

Employee Absenteeism



December 31, 2018

Contents

[Chapter 1 2](#_Toc534039295)

[**1.1 Introduction** 2](#_Toc534039296)

[**1.2 Attributes description** 2](#_Toc534039297)

[**1.3 Read data & description** 3](#_Toc534039298)

[**1.4 Distribution of variable** 4](#_Toc534039299)

[**Chapter 2** 5](#_Toc534039300)

[**2.1 Missing value analysis and treatment** 5](#_Toc534039301)

[**2.2 Outlier Analysis** 7](#_Toc534039302)

[**Chapter 3** 12](#_Toc534039303)

[**3.1 Relationship & Prediction** 12](#_Toc534039304)

[**3.2 Possible treatments** 13](#_Toc534039305)

[**3.3 Forecasting** 16](#_Toc534039306)

[**Chapter 4** 19](#_Toc534039307)

[**4.1 Feature Selection** 19](#_Toc534039308)

[**4.2 Model Development** 22](#_Toc534039309)

[**4.3 Evaluation model** 23](#_Toc534039310)

[**4.4 K-Fold cross validation** 24](#_Toc534039311)

[**Python Code** 25](#_Toc534039312)

[**R Code** 38](#_Toc534039313)

# Chapter 1

## **1.1 Introduction**

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?  
Build suitable model (both R and Python) to answer the above two questions with a  
proper report

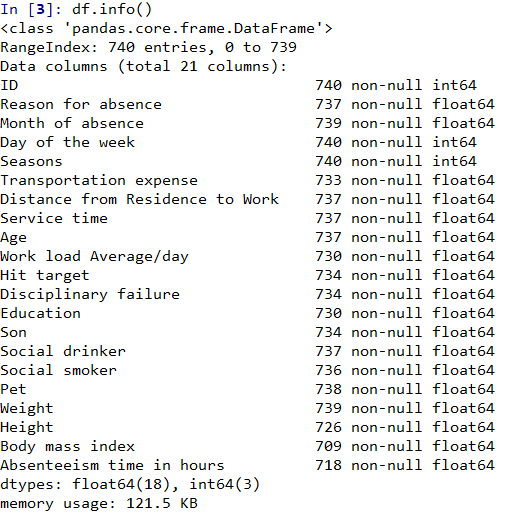
## **1.2 Attributes description**

**Dataset Details:**Dataset Characteristics: Timeseries MultivariantNumber of Attributes: 21Missing Values : Yes **Attribute Information:**1. Individual identification (ID)2. Reason for absence (ICD).Absences attested by the International Code of Diseases (ICD) stratified into 21categories (I to XXI) as follows:I Certain infectious and parasitic diseasesII NeoplasmsIII Diseases of the blood and blood-forming organs and certain disorders involving theimmune mechanismIV Endocrine, nutritional and metabolic diseasesV Mental and behavioural disordersVI Diseases of the nervous systemVII Diseases of the eye and adnexaVIII Diseases of the ear and mastoid processIX Diseases of the circulatory systemX Diseases of the respiratory systemXI Diseases of the digestive systemXII Diseases of the skin and subcutaneous tissueXIII Diseases of the musculoskeletal system and connective tissueXIV Diseases of the genitourinary systemXV Pregnancy, childbirth and the puerperiumXVI Certain conditions originating in the perinatal periodXVII Congenital malformations, deformations and chromosomal abnormalitiesXVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhereclassifiedXIX Injury, poisoning and certain other consequences of external causesXX External causes of morbidity and mortalityXXI Factors influencing health status and contact with health services.And 7 categories without (CID) patient follow-up (22), medical consultation (23), blooddonation (24), laboratory examination (25), unjustified absence (26), physiotherapy (27),dental consultation (28).3. Month of absence4. 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 expense7. Distance from Residence to Work (kilometers)8. Service time9. Age10. Work load Average/day11. Hit target12. 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. Weight19. Height20. Body mass index21. Absenteeism time in hours (target)

## **1.3 Read data & description**

*df = pd.read\_excel("Absenteeism\_at\_work\_Project.xls")*

*df. info()*



## **1.4 Distribution of variable**

Here, we are distributing variable into two categories, namely continuous variable and categorical variable

