PROJECT REPORT

EMPLOYEE ABSENTEEISM

Author: Prashant Raikwar

INDEX

Chapter 1

Introduction

Employee Absenteeism is the absence of an employee from work. It's a major problem faced by almost all employers of today. Employees are absent from work and thus the work suffers. Absenteeism of employees from work leads to back logs, piling of work and thus work delay.

1.1Problem 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

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case XYZ company has provided a data set with following features, we need to go through each and every variable of it to understand and for better functioning.
Dataset Characteristics: Timeseries Multivariant
Size of Dataset Provided: - 740 rows, 21 Columns (including dependent variable)
Missing Values: Yes

Below mentioned is a list of all the variable names with their meanings:

Variables	Description
Individual identification (ID)	ID of each employee
Reason for absence (ICD).	Absences attested by the International Code of Diseases (ICD) stratified into 21 (all of them mentioned below this table)
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	-

Service time	-
Age	-
Work load Average/day	-
Hit target	-
Disciplinary failure	yes=1; no=0
Education	-
Son	number of children
Social drinker	yes=1; no=0
Social smoker	yes=1; no=0
Pet	number of pets
Weight	-
Height	-
Body mass index	-
Absenteeism time in hours	**Target Variable**

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 behavioral disorders

VI. Diseases of the nervous system

VII. Diseases of the eye and adnexa

VIII. Diseases of the ear and mastoid process

IX. Diseases of the circulatory system

X. Diseases of the respiratory system

XI. Diseases of the digestive system

XII. Diseases of the skin and subcutaneous tissue

XIII. Diseases of the musculoskeletal system and connective tissue

XIV. Diseases of the genitourinary system

XV. Pregnancy, childbirth and the puerperium

XVI. Certain conditions originating in the perinatal period

XVII. Congenital malformations, deformations and chromosomal abnormalities

XVIII. Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified

XIX. Injury, poisoning and certain other consequences of external causes

XX. External causes of morbidity and mortality

XXI. Factors influencing health status and contact with health services

And 7 categories without (CID) patient follow-up (22), medical consultation (23), blood donation (24), laboratory examination (25), unjustified absence (26), physiotherapy (27), dental consultation (28).

For model simplification we have created broad categories of the above-mentioned reasons of absence Under five categories

Categories	Reasons for absense codes
1 -Code of Diseases	1 to 20 & 28
2 - medical consultation,	21, 22 & 27
3 - laboratory consultation,	23 & 24
4 - unjustified absence and	25
5 -Physiotherapy	26

Chapter 2

Methodology

➢ Pre-Processing

When we required to build a predictive model, we require to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps is combined under one shed which is **Exploratory Data Analysis**, which includes following steps:

- Looking into the data and analyzing various variables
- Missing Value Analysis
  - Imputation of missing values using median
  - Imputation of missing values using mode for categorical variables
- Outlier Analysis
  - Box plot to detect outliers
  - Elimination and imputation of outliers presented
- Feature Selection
  - Correlation Analysis
  - ANOVA (Two way Annova)
- Features Scaling
  - Normalization
- Dummy variable analysis

➢ Modelling

Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is modelling. Modelling plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. As per our problem statement and dataset, we will try some models on our preprocessed data and post comparing the output results we will select the best suitable model for our problem. As per our data set following models need to be tested:

- Decision Tree
- Random forest,
- Linear regression
- Gradient Boosting

❖ We have also used **Principal component analysis** is also used and post reducing the dimensions using PCA above models are again tested.

❖ Post applying PCA we have used hyper tuning technique with aim to improve the results

➢ Model Selection

The final step of our methodology will be the selection of the model based on the different output and results shown by different models. We have multiple parameters which we will study further in our report to test whether the model is suitable for our problem statement or not.

Chapter 3

Pre-Processing

3.1 Looking into the data and analyzing various variables

In this report we are trying to find out a solution to our problem of employee absenteeism for XYZ courier company. So here we have a data set of 21 variables containing 20 independent variables and one dependent variable.

Below are the names of Independent variables:
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, Disciplinary failure, Education, Son, Social drinker, Social smoker, Pet, Weight, Height, Body mass index

Our Dependent variable is: **Absenteeism time in hours**

We have also tried to find out the category of the variables:

Variable Name	**Variable Type**
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

Here **int64** means categorical or non-numeric variable and **float64** means numeric variables.

3.2 Uniqueness in Variable

We need to look at the unique number in the variables which help us to decide whether the variable is categorical or numeric. So, by using python script ‘nunique’ we tried to find out the unique values in each variable. We have also added the table below:

Variable Name	Unique Counts
ID	36
Reason for absence	28
Month of absence	13
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
Pet	6
Weight	26
Height	14
Body mass index	17
Absenteeism time in hours	19

Following are few points which are important for data understanding point:

- Since month variable can contain maximum 12 values, so here replace 0 with NA.
- We have divided the Work load Average/day variable by 1000.
- We have removed variable name ‘ID’ before processing as this variable is not carrying any relevant information.
- We have concluded that there are 10 continuous variable and 10 categorical variables in nature.

<u>Continuous variables</u> - 'Distance from Residence to Work', 'Service time', 'Age', 'Work load Average per day ', 'Transportation expense', 'Hit target', 'Weight', 'Height', 'Body mass index', 'Absenteeism time in hours'.

<u>Categorical Variables</u> - 'Reason for absence', 'Month of absence', 'Day of the week', 'Seasons', 'Disciplinary failure', 'Education', 'Social drinker', 'Social smoker', 'Pet', 'Son'.

3.3 Missing Value Analysis

Missing values in data is a common phenomenon in real world problems. It can have a significant effect on the conclusions that can be drawn from the data.

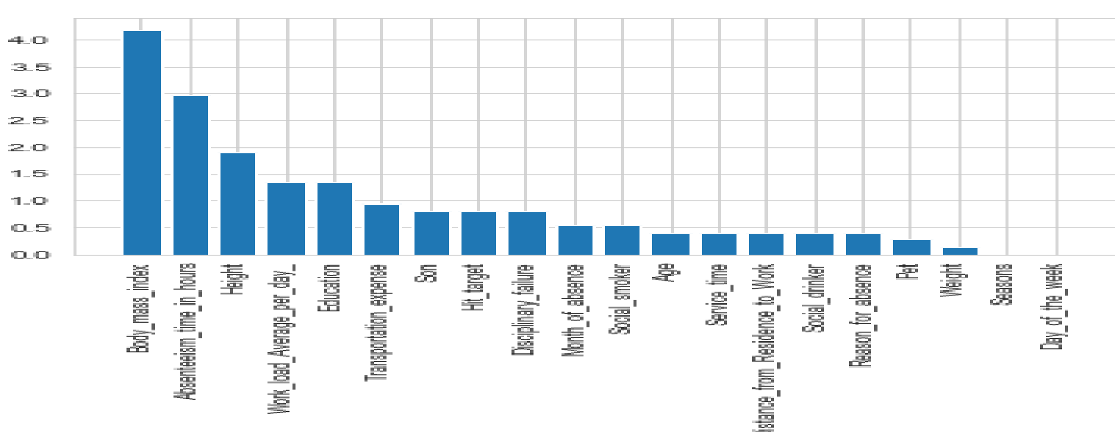
The most common reason for missing values in our data could be:

- Human error
- Refuse to answer
- Option to fill

As per the industry standards if a column has more than 30% of data as missing value either we ignore the entire column, or we ignore those observations.

Below table shows the counts of missing values of all the variables.

Variable Names	Count of Missing Values
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
Education	10
Son	6
Social drinker	3
Social smoker	4
Pet	2
Weight	1
Height	14
Body mass index	31
Absenteeism time in hours	22



In order to treat the missing value, we have multiple techniques through which we can impute the missing values of our data set. Following are the methods:

- Mean Imputation: Best suitable for continuous variables.
- Mode Imputation: Best suitable for binomial or categorical variables
- Median Imputation: When outliers are present this is best suitable.
- KNN Imputation: When segregation is required between two or more groups KNN is best suitable.

To check the best suitability we will try and test all above method except mode imputation on a single variable after intentionally replacing any of the provided value as 'NA' and try to impute using mean, median and KNN imputation techniques and compare the values of all imputation technique with the actual value and then we will freeze that imputation method whose value is the nearest to the actual value.

Here we have removed the rows of the missing values of our target variables because if we do not remove the missing values of target class and try to impute those values, we could have end up imputing biased or wrong values in our target variable which could impact the final result.

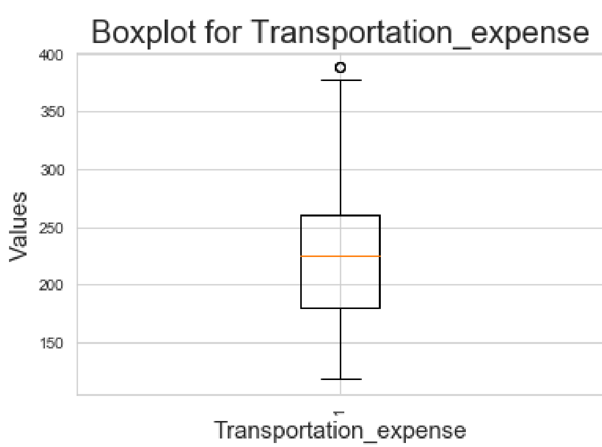
In our case we have chosen Median imputation for imputation of missing values as median imputation shows the nearest value to the actual value. And for categorical variables we have used mode imputation, as a standard procedure for imputation of categorical variables. Now further we go for our next step.

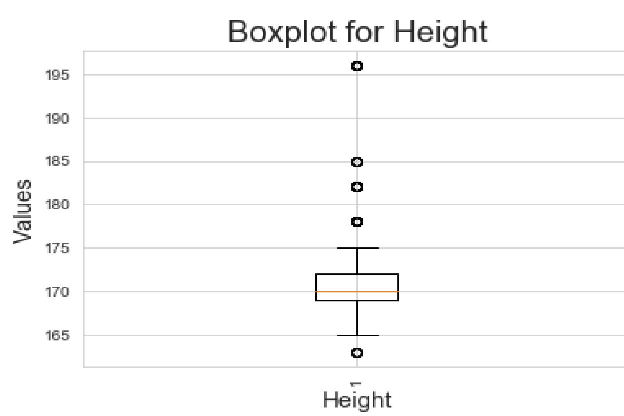
3.4 Outlier Analysis

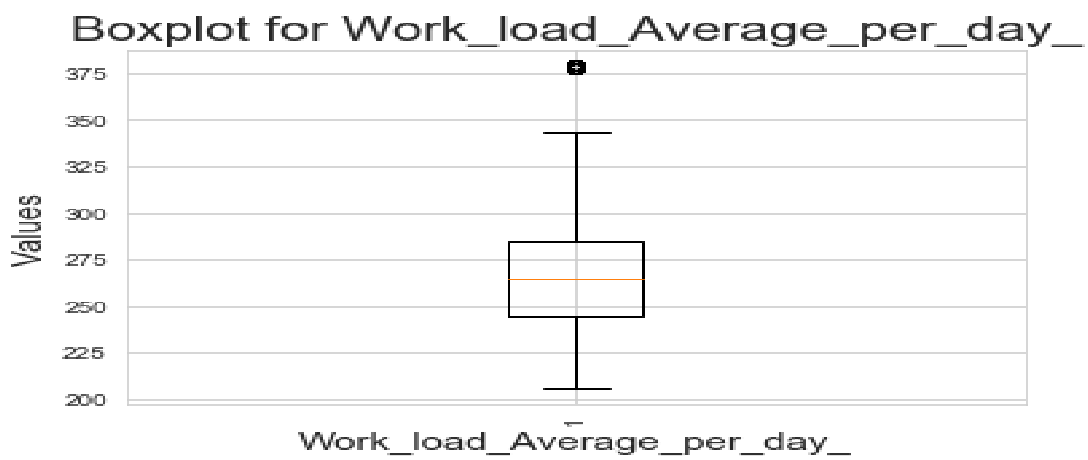
The next step of Preprocessing Technique is **Outliers Analysis**, An Outlier is a rare chance of occurrence within a given data set. In Data Science, an Outlier is an observation point that is distant from other observations. An Outlier may be due to variability in the measurement or it may indicate experimental error.

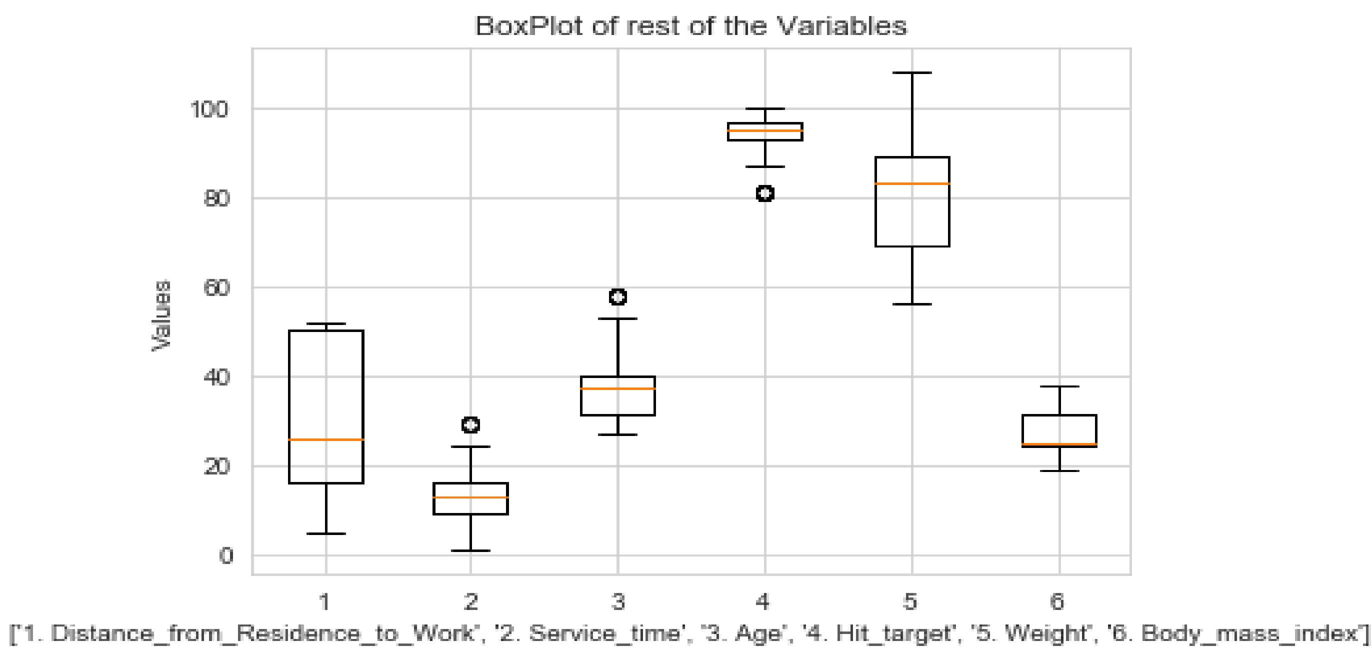
We will use boxplot technique to see whether outliers is present in continuous variables or not.

