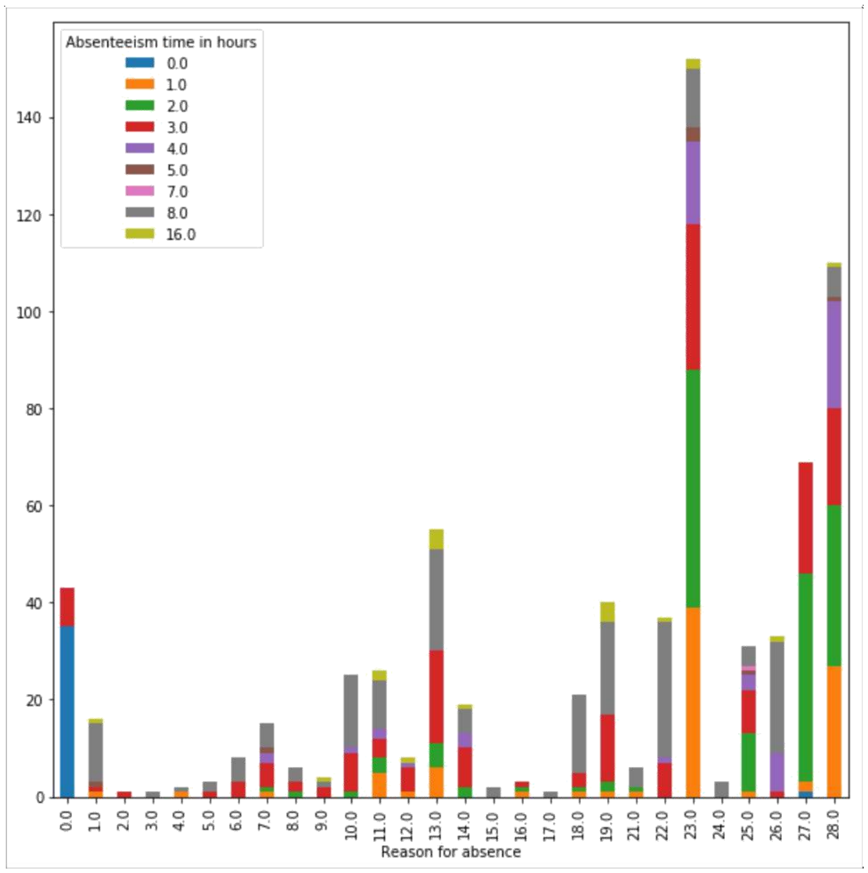
Fig 4

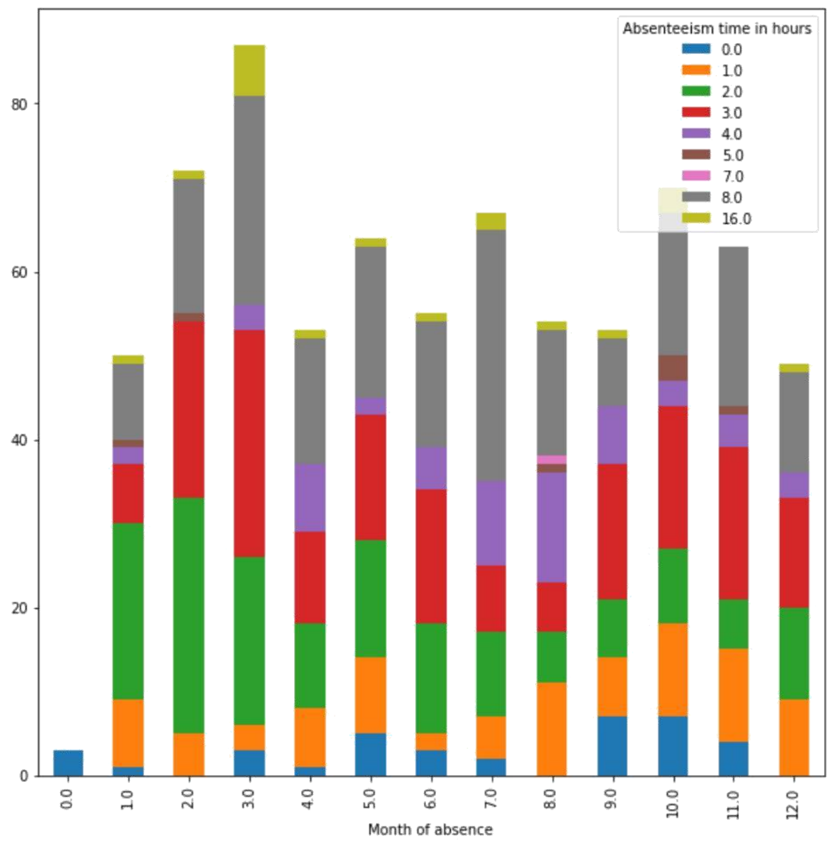
The boxplot of the variables have given the outliers for Transportation expense, Service time, Age, Work load Average/day, Hit target, Pet, Height, Absenteeism time in hours. Since the outliers effects the ANOVA and linear regression models they either have to be deleted or imputed by using NAN method. We use the NAN method and impute the outliers removed using median method.

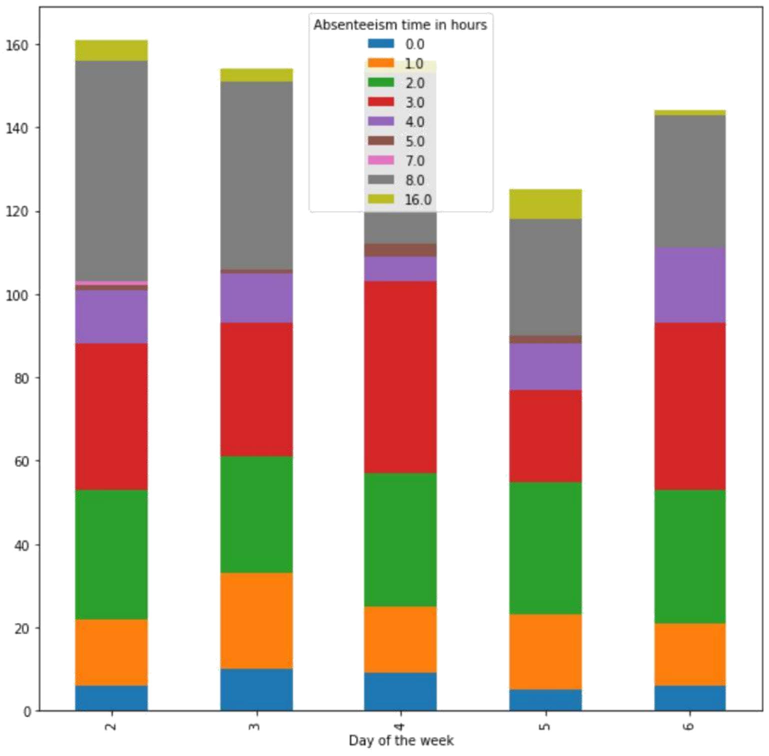
2.1.3 Feature Selection

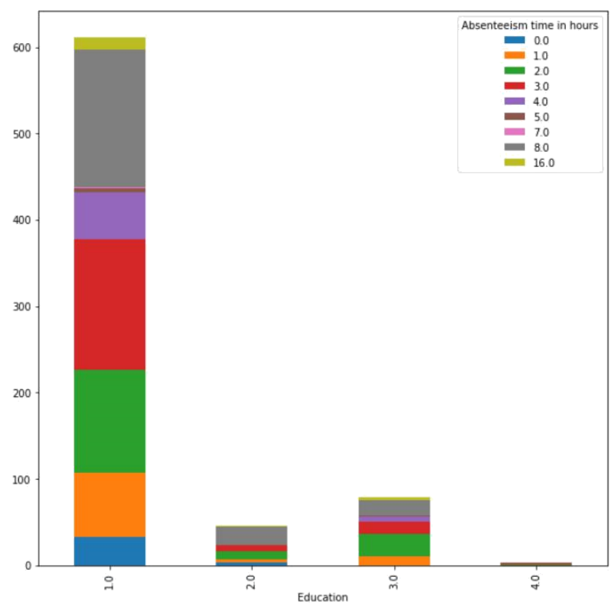
Variable selection is an important aspect of model building. It helps in building predictive models free from correlated variables, biases and unwanted noise. It helps in selecting a subset of relevant features (variables, predictors) for use in model construction and subset of a learning algorithm's input variables upon which it should focus attention, while ignoring the rest. Let's first see how our target variable is behaving with categorical variables in dataset.