*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']*

# **Chapter 2**

**Methodology**

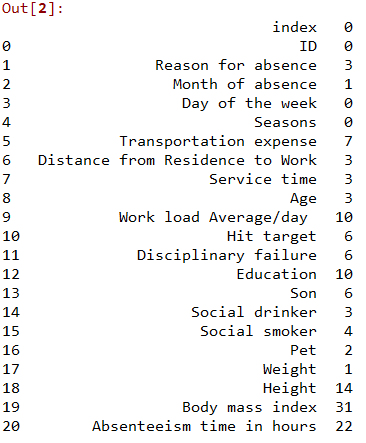
## **2.1 Missing value analysis and treatment**

The concept of missing values is important to understand in order to successfully manage data.  If the missing values are not handled properly by the researcher, then he/she may end up drawing an inaccurate inference about the data.  Due to improper handling, the result obtained by the researcher will differ from ones where the missing values are present.

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

*missing\_val = pd.DataFrame(df.isnull().sum()).reset\_index()*

*missing\_val*

**

*#Rename variable for missing value of dataframe*

*missing\_val = missing\_val.rename(columns = {'index': 'Variables', 0: 'Missing\_percentage'})*

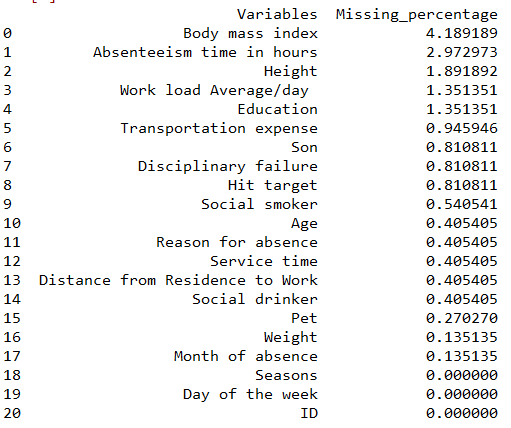
*#Percentage finding*

*missing\_val['Missing\_percentage'] = (missing\_val['Missing\_percentage']/len(df))\*100*

*#Sort missing values*

*missing\_val = missing\_val.sort\_values('Missing\_percentage', ascending = False).reset\_index(drop = True)*

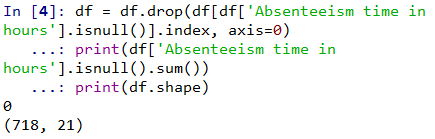
*Output of missing\_val*

**

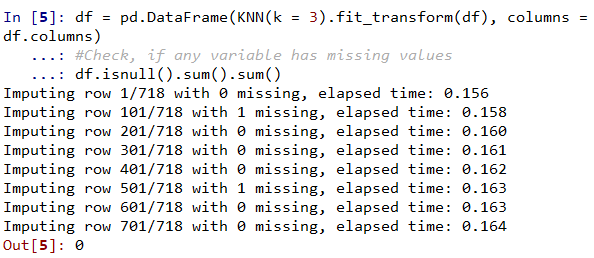
From above figure, we can see that Reason.for.absence", "Month.of.absence", "Day.of.the.week", "Seasons", "Education", "ID", "Age", "Weight","Height","Body.mass.index" had ‘0’ value in the observation .

As per figure, we can see that no column has more than 30% missing values. Therefore, we will not drop any column from dataset.

Although, we have not impute any value to the target variable. We have drop the observation, those target value has missing value.



For imputing the missing value, there are many methods such as mean, mode and KNN imputation. Here we have used KNN imputation method to fill missing data.