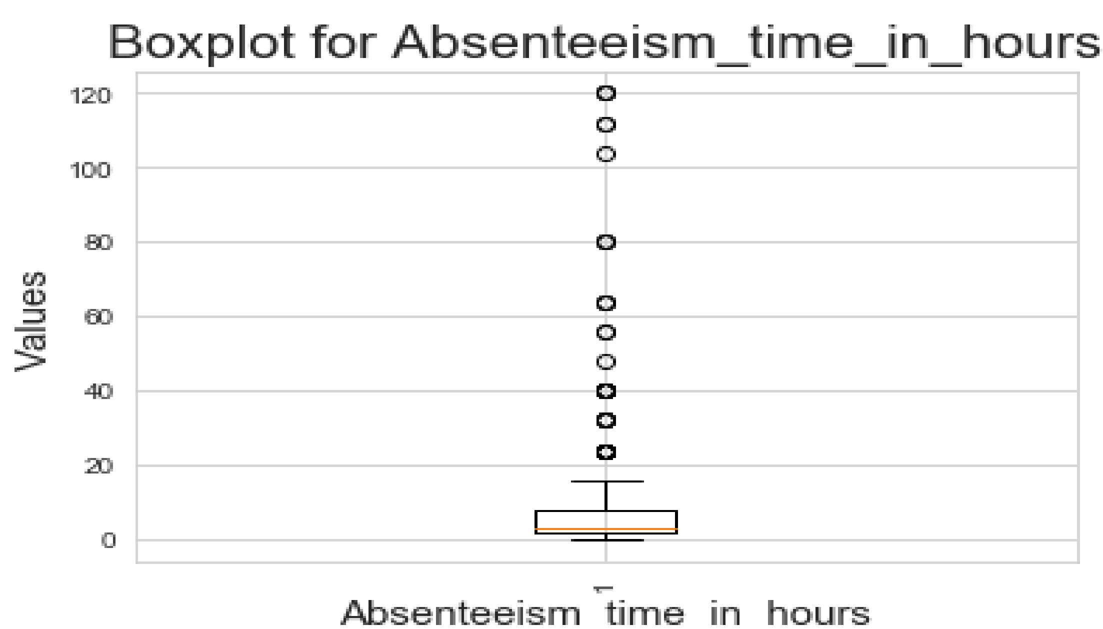
- **Boxplot: - It is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles**.






Boxplot for Work_load_Average_per_day_
Values
375
350
325
300
275
250
225
200
1
Work_load_Average_per_day_


BoxPlot of rest of the Variables
Values
100
80
60
40
20
0
1
2
3
4
5
6
['1. Distance_from_Residence_to_Work', '2. Service_time', '3. Age', '4. Hit_target', '5. Weight', '6. Body_mass_index']

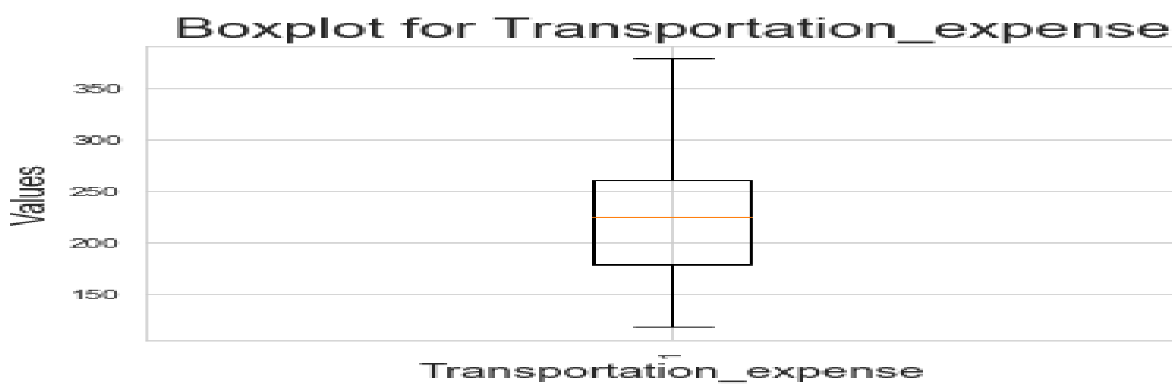

Boxplot for Absenteeism_time_in_hours
Values
120
100
80
60
40
20
0
1
Absenteeism_time_in_hours

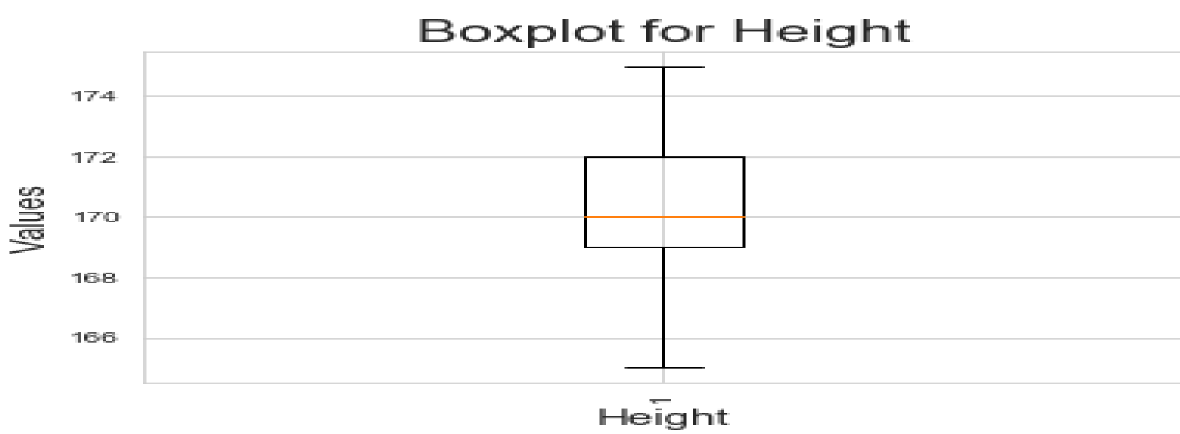
From the above boxplots we can see that variables "'Distance from Residence to Work', 'Weight' and 'Body mass index contains no outliers so in the treatment of outliers we are going to exclude these variables. By creating a subset of excluding the variables which are mentioned above and do not have outliers.

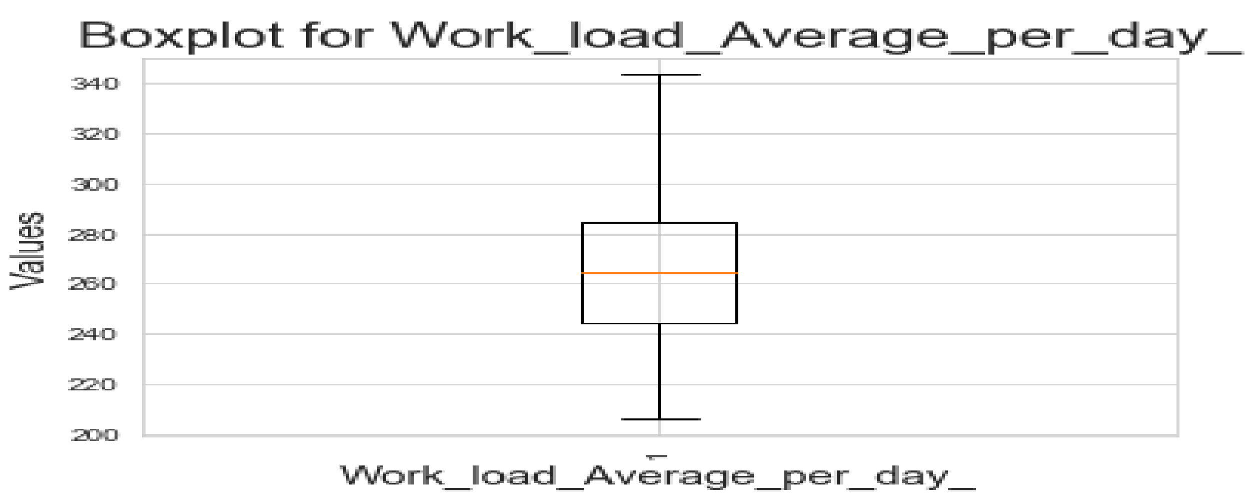
Treatment of outliers:

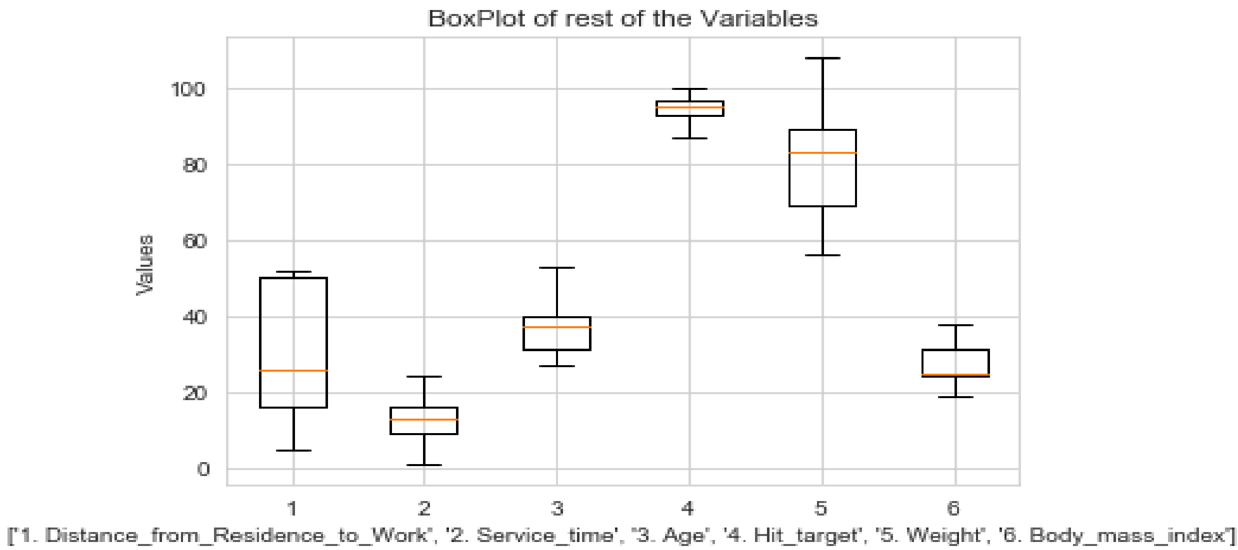
1. Firstly, we will remove the outlier figures presented in the variables and replace it with NA values.
2. After that we will treat the NA values as missing impute those values using same concept, we used in missing values.
3. We will check whether there is NA or missing values to impute.

<u>Below are the box plots after treating the outliers:</u>









As we can see all the outlier figures has been successfully imputed.

3.5 Feature Selection

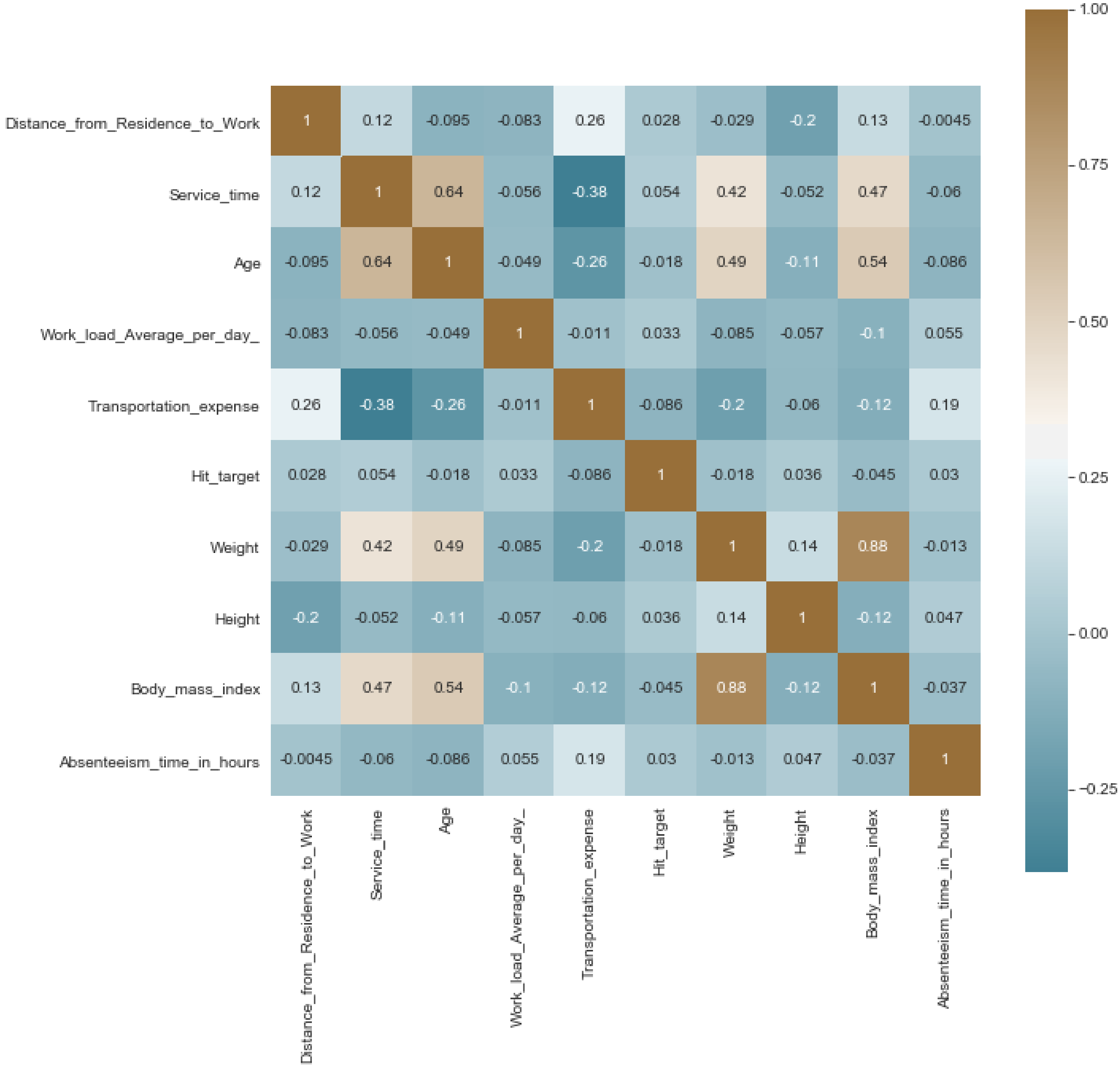
Feature selection is also called variable selection or attribute selection. Basic rule on which Machine learning works is– *if you put garbage in, you will only get garbage to come out*. By garbage here, we mean to say noise in the data.

