**Project Report**

**Employee Absenteeism**

Stuti Ghildiyal

12 July 2019

# Contents

1. [**Introduction**](#_bookmark0)
   1. [Problem Statement](#_bookmark1) 3
   2. [Data](#_bookmark2) 3
   3. Exploratory Data Analysis 5
2. [**Methodology**](#_bookmark3)
   1. [Preprocessing](#_bookmark4) 7
      1. Missing Value Analysis 7
      2. Data Conversion 9
      3. Univariate and Bivariate Analysis 10
      4. Outlier Analysis 15
      5. Feature Selection 19

2.1.5.1 Correlation Analysis 20

2.1.5.2 Anova Test 20

* + 1. Feature Scaling 21
    2. Principal Component Analysis 25
  1. Modeling 26

2.2.1 Decision Tree 26

2.2.2 Random Forest 27

2.2.3 Linear Regression 27

1. **Conclusion** 
   1. Model Evaluation 28
   2. Model Selection 28
   3. Answers of asked questions 28

**Appendix A -R - Code 33**

**Appendix B -Python Code 48**

**Chapter 1**

**Introduct****ion**

* 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?

Employee Absenteeism is referred to herein as failure of employees to report for work when they are scheduled to work. It is one of the major problems faced by most of the employers of today. Absenteeism of employees from work leads to back logs, piling of work and thus work delay. The main objective of this project is to resolve above two issues.

* 1. **Data**

There are 21 variables in our data out of which 20 are independent variables and 1 (Absenteeism time in hours) is dependent variable. Since our target variable is continuous in nature, this is a regression problem.

**Variables Information:**

**1.** Individual identification (ID)

**2.** Reason for absence (ICD) -

Absences attested by the **International Code of Diseases** (ICD) stratified into 21 categories (I to XXI) as follows:

**I**. Certain infectious and parasitic diseases

**II**. Neoplasms

**III.** Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism

**IV**. Endocrine, nutritional and metabolic diseases

**V**. Mental and behavioral disorders

**VI**. Diseases of the nervous system

**VII**. Diseases of the eye and adnexa

**VIII**. Diseases of the ear and mastoid process

**IX**. Diseases of the circulatory system

**X**. Diseases of the respiratory system

**XI**. Diseases of the digestive system

**XII**. Diseases of the skin and subcutaneous tissue

**XIII**. Diseases of the musculoskeletal system and connective tissue

**XIV**. Diseases of the genitourinary system

**XV**. Pregnancy, childbirth and the puerperium

**XVI**. Certain conditions originating in the perinatal period

**XVII**. Congenital malformations, deformations and chromosomal abnormalities

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

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

**XX.** External causes of morbidity and mortality

**XXI**. Factors influencing health status and contact with health services

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

**3.** Month of absence

**4.** Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))

**5.** Seasons (summer (1), autumn (2), winter (3), spring (4))

**6.** Transportation expense

**7.** Distance from Residence to Work (kilometers)

**8.** Service time

**9.** Age

**10.** Work load Average/day

**11.** Hit target

**12.** Disciplinary failure (yes=1; no=0)

**13.** Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))

**14.** Son (number of children)

**15.** Social drinker (yes=1; no=0)

**16.** Social smoker (yes=1; no=0)

**17.** Pet (number of pet)

**18.** Weight

**19.** Height

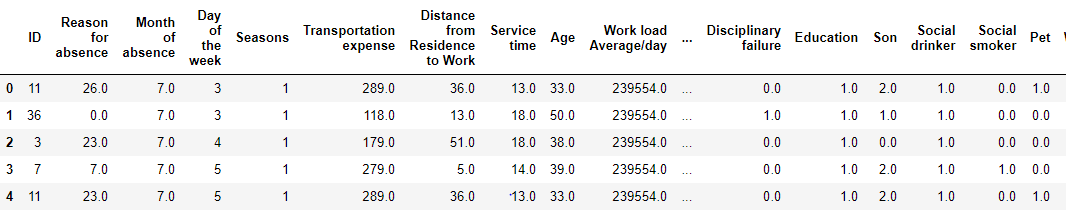
**20.** Body mass index

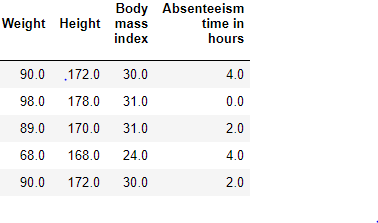
**21**. Absenteeism time in hours (target)

* 1. **Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics. In the given data set there are 21 variables and data types of all variables are either float64 or int64. There are 740 observations and 21 columns in our data set. Missing value is also present in our data.

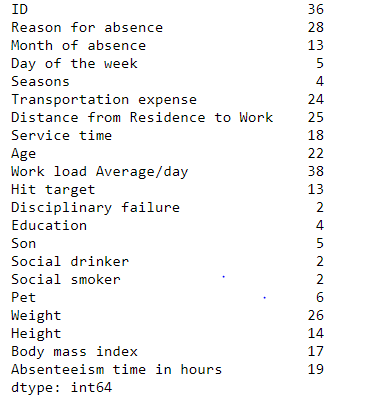
**Sample Data:**

****



**Fig 1.3 – First five rows of data**

**Unique Count:** Below figure shows the unique count of all the variables present in the data.



By seeing the data itself, we can find that we have few categorical variables also like: ID, day of week, Social smoker etc. We can classify our independent variables as below:

|  |  |
| --- | --- |
| Categorical independent variables | Continuous independent variables |
| ID | Transportation expense |
| Reason for absence | Distance from Residence to Work (kilometers) |
| Month of absence | Service time |
| Day of the week | Age |
| Seasons | Work load Average/day |
| Disciplinary failure (yes=1; no=0) | Hit target |
| Education | Weight |
| Social drinker (yes=1; no=0) | Height |
| Social smoker (yes=1; no=0) | Body mass index |
| Son |  |
| Pet |  |

**Chapter 2**

**Methodology**

Before feeding the data to the model we need to clean the data and convert it to a proper format. It is the most crucial part of data science project.

**2.1 Pre Processing**

**2.2.1Missing Value Analysis**

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data. If a column has more than 30% of data as missing value, then either we ignore the entire column or we ignore those observations. In the given data the maximum percentage of missing value is 4.189% for body mass index column. So we will compute missing value for all the columns.



As we know we have data for 36 unique ID which means we have data for 36 employees, so, we can impute few of the variables through ID.

**Transportation expense**

'ID' column has been used to impute missing value for 'Transportation expense'.

**Distance from Residence to Work**

'ID' column has been used to impute missing value for 'Distance from Residence to Work'.

**Service time**

'ID' column has been used to impute missing value for ‘Service time’.

**Age**

'ID' column has been used to impute missing value for ‘Age’.

**Education**

'ID' column has been used to impute missing value for ‘Education’.

**Son**

'ID' column has been used to impute missing value for ‘Son’.

**Social drinker**

'ID' column has been used to impute missing value for ‘Social drinker’.

**Social smoker**

'ID' column has been used to impute missing value for ‘Social smoker’.

**Pet**

'ID' column has been used to impute missing value for ‘Pet’.

**Weight**

'ID' column has been used to impute missing value for ‘Weight’.

**Height**

'ID' column has been used to impute missing value for ‘Height’.

**Body mass index**

'ID' column has been used to impute missing value for ‘Body mass index’.

For other variables, we can impute missing variables through mean, median or KNN imputation. We need to select the best method.

After applying all the method find, KNN imputation is the best method to fill missing values with KNN = 3.

**R-Code**

|  |
| --- |
| library(DMwR)  emp\_df = knnImputation(emp\_df,k=3) |

**Python –code**

|  |
| --- |
| #Apply KNN imputation algorithm  emp\_df = pd.DataFrame(KNN(k = 3).fit\_transform(emp\_df), columns = emp\_df.columns) |

After applying KNN imputation, check the presence of missing values again.

**2.2.2 Data conversion**

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.

To start this process, we will first do data conversion.

The data types can be converted using below codes:

**R-code:**

|  |
| --- |
| emp\_df$ID = as.factor(as.character(emp\_df$ID))  emp\_df$Day.of.the.week = as.factor(as.character(emp\_df$Day.of.the.week))  emp\_df$Education = as.factor(as.character(emp\_df$Education))  emp\_df$Social.drinker = as.factor(as.character(emp\_df$Social.drinker))  emp\_df$Social.smoker = as.factor(as.character(emp\_df$Social.smoker))  emp\_df$Reason.for.absence = as.factor(as.character(emp\_df$Reason.for.absence))  emp\_df$Seasons = as.factor(as.character(emp\_df$Seasons))  emp\_df$Month.of.absence = as.factor(as.character(emp\_df$Month.of.absence))  emp\_df$Disciplinary.failure = as.factor(as.character(emp\_df$Disciplinary.failure))  emp\_df$Son = as.factor(as.character(emp\_df$Son))  emp\_df$Pet = as.factor(as.character(emp\_df$Pet)) |

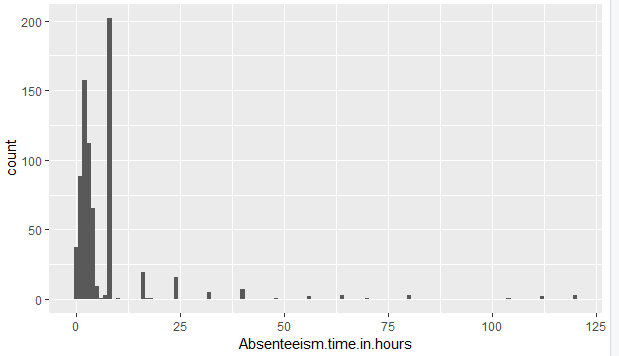
**Python Code:**

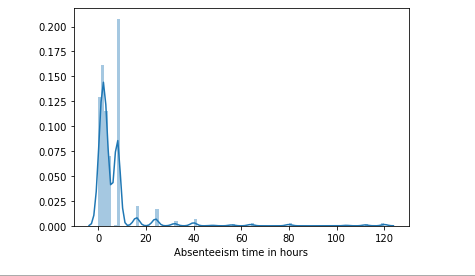
|  |
| --- |
| emp\_df['ID']=emp\_df['ID'].astype('category')  emp\_df['Reason for absence']=emp\_df['Reason for absence'].astype('category')  emp\_df['Month of absence']=emp\_df['Month of absence'].astype('category')  emp\_df['Day of th eweek']=emp\_df['Day of the week'].astype('category')  emp\_df['Seasons']=emp\_df['Seasons'].astype('category')  emp\_df['Disciplinary failure']=emp\_df['Disciplinary failure'].astype('category')  emp\_df['Education']=emp\_df['Education'].astype('category')  emp\_df['Son']=emp\_df['Son'].astype('category')  emp\_df['Social drinker']=emp\_df['Social drinker'].astype('category')  emp\_df['Social smoker']=emp\_df['Social smoker'].astype('category')  emp\_df['Pet']=emp\_df['Pet'].astype('category') |