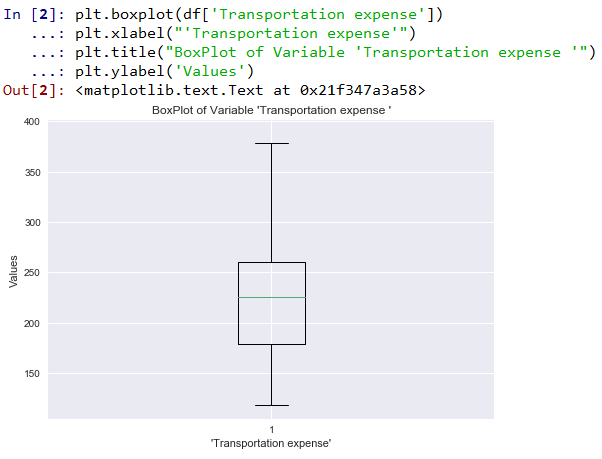
## **2.2 Outlier Analysis**

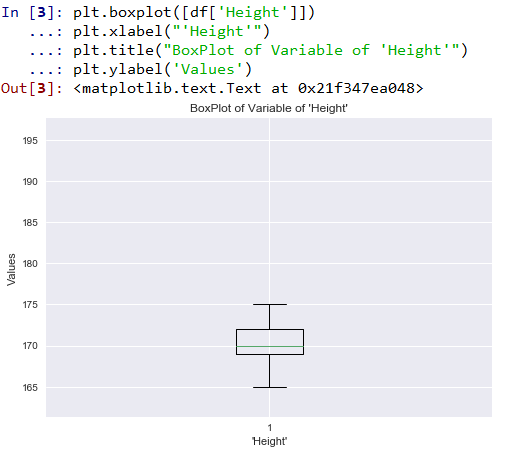
While *Outliers*, are attributed to a rare chance and may not necessarily be fully explainable, *Outliers* in data can distort predictions and affect the accuracy, if you don’t detect and handle them.

The contentious decision to consider or discard an *outlier* needs to be taken at the time of building the model. *Outliers* can drastically bias/change the fit estimates and predictions. It is left to the best judgement of the analyst to decide whether treating *outliers* is necessary and how to go about it.

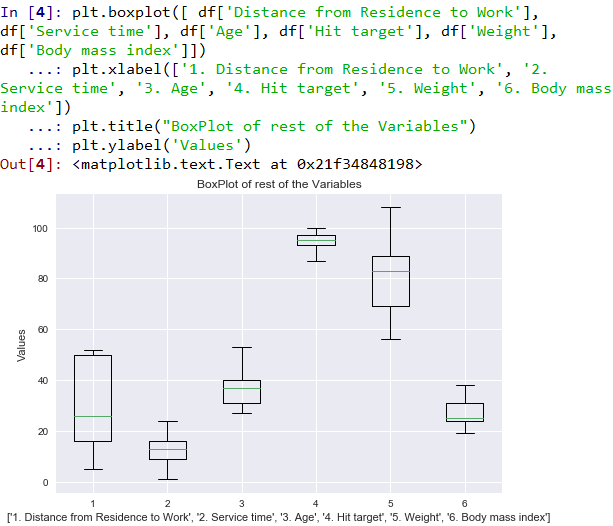
Treating or altering the *outlier*/extreme values in genuine observations is not a standard operating procedure. If a data point (or points) is excluded from the data analysis, this should be clearly stated on any subsequent report.

We have analyzed continuous and categorical variable for outlier analysis.



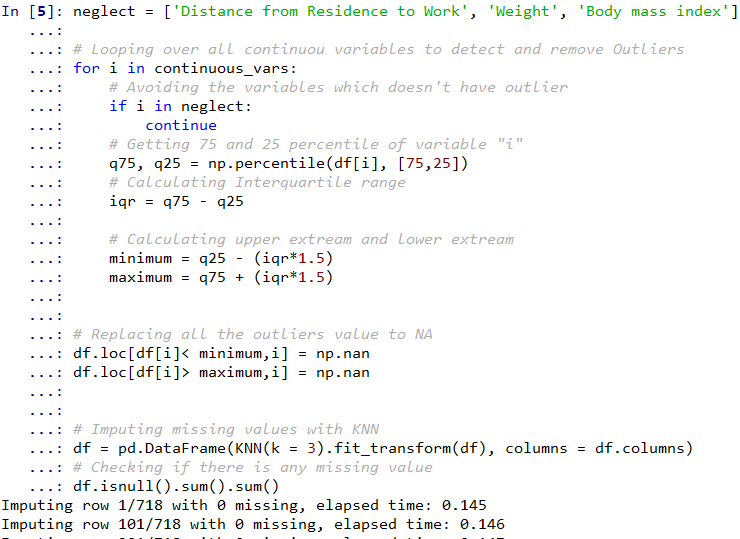


In report, we are showing only few variables box plot for outlier analysis. But in code, we have implemented required analysis



**Treatment of outliers,**

As per analysis, we can observe that few variables doesn’t have outliers. Therefore, we will not treat them. But for others, we have implemented strategy where, it creates first empty value and then after, using KNN imputation method, we will filled data.