When the number of features is very large, this rule becomes even more important. We need not use every feature at our disposal for creating an algorithm. We can assist our algorithm by feeding in only those features that are really important. We have witnessed feature subsets giving better results than complete set of features for the same algorithm or – *"Sometimes, less is better!"*.

The selection of features or variables is based on following two conditions:

1. The relationship between two independent variables should be less and
2. The relationship between Independent and Target variables should be high.

A **correlogram** or **correlation** matrix allows to analyze the relationship between each pair of numerical variables of a matrix.



The only variable which we are going to drop is '**weight'** as it is showing high correlation with '**Body mass Index'** variable which means both the variables are carrying the similar information so there would be no need to continue with both the variable, we can proceed further with only one variable out of these two. So, we are proceeding with **Body mass Index** variable.

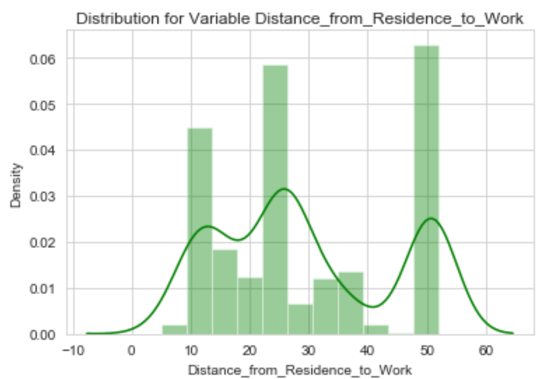
In this project we have selected Correlation Analysis for numerical variable and ANOVA testing categorical variable. So as per our hypothesis which is:
H0 – all the variables are independent of each other
H1 – all the variables are not independent of each other
We will drop categorical variables days of the week, seasons, education and social smoker for our further modelling steps basis our ANOVA results.

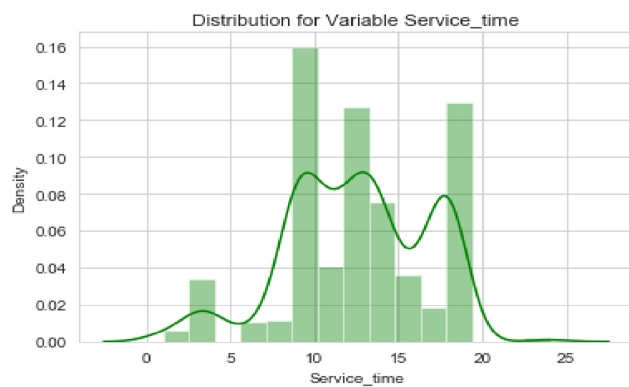
3.6 Features Scaling

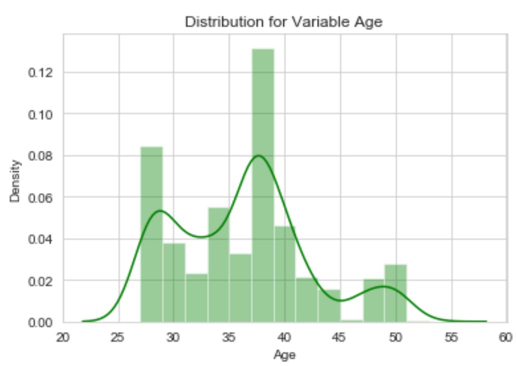
Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we tried to show the skewness of our variables and we find that our target variable absenteeism in hours having is one sided skewed so by using log transform technique we tried to reduce the skewness of the same.

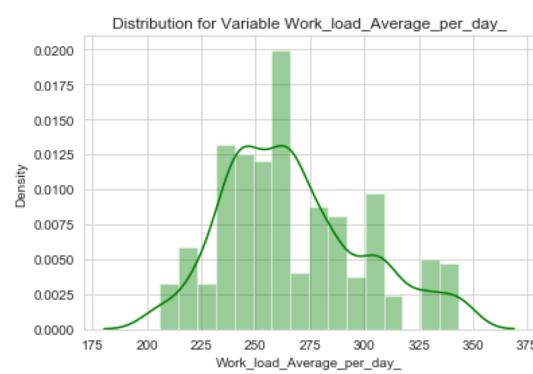
In the simplest cases, **normalization** of ratings means adjusting values measured on different scales to a notionally common scale, often prior to averaging. In more complicated cases, normalization may refer to more sophisticated adjustments where the intention is to bring the entire probability distributions of adjusted values into alignment. In the case of normalization of scores in educational assessment, there may be an intention to align distributions to a normal distribution. A different approach to normalization of probability distributions is quantile normalization, where the quantiles of the different measures are brought into alignment.

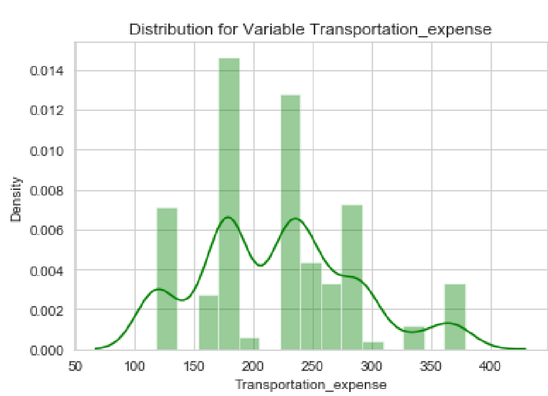
Below mentioned graphs shows the probability distribution plot to check distribution:

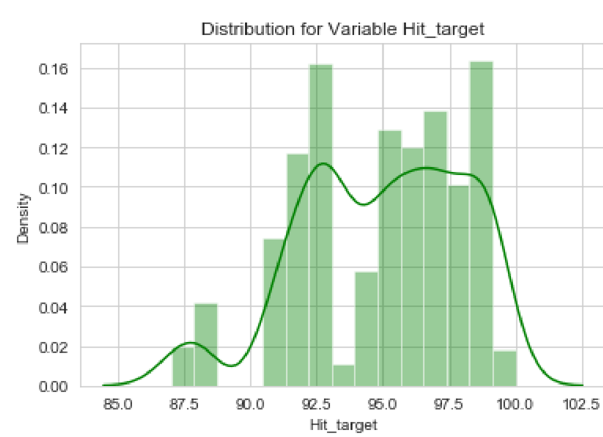


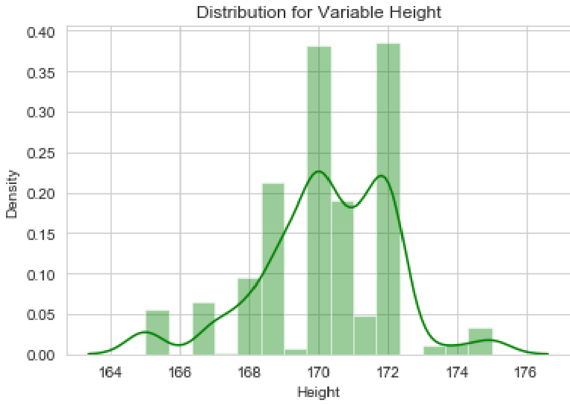


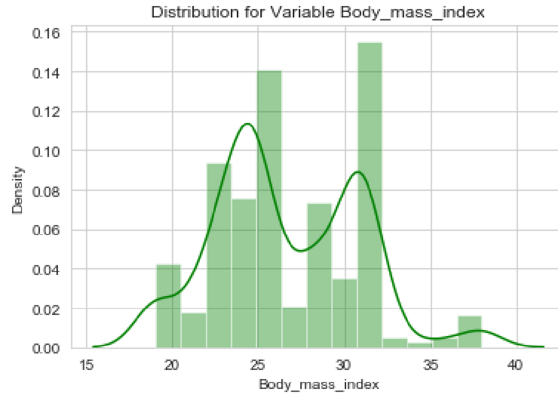


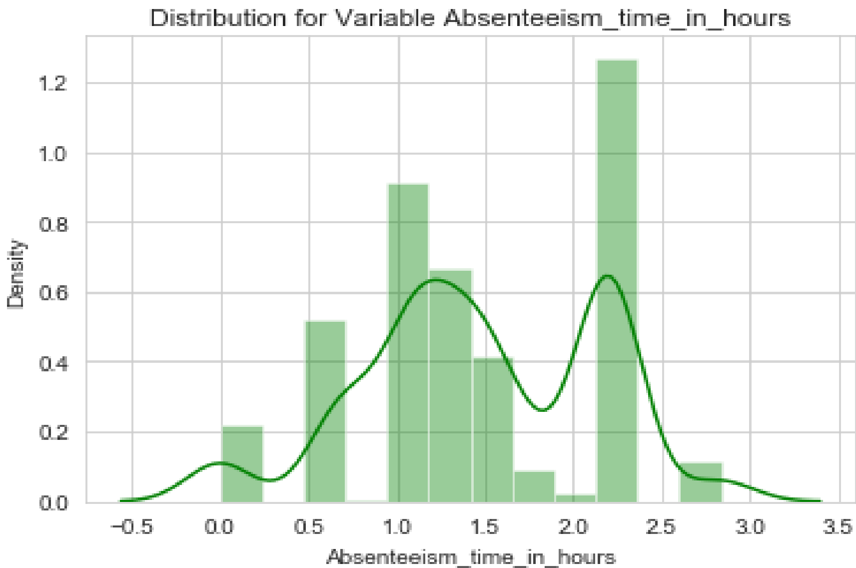








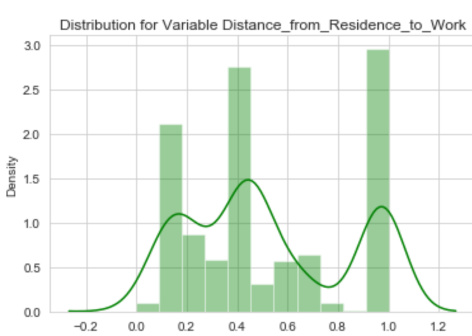


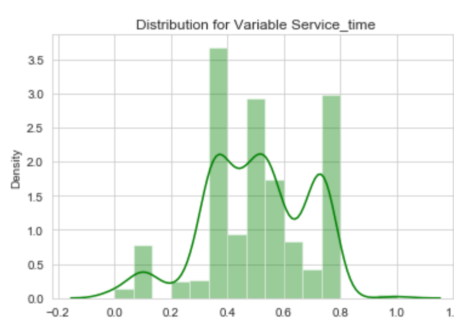


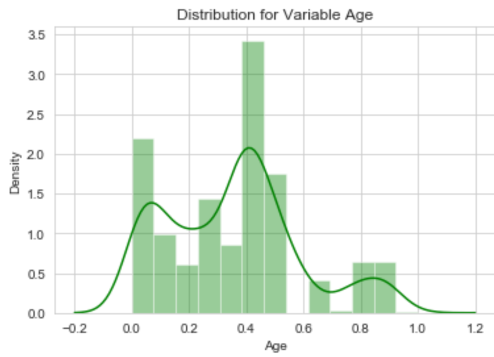
As we can see that our continuous variables are not normally distributed which means that we have to use normalization technique to normalize the values of each continuous variable to proceed for modelling stage. Rescaling data to have values between **0 and 1**. This is usually called feature scaling. One possible formula to achieve this is:

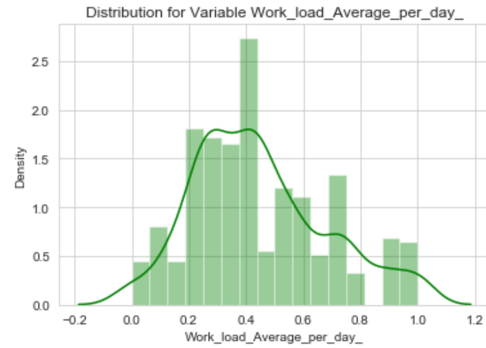
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

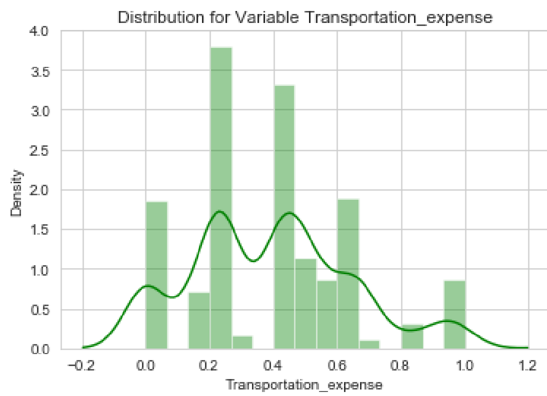
After normalization plot we have created distribution below:

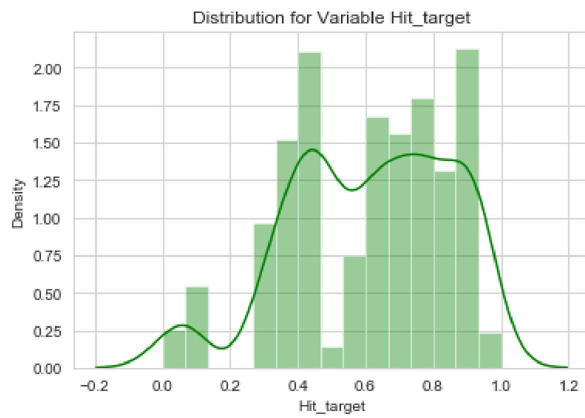


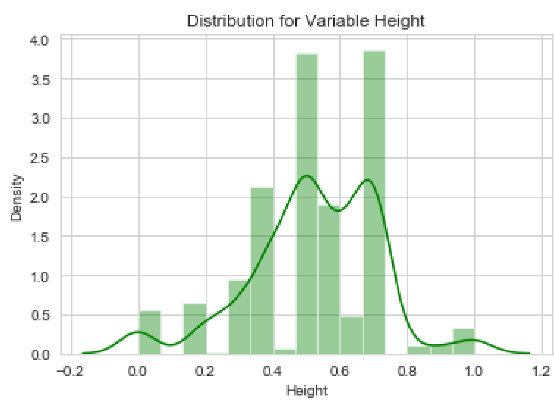


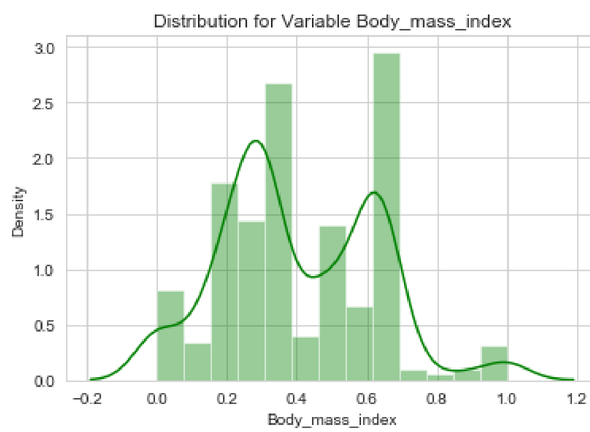

Distribution for Variable Age
Density
Age

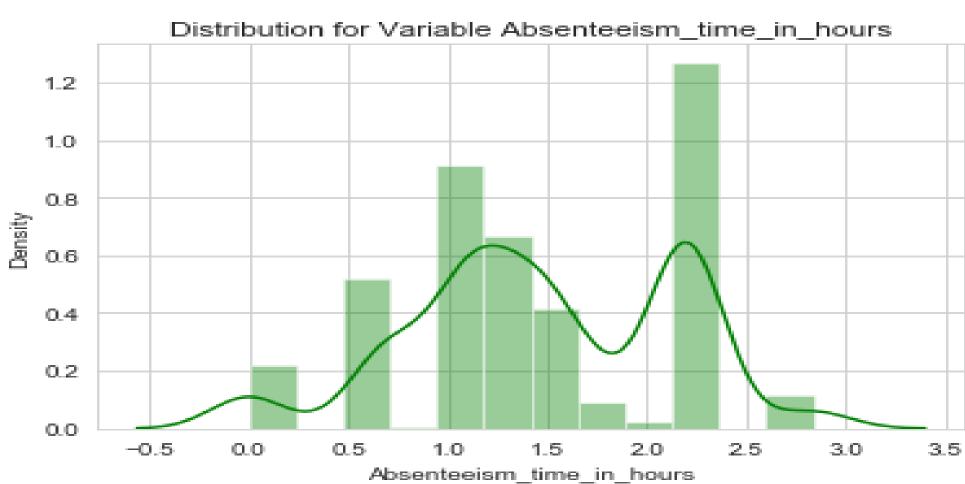

Distribution for Variable Work_load_Average_per_day_
Density
Work_load_Average_per_day_


Distribution for Variable Transportation_expense
Density
Transportation_expense


Distribution for Variable Hit_target
Density
Hit_target


Distribution for Variable Height
Density
Height


Distribution for Variable Body_mass_index
Density
Body_mass_index


Distribution for Variable Absenteeism_time_in_hours
Density
Absenteeism_time_in_hours

3.7 Dummy variable analysis

A dummy variable is a numerical variable used in regression analysis to represent the importance of categorical variables. Here we can see there are 6 categorical variables so we have created dummies for each categorical variable (there unique values). These dummy variables will interact with our model and show the significance of categorical variables in quantitative terms.

Below is the screenshot of the query we executed:

Creating Dummy variales

In [281]:
```
#### before going for algorithms we will use dummy variables analysis for categorical variables

dummy_emp_data = pd.get_dummies(data = emp_data, columns = cat)

# Copying dataframe
emp_data3 = dummy_emp_data.copy()
```

Chapter 4

Modelling

After a thorough preprocessing, we will use some regression models on our processed data to predict the target variable. Following are the models which we have built –

- Decision Tree
- Random Forest
- Linear Regression
- Gradient Boosting

Before we start building our models, we would like to know how our model will be evaluated.

4.1 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

Before running any model, we will split our data into two parts which is train and test data. Here in our case we have taken 80% of the data as our train data. Below is the snipped image of the split of train test.

Machine Learning algorithems

###train test split

In [284]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
```

In [285]:
```
##train test split for further modelling
X_train, X_test, y_train, y_test = train_test_split( dummy_emp_data.iloc[:, dummy_emp_data.columns != 'Absenteeism_time_in_hours
                                   dummy_emp_data.iloc[:, 8], test_size = 0.20, random_state = 1)
```

In [286]:
```
#checking the shape of the train and test
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(574, 40) (574,)
(144, 40) (144,)
```

Below is the screenshot of the query we executed and the result shown, we will compare the results of each model in a combined table later on.

Decision Tree Algorithm

###Decsion Tree

```
In [287]: fit_DT = DecisionTreeRegressor(max_depth = 2).fit(X_train,y_train)
```

```
In [288]: #prediction on train data
          pred_train_DT = fit_DT.predict(X_train)

          #prediction on test data
          pred_test_DT = fit_DT.predict(X_test)
```

```
In [289]: ##calculating RMSE for train data
          RMSE_train_DT = np.sqrt(mean_squared_error(y_train, pred_train_DT))

          ##calculating RMSE for test data
          RMSE_test_DT = np.sqrt(mean_squared_error(y_test, pred_test_DT))
```

```
In [290]: print("Root Mean Squared Error For Training data = "+str(RMSE_train_DT))
          print("Root Mean Squared Error For Test data = "+str(RMSE_test_DT))

          Root Mean Squared Error For Training data = 0.547396187805271
          Root Mean Squared Error For Test data = 0.5478922461592647
```

```
In [291]: ## R^2 calculation for train data
          r2_score(y_train, pred_train_DT)

Out[291]: 0.3169795409221665
```

```
In [292]: ## R^2 calculation for test data
          r2_score(y_test, pred_test_DT)

Out[292]: 0.26315319756909017
```

4.2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other task, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Below is a screenshot of the model we build and its output:

###Random Forest

```
In [293]: # Importing libraries for Random Forest
          from sklearn.ensemble import RandomForestRegressor

In [294]: # Building model on top of training dataset
          #here we have chose n_estimators = 200, we will further try to evaluate the performance of the model by tuning
          fit_RF = RandomForestRegressor(n_estimators = 200).fit(X_train,y_train)

In [295]: #prediction on train data
          pred_train_RF = fit_RF.predict(X_train)
          #prediction on test data
          pred_test_RF = fit_RF.predict(X_test)

In [296]: ##calculating RMSE for train data
          RMSE_train_RF = np.sqrt(mean_squared_error(y_train, pred_train_RF))
          ##calculating RMSE for test data
          RMSE_test_RF = np.sqrt(mean_squared_error(y_test, pred_test_RF))
```

```
In [297]: print("Root Mean Squared Error For Training data = "+str(RMSE_train_RF))
          print("Root Mean Squared Error For Test data = "+str(RMSE_test_RF))

          Root Mean Squared Error For Training data = 0.2433953880317634
          Root Mean Squared Error For Test data = 0.49696442270196534

In [298]: #calculate R^2 for test data
          r2_score(y_test, pred_test_RF)

Out[298]: 0.3937699042434335

In [299]: ## calculate R^2 for train data

          r2_score(y_train, pred_train_RF)

Out[299]: 0.8649623520053085
```

4.3 Linear Regression

Multiple linear regression is the most common form of linear regression analysis. Multiple regression is an extension of simple linear regression. It is used as a predictive analysis, when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable).

Below is a screenshot of the model we build and its output:

###Linear Regression

In [300]:
```python
# Importing libraries for Linear Regression
from sklearn.linear_model import LinearRegression
```

In [301]:
```python
# Building model on top of training dataset
fit_LR = LinearRegression().fit(X_train , y_train)
```

In [302]:
```python
#prediction on train data
pred_train_LR = fit_LR.predict(X_train)

#prediction on test data
pred_test_LR = fit_LR.predict(X_test)
```

In [303]:
```python
##calculating RMSE for train data
RMSE_train_LR = np.sqrt(mean_squared_error(y_train, pred_train_LR))

##calculating RMSE for test data
RMSE_test_LR = np.sqrt(mean_squared_error(y_test, pred_test_LR))
```

In [304]:
```python
print("Root Mean Squared Error For Training data = "+str(RMSE_train_LR))
print("Root Mean Squared Error For Test data = "+str(RMSE_test_LR))
```

```
Root Mean Squared Error For Training data = 0.49248961045203155
Root Mean Squared Error For Test data = 0.5337466860199246
```

In [305]:
```python
#calculate R^2 for train data
r2_score(y_train, pred_train_LR)
```

Out[305]: 0.447128348982633

In [306]:
```python
#calculate R^2 for test data

r2_score(y_test, pred_test_LR)
```

Out[306]: 0.3007100644511098

In [307]:
```python
#Linear Regression model for regression-
LR_model= sm.OLS(y_test,X_test).fit()
print(LR_model.summary())
```


In [307]:
```python
#Linear Regression model for regression-
LR_model= sm.OLS(y_test,X_test).fit()
print(LR_model.summary())
```


```
                             OLS Regression Results
==================================================================================
Dep. Variable:     Absenteeism_time_in_hours   R-squared:                  0.563
Model:                                   OLS   Adj. R-squared:             0.426
Method:                        Least Squares   F-statistic:                4.126
Date:                       Wed, 27 Mar 2019   Prob (F-statistic):      9.58e-09
Time:                               00:21:52   Log-Likelihood:           -80.107
No. Observations:                        144   AIC:                        230.2
Df Residuals:                            109   BIC:                        334.2
Df Model:                                 34
Covariance Type:                   nonrobust
```

4.4 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Below is a screenshot of the model we build and its output:

```
##Gradient Boosting

In [308]: # Importing library for GradientBoosting
          from sklearn.ensemble import GradientBoostingRegressor

In [309]: # Building model on top of training dataset
          fit_GB = GradientBoostingRegressor().fit(X_train, y_train)

In [310]: #prediction on train data
          pred_train_GB = fit_GB.predict(X_train)

          #prediction on test data
          pred_test_GB = fit_GB.predict(X_test)

In [311]: ##calculating RMSE for train data
          RMSE_train_GB = np.sqrt(mean_squared_error(y_train, pred_train_GB))

          ##calculating RMSE for test data
          RMSE_test_GB = np.sqrt(mean_squared_error(y_test, pred_test_GB))

In [312]: print("Root Mean Squared Error For Training data = "+str(RMSE_train_GB))
          print("Root Mean Squared Error For Test data = "+str(RMSE_test_GB))

          Root Mean Squared Error For Training data = 0.40159388901327125
          Root Mean Squared Error For Test data = 0.4680004269864299

In [313]: #calculate R^2 for test data
          r2_score(y_test, pred_test_GB)

Out[313]: 0.4623750771155887

In [314]: #calculate R^2 for train data
          r2_score(y_train, pred_train_GB)