**2.1.3 Univariate and Bivariate analysis:**

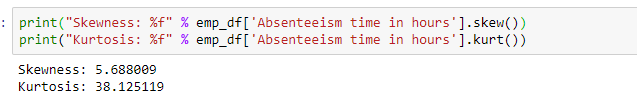
The analysis of univariate data is the simplest form of analysis since the information deals with only one quantity that changes. Here we will try to describe and find pattern in our target value. It seems that 'Absenteeism time in hours' is not normally distributed and most of the absenteeism hours lies between 0-20 hrs. and few member having absent hours more than 100

**Univariate:**



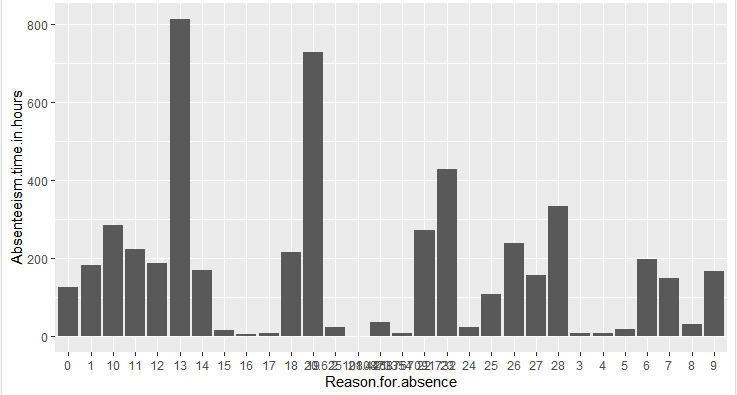


**Skewness** is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point. **Kurtosis** is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution.



**Bivariant**

1. Let’s see how reason of absence is affecting absence time



From the above fig we see that, main reason for abseentism is 13(Diseases of the

Musculoskeletal system and connective tissue) which means that the employee might

have to do heavy work which results in connective tissue or bone issues. So, there should be

a bone checkup and first aid maintained for the employers.

Another reason is 19(Injury, poisoning and certain other consequences of external

Causes) which means same that because of work the employee are suffering and there are

not enough medication or aid maintained to heal them. So, the employee has to take

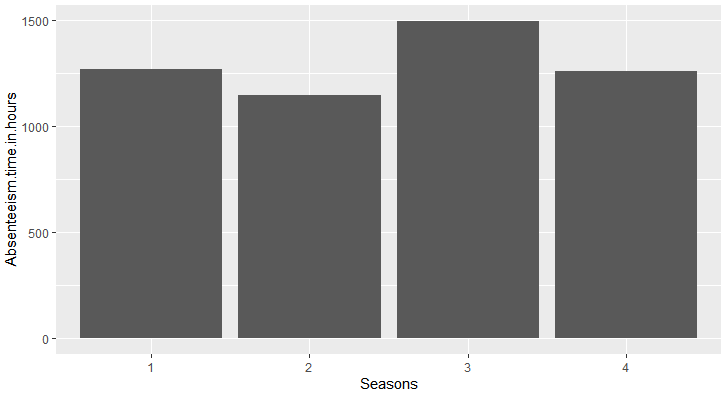
external treatment which results in absenteeism.

So, the workload should be reduced and also a team should be there to look after the health if the

Employee gets any injury and should be treated.

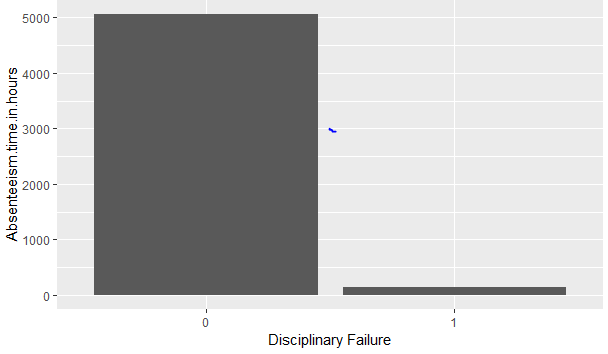
2. **Seasons VS Absenteeism in time**

There is no much difference in absentees’ time based on season.



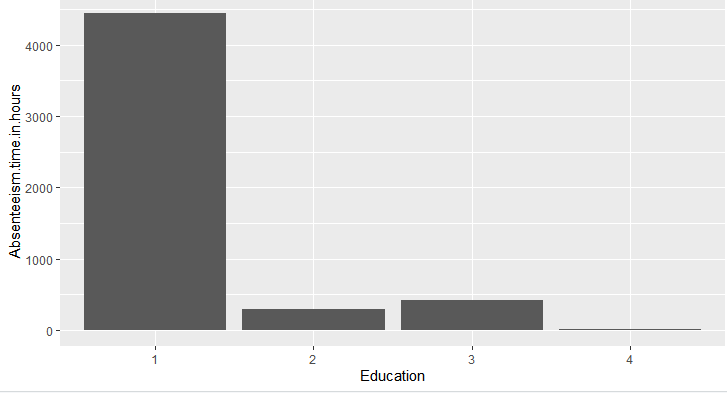
1. **Disciplinary failure Vs Absenteeism time in hours**

The plot clearly shows that the data is more biased when Disciplinary failure is ‘NO’



4) **Education Vs Absenteeism time in hours**

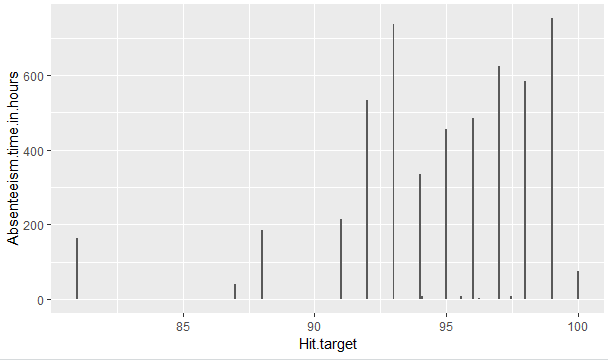
The plot clearly shows that the data is more biased towards one category. Employee having education degree till high school only shows trends of taking more leaves as compared to other employee. This shows may be people till high school degree are working here as a part time worker.



1. **Hit target VS Absenteeism time in hours**

We see that, more number of absenteeism comes under 90-100, which means that once an employee hits their target, they remain absent.

So, there should be a system assigning work once the employee finishes its assigned work. So, that he doesn’t have to be absent thinking that he has no target to complete.

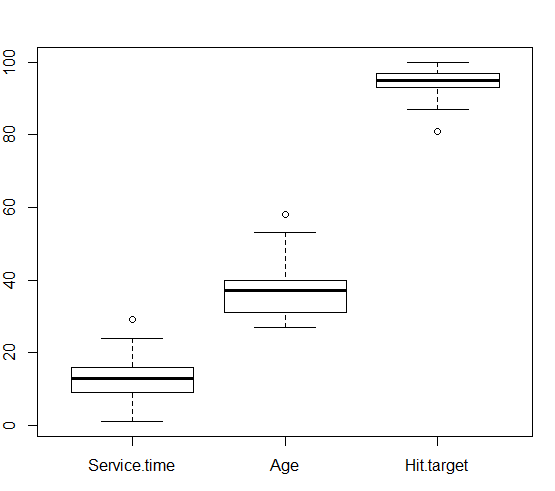


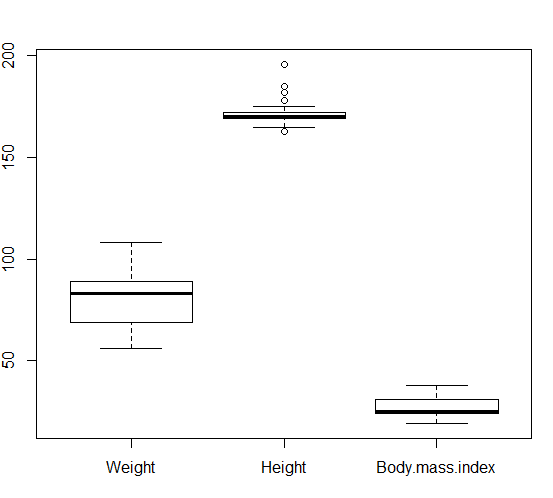
**2.1.4 Outlier Analysis**

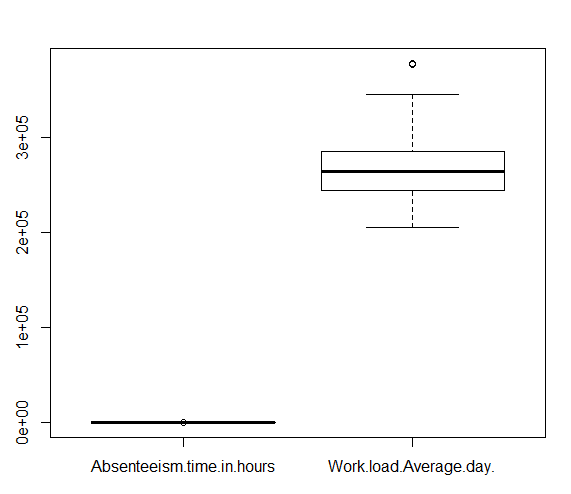
Outliers are the unwanted abnormal values that may get generated due to rough handling of data or an out of the range value in which most of the point lies.

For this dataset I am using Boxplot method to get if the outliers are present.