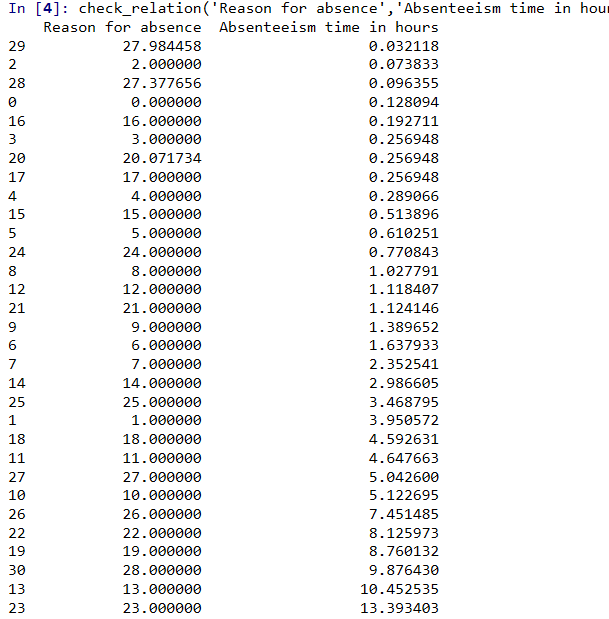


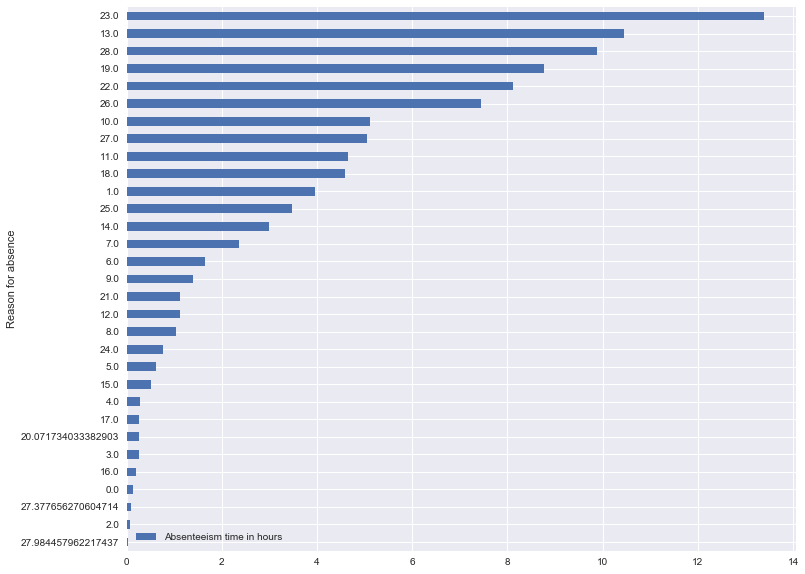
# **Chapter 3**

## **3.1 Relationship & Prediction**

We will check relationship between categorical variable vs dependent variable

Check relation between Reason for absence and Absenteeism time in hours



-

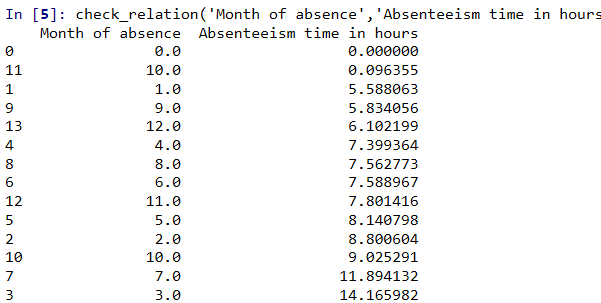
Top 3 categories in order of ‘Absenteeism time in hours’ are:

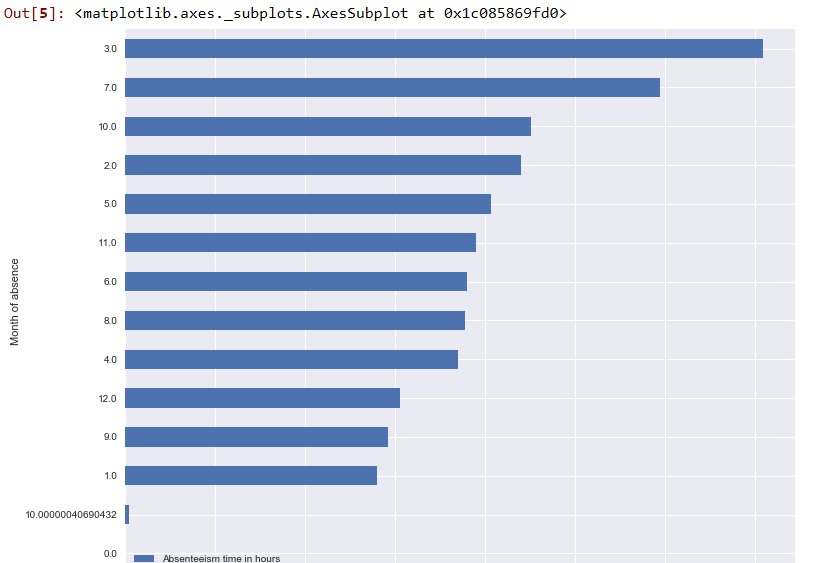
1. Category 23: :medical consultation – around 13 % of total time
2. Category 13 – Diseases of the musculoskeletal system and connective tissue around 11.17 % of total time
3. Category 19: dental consultation - 10 % of total time
4. Category 28: - Injury, poisoning and certain other consequences of external causes around 8% 0f total time.