Out[314]: 0.6323755677958434
```

4.5 Principal Component Analysis

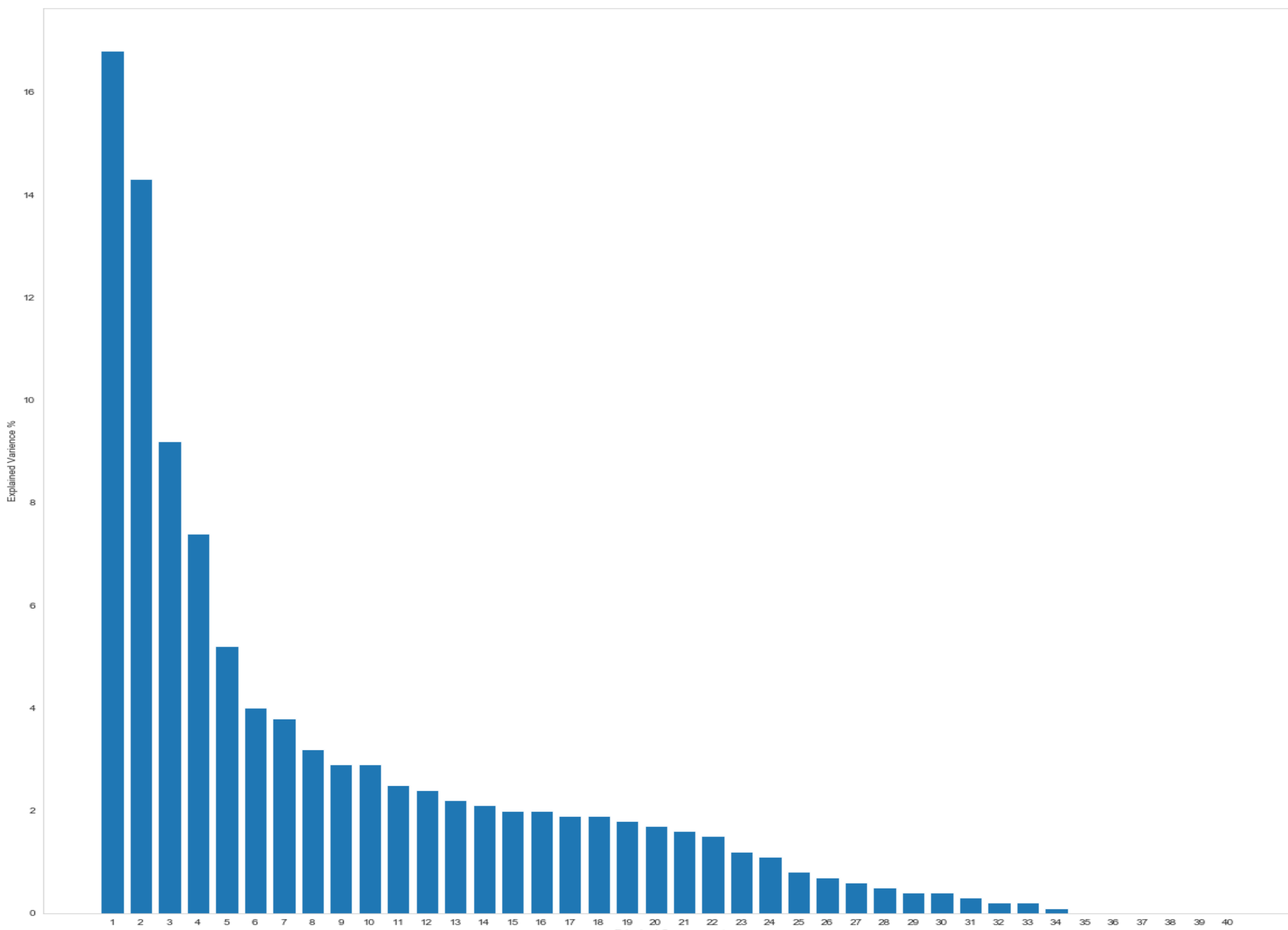
Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.

• Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.

• The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

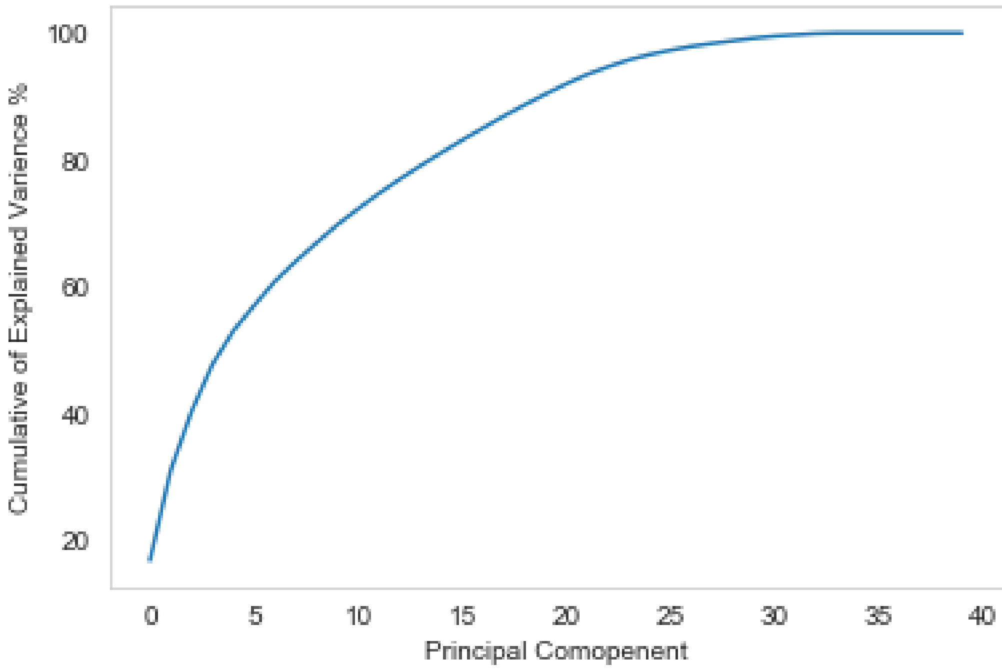
• Principal components analysis is similar to another multivariate procedure called Factor Analysis. They are often confused and many scientists do not understand the difference between the two methods or what types of analyses they are each best suited.

Below is a graph shows the number of variables explains the dependent variable:



This curve shows the % increase in the explained variance by increasing the number of factors or variables. Here we can see more than 95% of the variance is explained by 25 principal components or variables.

So here we can reduce our model dimensions from 41 variables to 25 variables with the focus of optimization of model.



We will mention the model output post using the Principal Component Analysis and check whether there is any significant change in the result is noticed in a combined table while evaluation of model.

4.6 Hyper Parameters Tunings on PCA results

Model hyperparameters are set by the data scientist ahead of training and control implementation aspects of the model. The weights learned during training of a linear regression model are parameters while the number of trees in a random forest is a model hyperparameter because this is set by the data scientist. Hyperparameters can be thought of as model settings. These settings need to be tuned for each problem because the best model hyperparameters for one particular dataset will not be the best across all datasets. The process of hyperparameter tuning (also called hyperparameter optimization) means finding the combination of hyperparameter values for a machine learning model that performs the best - as measured on a validation dataset - for a problem.

Here we have used two hyper parameters tuning techniques

- Random Search CV
- Grid Search CV

1. **Random Search CV**: This algorithm set up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources.

2. **Grid Search CV**: This algorithm set up a grid of hyperparameter values and for each combination, train a model and score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient.

   Check results post hyper parameter tuning on post PCA models in the model valuation section.

Chapter 5

Conclusion

5.1 Model Evaluation

The main concept of looking at what is called residuals or difference between our predictions f(x[I,]) and actual outcomes y[i].

In general, most data scientists use two methods to evaluate the performance of the model:

I. **RMSE** (Root Mean Square Error): is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}}$$

II. **R Squared(R^2):** is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In other words, we can say it explains as to how much of the variance of the target variable is explained.

III. We have shown both train and test data results, the main reason behind showing both the results is to check whether our data is overfitted or not.