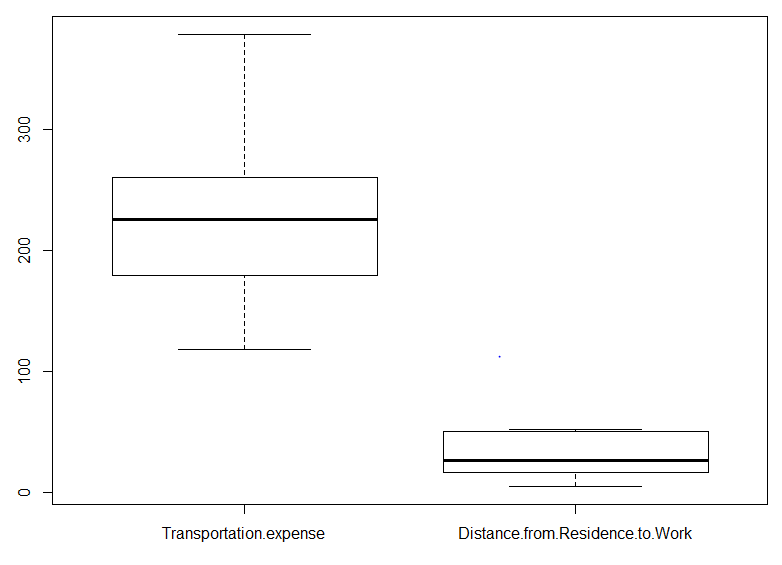


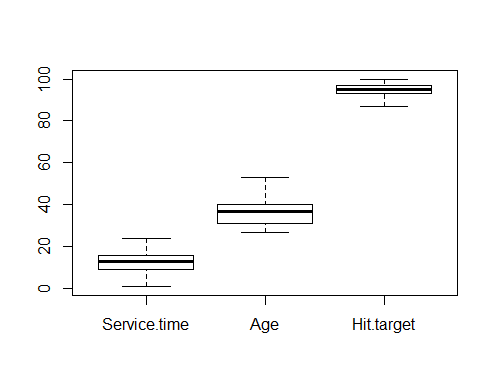


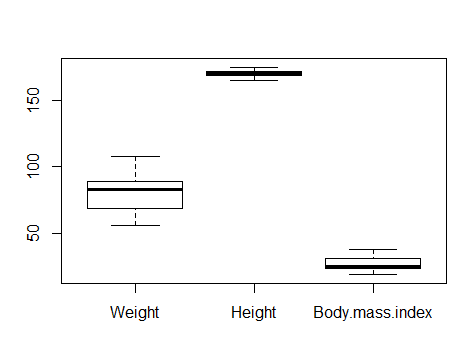


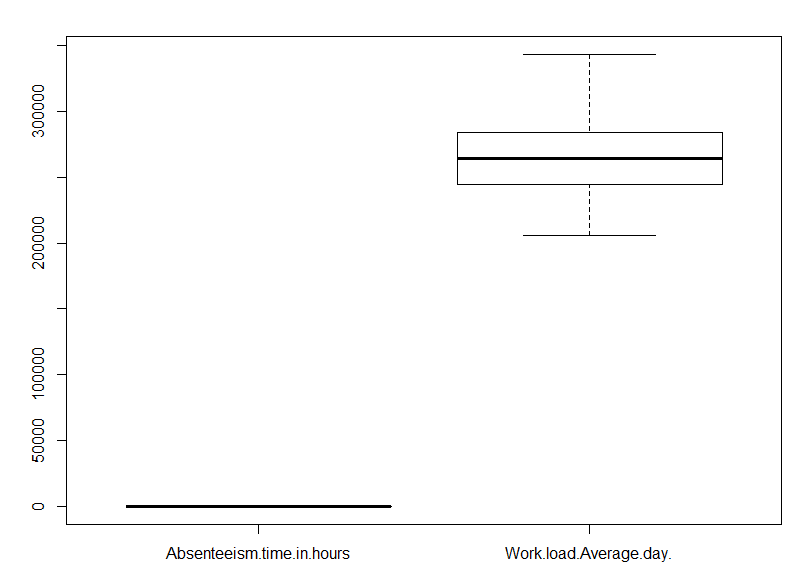


From the box plot almost all the variables except “Distance from residence to work”, “Weight” and “Body mass index” consists of outliers. In order to remove the outliers first we need to replace each of the outlier values with NA and then impute using already selected most suitable method of imputation KNN.







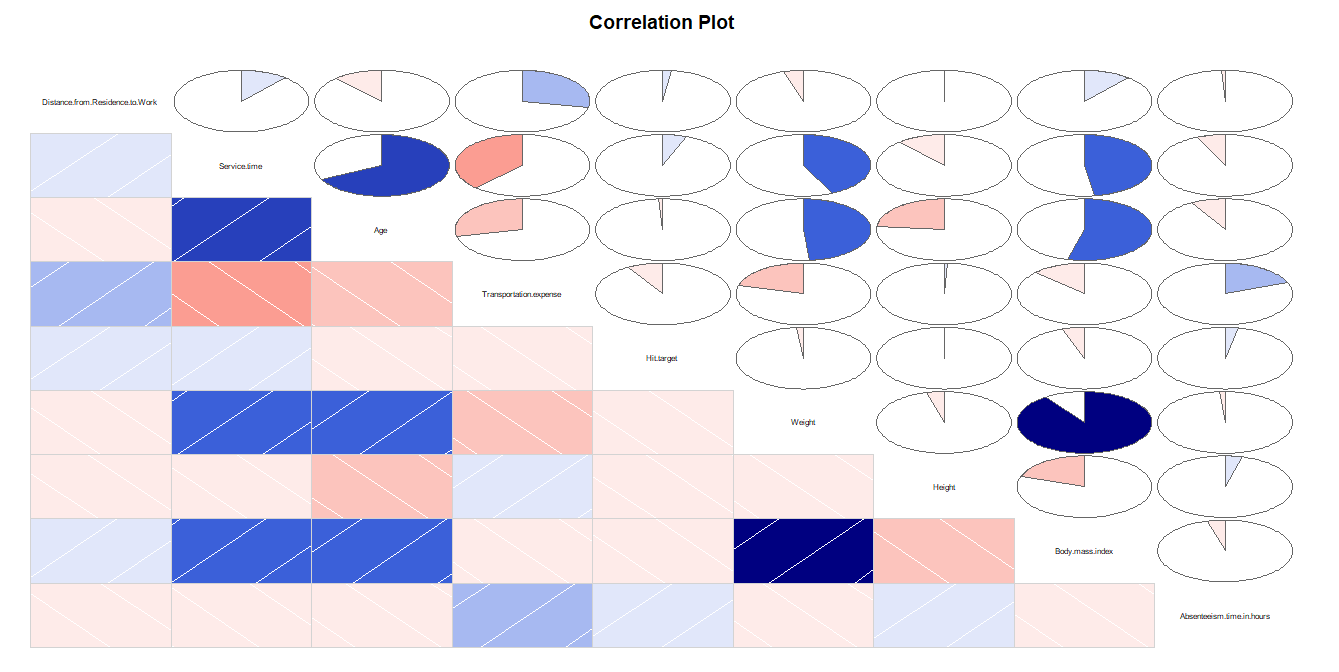


**2.1.5 Feature Selection**

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem. Selecting subset of relevant columns for the model construction is known as Feature Selection. We cannot use all the features because some features may be carrying the same information or irrelevant information which can increase overhead. To reduce overhead we adopt feature selection technique to extract meaningful features out of data. This in turn helps us to avoid the problem of multi collinearity. In this project we have selected Correlation Analysis for numerical variable and ANOVA (Analysis of variance) for categorical variable.

**2.1.5.1Correlation Analysis**

Correlation plot is used to find out if there is any multicollinearity between variables. The highly collinear variables are dropped and then the model is executed.



From the above figure we can see that weight is highly correlated to body mass index. Also,

Service time is slightly correlated to age. So, Body mass index and service time are removed from our data.

**2.1.5.2 Anova Test**

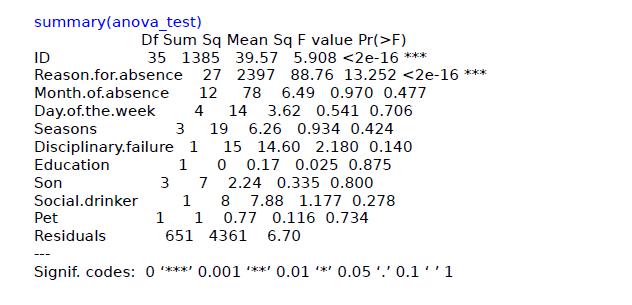
After performing anova test, we can see that reason for absence and ID is less than 0.05 so we

Consider these variables but ID does not explain much about the target variable so we

Consider only reason for absence and remove all other categorical variables.

Here one point must be considered. We have to do our calculation according to month in which employees are making absence.

Thus I am not deleting the Month of absence variable despite having p value greater than 0.05.



**2.2.6Feature Scaling**

Feature scaling is a method used to standardize the range of variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

The dataset which we have contains data of variables having quite different range of values. For instance, age lies between 30 to 60 whereas Work load Average/day lies between 280000 to 300000. These range values must be treated otherwise they would make the model bias and inefficient.

In order to scale the features there are usually two methods: -

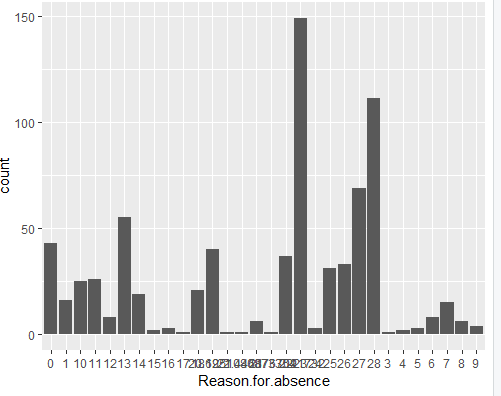
1. Standardization

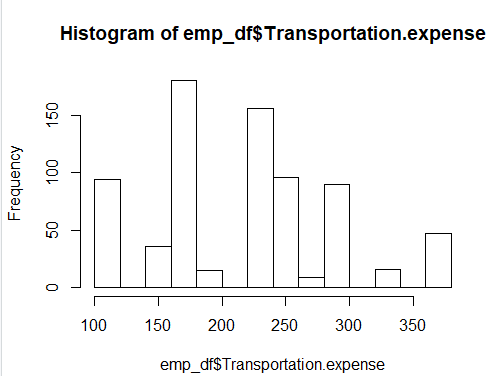
2. Normalization

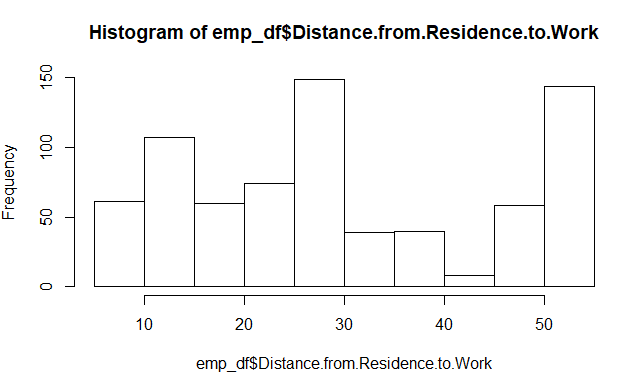
The method of standardization works only on the data which is normally distributed.

The Normalization method can work for any kind of distribution either positive skewed data or be it negative skewed data.