## **3.2 Possible treatments**

1. Medical consultation may be brought down by optimizing workloads.
2. Musculoskeletal system disease is the major reason of absenteism\_dataeeism. Bad working posture & high workload are possible reasons for the high incidence of musculoskeletal disease. Company should conduct a study on the working postures of people and go for more ergonomic workplace design. Company should try to optimize workload keeping in mind occupational health of working people.
3. Dental consultation time may be reduced by informing employees of the dental health guidelines so that they can take better care of their teeth.
4. Injury incidence may be reduced by creating proper ergonomic working setup.

**Check relation between 'Month of absence' Vs. 'Absenteeism time in hours'**

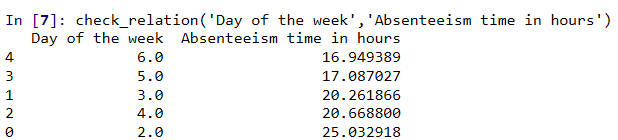


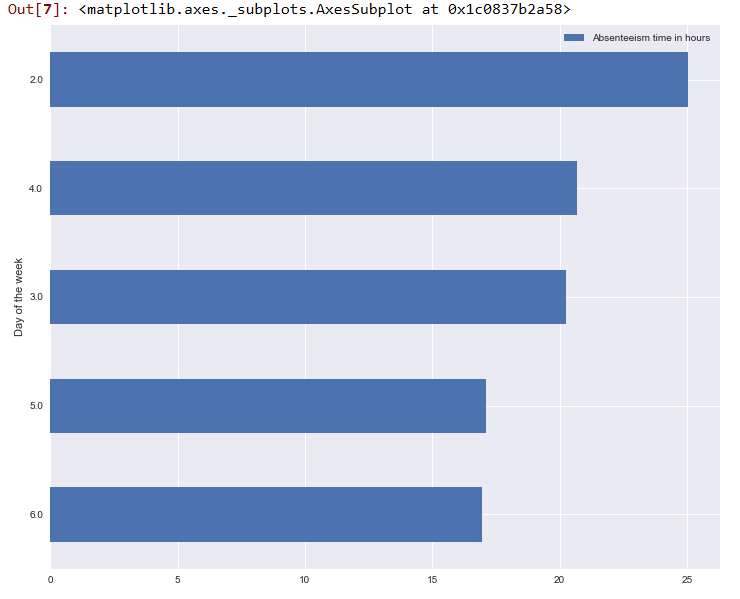


From above figure, we can see that top 3 months for absence are

1. Month 3, which has around 14%
2. Month 7 that has around 11%
3. Month 10 has around 9%

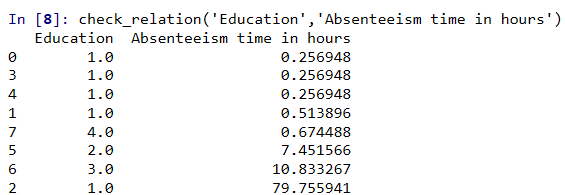
**Check relation between Day of the week and target variable**

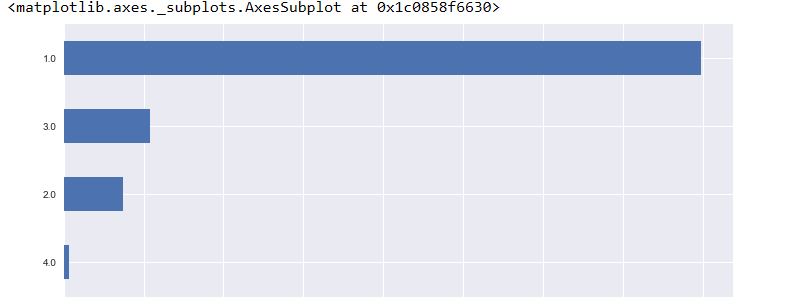




From above figure, we can clearly see that first day of week i.e. Monday has highest time hour and it is around25%. On Wednesday, it is around 21% hours are absence reported and Tuesday around 20%.