Below table shows the model results before applying PCA:

Model Name	**RMSE**		**R Squared**	
	Train	**Test**	**Train**	**Test**
Decision Tree	0.54	0.54	0.31	0.26
Random Forest model	0.24	0.49	0.86	0.39
Linear Regression	0.49	0.53	0.44	0.30
Gradient Boosting	0.40	0.46	0.63	0.45

Below table shows the model results after applying PCA:

Model Name	**RMSE**		**R Squared**	
	Train	**Test**	**Train**	**Test**
Decision Tree (PCA)	0.57	0.57	0.23	0.19
Random Forest (PCA)	0.24	0.54	0.86	0.27
Linear Regression (PCA)	0.49	0.52	0.43	0.31
Gradient Boosting (PCA)	0.32	0.52	0.76	0.33

Below table shows results post using hyper parameter tuning techniques:

Model Name	**Random Search CV**		**Grid Search CV**	
	RMSE (Test)	**R Squared (Test)**	**RMSE (Test)**	**R Squared (Test)**
Random Forest (PCA)	0.51	0.34	0.51	0.34
Gradient Boosting (PCA)	0.51	0.35	0.51	0.34

Above table shows the results after tuning the parameters of our two best suited models i.e. Random Forest and Gradient Boosting. For tuning the parameters, we have used Random Search CV and Grid Search CV under which we have given the range of n_estimators, depth and CV folds.

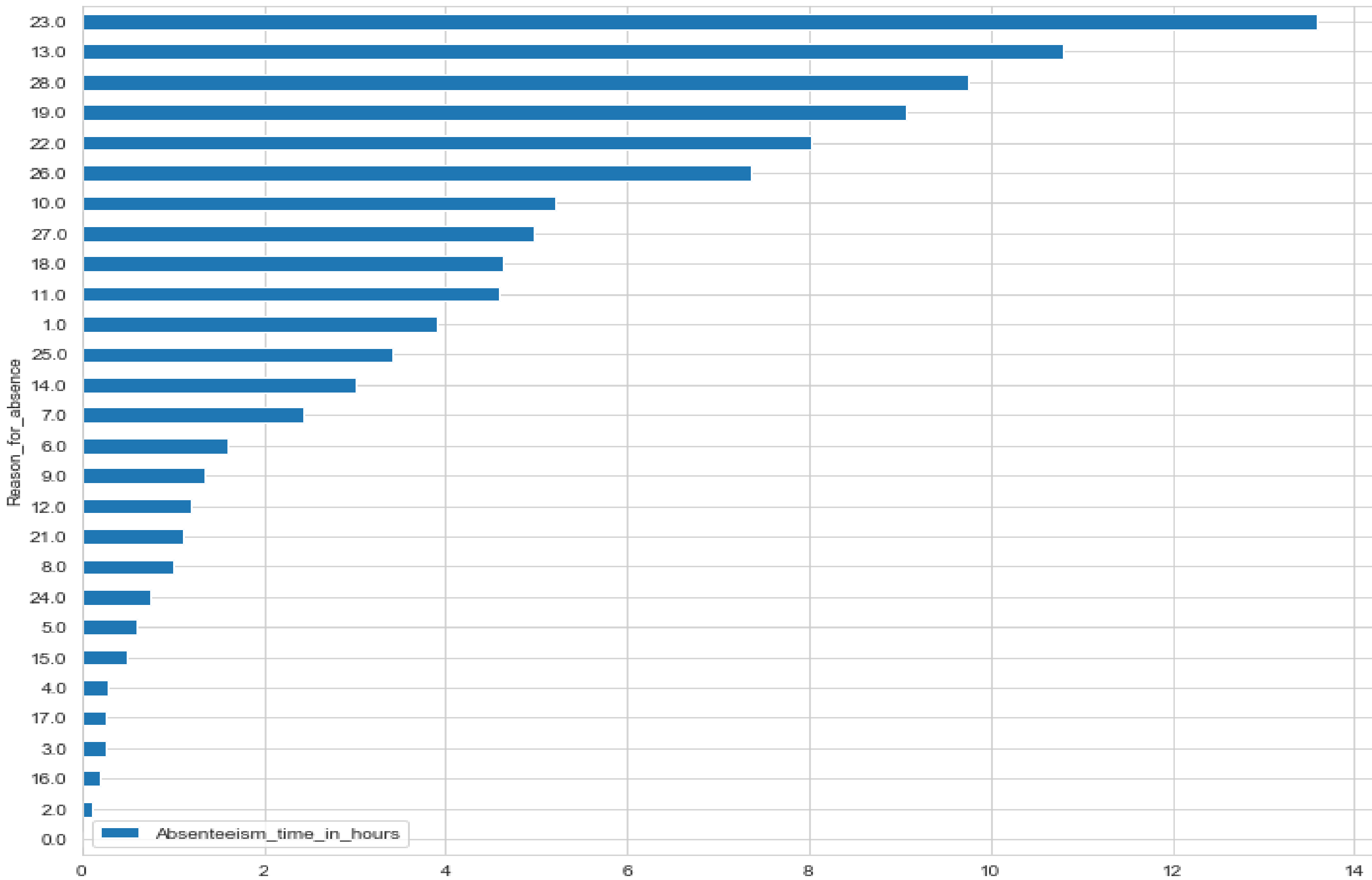
5. 2 Model Selection

On the basis RMSE and R Squared results a good model should have least RMSE and max R Squared value. So, from above tables we can see:

- Gradient Boosting Regressor model shows best result among all before applying the PCA.
- After applying PCA also gradient boosting model shows best results compared to rest of three.
- And also, after tuning the parameters of both random forest and gradient boost models, we get the gradient boosting shows better results compared to random forest after both parameter tuning techniques.
- So finally, we can say that Gradient Boosting Regressor is the best method to solve our problem with highest explained variance of the target variables and lowest error chances.

5. 3 What changes company should bring to reduce the number of absenteeism?

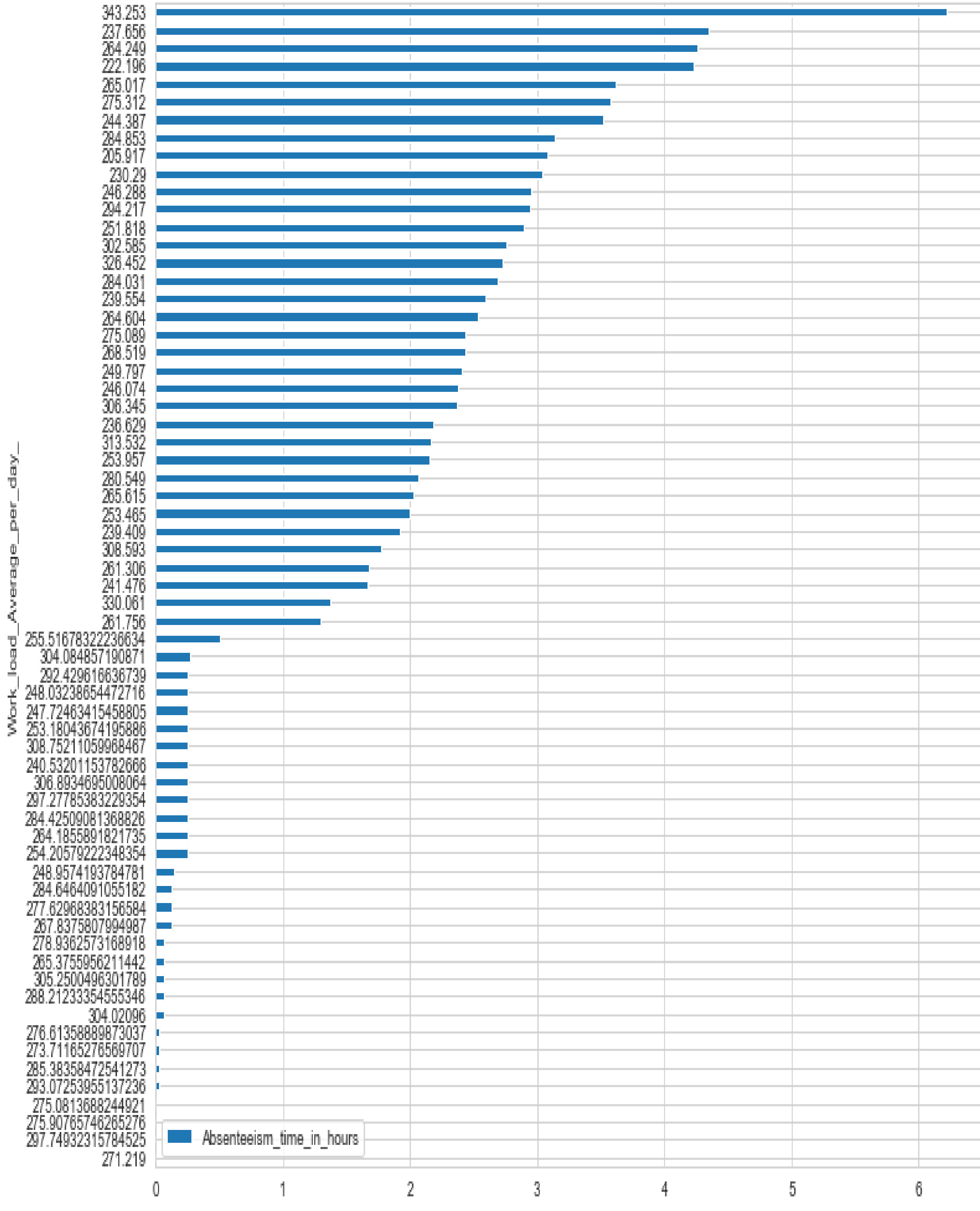
As we see the below graph tells about the relationship of reason of absence with the absenteeism hours:



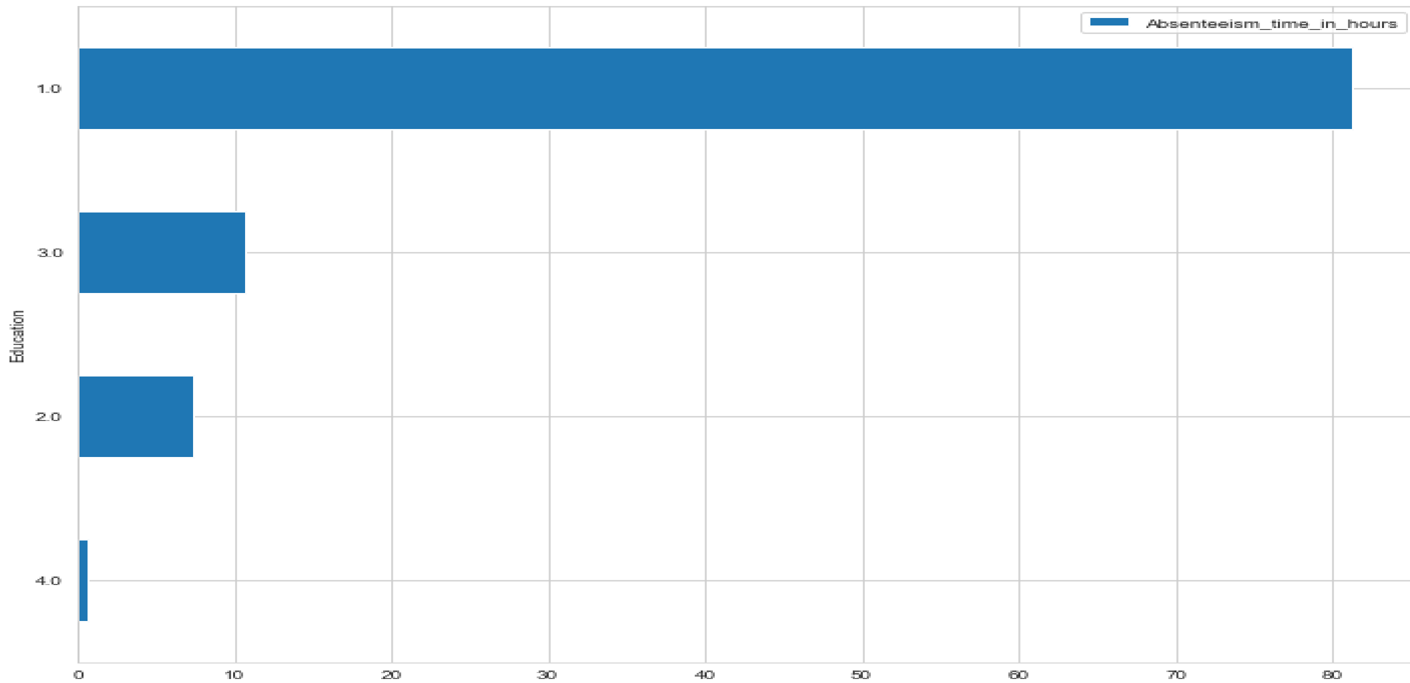
From the above graph we get to about top 5 reasons reported for absenteeism of employees:

- Category-13: Diseases of the musculoskeletal system and connective tissue - 12.79 % of total time.
- Category-23: medical consultation - 11.22 % of total time.
- Category-19: Injury, poisoning and certain other consequences of external causes - 10.63 % of total time.
- Category 28: dental consultation - 8.54 % 0f total time.
- Category 26: unjustified absence - 7.66 % of total time.

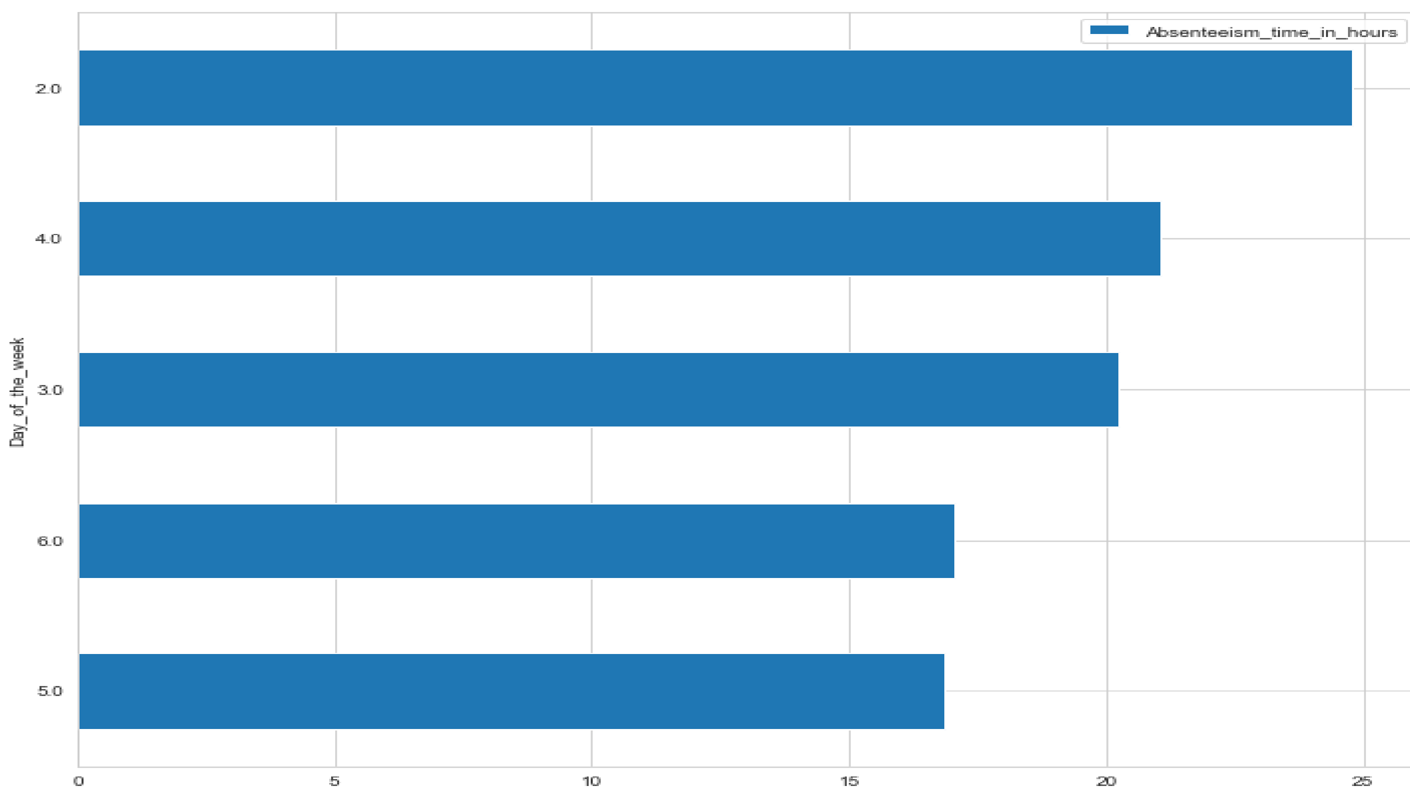
Below is another graph showing the relationship between count of work load with the hours of absenteeism of employees: highest work load employees has highest hours of absenteeism



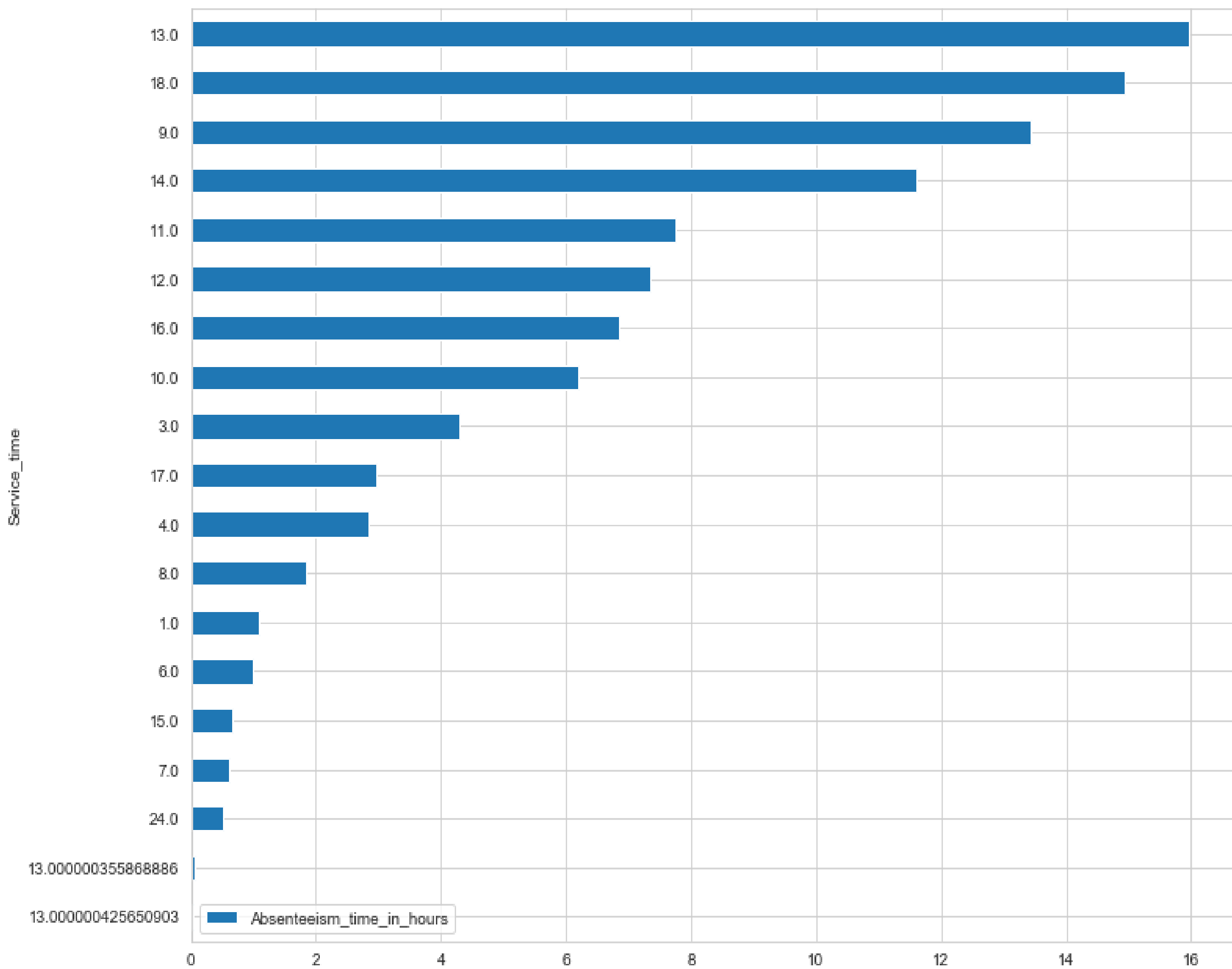
Below graph shows the relationship between education of the employees with hours of absenteeism: Here we can see that max number of employees have high school education only.