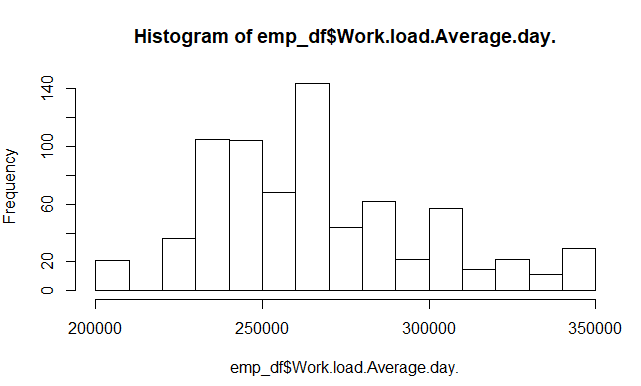
First we will check the distribution of our data

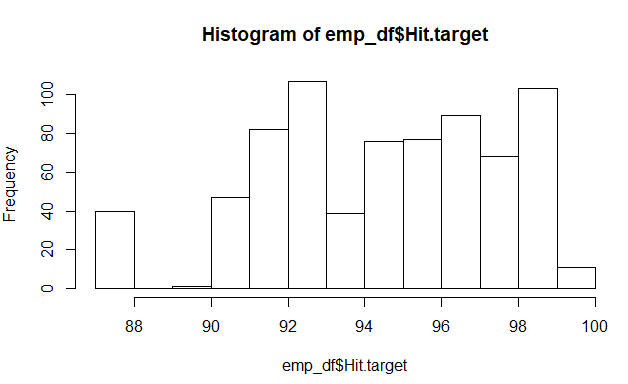


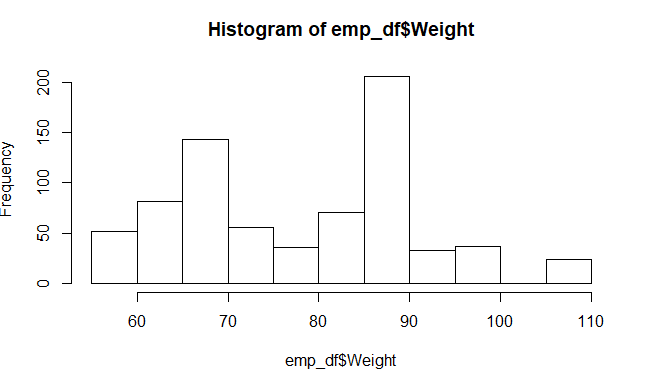
****

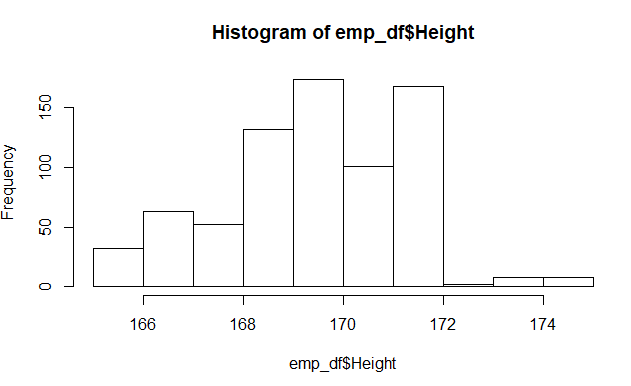
****











The plots are indicating that the data distribution is not normally distributed.

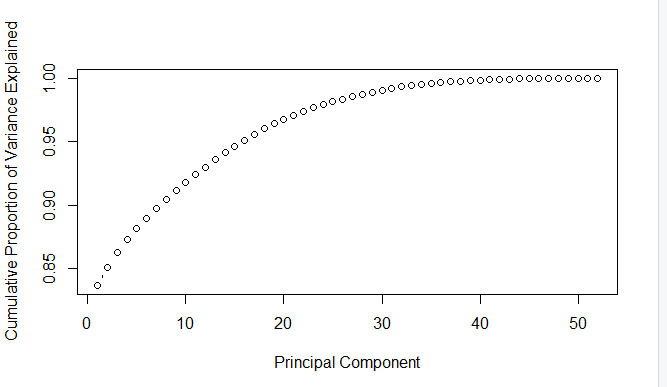
In this case we have to use Normalization to scale our features.

Formulae used for normalization is



**2.2.7Principal Component Analysis**

Principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible. With fewer variables, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher dimensional data. After creating dummy variable of categorical variables the shape of our data became 52 columns and 740 observations, this high number of columns leads to bad accuracy.



We have applied PCA algorithm on our data and from the above graph we have concluded that 20 variables out of 52 explains more than 95% of data. So we have selected only those 20 variables to feed our models.

**2.2 Model Selection**

After a thorough pre-processing we will be using some regression models on our processed data to predict the target variable. The target variable in our model is a continuous variable i.e., Absenteeism time in hours. Hence the models that we choose are Linear Regression, Decision Tree and Random Forest. The error metric chosen for the given problem statement is Root Mean Square Error (RMSE).

**2.2.1 Decision Tree**

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance of event outcomes, resource costs, and utility. 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. The RMSE value and R^2 value for our project in R and Python are –

|  |  |  |
| --- | --- | --- |
| **Decision Tree** | **R** | **PYTHON** |
| **RMSE** | 0.4098639 | 0.10912160465887401 |
| **R^2** | 0.9872817 | 0.9989825368670852 |

**2.2.2 Random Forest**

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations to build each decision tree. It means to build each decision tree on random forest we are not going to use the same data. The RMSE value and R^2 value for our project in R and Python are –

|  |  |  |
| --- | --- | --- |
| **Random Forest** | **R** | **PYTHON** |
| **RMSE** | 0.7057195 | 0.06047967073385911 |
| **R^2** | 0.9732212 | 0.9996874524992397 |

**2.2.3 Liner Regression**

Linear Regression is one of the statistical methods of prediction. It is applicable only on continuous data.

|  |  |  |
| --- | --- | --- |
| **Linear Regression** | **R** | **PYTHON** |
| **RMSE** | 0.007397825 | 2.578537524976396e-15 |
| **R^2** | 0.999995826 | 1.0 |

**Chapter 3**

In this chapter we are going to evaluate our models, select the best model for our dataset and try to get answers of the asked questions.

**3.1 Model Evaluation**

In the previous chapter we have seen the Root Mean Square Error (RMSE) and R-Squared Value of different models. Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE and higher value of R-Squared Value indicate better fit.

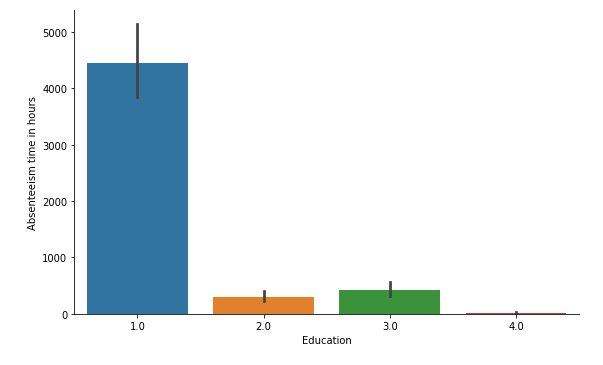
**3.2 Model Selection**

From the observation of all RMSE Value and R-Squared Value we have concluded that Random Forest Model has minimum value of RMSE and its R-Squared Value is also maximum.

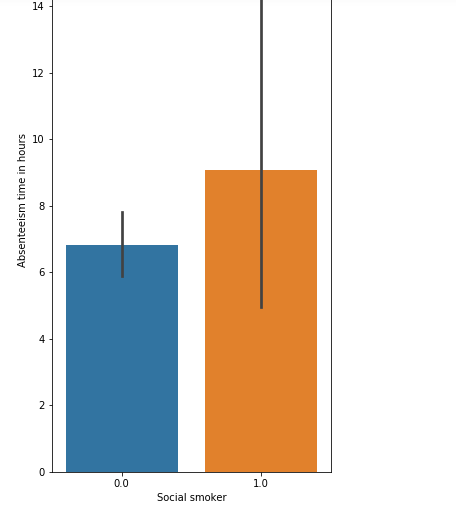
**3.3 Answers of asked questions**

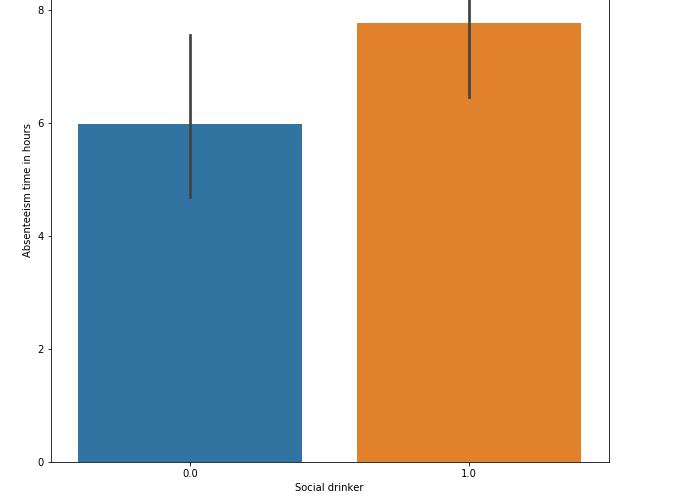
1. **The Changes which company should bring to reduce the number of absenteeism –**

It is observed that employee with low education have maximum absentee time. So, the company can either hire employees who have at least graduated from college.

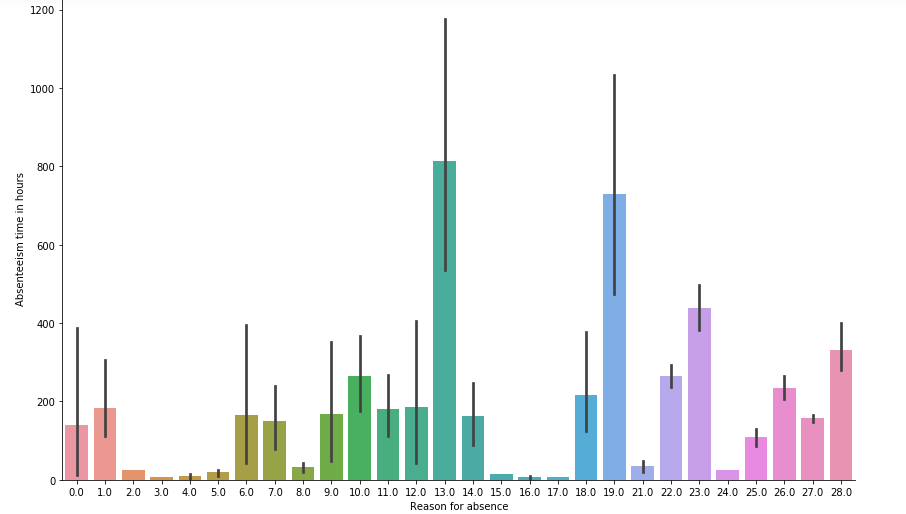
****

Employees who are social smoker have more absentee hour than who are not social smoker. Same goes with social drinkers.