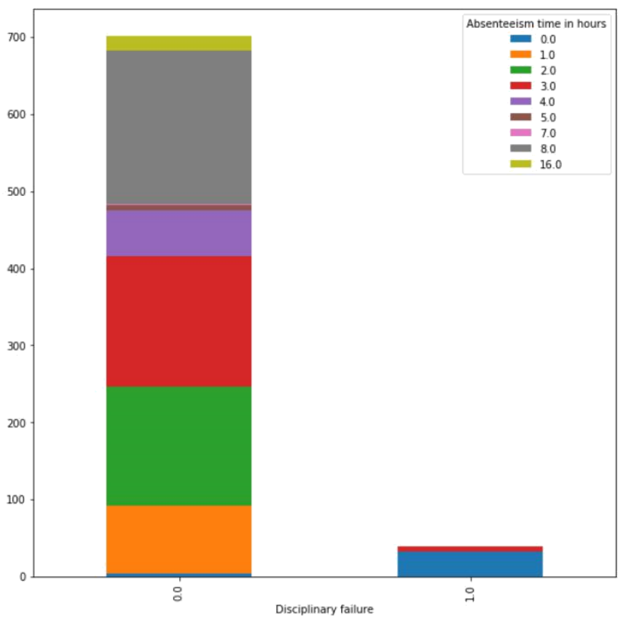


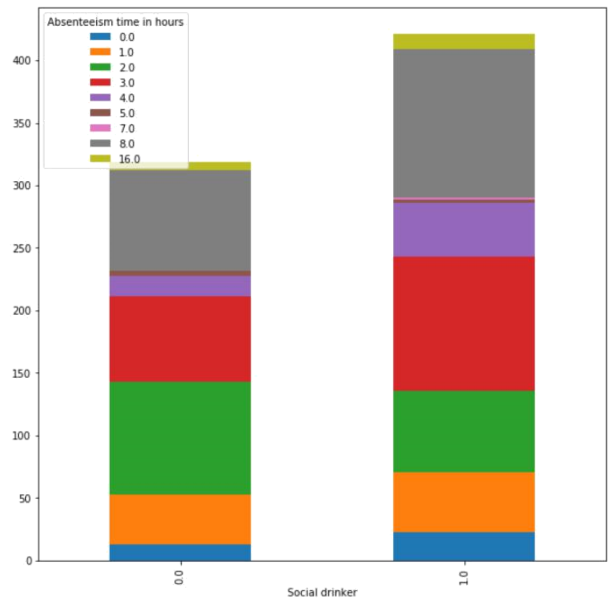


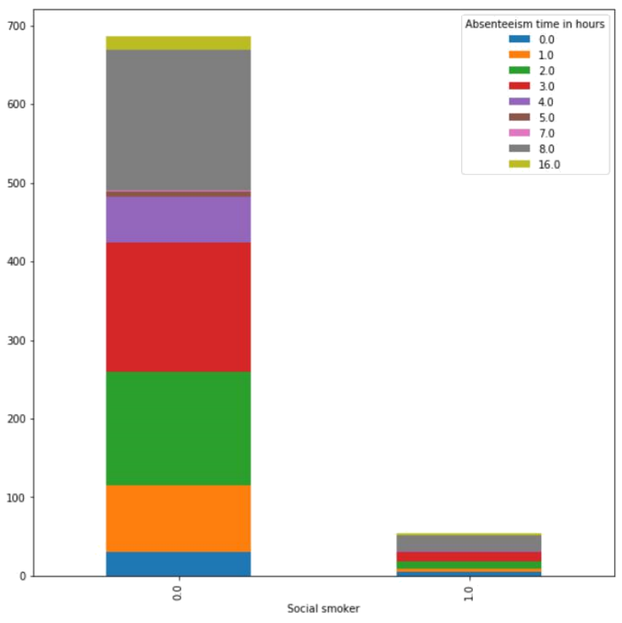

Absenteeism time in hours
0.0
1.0
2.0
3.0
4.0
5.0
7.0
8.0
16.0
0
20
40
60
80
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
Month of absence


Absenteeism time in hours
0.0
1.0
2.0
3.0
4.0
5.0
7.0
8.0
16.0
0
20
40
60
80
100
120
140
160
2
3
4
5
6
Day of the week









Categorical variables have already been assigned with levels. Details regarding the level could be seen in 1.2 Data. But the data provided is numerical so we have to convert them to categorical. Code for converting can be seen in appendix.

Correlation analysis

A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to see which pairs have the highest correlation. Checking at the correlation of each numerical variable has given below result.

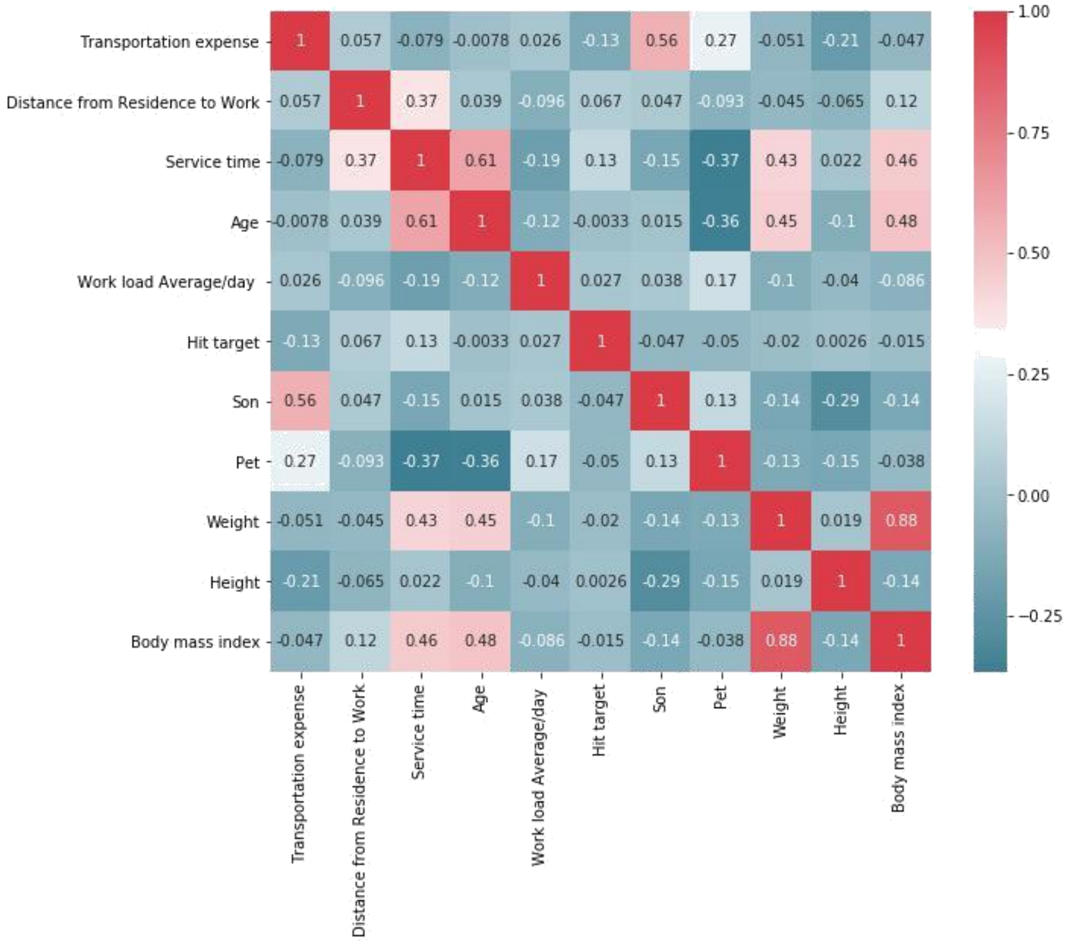


Fig 6 Correlation Matrix

Our correlation matrix shows some interesting results as follows

1. Only weight and body mass index are correlated maximum.
2. Service time and age are slightly correlated

We can now remove one of the highly correlated variable so that our model can perform well with much accuracy.