Below graphs shows the relationship between day of week with absenteeism hours: Maximum number of employees absent on Monday.



Below graph shows the relationship between service tenure of employees and absenteeism hours: long tenured employees are more often in the list of absenteeism



Points on which company needs to focus:

- Changes to be made is more college graduates to be hired in place of high school degree holders.
- Employees with highest work load have the highest number of hours absenteeism, so appropriate amount of work load should be assigned to each employee.
- Maximum number of times the reasons which are noted for absenteeism is Musculoskeletal system disease and blood donation
- One possible reason for the high incidence of musculoskeletal disease is repetitive movement strain due to high workload.
- Blood donation is the second highest reported reason for absenteeism, for this company can arrange blood donation camps in the office premises, so that employees can donate blood their itself.
- Companies needs to make changes in the roles of the employees who have been a long time in the company with the same roles.

5. 3 How much losses every month can we project in 2011 if same trend of absenteeism continues?

As we don't have the data of 2011 years so we can use the current data assuming that the same trend gets continue we can predict the looses of every month of 2011 by following method:

Assuming that workload average per day is the targeted work per employee for a particular day and the absenteeism hours are the days employees will not present in 2011 we can calculate loss of work by the following formula:

$$\text{Loss in work} = \left(\frac{\text{Workload Per Day}}{24}\right) * (\text{Absenteeism in Hours})$$

Below is the projected loss in work on monthly basis for the year 2011 if the same trend continues:

Out[213]:

	Total Absenteeism time in a month (hrs)	Work loss per month
Month_of_absence		
1.0	171.685945	2252.982436
2.0	279.399772	3167.922989
3.0	443.695125	5204.045042
4.0	239.910496	2732.008531
5.0	259.744293	2648.590871
6.0	240.571077	2711.681699
7.0	370.688157	3914.799769
8.0	237.163987	2332.985366
9.0	186.356411	2113.592214
10.0	281.000014	3155.595034
11.0	245.693580	2920.234946
12.0	199.424518	2154.100979

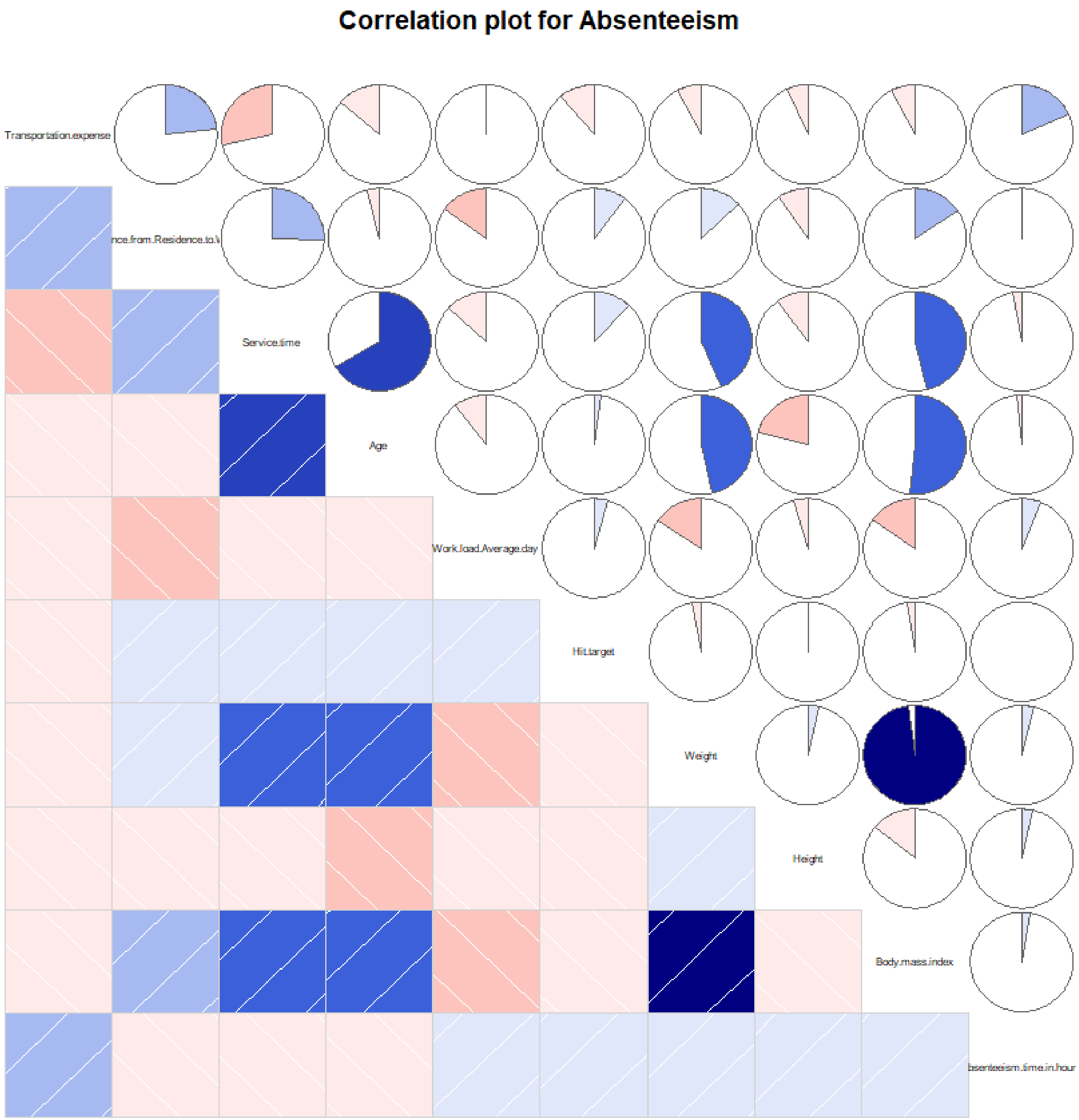
End of Report

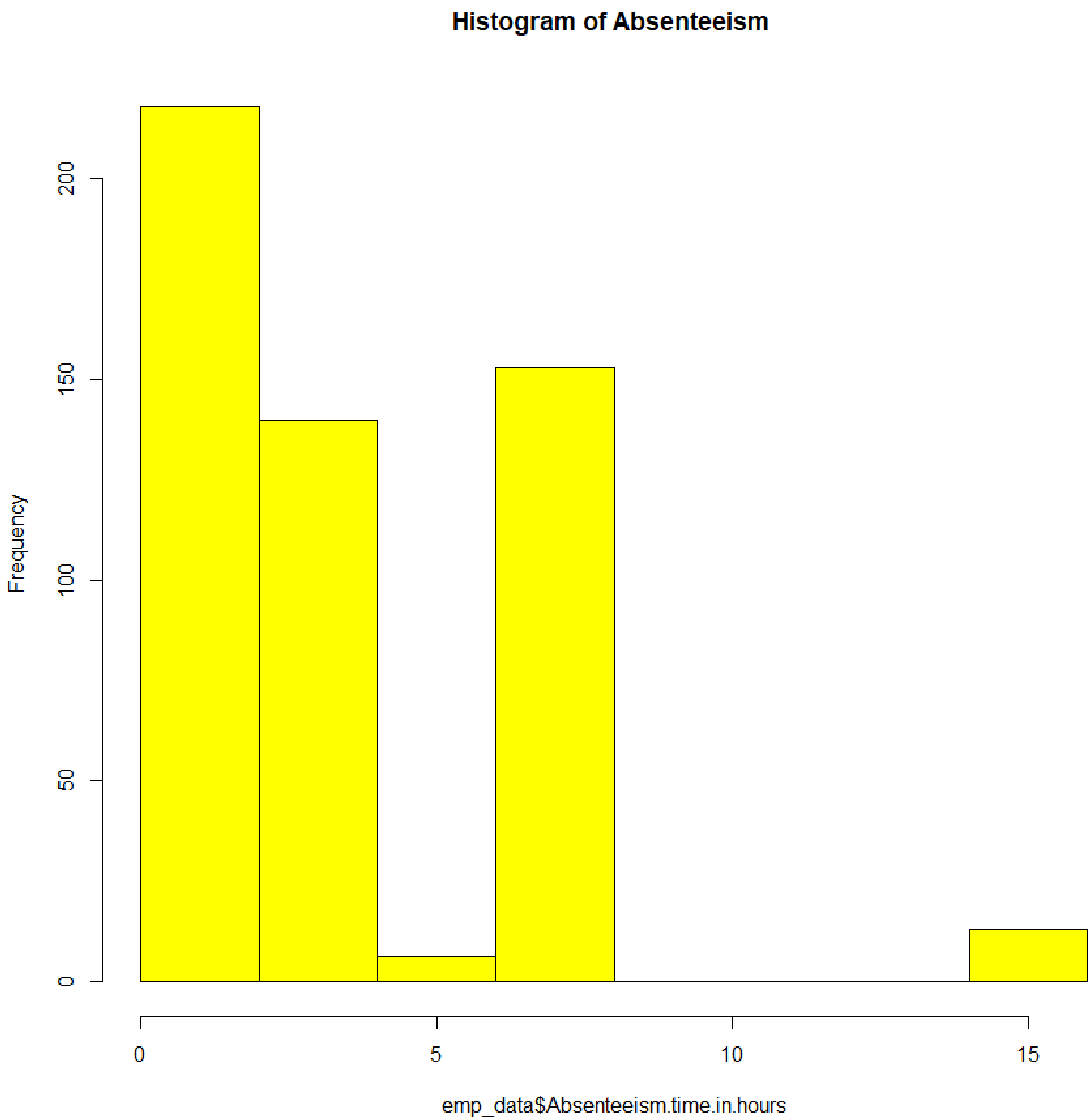
References

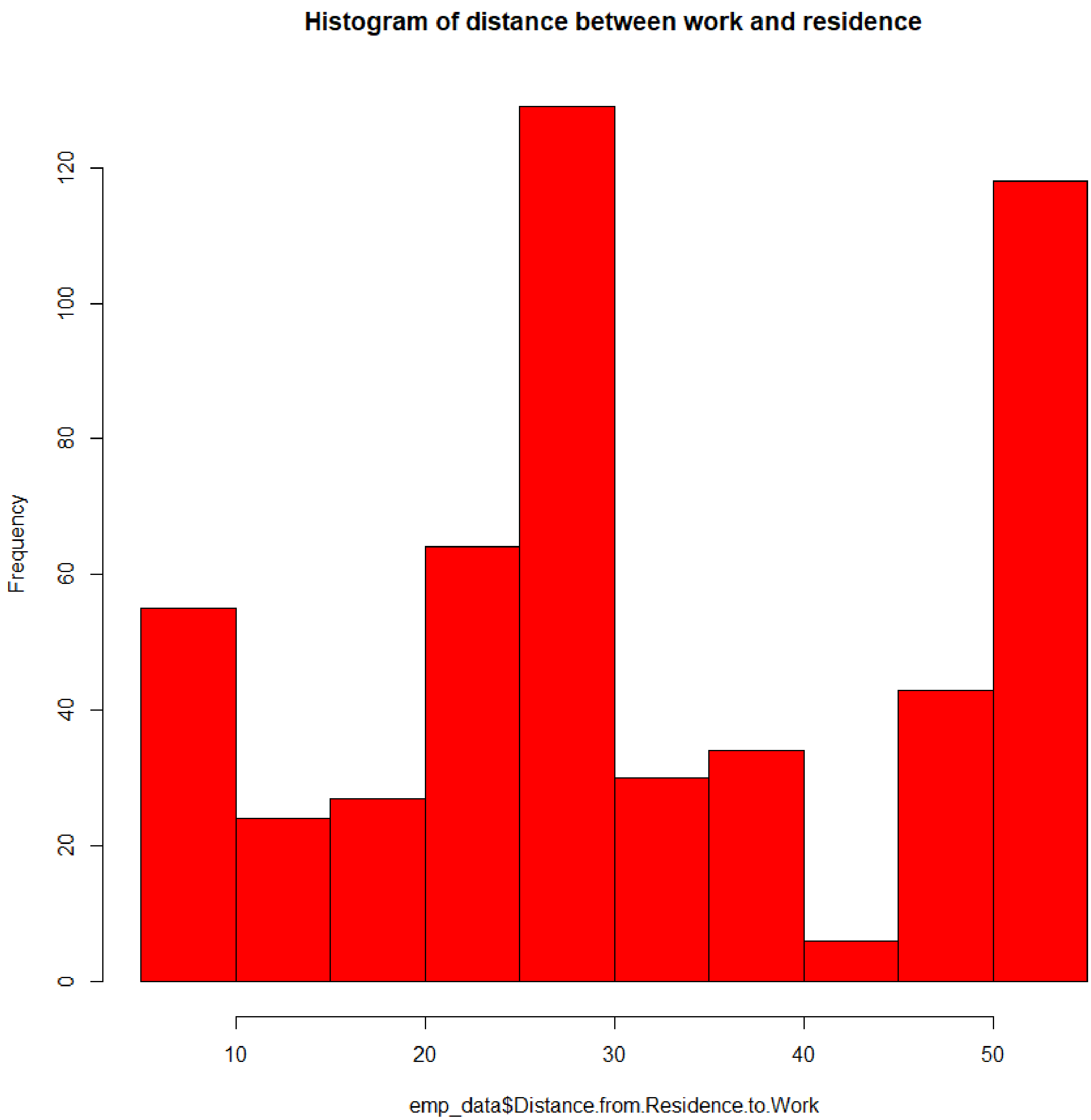
1. For Data Cleaning and Model Development - https://edwisor.com/career-data-scientist
2. For other code related queries - https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
3. For Visualization – https://www.udemy.com/python-for-data-science-and-machine-learning-bootcamp/
4. https://towardsdatascience.com/
5. https://stackoverflow.com/

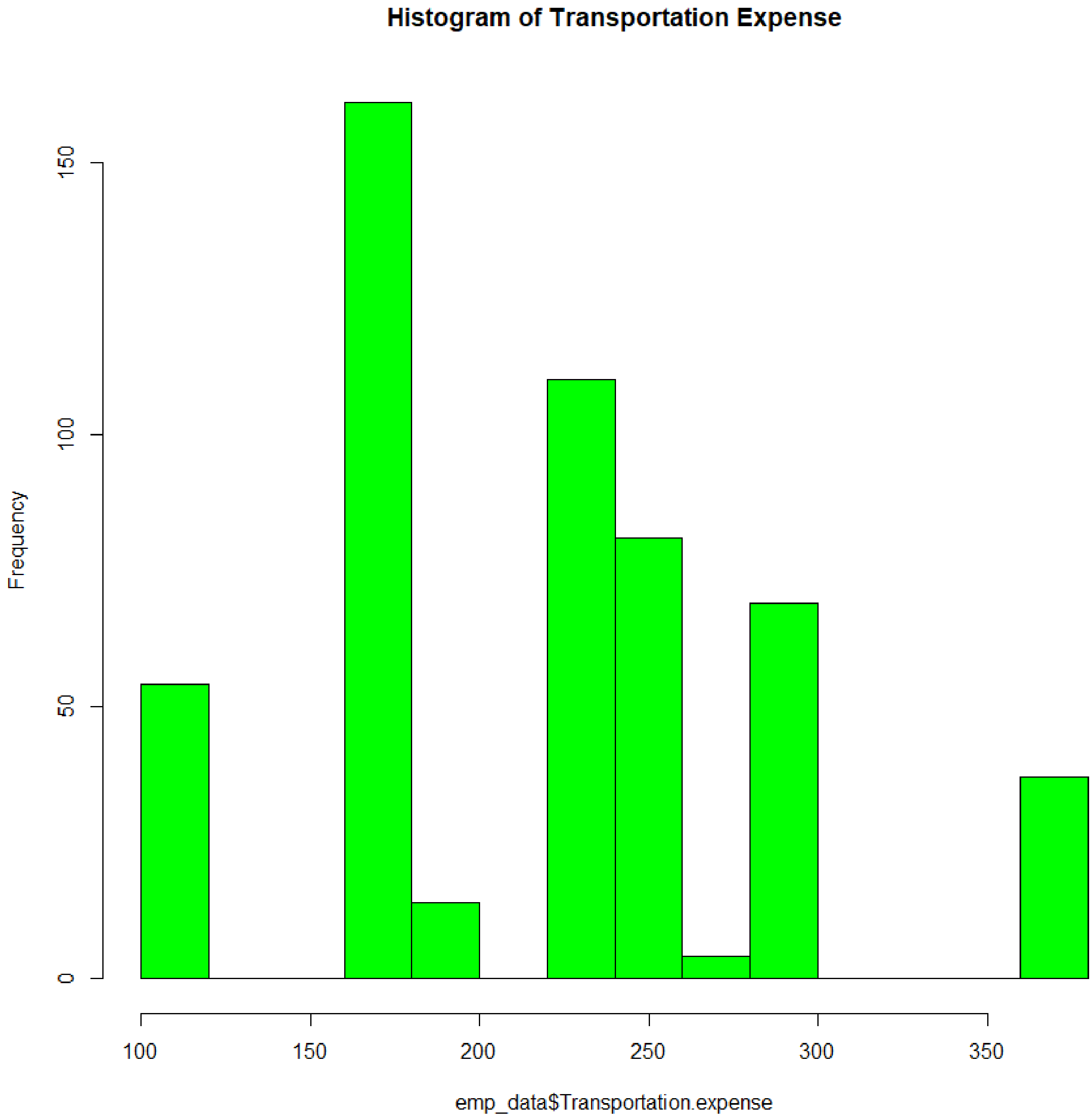
Appendix

R figures:




Histogram of Absenteeism
Frequency
0
50
100
150
200
0
5
10
15
emp_data$Absenteeism.time.in.hours


Histogram of distance between work and residence
Frequency
0
20
40
60
80
100
120
10
20
30
40
50
emp_data$Distance.from.Residence.to.Work



R code:

```
# Clearing the environment
rm(list=ls(all=T))


# Setting working directory


setwd("C:/Users/PRASHANT/Desktop/edwsior project new")
```

```r
getwd()


# Loading libraries

library(ggplot2)

library(corrgram)

library(DMwR)

library(caret)

library(randomForest)

library(unbalanced)

library(dummies)

library(e1071)

library(Information)

library(MASS)

library(rpart)

library(gbm)

library(ROSE)

library(xlsx)

library(DataCombine)

library(rpart)





x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071",
"Information",

     "MASS", "rpart", "gbm", "ROSE", 'xlsx', 'DataCombine', 'rpart')

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

rm(x)


## Reading the data

emp_data = read.csv('Absenteeism_at_work_Project_Cs.csv')
```

```
# Work.load.Average.day variable values have commas which need to be removed.

emp_data$Work.load.Average.day = gsub(',','',emp_data$Work.load.Average.day)



#--------------------------------------------------Exploratory Data Analysis-----------------------------------------------------
# Shape of the data
dim(emp_data)
# Viewing data
# View(df)
# Structure of the data
str(emp_data)
# Variable namesof the data
colnames(emp_data)

#from the data summary we can see variable ID, which is not useful variable
#in modeling further, so here removing variable ID from dataset.

emp_data= subset(emp_data,select=-c(ï..ID))

# From the above EDA and problem statement categorising data in 2 category "continuous" and "catagorical"
cont = c('Distance.from.Residence.to.Work', 'Service.time', 'Age',
         'Work.load.Average.day.', 'Transportation.expense',
         'Hit.target', 'Weight', 'Height',
         'Body.mass.index', 'Absenteeism.time.in.hours')

cat = c('Reason.for.absence','Month.of.absence','Day.of.the.week',
         'Seasons','Disciplinary.failure', 'Education', 'Social.drinker',
         'Social.smoker', 'Son', 'Pet')
```

```
#as we know that month variable can contain maximum 12 values, so here replace 0 with NA-
emp_data$Month.of.absence[emp_data$Month.of.absence %in% 0]= NA


#Dividing Work.load.Average.day by 1000 (As per discussion on the support forum)
emp_data$Work.load.Average.day= as.numeric(emp_data$Work.load.Average.day)/1000


View(emp_data)


###changing the required data type:

emp_data$Distance.from.Residence.to.Work = as.numeric(emp_data$Distance.from.Residence.to.Work)
emp_data$Service.time = as.numeric(emp_data$Service.time)
emp_data$Age = as.numeric(emp_data$Age)
emp_data$Work.load.Average.day. = as.numeric(emp_data$Work.load.Average.day.)
emp_data$Transportation.expense= as.numeric(emp_data$Transportation.expense)
emp_data$Hit.target = as.numeric(emp_data$Hit.target)
emp_data$Weight = as.numeric(emp_data$Weight)
emp_data$Height = as.numeric(emp_data$Height)
emp_data$Body.mass.index = as.numeric(emp_data$Body.mass.index)
emp_data$Absenteeism.time.in.hours = as.numeric(emp_data$Absenteeism.time.in.hours)



emp_data$Reason.for.absence = as.factor(emp_data$Reason.for.absence)
emp_data$Reason.for.absence = as.factor(emp_data$Reason.for.absence)
emp_data$Month.of.absence = as.factor(emp_data$Month.of.absence)
```

```
emp_data$Day.of.the.week = as.factor(emp_data$Day.of.the.week )

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

emp_data$Disciplinary.failure = as.factor(emp_data$Disciplinary.failure)

emp_data$Education = as.factor(emp_data$Education)

emp_data$Social.drinker = as.factor(emp_data$Social.drinker)

emp_data$Social.smoker= as.factor(emp_data$Social.smoker)
emp_data$Son = as.factor(emp_data$Son)
emp_data$Pet = as.factor(emp_data$Pet)



str(emp_data)






#===========================Data Pre-processing==================================================#

#--------------------------Missing Values Analysis----------------------------------------------#


###remove the rows which contains NA values in our target variable which is Absenteeism in hours
emp_data = emp_data[(!emp_data$Absenteeism.time.in.hours %in% NA),]
```

```
#Creating dataframe with missing values present in each variable

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

missing_val$Columns = row.names(missing_val)

names(missing_val)[1] =  "Missing_percentage"


#Calculating percentage missing value

missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(emp_data)) * 100


# Sorting missing_val in Descending order

missing_val = missing_val[order(-missing_val$Missing_percentage),]

row.names(missing_val) = NULL


# Reordering columns

missing_val = missing_val[,c(2,1)]


# Saving output result into csv file

write.csv(missing_val, "Missing_perc_R.csv", row.names = F)


# # Plot

ggplot(data = missing_val[1:18,], aes(x=reorder(Columns, -Missing_percentage),y = Missing_percentage))+

geom_bar(stat = "identity",fill = "grey")+xlab("Variables")+

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



#Number of missing values


as.data.frame(colSums(is.na(emp_data)))


#Reason.for.absence                    3

#Month.of.absence                      4
```

```
#Day.of.the.week                    0
#Seasons                            0
#Transportation.expense             6
#Distance.from.Residence.to.Work    3
#Service.time                       3
#Age                                2
#Work.load.Average.day              8
#Hit.target                         6
#Disciplinary.failure               5
#Education                          10
#Son                                6
#Social.drinker                     3
#Social.smoker                      4
#Pet                                2
#Weight                             1
#Height                             14
#Body.mass.index                    29
#Absenteeism.time.in.hours          0



# Actual Value = 27
# Mean = 26.68
# Median = 25
# KNN = 27

##checking actual value of body mass cell number 23
emp_data$Body.mass.index[23]


##converting the known value as na
```

```
emp_data[23,19] = NA

emp_data[23,19]



#Mean Method
emp_data$Body.mass.index[is.na(emp_data$Body.mass.index)] = mean(emp_data$Body.mass.index, na.rm = T)

#Median Method
emp_data$Body.mass.index[is.na(emp_data$Body.mass.index)] = median(emp_data$Body.mass.index, na.rm = T)

# kNN Imputation methond
##df = knnImputation(df, k = 3)
emp_data = kNN(emp_data, k=3)



#for categorical variable we will use mode imputation
mode=function(v){
  uniqv=unique(v)
  uniqv[which.max(tabulate(match(v,uniqv)))]
}

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

##for continous variables we will use KNN imputation
# KNN Imputation
```

```
install.packages("bitops")
install.packages("DMwR")

emp_data= knnImputation(emp_data,k=3)



# Checking for missing value
sum(is.na(emp_data))




#-------------------------------------Outlier Analysis-------------------------------------#
# BoxPlots - Distribution and Outlier Check

# Boxplot for continuous variables
numeric_index = sapply(emp_data,is.numeric) #selecting only numeric

numeric_data = emp_data[,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(emp_data))+
        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 of Absenteeism for",cnames[i])))
}

## Plotting plots together
gridExtra::grid.arrange(gn1,gn5,gn2,ncol=3)
gridExtra::grid.arrange(gn6,gn7,ncol=2)
gridExtra::grid.arrange(gn8,gn9,gn10,ncol=3)



#Remove outliers from dataset-
for(i in cnames){
  print(i)
  outlier= emp_data[,i][emp_data[,i] %in% boxplot.stats(emp_data[,i])$out]
  print(length(outlier))
  emp_data = emp_data[which(!emp_data[,i] %in% outlier),]
}



#Replace outliers with NA and impute using KNN method-
for(i in cnames){
  print(i)
  outlier= emp_data[,i][emp_data[,i] %in% boxplot.stats(emp_data[,i])$out]
  print(length(outlier))
  emp_data[,i][emp_data[,i] %in% outlier]=NA
}
```

```
 sum(is.na(emp_data))


 #KNN-
 emp_data= knnImputation(emp_data,k=3)
 sum(is.na(emp_data))



#----------------------------------Feature Selection----------------------------------------#

#Correlation Analysis for continuous variables-
library(corrgram)    #Library for correlation plot

corrgram(data[,cnames],order=FALSE,upper.panel = panel.pie,
        text.panel = panel.txt,font.labels =1,
        main="Correlation plot for Absenteeism")


#Correlated variable= weight

#Anova Test for categorical variable-

for(i in cat_cnames){
  print(i)
  Anova_result= summary(aov(formula = Absenteeism.time.in.hours~data[,i],data))
  print(Anova_result)
}

#redudant categorical variables- Pet,Social.smoker,Education,Seasons,Month.of.absence
```

```
#Dimensionity Reduction

data= subset(data,select=-c(Weight,Pet,Social.smoker,Education,Seasons,Month.of.absence))

dim(data)


#--------------------------------Feature Scaling--------------------------------#

#update the continuous variable dataframe after dimension reduction-

cnames1= c('Distance.from.Residence.to.Work', 'Service.time', 'Age',

           'Work.load.Average.day.', 'Transportation.expense',

           'Hit.target','Height', 'Body.mass.index')



#summary of data to check min and max values of numeric variables-

summary(emp_data)



#Normality check

hist(emp_data$Absenteeism.time.in.hours,col="Yellow",main="Histogram of Absenteeism ")

hist(emp_data$Distance.from.Residence.to.Work,col="Red",main="Histogram of distance between work and
residence")

hist(emp_data$Transportation.expense,col="Green",main="Histogram of Transportation Expense")


#From all above histogram plot we can say that data is not uniformaly distributed,

#So best method for scaling will be normalization-


#Skewness of numeric variables-

library(propagate)


for(i in cnames){

  skew = skewness(emp_data[,i])

  print(i)

  print(skew)
```

```
}

#log transform

emp_data$Absenteeism.time.in.hours = log1p(emp_data$Absenteeism.time.in.hours)

#Normalization-

for(i in cnames1){

  print(i)

  emp_data[,i]= (emp_data[,i]-min(emp_data[,i]))/(max(emp_data[,i]-min(emp_data[,i])))

  print(emp_data[,i])

}

##due to some error need to normalized without loop some of the variables

emp_data$Transportation.expense = (emp_data$Transportation.expense - min
(emp_data$Transportation.expense))/(max(emp_data$Transportation.expense -
min(emp_data$Transportation.expense)))

emp_data$Work.load.Average.day = (emp_data$Work.load.Average.day - min
(emp_data$Work.load.Average.day))/(max(emp_data$Work.load.Average.day -
min(emp_data$Work.load.Average.day)))

emp_data$Hit.target = (emp_data$Hit.target - min (emp_data$Hit.target))/(max(emp_data$Hit.target -
min(emp_data$Hit.target)))

emp_data$Height = (emp_data$Height - min (emp_data$Height))/(max(emp_data$Height - min(emp_data$Height)))

emp_data$Body.mass.index = (emp_data$Body.mass.index - min
(emp_data$Body.mass.index))/(max(emp_data$Body.mass.index - min(emp_data$Body.mass.index)))

#Summary of data after all preprocessing-

summary(emp_data)
```

```
write.csv(emp_data,"Absenteeism_Pre_processed_Data.csv",row.names=FALSE)




#------------------------------------------Model Development-------------------------------------------#

#Clean the Environment-
library(DataCombine)
rmExcept("emp_data")

#Data Copy for refrance-
df=emp_data
emp_data=df



#categorical variables-
cat_cnames = c('Reason.for.absence','Day.of.the.week','Disciplinary.failure',
            'Social.drinker','Son')

#create dummies for categorical variables-
library(dummies)
data= dummy.data.frame(emp_data,cat_cnames)
dim(emp_data)
colnames(emp_data)

#Divide the data into train and test-

set.seed(123)
```

```
train_index= sample(1:nrow(data),0.8*nrow(data))

train= data[train_index,]

test= data[-train_index,]


#--------------------------Decision Tree for Regression-----------------------------------------#


#Model devlopment for train data-

library(rpart)    #Library for regression model

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

DT_model


#Prediction for test data-

DT_test=predict(DT_model,test[-55])


#Error metrics to calculate the performance of model-

rmse= function(y,y1){

  sqrt(mean(abs(y-y1)^2))

}

#RMSE calculation for test data-

rmse(test[,55],DT_test)

##RMSE Result : 0.5197763



#r-square calculation-

#function for r-square-

rsquare=function(y,y1){

  cor(y,y1)^2

}


#r-square calculation for test data-

rsquare(test[,55],DT_test)
```

```
##Rsquare result : 0.4006544


#---------------------------Random Forest for Regression----------------------------------------#


library(randomForest)  #Library for randomforest machine learning algorithm
library(inTrees)       #Library for intree transformation
RF_model= randomForest(Absenteeism.time.in.hours~.,train,ntree=500,method="anova")


#transform ranfomforest model into treelist-
treelist= RF2List(RF_model)


#Extract rules-
rules= extractRules(treelist,train[-55])
rules[1:5,]
#covert rules into redable format-
readable_rules= presentRules(rules,colnames(train))
readable_rules[1:5,]
#Get Rule metrics-
rule_metrics= getRuleMetric(rules,train[-55],train$Absenteeism.time.in.hours)
rule_metrics= presentRules(rule_metrics,colnames(train))
rule_metrics[1:10,]
summary(rule_metrics)


#Check model performance on test data-
RF_test= predict(RF_model,test[-55])


#RMSE calculation for test data-
rmse(test[,55],RF_test)


#RMSE_test=  0.4949839
```

```
#r-square calculation for test data-
rsquare(test[,55],RF_test)
#r-square=0.450954

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


#Linear regression model-
lr_model= lm(Absenteeism.time.in.hours~.,train)
summary(lr_model)

#check model performance on test data-
lr_test= predict(lr_model,test[-55])

#RMSE calculation for test data-
rmse(test[,55],lr_test)

#RMSE_test=0.507

#r-square calculation for test data-
rsquare(test[,55],lr_test)

#r-square=0.446
```