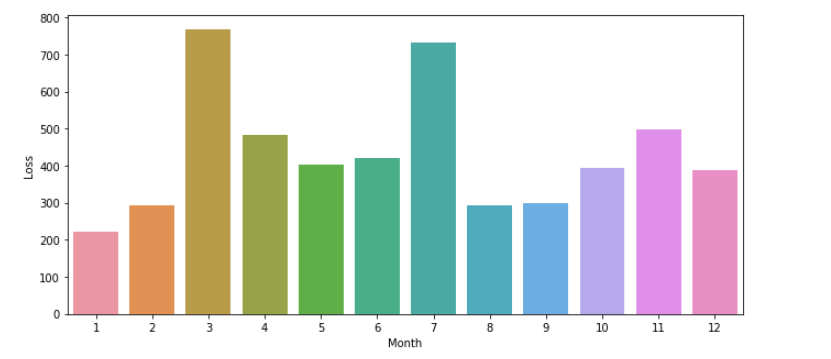
****

****

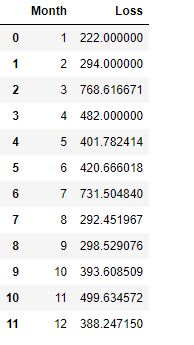
Most often Reason for absence are medical consultation and dental consultation, company should take care of it.

****

**Loss Prediction:**



From the above graph we can say that employees will take more leaves on month of march and July as compared to other months.