ANOVA

Analysis of variance (ANOVA) is a statistical technique that is used to check if the means of two or more groups are significantly different from each other. ANOVA checks the impact of one or more factors by comparing the means of different samples. As our target variable is numerical we will use ANOVA for feature selection technique to see whether any categorical variable is related to target variable. The result of anova is as follows:

```
                       Df Sum Sq Mean Sq F value Pr(>F)
ID                     35   1152   32.92   4.778<2e-16 ***
Reason_for_absence     27   2264   83.84  12.167<2e-16 ***
Month_of_absence       12     83    6.88   0.998 0.449
Day_of_week             4     30    7.38   1.071 0.370
Seasons                 3     18    6.07   0.881 0.451
Disciplinary_failure    1      3    2.83   0.410 0.522
Education               1      0    0.26   0.037 0.847
Social_drinker          1      9    9.43   1.369 0.242
Residuals             655   4513    6.89
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

If the p value of the categorical variable is less than 0.05 then we will consider that variable that is the target variable is dependent on the categorical variable for which we confirm the null hypothesis.

From the above result we can see that only one variable is very much related to target variable but we shouldn’t delete all the variables. So with some domain knowledge we delete the below mentioned variables

Therefore from both the correlation analysis and ANOVA we got some variable which we shouldn’t consider for further processing. The variables that could be deleted are as follows

Numerical: Hit target, Son, Pet, Height, Body mass index
Categorical: ID, Seasons, Education, Disciplinary failure

2.1.4 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing steps. If training an algorithm using different features and some of them are off the scale in their magnitude, then the results might be dominated by them. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. We use normalization here for feature scaling.

Normalization also called Min-Max scaling. It is the process of reducing unwanted variation either within or between variables. Normalization brings all of the variables into proportion with one another. It transforms data into a range between 0 and 1. We have to see the variables that are scattered highly and apply normalization. We normalize the following variables in our data so that we can process to the modeling phase. Normality check for variables is in appendix

Variables

Transportation expense, Workload Average/day, Distance from Residence to Work, Service time, Weight

Formulae used for normalization is

$$Value_{new} = \frac{Value - minValue}{maxValue - minValue}$$

Now our data is ready for applying models. We will use several machine learning regression models that reads the employee behaviour data and provide less error rate so that our prediction will be accurate

2.2 Modeling

2.2.1 Model Selection

After a thorough pre processing we will be using some regression models on our processed data to predict the target variable.

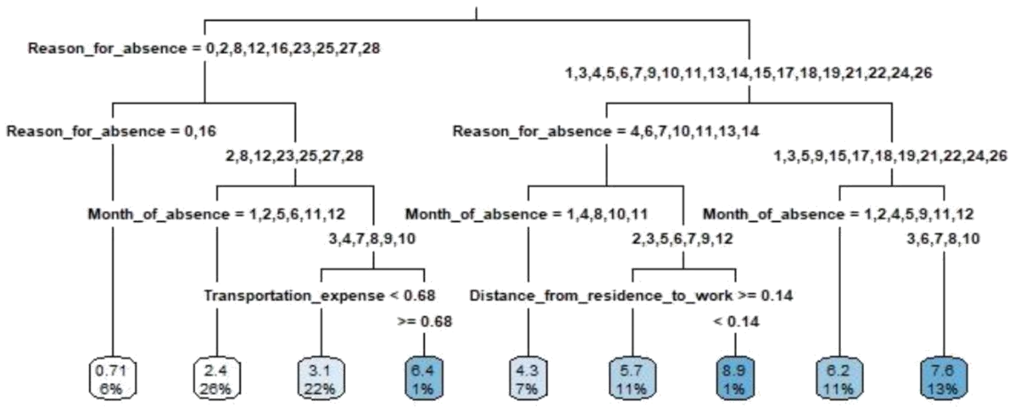
Decision Tree: Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. It can be used for classification and regression. It is a Supervised machine learning algorithm. Accept continuous and categorical variables as independent variables. Extremely easy to understand by the business users. Split of decision tree is seen in the below tree. Decision tree regression is as follows

```
n= 592

node), split, n, deviance, yval
      * denotes terminal node

 1) root 592 6618.18800 4.1875000
   2) Reason_for_absence=0,2,8,12,16,23,25,27,28 333 1485.86200 2.5645650
     4) Reason_for_absence=0,16 38   57.81579 0.7105263 *
     5) Reason_for_absence=2,8,12,23,25,27,28 295 1280.59700 2.8033900
      10) Month_of_absence=1,2,5,6,11,12 154  242.15580 2.3766230 *
      11) Month_of_absence=3,4,7,8,9,10 141  979.75890 3.2695040
        22) Transportation_expense< 0.6826923 133  720.09020 3.0827070 *
        23) Transportation_expense>=0.6826923 8  177.87500 6.3750000 *
   3) Reason_for_absence=1,3,4,5,6,7,9,10,11,13,14,15,17,18,19,21,22,24,26 25
9 3127.53700 6.2741310
     6) Reason_for_absence=4,6,7,10,11,13,14 114 1512.18400 5.4473680
      12) Month_of_absence=1,4,8,10,11 41  278.87800 4.3170730 *
      13) Month_of_absence=2,3,5,6,7,9,12 73 1151.50700 6.0821920
        26) Distance_from_residence_to_work>=0.1382979 65  918.55380 5.738462
0 *
        27) Distance_from_residence_to_work< 0.1382979 8  162.87500 8.8750000
*
     7) Reason_for_absence=1,3,5,9,15,17,18,19,21,22,24,26 145 1476.16600 6.9
241380
      14) Month_of_absence=1,2,4,5,9,11,12 67  528.47760 6.1940300 *
      15) Month_of_absence=3,6,7,8,10 78  881.29490 7.5512820 *
```

For the better understanding of the details let’s see the tree structure:

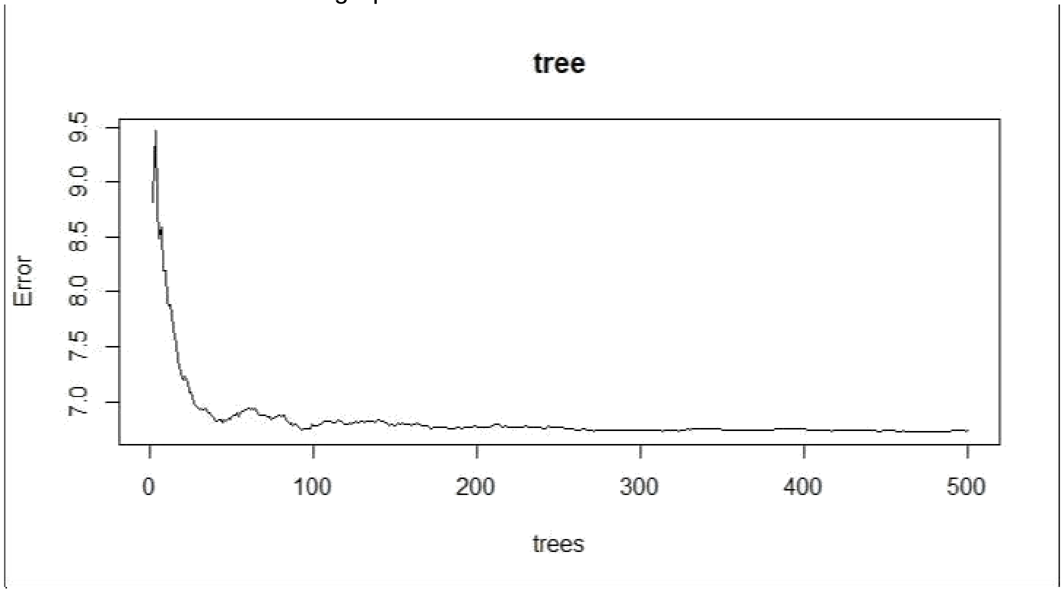

Reason_for_absence = 0,2,8,12,16,23,25,27,28
1,3,4,5,6,7,9,10,11,13,14,15,17,18,19,21,22,24,26
Reason_for_absence = 0,16
2,8,12,23,25,27,28
Reason_for_absence = 4,6,7,10,11,13,14
1,3,5,9,15,17,18,19,21,22,24,26
Month_of_absence = 1,2,5,6,11,12
3,4,7,8,9,10
Month_of_absence = 1,4,8,10,11
2,3,5,6,7,9,12
Month_of_absence = 1,2,4,5,9,11,12
3,6,7,8,10
Transportation_expense < 0.68
>= 0.68
Distance_from_residence_to_work >= 0.14
< 0.14
0.71 6%
2.4 26%
3.1 22%
6.4 1%
4.3 7%
5.7 11%
8.9 1%
6.2 11%
7.6 13%

Random Forest: Random Forest or decision tree forests is an ensemble learning method for classification, regression and other tasks. It consists of an arbitrary number of simple trees, which are used to determine the final outcome. In the regression problem, their responses are averaged to obtain an estimate of the dependent variable. Using tree ensembles can lead to significant improvement in prediction accuracy (i.e., better ability to predict new data cases). The goal of using a large number of trees is to train enough that each feature has a chance to appear in several models. We can see certain rules of random forest in the R code.