**Checking relationship between education and target variable**





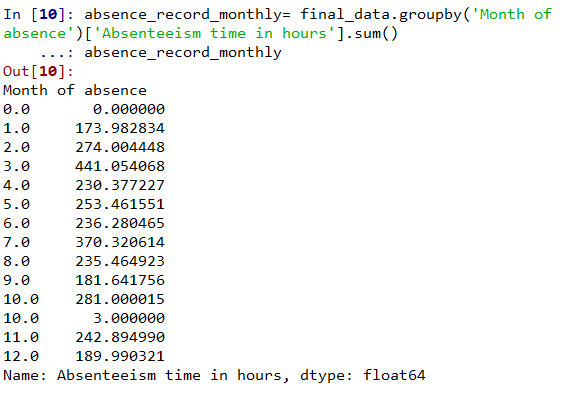
As per above figure, we can observe that we has having high school education, which leads to more absence % in hours.

## **3.3 Forecasting**

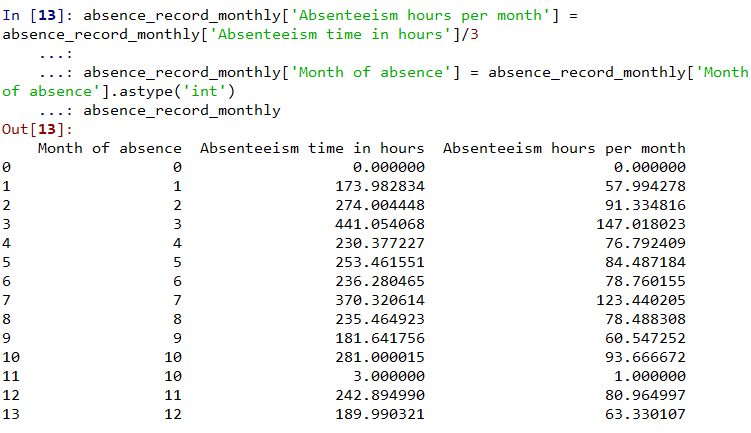
As per objective of question that, How much losses every month can we project in 2011 if same trend of absenteeism continues?

We tried to answer the same in different approach. Understanding the original data set us understand that it is the data for 3 years. Since this information is not provided to us in the original question but months are repeated thrice hence this understanding. Divinding absence hrs. by 3 since this data is of 3 years from 2007 to 2010.

In first step, we have find aggregate absence hours by month.



As we discussed above, we have divided dataset into three sets.



# **Chapter 4**

## **4.1 Feature Selection**

Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data.

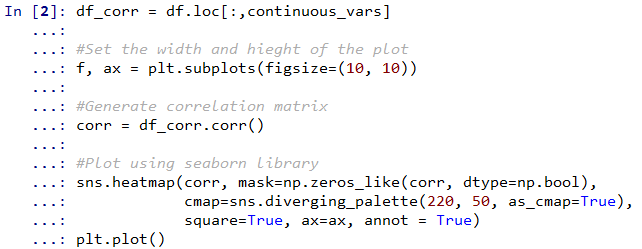
This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results than complete set of feature for the same algorithm. Or as [Rohan Rao](https://www.analyticsvidhya.com/blog/2016/10/exclusive-interview-ama-with-data-scientist-rohan-rao-analytics-vidhya-rank-4/) puts it – “Sometimes, less is better!”