**Appendix – R code**

|  |
| --- |
| #Clear the environment  rm(list = ls())  #Set working Directory  setwd("C:/Users/HP/Desktop/edwisor/Project 1")  getwd()  #Load Libraries  x = c("ggplot2", "corrgram", "caret", "randomForest", "dummies","rpart.plot",  "rpart", 'sampling')  #install.packages(x)  lapply(x, require, character.only = TRUE)  rm(x)  #Load libraries  library("xlsx")  library(ggplot2)  library(gplots)  #Read the data  emp\_df = read.xlsx("Absenteeism\_at\_work\_Project.xls", sheetIndex = 1)  #####################################Exploratory Data Analysis##############################  #observe the class of data object  class(emp\_df)  #Observe the dimension of data  dim(emp\_df)  #get the names of variable  colnames(emp\_df)  #Check the structure of the dataset variables  str(emp\_df)  #Observe top 5 rows  head(emp\_df)  #no of unique values in each variables  apply(emp\_df, 2,function(x) length(table(x)))  # From the above EDA and problem statement categorising data in 2 category "continuous" and "catagorical"  continuous\_vars = 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')  categorical\_vars = c('ID','Reason.for.absence','Month.of.absence','Day.of.the.week',  'Seasons','Disciplinary.failure', 'Education', 'Social.drinker',  'Social.smoker', 'Son', 'Pet')  #########################################Pre Processing ###################################################  #Missing value Analysis  #Creating dataframe with missing values present in each variable  missing\_val = data.frame(apply(emp\_df,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$Missing\_percentage/nrow(emp\_df)) \* 100  # Sorting missing\_val in Descending order  missing\_val = missing\_val[order(-missing\_val$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()  #--> Since the calculated missing percentage is less than 5%  #--> Thus there is no need to drop any variable due to less number of data  #Now generate a missing value in the dataset and try various imputation methods on it  #make a backup for data object  data\_back = emp\_df  #emp\_df = data\_back  #Finding ID column values for which there are missing values in Transportation.expense  emp\_df$ID[is.na(emp\_df$Transportation.expense)]  # [1] 10 3 1 15 20 22 20  for (i in c(1,3,10,15,20,22)){  emp\_df$Transportation.expense[is.na(emp\_df$Transportation.expense) & emp\_df$ID==i] = mean(emp\_df$Transportation.expense[emp\_df$ID==i],na.rm = T)  }  as.data.frame(colSums(is.na(emp\_df)))  #Finding ID column values for which there are missing values in Distance.from.Residence.to.Work  emp\_df$ID[is.na(emp\_df$Distance.from.Residence.to.Work)]  # [1] 34 22 28  for (i in c(34,22,28)){  emp\_df$Distance.from.Residence.to.Work[is.na(emp\_df$Distance.from.Residence.to.Work) & emp\_df$ID==i] = mean(emp\_df$Distance.from.Residence.to.Work[emp\_df$ID==i],na.rm = T)  }  as.data.frame(colSums(is.na(emp\_df)))  #Finding ID column values for which there are missing values in Service.time  emp\_df$ID[is.na(emp\_df$Service.time)]  # [1] 28 34 34  for (i in c(34,28)){  emp\_df$Service.time[is.na(emp\_df$Service.time) & emp\_df$ID==i] = mean(emp\_df$Service.time[emp\_df$ID==i],na.rm = T)  }  as.data.frame(colSums(is.na(emp\_df)))  #Finding ID column values for which there are missing values in Age  emp\_df$ID[is.na(emp\_df$Age)]  # [1] 28 24 24  for (i in c(24,28)){  emp\_df$Age[is.na(emp\_df$Age) & emp\_df$ID==i] = mean(emp\_df$Age[emp\_df$ID==i],na.rm = T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Education  emp\_df$ID[is.na(emp\_df$Education)]  # [1] 11 10 34 34 14 34 34 34 10 24  for (i in c(10,11,14,24,34)){  emp\_df$Education[is.na(emp\_df$Education) & emp\_df$ID==i] = mean(emp\_df$Education[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Son  emp\_df$ID[is.na(emp\_df$Son)]  # [1] 20 14 34 34 27 1  for (i in c(1,14,20,27,34)){  emp\_df$Son[is.na(emp\_df$Son) & emp\_df$ID==i] = mean(emp\_df$Son[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Social.drinker  emp\_df$ID[is.na(emp\_df$Social.drinker)]  # [1] 10 14 17  for (i in c(10,14,17)){  emp\_df$Social.drinker[is.na(emp\_df$Social.drinker) & emp\_df$ID==i] = mean(emp\_df$Social.drinker[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Social.smoker  emp\_df$ID[is.na(emp\_df$Social.smoker)]  # [1] 34 1 11 15  for (i in c(34,1,11,15)){  emp\_df$Social.smoker[is.na(emp\_df$Social.smoker) & emp\_df$ID==i] = mean(emp\_df$Social.smoker[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Pet  emp\_df$ID[is.na(emp\_df$Pet)]  # [1] 1 13  for (i in c(1,13)){  emp\_df$Pet[is.na(emp\_df$Pet) & emp\_df$ID==i] = mean(emp\_df$Pet[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Weight  emp\_df$ID[is.na(emp\_df$Weight)]  # [1] 27  for (i in c(27)){  emp\_df$Weight[is.na(emp\_df$Weight) & emp\_df$ID==i] = mean(emp\_df$Weight[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Height  emp\_df$ID[is.na(emp\_df$Height)]  # [1] 20 10 28 34 34 27 10 11 5 22 13 24 32 28  for (i in c(20,10,28,34,27,11,5,22,13,24,32)){  emp\_df$Height[is.na(emp\_df$Height) & emp\_df$ID==i] = mean(emp\_df$Height[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  #ID column has been used to impute missing value for Body.mass.index  emp\_df$ID[is.na(emp\_df$Body.mass.index)]  # [1] 3 24 11 30 2 19 34 3 28 34 28 3 13 36 11 14 34 36 28 20 28 28 18 28 17 15 20 22 24 11 5  for (i in c(3,24,11,30,2,19,34,28,13,36,14,20,18,17,15,22,5)){  emp\_df$Body.mass.index[is.na(emp\_df$Body.mass.index) & emp\_df$ID==i] = mean(emp\_df$Body.mass.index[emp\_df$ID==i],na.rm=T)  }  as.data.frame(colSums(is.na(emp\_df)))  ## create a missing value in any variable  #emp\_df$Reason.for.absence[7] #22  #emp\_df$Reason.for.absence[7] = NA  ##Now apply mean method to impute this generated value and observe the result  #emp\_df$Reason.for.absence[is.na(emp\_df$Reason.for.absence)] = mean(emp\_df$Reason.for.absence, na.rm = T)# 19.18478  #emp\_df$Reason.for.absence[7]  #emp\_df$Reason.for.absence[7] = NA  ##Now apply median method to impute the missing value  #emp\_df$Reason.for.absence[is.na(emp\_df$Reason.for.absence)] = median(emp\_df$Reason.for.absence, na.rm = T)#23  #emp\_df$Reason.for.absence[7]  #emp\_df$Reason.for.absence[7] = NA  #now apply KNN method to impute  library(DMwR)  emp\_df = knnImputation(emp\_df,k=3) #22.21795  emp\_df$Reason.for.absence[7]  #emp\_df=data\_back  #emp\_df$Reason.for.absence[7] = NA  #emp\_df = knnImputation(emp\_df,k=5)  #emp\_df$Reason.for.absence[7] #22.34878  #After applying all the method find,KNN imputation is the best method to fill missing values wih KNN= 3  #Now load the data again and impute the missing value by KNN  #Check presence of missing values once to confirm  apply(emp\_df,2, function(x){sum(is.na(x))})  #NO missing value is found  ############################## Data conversion################################  emp\_df$ID = as.factor(as.character(emp\_df$ID))  emp\_df$Day.of.the.week = as.factor(as.character(emp\_df$Day.of.the.week))  emp\_df$Education = as.factor(as.character(emp\_df$Education))  emp\_df$Social.drinker = as.factor(as.character(emp\_df$Social.drinker))  emp\_df$Social.smoker = as.factor(as.character(emp\_df$Social.smoker))  emp\_df$Reason.for.absence = as.factor(as.character(emp\_df$Reason.for.absence))  emp\_df$Seasons = as.factor(as.character(emp\_df$Seasons))  emp\_df$Month.of.absence = as.factor(as.character(emp\_df$Month.of.absence))  emp\_df$Disciplinary.failure = as.factor(as.character(emp\_df$Disciplinary.failure))  emp\_df$Son = as.factor(as.character(emp\_df$Son))  emp\_df$Pet = as.factor(as.character(emp\_df$Pet))  #\_  #########################################Univarant and Bivarant ############################  #plotting boxplot of Reason.for.absence Vs. Absenteeism.time.in.hours  ggplot(emp\_df,aes\_string(emp\_df$Absenteeism.time.in.hours))+geom\_histogram(stat='bin',binwidth = 1)+xlab('Absenteeism.time.in.hours')  ggplot(emp\_df,aes\_string(y=emp\_df$Absenteeism.time.in.hours,x=as.factor(emp\_df$Reason.for.absence)))+geom\_bar(stat='identity')+xlab('Reason.for.absence')+ylab('Absenteeism.time.in.hours')  ggplot(emp\_df,aes\_string(y=emp\_df$Absenteeism.time.in.hours,x=as.factor(emp\_df$Seasons)))+geom\_bar(stat='identity')+xlab('Seasons')+ylab('Absenteeism.time.in.hours')  ggplot(emp\_df,aes\_string(y=emp\_df$Absenteeism.time.in.hours,x=as.factor(emp\_df$Disciplinary.failure)))+geom\_bar(stat='identity')+xlab('Disciplinary Failure')+ylab('Absenteeism.time.in.hours')  ggplot(emp\_df,aes\_string(y=emp\_df$Absenteeism.time.in.hours,x=as.factor(emp\_df$Education)))+geom\_bar(stat='identity')+xlab('Education')+ylab('Absenteeism.time.in.hours')  ggplot(emp\_df,aes\_string(y=emp\_df$Absenteeism.time.in.hours,x=emp\_df$Hit.target))+geom\_bar(stat='identity')+xlab('Hit.target')+ylab('Absenteeism.time.in.hours')  #######################Outlier Analysis#####################################  #let us first check outliers  #Transportation.expense, Distance.from.Residence.to.Work  boxplot(emp\_df[,c('Transportation.expense','Distance.from.Residence.to.Work')])  #boxplot for Service.time, Age, Hit.target  boxplot(emp\_df[,c( 'Service.time', 'Age','Hit.target')])  #boxplot for Weight,Height,Body.mass.index  boxplot(emp\_df[,c('Weight', 'Height', 'Body.mass.index')])  #boxplot for Absenteeism.time.in.hours ,Work.load.Average.day  boxplot(emp\_df[,c('Absenteeism.time.in.hours','Work.load.Average.day.')])  for (i in continuous\_vars) {  print(i)  val = emp\_df[,i][emp\_df[,i] %in% boxplot.stats(emp\_df[,i])$out]  print(length(val))  print(val)  }  #Make each outlier as NA  for (i in continuous\_vars) {  val = emp\_df[,i][emp\_df[,i] %in% boxplot.stats(emp\_df[,i])$out]  emp\_df[,i][emp\_df[,i] %in% val] = NA  }  #Check number of missing values  sum(is.na(emp\_df))  #Get number of missing values in each variable  for (i in continuous\_vars) {  print(i)  print(sum(is.na(emp\_df[,i])))  }  #Impute the values using KNN method  emp\_df = knnImputation(emp\_df, k=3)  #Check again for missing value if present in case  sum(is.na(emp\_df))  ####################################Feature Selection ###################################  #Check for multicollinearity using corelation graph  library(corrgram)  corrgram(emp\_df[,continuous\_vars], order = F,  upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")  #Variable Reduction  emp\_df = subset.data.frame(emp\_df, select = -c(Body.mass.index,Service.time))  #Make a copy of Clean Data  clean\_data = emp\_df  write.xlsx(clean\_data, "clean\_data.xlsx", row.names = F)  ## ANOVA test for Categorical variable  anova\_test=aov(Absenteeism.time.in.hours~ID+Reason.for.absence+Month.of.absence+Day.of.the.week+Seasons+  Disciplinary.failure+Education+Son+Social.drinker+Social.smoker+Pet,data = emp\_df)  summary(anova\_test)  ## remove all categorical variables except Reason for absence and Month of absence  ##Removing Highly Corelated var  emp\_df = subset(emp\_df, select = -c(ID,Day.of.the.week,Seasons,  Disciplinary.failure,Education,Son,Social.drinker,Social.smoker,Pet))  #Make a backup of features selected  feature\_selected = emp\_df  write.csv(emp\_df,"Selected\_Features.csv", row.names = F)  colnames(emp\_df)  ######################################Feature Scaling ######################################  #draw histogram of each variable to know data is normal distributed or not  ggplot(emp\_df,aes\_string(emp\_df$Reason.for.absence))+geom\_histogram(stat='count',binwidth = 1)+xlab('Reason.for.absence')  hist(emp\_df$Transportation.expense)  hist(emp\_df$Distance.from.Residence.to.Work)  hist(emp\_df$Age)  hist(emp\_df$Work.load.Average.day.)  hist(emp\_df$Hit.target)  hist(emp\_df$Weight)  hist(emp\_df$Height)  #Now select numerical variables to perform normalization  print(sapply(emp\_df,is.numeric))  num\_names = c("Transportation.expense", "Distance.from.Residence.to.Work","Age","Work.load.Average.day.","Hit.target","Weight","Height")  factor\_columns = c("Reason.for.absence","Month.of.absence")  #Feature Scaling  #num\_names object contains all the numerical features of selected features  for (i in num\_names) {  print(i)  emp\_df[,i] = ((emp\_df[,i] - min(emp\_df[,i])) /  (max(emp\_df[,i]) - min(emp\_df[,i])))  }  back\_df = emp\_df  #Create dummy variables of factor variables  emp\_df = dummy.data.frame(emp\_df, factor\_columns)  head(emp\_df)  dim(emp\_df)  ###################### MODEL DEVELOPMENT #################################  #Cleaning the environment  #rmExcept("emp\_df")  ## Divide the data into train and test data using simple random sampling  #Splitting data into train and test data  library("rpart")  set.seed(7)  train\_index = sample(1:nrow(emp\_df), 0.8\* nrow(emp\_df))  train = emp\_df[train\_index,]  test = emp\_df[-train\_index,]  ########################################DECISION TREE########################################  #RMSE: 2.9360973  #MAE: 1.9640741  #R squared: 0.3401192  #Build decsion tree using rpart  dt\_model = rpart(Absenteeism.time.in.hours ~ ., data = train, method = "anova")  #Perdict for test cases  dt\_predictions = predict(dt\_model, test[,-52])  #Create data frame for actual and predicted values  df\_pred = data.frame("actual"=test[,52], "dt\_pred"=dt\_predictions)  head(df\_pred)  #Calcuate MAE, RMSE, R-sqaured for testing data  print(postResample(pred = dt\_predictions, obs = test[,52]))  #Plot a graph for actual vs predicted values  plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")  lines(dt\_predictions,col="blue")  #write rules into disk  write(capture.output(summary(dt\_model)), "Rules.txt")  #Summary of DT model  summary(dt\_model)  ########################################RANDOM FOREST########################################  #RMSE: 2.6872313  #MAE: 1.7624767  #R squared: 0.4490218  ##Train the model using training data  rf\_model = randomForest(Absenteeism.time.in.hours~., data = train, ntree = 500)  #Predict the test cases  rf\_predictions = predict(rf\_model, test[,-52])  #Create dataframe for actual and predicted values  df\_pred = cbind(df\_pred,rf\_predictions)  head(df\_pred)  #Calcuate MAE, RMSE, R-sqaured for testing data  print(postResample(pred = rf\_predictions, obs = test[,52]))  #Plot a graph for actual vs predicted values  plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="red")  lines(rf\_predictions,col="blue")  ########################################LINEAR REGRESSION########################################  #RMSE: 2.9470610  #MAE: 1.9320873  #R squared: 0.3350476  ##Train the model using training data  lr\_model = lm(formula = Absenteeism.time.in.hours~., data = train)  #Get the summary of the model  summary(lr\_model)  #Predict the test cases  lr\_prediction = predict(lr\_model, test[,-52])  #Create dataframe for actual and predicted values  df\_pred = cbind(df\_pred,lr\_prediction)  #df\_pred = subset.data.frame(df\_pred, select = -c(lr\_prediction))  head(df\_pred)  #Calcuate MAE, RMSE, R-sqaured for testing data  print(postResample(pred = lr\_prediction, obs = test[,52]))  #Plot a graph for actual vs predicted values  plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")  lines(lr\_prediction,col="red")  #########################################DIMENSION REDUCTION USING PCA########################################  #Principal component analysis  prin\_comp = prcomp(train)  #Compute standard deviation of each principal component  pr\_stdev = prin\_comp$sdev  #Compute variance  pr\_var = pr\_stdev^2  #Proportion of variance explained  prop\_var = pr\_var/sum(pr\_var)  #Cumulative scree plot  plot(cumsum(prop\_var), xlab = "Principal Component",  ylab = "Cumulative Proportion of Variance Explained",  type = "b")  #Add a training set with principal components  train.data = data.frame(Absenteeism.time.in.hours = train$Absenteeism.time.in.hours, prin\_comp$x)  # From the above plot selecting 20 components since it explains almost 95+ % data variance  train.data =train.data[,1:20]  #Transform test data into PCA  test.data = predict(prin\_comp, newdata = test)  test.data = as.data.frame(test.data)  #Select the first 20 components  test.data=test.data[,1:20]  ########################################DECISION TREE########################################  #RMSE:0.4098639  #MAE: 0.2650369  #R squared: 0.9872817  #Build decsion tree using rpart  dt\_model = rpart(Absenteeism.time.in.hours ~., data = train.data, method = "anova")  #Predict the test cases  dt\_predictions = predict(dt\_model,test.data)  #Create data frame for actual and predicted values  df\_preds = data.frame("actual"=test[,52], "dt\_pred"=dt\_predictions)  head(df\_preds)  #Calcuate MAE, RMSE, R-sqaured for testing data  print(postResample(pred = dt\_predictions, obs = test$Absenteeism.time.in.hours))  #Plot a graph for actual vs predicted values  plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="red")  lines(dt\_predictions,col="blue")  ########################################RANDOM FOREST########################################  #RMSE:0.7057195  #MAE: 0.3670694  #R squared: 0.9732212  #Train the model using training data  rf\_model = randomForest(Absenteeism.time.in.hours~., data = train.data, ntrees = 500)  #Predict the test cases  rf\_predictions = predict(rf\_model,test.data)  #Create dataframe for actual and predicted values  df\_preds = cbind(df\_preds,rf\_predictions)  head(df\_preds)  #Calcuate MAE, RMSE, R-sqaured for testing data  print(postResample(pred = rf\_predictions, obs = test$Absenteeism.time.in.hours))  #Plot a graph for actual vs predicted values  plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")  lines(rf\_predictions,col="blue")  ########################################LINEAR REGRESSION########################################  #RMSE: 0.007397825  #MAE: 0.005452601  #R squared: 0.999995826  #Train the model using training data  lr\_model = lm(Absenteeism.time.in.hours ~ ., data = train.data)  #Get the summary of the model  summary(lr\_model)  #Predict the test cases  lr\_predictions = predict(lr\_model,test.data)  #Create dataframe for actual and predicted values  df\_preds = cbind(df\_preds,lr\_predictions)  head(df\_preds)  #Calcuate MAE, RMSE, R-sqaured for testing data  print(postResample(pred = lr\_predictions, obs =test$Absenteeism.time.in.hours))  #Plot a graph for actual vs predicted values  plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="red")  lines(lr\_predictions,col="blue") |