```
Call:
 randomForest(formula = Absenteeism_time_in_hours ~ ., data = train, ntrees = 
100)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 6.739174
                    % Var explained: 33.72
```

Error vs number of trees to be used graph is as follows



Linear Regression: Linear regression is the most basic type of regression and commonly used predictive analysis. Linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.
In the simple linear regression
• One variable, denoted x, is regarded as the predictor, explanatory, or independent variable.
• The other variable, denoted y, is regarded as the response, outcome, or dependent variable. The equation expressing this relationship is the line:

y = bo +b1x

Where bo = intercept, b1 = coefficient of variable (predictor) x

- **VIF:** The variance inflation factor (VIF) is the ratio of variance in a model with multiple terms, divided by the variance of a model with one term alone. It quantifies the severity of multicollinearity in an ordinary least squares regression analysis. It provides an index that measures how much the variance (the square of the estimate's standard deviation) of an estimated regression coefficient is increased because of collinearity. VIF for our data can be seen as follows. If VIF is 0 then we can use that data for linear regression model

```
No variable from the 10 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( Weight ~ Reason_for_absence ):  0.0006567935
max correlation ( Social_drinker ~ Distance_from_residence_to_work ):  0.4498
013

---------- VIFs of the remained variables --------
                          Variables      VIF
1               Reason_for_absence 1.087723
2                  Month_of_absence 1.070228
3                       Day_of_week 1.047472
4           Transportation_expense 1.516734
5  Distance_from_residence_to_work 1.454009
6                     Service_time 1.774908
7        Work_load_Average_per_day 1.056385
8                   Social_drinker 1.827485
9                    Social_smoker 1.129598
10                          Weight 1.503382
```

The summary of linear model can be seen as following:

```
Call:
lm(formula = Absenteeism_time_in_hours ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.5120 -1.3434 -0.0928  1.0988 12.8218 


Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)                       -0.41326    1.57166  -0.263 0.792692    
Reason_for_absence1                7.62096    0.94118   8.097 3.74e-15 ***
Reason_for_absence2                3.19057    2.56613   1.243 0.214282    
Reason_for_absence3                7.88183    2.57909   3.056 0.002353 ** 
Reason_for_absence4                4.10344    1.84921   2.219 0.026900 *  
Reason_for_absence5                8.43784    1.84583   4.571 6.01e-06 ***
Reason_for_absence6                6.07226    1.12748   5.386 1.08e-07 ***
```

```
Reason_for_absence7              4.78997    0.86558   5.534 4.89e-08 ***
Reason_for_absence8              4.47927    1.22311   3.662 0.000275 ***
Reason_for_absence9              4.82493    1.52174   3.171 0.001607 **
Reason_for_absence10             6.34788    0.71990   8.818  < 2e-16 ***
Reason_for_absence11             5.60355    0.71202   7.870 1.95e-14 ***
Reason_for_absence12             2.58765    1.23047   2.103 0.035929 *
Reason_for_absence13             5.25823    0.62273   8.444 2.84e-16 ***
Reason_for_absence14             4.00587    0.77859   5.145 3.75e-07 ***
Reason_for_absence15             7.33078    1.83707   3.990 7.50e-05 ***
Reason_for_absence16             1.93273    1.55957   1.239 0.215782
Reason_for_absence17             5.96669    2.59892   2.296 0.022067 *
Reason_for_absence18             6.83105    0.79049   8.642  < 2e-16 ***
Reason_for_absence19             5.86411    0.66231   8.854  < 2e-16 ***
Reason_for_absence21             5.25472    1.23542   4.253 2.48e-05 ***
Reason_for_absence22             5.99871    0.66754   8.986  < 2e-16 ***
Reason_for_absence23             2.60187    0.52103   4.994 8.00e-07 ***
Reason_for_absence24             7.67892    1.51602   5.065 5.61e-07 ***
Reason_for_absence25             2.89505    0.70739   4.093 4.92e-05 ***
Reason_for_absence26             7.26235    0.68091  10.666  < 2e-16 ***
Reason_for_absence27             2.27143    0.65151   3.486 0.000529 ***
Reason_for_absence28             2.69161    0.54471   4.941 1.04e-06 ***
Month_of_absence1               -0.64550    1.59160  -0.406 0.685220
Month_of_absence2                0.20235    1.55745   0.130 0.896673
Month_of_absence3                0.89393    1.55007   0.577 0.564378
Month_of_absence4               -0.06616    1.56185  -0.042 0.966226
Month_of_absence5               -0.31613    1.55515  -0.203 0.838994
Month_of_absence6               -0.38389    1.55522  -0.247 0.805125
Month_of_absence7                0.08287    1.55816   0.053 0.957605
Month_of_absence8                0.07131    1.58128   0.045 0.964047
Month_of_absence9               -0.24948    1.54687  -0.161 0.871931
Month_of_absence10               0.13137    1.53689   0.085 0.931913
Month_of_absence11              -0.34156    1.54826  -0.221 0.825479
Month_of_absence12              -0.22528    1.56906  -0.144 0.885887
Day_of_week3                     0.03602    0.33168   0.109 0.913556
Day_of_week4                    -0.26884    0.32709  -0.822 0.411496
Day_of_week5                     0.20787    0.34379   0.605 0.545668
Day_of_week6                    -0.27410    0.34097  -0.804 0.421825
Transportation_expense           0.62128    0.56219   1.105 0.269605
Distance_from_residence_to_work  0.03688    0.41871   0.088 0.929843
Service_time                    -0.48042    0.82909  -0.579 0.562525
Work_load_Average_per_day        0.90007    0.56924   1.581 0.114423
Social_drinker1                  0.44620    0.29469   1.514 0.130575
Social_smoker1                   0.79909    0.44409   1.799 0.072513 .
Weight                          -0.05523    0.53646  -0.103 0.918036
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.48 on 541 degrees of freedom
Multiple R-squared:  0.4473,  Adjusted R-squared:  0.3962
F-statistic: 8.755 on 50 and 541 DF,  p-value: < 2.2e-16
```

Ordinary Least Square Regression:

Dep. Variable:	Absenteeism time in hours	**R-squared:**	0.618
Model:	OLS	**Adj. R-squared:**	0.612
Method:	Least Squares	**F-statistic:**	94.28
Date:	Sun, 24 Jun 2018	**Prob (F-statistic):**	8.70e-115
Time:	11:12:30	**Log-Likelihood:**	-1556.1
No. Observations:	592	**AIC:**	3132.
Df Residuals:	582	**BIC:**	3176.
Df Model:	10		
Covariance Type:	Nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Reason for absence	-0.0166	0.016	-1.048	0.295	-0.048	0.015
Month of absence	0.0466	0.039	1.207	0.228	-0.029	0.123
Day of the week	-0.0097	0.094	-0.103	0.918	-0.195	0.175
Transportation expense	3.5625	0.794	4.487	0.000	2.003	5.122
Distance from Residence to Work	0.7127	0.547	1.304	0.193	-0.361	1.786
Service time	2.2732	1.125	2.021	0.044	0.064	4.483
Work load Average/day	3.4099	0.673	5.066	0.000	2.088	4.732
Social drinker	-0.3845	0.391	-0.983	0.326	-1.153	0.384
Social smoker	0.6959	0.590	1.179	0.239	-0.463	1.855
Weight	0.6376	0.674	0.945	0.345	-0.687	1.962

Omnibus:	115.747	**Durbin-Watson:**	1.983
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	214.203
Skew:	1.130	**Prob(JB):**	3.06e-47
Kurtosis:	4.892	**Cond. No.**	201.

Clearly the R squared value is less that is only 61%. This indicates that our target variable is very much dependent on the other variables in the dataset.

Chapter 3

Conclusion

3.1 Model Evaluation

Model evaluation is done on basis of evaluation metrics or error metrics. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results. Simply, building a predictive model is not our motive. But, creating and selecting a model which gives high accuracy on out of sample data. Hence, it is crucial to check accuracy or other metric of the model prior to computing predicted values. In our data as we applied regression models we have error metrics like Mean square error(MSE), MAPE, Root mean square error (RMSE), Mean absolute error (MAE) As out data is time variant data RMSE and MSE are the best error metrics to explain the accuracy of the models applied.

Decision Tree Regression:

```
MSE: 9.438635530514587
RMSE: 3.0722362426276053
```

Random Forest Regression

```
MSE: 7.494515821360468
RMSE: 2.737611334970775
```

Linear Regression Model:

```
MSE: 9.933191614028635
RMSE: 3.15169662468148
```

3.2 Model Selection

We can see that all models perform comparatively on average and therefore we select random forest classifier models for better prediction.

3.3 Answers

1. Company must bring the following changes for reducing absenteeism.

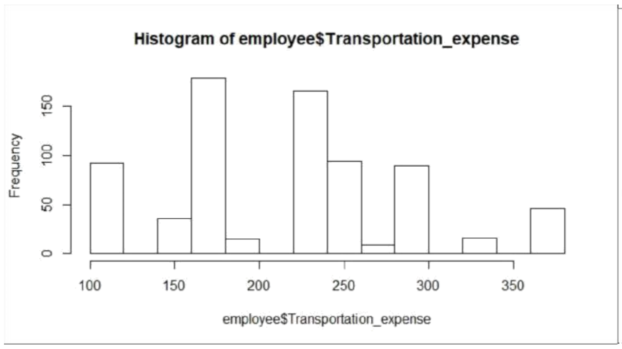
   We have 6% of absent with no reason. So some rules must be brought for changing this. We have 51% of absenteeism for blood donations, medical checkups. Company should provide medical checkups once in 6 months for employees so that the absenteeism could be reduced. 1% of people are absent due to smoking so it'd be better for company to bring a smoke zone. 10% of people are absent due to transportation expenses. So company must provide transportation facility and ask for little fee from employee so that this could be profitable for company help in reducing absenteeism.

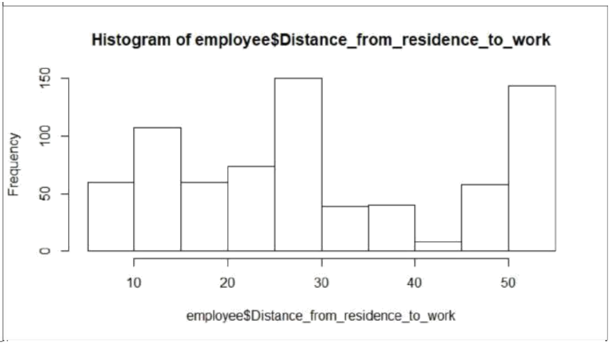
2. Loss for the compant in terms of work if same trend of absent continues.

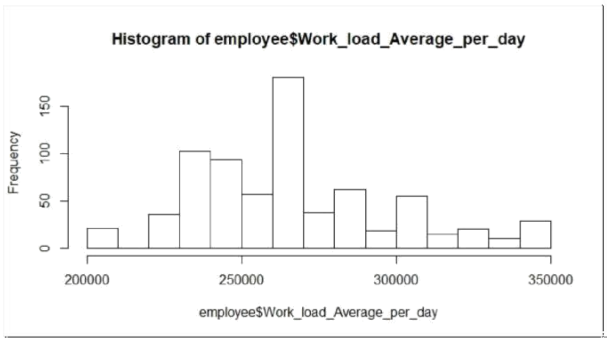
	Work Load Loss/Month
No Absent	0
Janaury	4606962
Febraury	5621917
March	9460589
April	4979442
May	4977071
June	4853828
July	7382847
August	4573790
September	3697632
October	6659389
November	5650168
December	4237499

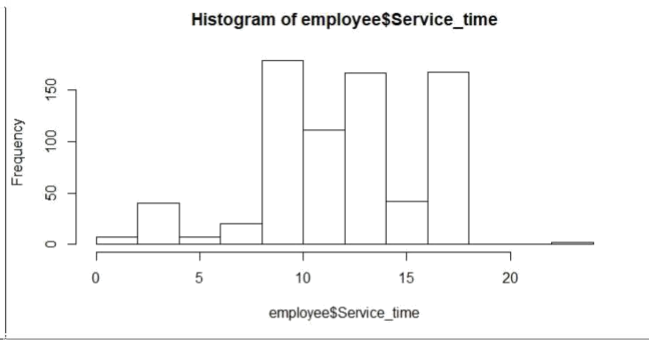
Appendix A - Extra Figures

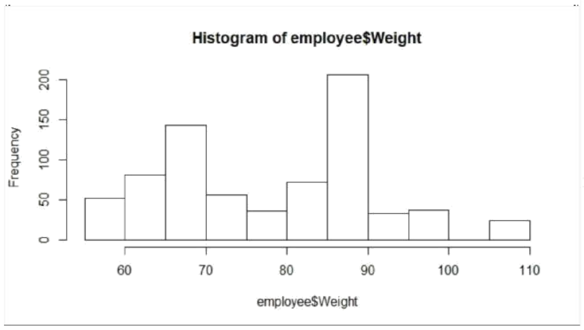
Normality check plots of various numerical variables:

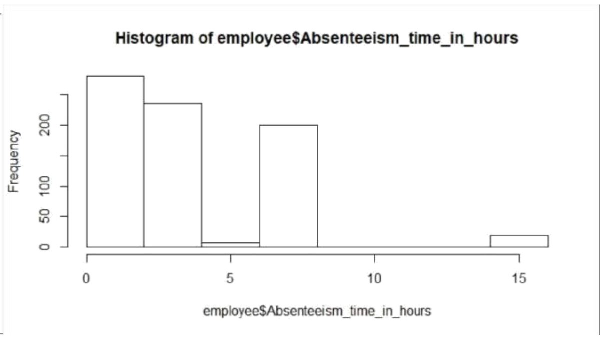












Appendix B - Code

R code

```
rm(list = ls())
setwd("C:/Users/chinna/Desktop/Data Science/Project/Employee Absenteeism")

#libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies",
"e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','readxl')

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

#data
employee = read_excel("Absenteeism.xls")
employee = as.data.frame(employee)

str(employee)

options(warn = -1)

#changing names of the variable.
colnames(employee)[colnames(employee) == 'Absenteeism time in hours'] =
'Absenteeism_time_in_hours'
colnames(employee)[colnames(employee) == 'Reason for absence'] = 'Reason_for_absence'
colnames(employee)[colnames(employee) == 'Month of absence'] = 'Month_of_absence'
colnames(employee)[colnames(employee) == 'Day of the week'] = 'Day_of_week'
colnames(employee)[colnames(employee) == 'Transportation expense'] = 'Transportation_expense'
colnames(employee)[colnames(employee) == 'Distance from Residence to Work'] =
'Distance_from_residence_to_work'
colnames(employee)[colnames(employee) == 'Service time'] = 'Service_time'
colnames(employee)[colnames(employee) == 'Work load Average/day'] =
'Work_load_Average_per_day'
colnames(employee)[colnames(employee) == 'Hit target'] = 'Hit_target'
colnames(employee)[colnames(employee) == 'Disciplinary failure'] = 'Disciplinary_failure'
colnames(employee)[colnames(employee) == 'Social drinker'] = 'Social_drinker'
colnames(employee)[colnames(employee) == 'Social smoker'] = 'Social_smoker'
colnames(employee)[colnames(employee) == 'Body mass index'] = 'Body_mass_index'

#missing value analysis
missing_val = data.frame(apply(employee,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$Missing_percentage/nrow(employee)) * 100
missing_val = missing_val[order(-missing_val$Missing_percentage),]
row.names(missing_val) = NULL
missing_val = missing_val[,c(2,1)]

#imputing missing vals
employee$`Reason_for_absence`[is.na(employee$`Reason_for_absence`)] =
median(employee$`Reason_for_absence`, na.rm = T)
employee$`Month_of_absence`[is.na(employee$`Month_of_absence`)] =
median(employee$`Month_of_absence`, na.rm = T)
```