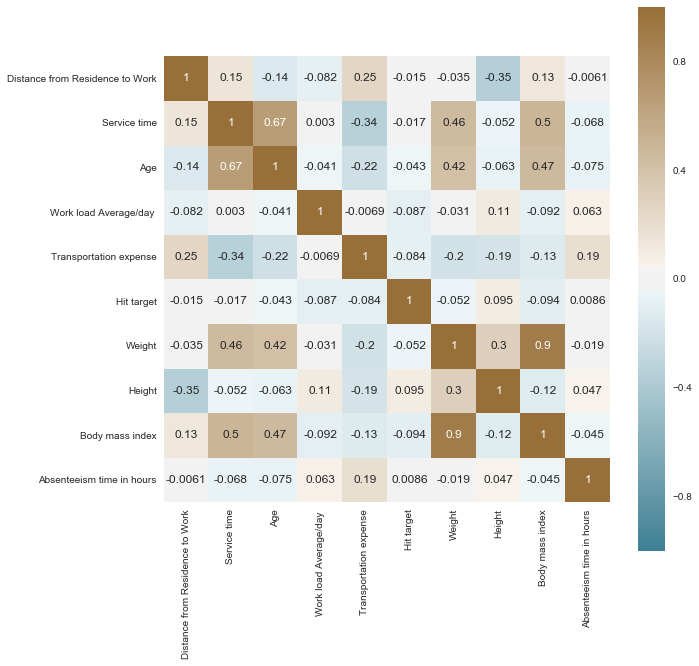
Not only in the competitions but this can be very useful in industrial applications as well. You not only reduce the training time and the evaluation time, you also have less things to worry about!

Top reasons to use feature selection are:

* It enables the machine learning algorithm to train faster.
* It reduces the complexity of a model and makes it easier to interpret.
* It improves the accuracy of a model if the right subset is chosen.
* It reduces overfitting.

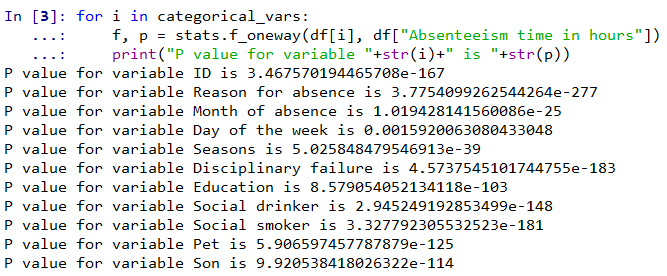
Here, we have used correlation matrix for continuous variable by using this formula



d

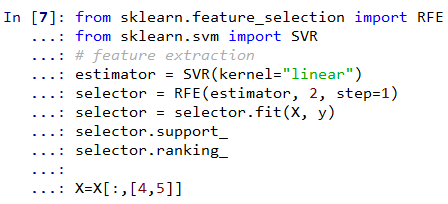
As we can see from figure, few variable has very/low similarity with each other. Therefore, we will omit those features from given data set. Therefore, we will drop weight, height, age, service time and ID from the features set

For categorical analysis, we have used Annova score since all values are categorical variables.



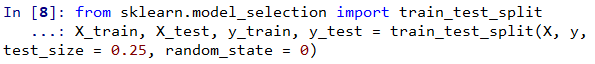
We will drop those variable which has hold more than 5% values. Therefore, we will drop Season, Education, pet and Son.

By reducing all features, we have now less independent variables. Although, there are many variable, So if we put many variable for training, we will not train our proper mode. For same, we have used recursive feature elimination technique to take few parameters.

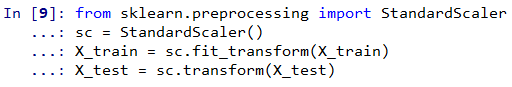


## **4.2 Model Development**

For training and testing our model, we have divided our data into two batch i.e. training and testing mode. We have split 80% for training and 20% test.

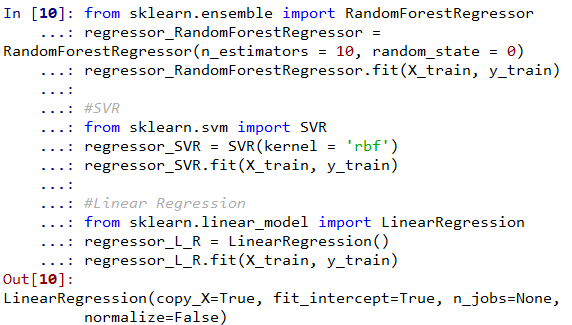


We have scale our data with standard scaler to normalize data

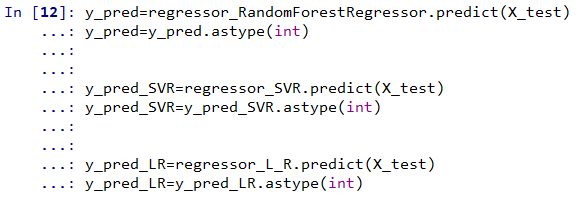


Training the model

We have used three different type of regression model to predict the outcome. We have used Random forest regressor, SVR and Linear Regression.