**Appendix B -Python Code**

|  |
| --- |
| # 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?  #  #  # In[1]:  #Import libraries  import os  import pandas as pd  import numpy as np  from sklearn.metrics import r2\_score  from fancyimpute import KNN  from sklearn.metrics import mean\_squared\_error  # In[2]:  #import libraries for plots  import matplotlib.pyplot as plt  import seaborn as sns  get\_ipython().run\_line\_magic('matplotlib', 'inline')  # In[3]:  # Setting working directory  os.chdir("C:/Users/HP/Desktop/edwisor/Project 1")  # Loading data  emp\_df = pd.read\_excel("Absenteeism\_at\_work\_Project.xls")  # Exploratory Data Analysis  # In[4]:  emp\_df.dtypes  # In[5]:  #first five rows  emp\_df.head(5)  # In[6]:  #Dimension  emp\_df.shape  # In[7]:  # Number of Unique values present in each variable  emp\_df.nunique()  # In[8]:  emp\_df.columns  # In[9]:  # From the EDA and problem statement file categorising the variables in two category " Continuos" and "Categorical"  continuous\_vars = ['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']  categorical\_vars = ['ID','Reason for absence','Month of absence','Day of the week',  'Seasons','Disciplinary failure', 'Education', 'Social drinker',  'Social smoker', 'Pet', 'Son']  # In[10]:  #Make a copy of dataframe  df\_copy = emp\_df.copy()  # Pre Processing  # In[11]:  # Missing value Analysis  emp\_df.isna().sum()  Missing\_Value = pd.DataFrame((emp\_df.isna().sum()/len(emp\_df)\*100))  Missing\_Value.reset\_index()  Missing\_Value = Missing\_Value.rename(columns = {'index': 'Variables', 0: 'Missing\_percentage'})  #Arranging Missing Values in Decreasing Order  Missing\_Value = Missing\_Value.sort\_values('Missing\_percentage', ascending = False)  Missing\_Value  # In[12]:  fig\_size = plt.rcParams['figure.figsize']  fig\_size[0] = 16  fig\_size[1] = 4  plt.rcParams['figure.figsize'] = fig\_size  sns.barplot(x=Missing\_Value.index,y='Missing\_percentage',data=Missing\_Value)  # Using ID to impute Distance from Residence to Work  # In[13]:  df = emp\_df[emp\_df['Distance from Residence to Work'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Distance from Residence to Work'].isnull()) & (emp\_df['ID']==i),'Distance from Residence to Work'] = emp\_df.loc[emp\_df['ID']==i,'Distance from Residence to Work'].mean()  # Using ID to impute Transportation expense  # In[14]:  df = emp\_df[emp\_df['Transportation expense'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Transportation expense'].isnull()) & (emp\_df['ID']==i),'Transportation expense'] = emp\_df.loc[emp\_df['ID']==i,'Transportation expense'].mean()  # Using ID to impute Service time  # In[15]:  df = emp\_df[emp\_df['Service time'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Service time'].isnull()) & (emp\_df['ID']==i),'Service time'] = emp\_df.loc[emp\_df['ID']==i,'Service time'].mean()  # Using ID to impute Age  # In[16]:  df = emp\_df[emp\_df['Age'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Age'].isnull()) & (emp\_df['ID']==i),'Age'] = emp\_df.loc[emp\_df['ID']==i,'Age'].mean()  # Using ID to impute Education  # In[17]:  df = emp\_df[emp\_df['Education'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Education'].isnull()) & (emp\_df['ID']==i),'Education'] = emp\_df.loc[emp\_df['ID']==i,'Education'].mean()  # Using ID to impute Son  # In[18]:  df = emp\_df[emp\_df['Son'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Son'].isnull()) & (emp\_df['ID']==i),'Son'] = emp\_df.loc[emp\_df['ID']==i,'Son'].mean()  # Using ID to impute Social drinker  # In[19]:  df = emp\_df[emp\_df['Social drinker'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Social drinker'].isnull()) & (emp\_df['ID']==i),'Social drinker'] = emp\_df.loc[emp\_df['ID']==i,'Social drinker'].mean()  # Using ID to impute Social smoker  # In[20]:  df = emp\_df[emp\_df['Social smoker'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Social smoker'].isnull()) & (emp\_df['ID']==i),'Social smoker'] = emp\_df.loc[emp\_df['ID']==i,'Social smoker'].mean()  # Using ID to impute Pet  # In[21]:  df = emp\_df[emp\_df['Pet'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Pet'].isnull()) & (emp\_df['ID']==i),'Pet'] = emp\_df.loc[emp\_df['ID']==i,'Pet'].mean()  # Using ID to impute Weight  # In[22]:  df = emp\_df[emp\_df['Weight'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Weight'].isnull()) & (emp\_df['ID']==i),'Weight'] = emp\_df.loc[emp\_df['ID']==i,'Weight'].mean()  # Using ID to impute Height  # In[23]:  df = emp\_df[emp\_df['Height'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Height'].isnull()) & (emp\_df['ID']==i),'Height'] = emp\_df.loc[emp\_df['ID']==i,'Height'].mean()  # Using ID to impute Body mass index  # In[24]:  df = emp\_df[emp\_df['Body mass index'].isna() == True]['ID']  print(df)  for i in df:  emp\_df.loc[(emp\_df['Body mass index'].isnull()) & (emp\_df['ID']==i),'Body mass index'] = emp\_df.loc[emp\_df['ID']==i,'Body mass index'].mean()  # In[25]:  emp\_df.isna().sum()  # For other variables, we can impute missing variables through mean, median or KNN imputation. We need to select the best method.  # In[26]:  #Actual value = 1  #Mean = 19.183423913043477  #Median = 23  #KNN = 3.4878048780487805  print(emp\_df['Reason for absence'].iloc[10])  #Set the value of first row in Reason for absence as NAN  #create missing value  emp\_df['Reason for absence'].iloc[10] = np.nan  # In[27]:  #Impute with mean  #emp\_df['Reason for absence'] = emp\_df['Reason for absence'].fillna(emp\_df['Reason for absence'].mean())  #Impute with median  #emp\_df['Reason for absence'].iloc[7] = np.nan  #emp\_df['Reason for absence'] = emp\_df['Reason for absence'].fillna(emp\_df['Reason for absence'].median())  #Apply KNN imputation algorithm  emp\_df = pd.DataFrame(KNN(k = 3).fit\_transform(emp\_df), columns = emp\_df.columns)  emp\_df['Reason for absence'].iloc[10]  # So KNN is the best method to impute mising values  # In[28]:  emp\_df.isna().sum()  # Data Conversion  # In[29]:  #converting appropriate categorical variable into actual categorical variable  emp\_df['ID']=emp\_df['ID'].astype('category')  emp\_df['Reason for absence']=emp\_df['Reason for absence'].astype('category')  emp\_df['Month of absence']=emp\_df['Month of absence'].astype('category')  emp\_df['Day of the week']=emp\_df['Day of the week'].astype('category')  emp\_df['Seasons']=emp\_df['Seasons'].astype('category')  emp\_df['Disciplinary failure']=emp\_df['Disciplinary failure'].astype('category')  emp\_df['Education']=emp\_df['Education'].astype('category')  emp\_df['Son']=emp\_df['Son'].astype('category')  emp\_df['Social drinker']=emp\_df['Social drinker'].astype('category')  emp\_df['Social smoker']=emp\_df['Social smoker'].astype('category')  emp\_df['Pet']=emp\_df['Pet'].astype('category')  # In[30]:  emp\_df.dtypes  # Univarant and Bivariant  # In[31]:  #Check whether target variable is normal or not  sns.distplot(emp\_df['Absenteeism time in hours']);  # In[32]:  print("Skewness: %f" % emp\_df['Absenteeism time in hours'].skew())  print("Kurtosis: %f" % emp\_df['Absenteeism time in hours'].kurt())  # In[33]:  #Reason for absence VS Absenteeism time in hours  sns.factorplot(x='Reason for absence', y="Absenteeism time in hours", kind= 'bar',data=emp\_df,height=7.5,aspect=12/7.5,estimator=np.sum)  # Seasons VS Absenteeism with hue = 'Day of the Week'  # In[34]:  sns.barplot(x='Seasons',y='Absenteeism time in hours',data=emp\_df,hue = 'Day of the week')  # Disciplinary Failure VS Absenteesim time in hours  # In[35]:  sns.factorplot(x='Disciplinary failure', y="Absenteeism time in hours", kind= 'bar',data=emp\_df,height=5,aspect=12/7.5,estimator=np.sum)  # Education VS Absenteesim time in hours  # In[36]:  sns.factorplot(x='Education', y="Absenteeism time in hours", kind= 'bar',data=emp\_df,height=5,aspect=12/7.5,estimator=np.sum)  # Hit Target VS Absenteeism time in hours  # In[37]:  sns.barplot(x='Hit target',y='Absenteeism time in hours',data=emp\_df)  # In[38]:  #fig\_size = plt.rcParams['figure.figsize']  #fig\_size[0] = 11  #fig\_size[1] = 5  #plt.rcParams['figure.figsize'] = fig\_size  # In[39]:  #Loss per month  #sum = list(emp\_df.groupby(['Month of absence'])['Absenteeism time in hours'].agg('sum'))  #del sum[0]  #del sum[9]  #sum  #month = list(range(1,13))  #month  #loss\_df=pd.DataFrame({'Month':month,'Loss':sum})  #loss\_df  #sns.barplot(x='Month',y='Loss',data=loss\_df)  # In[40]:  fig\_size = plt.rcParams['figure.figsize']  fig\_size[0] = 25  fig\_size[1] = 20  plt.rcParams['figure.figsize'] = fig\_size  # Outliers Analysis  # In[41]:  ######################################### Outlier Analysis ####################################################  plt.subplot(4,3,1)  plt.boxplot(emp\_df['Transportation expense'])  plt.title('Transportation expense')  plt.subplot(4,3,2)  plt.boxplot(emp\_df['Distance from Residence to Work'])  plt.title('Distance from Residence to Work')  plt.subplot(4,3,3)  plt.boxplot(emp\_df['Service time'])  plt.title('Service time')  plt.subplot(4,3,4)  plt.boxplot(emp\_df['Age'])  plt.title('Age')  plt.subplot(4,3,5)  plt.boxplot(emp\_df['Work load Average/day '])  plt.title('Work load Average/day ')  plt.subplot(4,3,6)  plt.boxplot(emp\_df['Hit target'])  plt.title('Hit target')  plt.subplot(4,3,7)  plt.boxplot(emp\_df['Weight'])  plt.title('Weight')  plt.subplot(4,3,8)  plt.boxplot(emp\_df['Height'])  plt.title('Height')  plt.subplot(4,3,9)  