```
employee$`Disciplinary_failure`[is.na(employee$`Disciplinary_failure`)] =
median(employee$`Disciplinary_failure`, na.rm = T)
employee$Education[is.na(employee$Education)] = median(employee$Education, na.rm = T)
employee$`Social_drinker`[is.na(employee$`Social_drinker`)] = median(employee$`Social_drinker`,
na.rm = T)
employee$`Social_smoker`[is.na(employee$`Social_smoker`)] = median(employee$`Social_smoker`,
na.rm = T)
employee$`Transportation_expense`[is.na(employee$`Transportation_expense`)] =
median.default(employee$`Transportation_expense`, na.rm = T)
employee$`Distance_from_residence_to_work`[is.na(employee$`Distance_from_residence_to_work`)
] = median(employee$`Distance_from_residence_to_work`, na.rm = T)
employee$`Service_time`[is.na(employee$`Service_time`)] = median(employee$`Service_time`, na.rm
= T)
employee$Age[is.na(employee$Age)] = median(employee$Age, na.rm = T)
employee$`Work_load_Average_per_day`[is.na(employee$`Work_load_Average_per_day`)] =
median(employee$`Work_load_Average_per_day`, na.rm = T)
employee$`Hit_target`[is.na(employee$`Hit_target`)] = median(employee$`Hit_target`, na.rm = T)
employee$Son[is.na(employee$Son)] = median(employee$Son, na.rm = T)
employee$Pet[is.na(employee$Pet)] = median(employee$Pet, na.rm = T)
employee$Weight[is.na(employee$Weight)] = median(employee$Weight, na.rm = T)
employee$Height[is.na(employee$Height)] = median(employee$Height, na.rm = T)
employee$`Body_mass_index`[is.na(employee$`Body_mass_index`)] =
median(employee$`Body_mass_index`, na.rm = T)
employee$`Absenteeism_time_in_hours`[is.na(employee$`Absenteeism_time_in_hours`)] =
median(employee$`Absenteeism_time_in_hours`, na.rm = T)


#converting variables to their types
employee$ID = as.factor(employee$ID)
employee$`Reason_for_absence`=as.factor(employee$`Reason_for_absence`)
employee$`Month_of_absence` = as.factor(employee$`Month_of_absence`)
employee$`Day_of_week` = as.factor(employee$`Day_of_week`)
employee$Seasons = as.factor(employee$Seasons)
employee$`Disciplinary_failure` = as.factor(employee$`Disciplinary_failure`)
employee$Education = as.factor(employee$Education)
employee$`Social_drinker` = as.factor(employee$`Social_drinker`)
employee$`Social_smoker`= as.factor(employee$`Social_smoker`)

str(employee)


############OUTLIER ANALYSIS#################

numeric_index = sapply(employee,is.numeric) #selecting only numeric
numeric_data = employee[,numeric_index] cnames =
colnames(numeric_data)

for (i in 1:length(cnames))
  {
    assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Absenteeism_time_in_hours"), data =
subset(employee))+
            stat_boxplot(geom = "errorbar", width = 0.5) +
            geom_boxplot(outlier.colour="red", fill = "blue" ,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 of responded for",cnames[i])))
}

#boxplot of outliers
```

```
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)

#outlier impute using NA technique
val = employee$`Transportation_expense`[employee$`Transportation_expense` %in%
boxplot.stats(employee$`Transportation_expense`)$out]
employee$`Transportation_expense`[(employee$`Transportation_expense` %in% val)] = NA
employee$`Transportation_expense`[is.na(employee$`Transportation_expense`)] =
median.default(employee$`Transportation_expense`, na.rm = T)

val = employee$`Service_time`[employee$`Service_time` %in%
boxplot.stats(employee$`Service_time`)$out]
employee$`Service_time`[(employee$`Service_time` %in% val)] = NA
employee$`Service_time`[is.na(employee$`Service_time`)] = median(employee$`Service_time`, na.rm
= T)

val = employee$Age[employee$Age %in% boxplot.stats(employee$Age)$out]
employee$Age[(employee$Age %in% val)] = NA
employee$Age[is.na(employee$Age)] = median(employee$Age, na.rm = T)

val = employee$`Work_load_Average_per_day`[employee$`Work_load_Average_per_day` %in%
boxplot.stats(employee$`Work_load_Average_per_day`)$out]
employee$`Work_load_Average_per_day`[(employee$`Work_load_Average_per_day` %in% val)] =
NA
employee$`Work_load_Average_per_day`[is.na(employee$`Work_load_Average_per_day`)] =
median(employee$`Work_load_Average_per_day`, na.rm = T)

val = employee$`Hit_target`[employee$`Hit_target` %in% boxplot.stats(employee$`Hit_target`)$out]
employee$`Hit_target`[(employee$`Hit_target` %in% val)] = NA
employee$`Hit_target`[is.na(employee$`Hit_target`)] = median(employee$`Hit_target`, na.rm = T)

val = employee$Pet[employee$Pet %in% boxplot.stats(employee$Pet)$out]
employee$Pet[(employee$Pet %in% val)] = NA
employee$Pet[is.na(employee$Pet)] = median(employee$Pet, na.rm = T)

val = employee$Height[employee$Height %in% boxplot.stats(employee$Height)$out]
employee$Height[(employee$Height %in% val)] = NA
employee$Height[is.na(employee$Height)] = median(employee$Height, na.rm = T)

val = employee$Absenteeism_time_in_hours[employee$Absenteeism_time_in_hours %in%
boxplot.stats(employee$Absenteeism_time_in_hours)$out]
employee$Absenteeism_time_in_hours[(employee$Absenteeism_time_in_hours %in% val)] = NA
employee$Absenteeism_time_in_hours[is.na(employee$Absenteeism_time_in_hours)] =
median(employee$Absenteeism_time_in_hours, na.rm = T)

rm(val)

###########FEATURE SELECTION##################

#correlation plot
corrgram(employee, order = F,
      upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")


str(employee)

df = employee
#employee = df
```

```
#ANOVA
result = aov(formula = Absenteeism_time_in_hours ~
ID+Reason_for_absence+Month_of_absence+Day_of_week+Seasons+Disciplinary_failure
          +Education+Social_drinker+Social_smoker , data = employee)
summary(result)


#library(randomForest)
#imp_var = randomForest(Absenteeism_time_in_hours ~ ., data = employee, ntree = 100,
#                    keep.forest = FALSE, importance = TRUE)
#importance(imp_var, type = 1)

#Dimentionality reduction
employee = subset(employee, select = -c(ID, Seasons, Education, Hit_target, Son, Pet, Height,
Body_mass_index,
                              Age, Disciplinary_failure))
############FEATURE SCALING##############
#Normality check
hist(employee$Transportation_expense)
hist(employee$Distance_from_residence_to_work)
hist(employee$Service_time)
hist(employee$Work_load_Average_per_day)
hist(employee$Weight)
hist(employee$Absenteeism_time_in_hours)

str(employee)

#Normalization
cnames =
c("Transportation_expense","Distance_from_residence_to_work","Service_time","Work_load_Averag
e_per_day",
        "Weight")

for(i in cnames){
  print(i)
  employee[,i] = (employee[,i] - min(employee[,i]))/
    (max(employee[,i] - min(employee[,i])))
}

##############MODEL IMPLEMENTATION

library(rpart)
library(MASS)
#install.packages("rpart.plot")
library(rpart.plot)

rmExcept(c("df","employee"))

train_index = sample(1:nrow(employee), 0.8 * nrow(employee))
train = employee[train_index,]
test = employee[-train_index,]

######DECISION TREE REGRESSION#####

fit = rpart(Absenteeism_time_in_hours ~ ., data = train, method = "anova")
fit
plt = rpart.plot(fit, type = 3, digits = 2, fallen.leaves = TRUE)

prediction_dt = predict(fit, test[-11])
```

```
actual = test[,11]
predicted = data.frame(prediction_dt)

error = actual - predicted

rmse <- function(error)
{
  sqrt(mean(error^2))
}

rmse(error)

#ERROR = 3.5147
#Accuracy = 96.4853

##### RANDOM FOREST REGRESSION#######
tree = randomForest(Absenteeism_time_in_hours ~ ., data = train, ntrees = 100)
tree
plot(tree)

prediction_rf = predict(tree, test[-11])

actual = test[,11]
predicted = data.frame(prediction_rf)

error = actual - predicted

rmse <- function(error)
{
  sqrt(mean(error^2))
}

rmse(error)

#ERROR = 2.7376
#Accuracy = 97.3624

library(usdm)
library(car)
library(VIF)

employee$`Reason_for_absence`=as.numeric(employee$`Reason_for_absence`)
employee$`Month_of_absence` = as.numeric(employee$`Month_of_absence`)
employee$`Day_of_week` = as.numeric(employee$`Day_of_week`)
employee$`Social_drinker` = as.numeric(employee$`Social_drinker`)
employee$`Social_smoker`= as.numeric(employee$`Social_smoker`)

str(employee)

vif(employee[,-11])
vifcor(employee[,-11], th = 0.9)

lm_model = lm(Absenteeism_time_in_hours ~ ., data = train)
summary(lm_model)

prediction_lr = predict(lm_model, test[,1:10])

#regr.eval(test[-14], prediction_dt, stats = c('mse','rmse','mape','mae'))
```

```
actual = test[,11]
predicted = data.frame(prediction_lr)

error = actual - predicted

rmse <- function(error)
{
  sqrt(mean(error^2))
}

rmse(error)

#ERROR = 3.1516
#Accuracy= 96.8484

str(employee)

########################## 2nd PART PREDICTION OF LOSS FOR THE COMPANY IN EACH
MONTH##############################

new = subset(df, select = c(Month_of_absence, Service_time, Absenteeism_time_in_hours,
Work_load_Average_per_day))

#Work loss = ((Work load per day/ service time)* Absenteeism hours)

new["loss"]=with(new,((new[,4]*new[,3])/new[,2]))

for(i in 1:12)
{
  d1=new[which(new["Month_of_absence"]==i),]
  print(sum(d1$loss))

}
```

Python Code

```
# coding: utf-8

# In[ ]:


#Load libraries
import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform


# In[ ]:
```

```
#Load data
os.chdir("C:/Users/chinna/Desktop/Data Science/Project/Employee Absenteeism")


# In[ ]:


#read data
employee = pd.read_excel("Absenteeism.xls")


# # Missing value Analysis

# In[ ]:


#Create dataframe with missing percentage
missing_val = pd.DataFrame(employee.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(employee))*100

#descending order
missing_val = missing_val.sort_values('Missing_percentage', ascending = False).reset_index(drop =
True)


# In[ ]:


missing_val


# In[ ]:


#details of data
employee.describe()


# In[ ]:


#Missing value Imputation using median method
employee['Reason for absence'] = employee['Reason for absence'].fillna(employee['Reason for
absence'].median())
employee['Month of absence'] = employee['Month of absence'].fillna(employee['Month of
absence'].median())
employee['Transportation expense'] = employee['Transportation
expense'].fillna(employee['Transportation expense'].median())
employee['Distance from Residence to Work'] = employee['Distance from Residence to
Work'].fillna(employee['Distance from Residence to Work'].median())
employee['Service time']= employee['Service time'].fillna(employee['Service time'].median())
employee['Age'] = employee['Age'].fillna(employee['Age'].median())
```

```
employee['Work load Average/day ']= employee['Work load Average/day '].fillna(employee['Work load
Average/day '].median())
employee['Hit target']= employee['Hit target'].fillna(employee['Hit target'].median())
employee['Disciplinary failure']= employee['Disciplinary failure'].fillna(employee['Disciplinary
failure'].median())
employee['Education']= employee['Education'].fillna(employee['Education'].median())
employee['Son']= employee['Son'].fillna(employee['Son'].median())
employee['Social drinker']= employee['Social drinker'].fillna(employee['Social drinker'].median())
employee['Social smoker']= employee['Social smoker'].fillna(employee['Social smoker'].median())
employee['Pet']= employee['Pet'].fillna(employee['Pet'].median())
employee['Weight']= employee['Weight'].fillna(employee['Weight'].median())
employee['Height']= employee['Height'].fillna(employee['Height'].median())
employee['Body mass index']= employee['Body mass index'].fillna(employee['Body mass
index'].median())
employee['Absenteeism time in hours']= employee['Absenteeism time in
hours'].fillna(employee['Absenteeism time in hours'].median())


# In[ ]:


df = employee.copy()


# # Outlier Analysis

# In[ ]:


#Converting variable to factors
employee['ID'] = employee['ID'].astype('category')
employee['Reason for absence'] = employee['Reason for absence'].astype('category')
employee['Month of absence'] = employee['Month of absence'].astype('category')
employee['Day of the week'] = employee['Day of the week'].astype('category')
employee['Seasons'] = employee['Seasons'].astype('category')
employee['Disciplinary failure'] = employee['Disciplinary failure'].astype('category')
employee['Education'] = employee['Education'].astype('category')
employee['Social drinker'] = employee['Social drinker'].astype('category')
employee['Social smoker'] = employee['Social smoker'].astype('category')


# In[ ]:


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


# In[ ]:


#boxplot of transportation
plt.boxplot(Numeric['Transportation expense'])


# In[ ]:


#boxplot of Distance from residence to work
```

```
plt.boxplot(Numeric['Distance from Residence to Work'])


# In[ ]:


#boxplot of service time
plt.boxplot(Numeric['Service time'])


# In[ ]:


#boxplot of age
plt.boxplot(Numeric['Age'])


# In[ ]:


#boxplot of workload
plt.boxplot(Numeric['Work load Average/day '])


# In[ ]:


#boxplot of hit target
plt.boxplot(Numeric['Hit target'])


# In[ ]:


#boxplot of son
plt.boxplot(Numeric['Son'])


# In[ ]:


#boxplot of pet
plt.boxplot(Numeric['Pet'])


# In[ ]:


#boxplot of weight
plt.boxplot(Numeric['Weight'])


# In[ ]:


#boxplot of height
plt.boxplot(Numeric['Height'])


# In[ ]:
```

```
#boxplot of bmi
plt.boxplot(Numeric['Body mass index'])


# In[ ]:


#boxplot of absenteeism
plt.boxplot(Numeric['Absenteeism time in hours'])


# In[ ]:


for i in Numeric:
    print(i)
    q75, q25 = np.percentile(Numeric.loc[:,i], [75 ,25])
    iqr = q75 - q25

    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)


# In[ ]:


#Replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Transportation expense'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Transportation expense'] < minimum,:'Transportation expense'] = np.nan
employee.loc[employee['Transportation expense'] > maximum,:'Transportation expense'] = np.nan


# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Service time'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)
```

```
#Replace with NA
employee.loc[employee['Service time'] < minimum,:'Service time'] = np.nan
employee.loc[employee['Service time'] > maximum,:'Service time'] = np.nan


# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Age'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Age'] < minimum,:'Age'] = np.nan
employee.loc[employee['Age'] > maximum,:'Age'] = np.nan


# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Work load Average/day '], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Work load Average/day '] < minimum,:'Work load Average/day '] = np.nan
employee.loc[employee['Work load Average/day '] > maximum,:'Work load Average/day '] = np.nan


# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Hit target'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Hit target'] < minimum,:'Hit target'] = np.nan
employee.loc[employee['Hit target'] > maximum,:'Hit target'] = np.nan
```

```
# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Pet'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Pet'] < minimum,:'Pet'] = np.nan
employee.loc[employee['Pet'] > maximum,:'Pet'] = np.nan


# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Height'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Height'] < minimum,:'Height'] = np.nan
employee.loc[employee['Height'] > maximum,:'Height'] = np.nan


# In[ ]:


#Detect and replace with NA
#Extract quartiles
q75, q25 = np.percentile(employee['Absenteeism time in hours'], [75 ,25])

#Calculate IQR
iqr = q75 - q25

#Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

#Replace with NA
employee.loc[employee['Absenteeism time in hours'] < minimum,:'Absenteeism time in hours'] =
np.nan employee.loc[employee['Absenteeism time in hours'] > maximum,:'Absenteeism time in hours']
= np.nan
```

```
# In[ ]:


#Impute replaced outliers
employee['Transportation expense'] = employee['Transportation
expense'].fillna(employee['Transportation expense'].median())
employee['Service time']= employee['Service time'].fillna(employee['Service time'].median())
employee['Height']= employee['Height'].fillna(employee['Height'].median())
employee['Pet']= employee['Pet'].fillna(employee['Pet'].median())
employee['Hit target']= employee['Hit target'].fillna(employee['Hit target'].median())
employee['Age'] = employee['Age'].fillna(employee['Age'].median())
employee['Work load Average/day ']= employee['Work load Average/day '].fillna(employee['Work load
Average/day '].median())
employee['Absenteeism time in hours']= employee['Absenteeism time in
hours'].fillna(employee['Absenteeism time in hours'].median())