plt.boxplot(emp\_df['Body mass index'])  plt.title('Body mass index')  plt.subplot(4,3,10)  plt.boxplot(emp\_df['Absenteeism time in hours'])  plt.title('Absenteeism time in hours')  plt.tight\_layout()  #We have outlerss in most of the variables  # In[42]:  #Deleting Outliers  cname = ['Transportation expense','Distance from Residence to Work', 'Service time','Age','Work load Average/day ','Hit target','Weight','Height','Body mass index','Absenteeism time in hours']  numeric\_data = emp\_df[cname]  for i in cname:  print(i)  q75, q25 = np.percentile(emp\_df.loc[:,i],[75,25])  iqr = q75- q25  min = q25 - (1.5\*iqr)  max = q75 + (1.5\*iqr)  emp\_df.loc[emp\_df.loc[:,i] < min , i] = np.nan  emp\_df.loc[emp\_df.loc[:,i] > max , i] = np.nan  # In[43]:  pd.isnull(emp\_df).sum() #after deletion we have 233 Missing values  # Impute missing value through KNN imputation  # In[44]:  #Apply KNN imputation algorithm  emp\_df = pd.DataFrame(KNN(k = 3).fit\_transform(emp\_df), columns = emp\_df.columns)  # In[45]:  pd.isnull(emp\_df).sum()  # Feature Selection  # In[46]:  emp\_df.to\_excel("new\_employee.xlsx",index=False)  #correlation graph for continous variables  #Get dataframe with all continuous variables  df\_corr = emp\_df.loc[:,continuous\_vars]  # In[47]:  #Check for multicollinearity using corelation graph  #Set the width and hieght of the plot  f, ax = plt.subplots(figsize=(10, 10))  #Generate correlation matrix  corr = df\_corr.corr()  #Plot using seaborn library  sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool),  cmap=sns.diverging\_palette(220, 50, as\_cmap=True),  square=True, ax=ax, annot = True)  plt.plot()  # In[48]:  #Variable Reduction  to\_drop = ['Body mass index','Service time']  emp\_df = emp\_df.drop(to\_drop, axis = 1)  # In[49]:  # Updating the Continuous and Categorical Variables  continuous\_vars.remove('Body mass index')  continuous\_vars.remove('Service time')  # In[50]:  #Make a copy of clean data  clean\_data = emp\_df.copy()  #emp\_df = clean\_data  # In[51]:  # now let us see anova test for categorical varibles  # In[52]:  #loop for ANOVA test Since the target variable is continuous  from scipy import stats  # In[53]:  #loop for ANOVA test Since the target variable is continuous  for i in categorical\_vars:  f, p = stats.f\_oneway(emp\_df[i], emp\_df["Absenteeism time in hours"])  print("P value for variable "+str(i)+" is "+str(p))  # In[54]:  #we see that reason of absense has valuess less than 0.05, rest all are above 0.05 so we drop all varibles except reason of absense  #we want to know loss per month so we are not droping Month of absence  # In[55]:  emp\_df=emp\_df.drop(['ID','Seasons','Pet','Son','Social drinker','Day of the week','Disciplinary failure','Education','Social smoker'],axis=1)  # In[56]:  categorical\_vars.remove('ID')  categorical\_vars.remove('Seasons')  categorical\_vars.remove('Pet')  categorical\_vars.remove('Son')  categorical\_vars.remove('Social drinker')  categorical\_vars.remove('Day of the week')  categorical\_vars.remove('Disciplinary failure')  categorical\_vars.remove('Education')  categorical\_vars.remove('Social smoker')  # In[57]:  emp\_df.head(10)  # Feature Scaling  # In[58]:  #we see that the data is skewed  #Scaling for Continious variable  plt.figure(figsize=(14,4))  plt.subplot(3,3,1)  sns.distplot(emp\_df['Transportation expense'])  plt.title('Transportation expense')  plt.subplot(3,3,2)  sns.distplot(emp\_df['Distance from Residence to Work'])  plt.title('Distance from Residence to Work')  plt.subplot(3,3,3)  sns.distplot(emp\_df['Work load Average/day '])  plt.title('Work load Average/day')  plt.subplot(3,3,4)  sns.distplot(emp\_df['Hit target'])  plt.title('Hit target')  plt.subplot(3,3,5)  sns.distplot(emp\_df['Age'])  plt.title('Age')  plt.subplot(3,3,6)  sns.distplot(emp\_df['Month of absence'])  plt.title('Month of absence')  plt.subplot(3,3,7)  sns.distplot(emp\_df['Reason for absence'])  plt.title('Reason for absence')  plt.subplot(3,3,8)  sns.distplot(emp\_df['Weight'])  plt.title('Weight')  plt.subplot(3,3,9)  sns.distplot(emp\_df['Height'])  plt.title('Height')  plt.tight\_layout()  # In[59]:  #Normalization of continuous variables  for i in continuous\_vars:  if i == 'Absenteeism time in hours':  continue  emp\_df[i] = (emp\_df[i] - emp\_df[i].min())/(emp\_df[i].max()-emp\_df[i].min())  # In[60]:  #Create dummy variables of factor variables  emp\_df = pd.get\_dummies(data = emp\_df, columns = categorical\_vars)  # Copying dataframe  df1 = emp\_df.copy()  # Modelling  # In[61]:  #Splitting data into train and test data  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split( emp\_df.iloc[:, emp\_df.columns != 'Absenteeism time in hours'],emp\_df.iloc[:, 8], test\_size = 0.20, random\_state = 1)  # Decison Tree  # In[62]:  # Importing libraries for Decision Tree  from sklearn.tree import DecisionTreeRegressor  #Build decsion tree using DecisionTreeRegressor  dt\_model = DecisionTreeRegressor(random\_state = 1).fit(X\_train,y\_train)  #Perdict for test cases  dt\_predictions = dt\_model.predict(X\_test)  #Create data frame for actual and predicted values  df\_dt = pd.DataFrame({'actual': y\_test, 'pred': dt\_predictions})  print(df\_dt.head())  #Define function to calculate RMSE  def RMSE(y\_actual,y\_predicted):  rmse = np.sqrt(mean\_squared\_error(y\_actual,y\_predicted))  return rmse  #Calculate RMSE and R-squared value  print("Root Mean Squared Error: "+str(RMSE(y\_test, dt\_predictions)))  print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, dt\_predictions)))  # In[63]:  # Importing libraries for Random Forest  from sklearn.ensemble import RandomForestRegressor  #Build random forest using RandomForestRegressor  rf\_model = RandomForestRegressor(n\_estimators = 500, random\_state = 1).fit(X\_train,y\_train)  #Perdict for test cases  rf\_predictions = rf\_model.predict(X\_test)  #Create data frame for actual and predicted values  df\_rf = pd.DataFrame({'actual': y\_test, 'pred': rf\_predictions})  print(df\_rf.head())  #Calculate RMSE and R-squared value  print("Root Mean Squared Error: "+str(RMSE(y\_test, rf\_predictions)))  print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, rf\_predictions)))  # In[64]:  # Importing libraries for Linear Regression  from sklearn.linear\_model import LinearRegression  #Train the model  lr\_model = LinearRegression().fit(X\_train , y\_train)  #Perdict for test cases  lr\_predictions = lr\_model.predict(X\_test)  #Create data frame for actual and predicted values  df\_lr = pd.DataFrame({'actual': y\_test, 'pred': lr\_predictions})  print(df\_lr.head())  #Calculate RMSE and R-squared value  print("Root Mean Squared Error: "+str(RMSE(y\_test, lr\_predictions)))  print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, lr\_predictions)))  # In[65]:  #Get the target variable  target = emp\_df['Absenteeism time in hours']  # In[66]:  #Get the number of rows and columns of data  emp\_df.shape  # In[67]:  #Import library for PCA  from sklearn.decomposition import PCA  plt.figure(figsize=(14,4))  #Converting data to numpy array  X = emp\_df.values  #Data has 54 variables so no of components of PCA = 53  pca = PCA(n\_components=53)  pca.fit(X)  #Proportion of variance explained  var= pca.explained\_variance\_ratio\_  #Cumulative scree plot  var1=np.cumsum(np.round(pca.explained\_variance\_ratio\_, decimals=4)\*100)  #Draw the plot  plt.plot(var1)  plt.show()  # In[68]:  #Selecting 20 components since it explains almost 95+ % data variance  pca = PCA(n\_components=20)  #Fitting the selected components to the data  pca.fit(X)  #Splitting data into train and test data  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,target, test\_size=0.2, random\_state = 1)  # In[69]:  #Build decsion tree using DecisionTreeRegressor  dt\_model = DecisionTreeRegressor(random\_state = 1).fit(X\_train,y\_train)  #Perdict for test cases  dt\_predictions = dt\_model.predict(X\_test)  #Create data frame for actual and predicted values  df\_dt = pd.DataFrame({'actual': y\_test, 'pred': dt\_predictions})  print(df\_dt.head())  #Calculate RMSE and R-squared value  print("Root Mean Squared Error: "+str(RMSE(y\_test, dt\_predictions)))  print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, dt\_predictions)))  # In[70]:  #Build random forest using RandomForestRegressor  rf\_model = RandomForestRegressor(n\_estimators = 500, random\_state = 1).fit(X\_train,y\_train)  #Perdict for test cases  rf\_predictions = rf\_model.predict(X\_test)  #Create data frame for actual and predicted values  df\_rf = pd.DataFrame({'actual': y\_test, 'pred': rf\_predictions})  print(df\_rf.head())  #Calculate RMSE and R-squared value  print("Root Mean Squared Error: "+str(RMSE(y\_test, rf\_predictions)))  print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, rf\_predictions)))  # In[71]:  # Importing libraries for Linear Regression  from sklearn.linear\_model import LinearRegression  #Train the model  lr\_model = LinearRegression().fit(X\_train , y\_train)  #Perdict for test cases  lr\_predictions = lr\_model.predict(X\_test)  #Create data frame for actual and predicted values  df\_lr = pd.DataFrame({'actual': y\_test, 'pred': lr\_predictions})  print(df\_lr.head())  #Calculate RMSE and R-squared value  print("Root Mean Squared Error: "+str(RMSE(y\_test, lr\_predictions)))  print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, lr\_predictions)))  # In[ ]:  # In[ ]: |