# In[ ]:


employee['ID'] = df['ID']
employee['Reason for absence'] = df['Reason for absence']
employee['Month of absence'] = df['Month of absence']
employee['Day of the week'] = df['Day of the week']
employee['Seasons'] = df['Seasons']
employee['Distance from Residence to Work'] = df['Distance from Residence to Work']
employee['Disciplinary failure'] = df['Disciplinary failure'] employee['Education'] =
df['Education']
employee['Son'] = df['Son']
employee['Social drinker'] = df['Social drinker']
employee['Social smoker'] = df['Social smoker']
employee['Weight'] = df['Weight']
employee['Body mass index'] = df ['Body mass index']


# In[ ]:


Missing = pd.DataFrame(employee.isnull().sum())
Missing


# In[ ]:


#converting datatypes
employee['ID'] = employee['ID'].astype('category')
employee['Reason for absence'] = employee['Reason for absence'].astype('category')
employee['Month of absence'] = employee['Month of absence'].astype('category')
employee['Day of the week'] = employee['Day of the week'].astype('category')
employee['Seasons'] = employee['Seasons'].astype('category')
employee['Disciplinary failure'] = employee['Disciplinary failure'].astype('category')
employee['Education'] = employee['Education'].astype('category')
employee['Social drinker'] = employee['Social drinker'].astype('category')
employee['Social smoker'] = employee['Social smoker'].astype('category')


# # Feature Selection

# In[ ]:
```

```
Numerical = employee[['Transportation expense', 'Distance from Residence to
       Work', 'Service time', 'Age', 'Work load Average/day ', 'Hit target', 'Son',
       'Pet', 'Weight', 'Height', 'Body mass index']]


# In[ ]:


#correlation plot
corr = Numerical.corr()
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(10, 8))

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=
sns.diverging_palette(220,10,as_cmap=True),annot=True,
            square=True, ax=ax)


# In[ ]:


df = employee.copy()


# In[ ]:


employee.groupby(["ID", "Absenteeism time in hours"]).size().unstack().plot(kind='bar', stacked=True,
figsize=(10,10))


# In[ ]:


employee.groupby(["Reason for absence", "Absenteeism time in
hours"]).size().unstack().plot(kind='bar', stacked=True, figsize=(10,10))


# In[ ]:


employee.groupby(["Month of absence", "Absenteeism time in hours"]).size().unstack().plot(kind='bar',
stacked=True, figsize=(10,10))


# In[ ]:


employee.groupby(["Day of the week", "Absenteeism time in hours"]).size().unstack().plot(kind='bar',
stacked=True, figsize=(10,10))


# In[ ]:


employee.groupby(["Education", "Absenteeism time in hours"]).size().unstack().plot(kind='bar',
stacked=True, figsize=(10,10))
```

```
# In[ ]:


employee.groupby(["Disciplinary failure", "Absenteeism time in
hours"]).size().unstack().plot(kind='bar', stacked=True, figsize=(10,10))


# In[ ]:


employee.groupby(["Social drinker", "Absenteeism time in hours"]).size().unstack().plot(kind='bar',
stacked=True, figsize=(10,10))


# In[ ]:


employee.groupby(["Social smoker", "Absenteeism time in hours"]).size().unstack().plot(kind='bar',
stacked=True, figsize=(10,10))


# In[ ]:


#Droppong variables
employee = employee.drop(['ID', 'Seasons','Education','Hit target','Son','Pet','Height','Body mass
                        index', 'Age','Disciplinary failure'], axis=1)


# In[ ]:


employee.shape


# # Feature Scaling

# In[ ]:


cnames = ['Transportation expense','Distance from Residence to Work','Service time','Work load
Average/day ','Weight']


# In[ ]:


employee['Transportation expense'] = employee['Transportation expense'].astype('int64')
employee['Distance from Residence to Work'] = employee['Distance from Residence to
Work'].astype('int64')
employee['Service time'] = employee['Service time'].astype('int64')
employee['Work load Average/day '] = employee['Work load Average/day '].astype('int64')
employee['Weight'] = employee['Weight'].astype('int64')


# In[ ]:
```

```
#Nomalisation
for i in cnames:
    print(i)
    employee[i] = (employee[i] - min(employee[i]))/(max(employee[i]) - min(employee[i]))


# # MODEL APPLICATION

# In[ ]:


#DECISION TREE REGRESSION
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor


# In[ ]:


train, test = train_test_split(employee, test_size = 0.2)


# In[ ]:


train.shape,test.shape


# In[ ]:


fit = DecisionTreeRegressor(max_depth = 2).fit(train.iloc[:,0:10],train.iloc[:,10])


# In[ ]:


prediction_dt = fit.predict(test.iloc[:,0:10])


# In[ ]:


#ERROR MERTICS
from sklearn import metrics


# In[ ]:


print('MSE:', metrics.mean_squared_error(test.iloc[:,10], prediction_dt))
print('RMSE:', np.sqrt(metrics.mean_squared_error(test.iloc[:,10], prediction_dt)))


# In[ ]:


#RANDOM FOREST REGRESSION
from sklearn.ensemble import RandomForestRegressor
```

```
# In[ ]:


rf = RandomForestRegressor(n_estimators = 1000, random_state =
42).fit(train.iloc[:,0:10],train.iloc[:,10])


# In[ ]:


prediction_rf = rf.predict(test.iloc[:,0:10])


# In[ ]:


#ERROR METRICS
print('MSE:', metrics.mean_squared_error(test.iloc[:,10], prediction_rf))
print('RMSE:', np.sqrt(metrics.mean_squared_error(test.iloc[:,10], prediction_rf)))


# In[ ]:


#LINEAR REGRESSION
import statsmodels.api as sm


# In[ ]:


employee['Reason for absence'] = employee['Reason for absence'].astype('float')
employee['Month of absence'] = employee['Month of absence'].astype('float')
employee['Day of the week'] = employee['Day of the week'].astype('float')
employee['Social drinker'] = employee['Social drinker'].astype('float')
employee['Social smoker'] = employee['Social smoker'].astype('float')


# In[ ]:


train, test = train_test_split(employee, test_size = 0.2)


# In[ ]:


model = sm.OLS(train.iloc[:,10],train.iloc[:,0:10]).fit()


# In[ ]:


model.summary()


# In[ ]:
```

```
prediction_lr = model.predict(test.iloc[:,0:10])


# In[ ]:


#ERROR METRICS
print('MSE:', metrics.mean_squared_error(test.iloc[:,10], prediction_lr))
print('RMSE:', np.sqrt(metrics.mean_squared_error(test.iloc[:,10], prediction_lr)))


# # LOSS FOR THE COMPANY EACH MONTH

# In[ ]:


new = df[['Month of absence','Service time','Work load Average/day ','Absenteeism time in hours']]


# In[ ]:


new["Loss"]=(new['Work load Average/day ']*new['Absenteeism time in hours'])/new['Service time']


# In[ ]:


new.head()


# In[ ]:


new["Loss"] = np.round(new["Loss"]).astype('int64')


# In[ ]:


No_absent = new[new['Month of absence'] == 0]['Loss'].sum()
January = new[new['Month of absence'] == 1]['Loss'].sum()
February = new[new['Month of absence'] == 2]['Loss'].sum()
March = new[new['Month of absence'] == 3]['Loss'].sum()
April = new[new['Month of absence'] == 4]['Loss'].sum()
May = new[new['Month of absence'] == 5]['Loss'].sum()
June = new[new['Month of absence'] == 6]['Loss'].sum()
July = new[new['Month of absence'] == 7]['Loss'].sum()
August = new[new['Month of absence'] == 8]['Loss'].sum()
September = new[new['Month of absence'] == 9]['Loss'].sum()
October = new[new['Month of absence'] == 10]['Loss'].sum()
November = new[new['Month of absence'] == 11]['Loss'].sum()
December = new[new['Month of absence'] == 12]['Loss'].sum()


# In[ ]:


data = {'No Absent': No_absent, 'Janaury': January,'Febraury': February,'March': March,
       'April': April, 'May': May,'June': June,'July': July,
    'August': August,'September': September,'October': October,'November': November,
    'December': December}


# In[ ]:


WorkLoss = pd.DataFrame.from_dict(data, orient='index')


# In[ ]:


WorkLoss.rename(index=str, columns={0: "Work Load Loss/Month"})
```