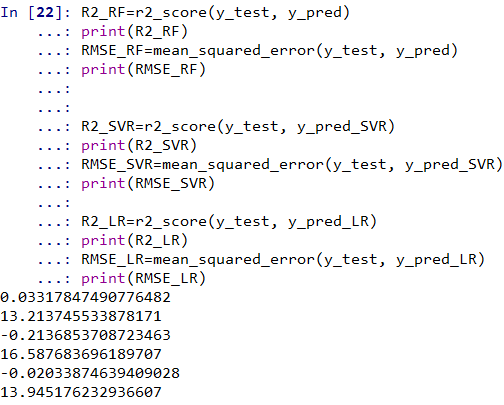


Once the model is train with three different type of regression. In next step, we will predict the outcomes using X\_test



## **4.3 Evaluation model**

We have used two parameters for evaluating our model i.e. R2 score and RMSE



As per above figure we can see that Random forest showing better performance in compare to SVR and Linear Regression.

## **4.4 K-Fold cross validation**

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

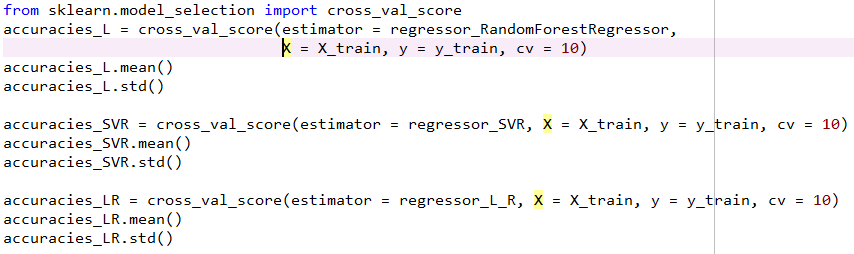
Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   * + 1. Take the group as a hold out or test data set
       2. Take the remaining groups as a training data set
       3. Fit a model on the training set and evaluate it on the test set
       4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.



## **Python Code**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Dec 25 10:00:08 2018

@author: Chintan

"""

# Importing Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from fancyimpute import KNN

from scipy import stats

# =============================================================================

# Read data

# =============================================================================

df = pd.read\_excel("Absenteeism\_at\_work\_Project.xls")

df.info()

# =============================================================================

# Variable's data type

# =============================================================================

df.dtypes

# =============================================================================

# Distributing Categorical and Continous variable

# =============================================================================

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']

# =============================================================================

# Missing value analysis and treatment

# =============================================================================

#Creating dataframe with missing values present in each variable

missing\_val = pd.DataFrame(df.isnull().sum()).reset\_index()

missing\_val

#Rename variable for missing value of dataframe

missing\_val = missing\_val.rename(columns = {'index': 'Variables', 0: 'Missing\_percentage'})

#Percentage finding

missing\_val['Missing\_percentage'] = (missing\_val['Missing\_percentage']/len(df))\*100

#Sort missing values

missing\_val = missing\_val.sort\_values('Missing\_percentage', ascending = False).reset\_index(drop = True)

missing\_val

# =============================================================================

# Imputing the missing value

# =============================================================================

# =============================================================================

# Dropping the observation, which have missing values in target variable

# =============================================================================

df = df.drop(df[df['Absenteeism time in hours'].isnull()].index, axis=0)

print(df['Absenteeism time in hours'].isnull().sum())

print(df.shape)

#Imputing with KNN imputation algorithm

df = pd.DataFrame(KNN(k = 3).fit\_transform(df), columns = df.columns)

#Check, if any variable has missing values

df.isnull().sum().sum()

# =============================================================================

# Outlier analysis for all features/variable

# Ploting BoxPlot of continuous variables

# =============================================================================

plt.boxplot(df['Transportation expense'])

plt.xlabel("'Transportation expense'")

plt.title("BoxPlot of Variable 'Transportation expense '")

plt.ylabel('Values')

plt.boxplot([df['Height']])

plt.xlabel("'Height'")

plt.title("BoxPlot of Variable of 'Height'")

plt.ylabel('Values')

plt.boxplot(df['Work load Average/day '])

plt.xlabel("Work load Average/day ")

plt.title("BoxPlot of Variable 'Work load Average/day '")

plt.ylabel('Values')

plt.boxplot([ df['Distance from Residence to Work'], df['Service time'], df['Age'], df['Hit target'], df['Weight'], df['Body mass index']])

plt.xlabel(['1. Distance from Residence to Work', '2. Service time', '3. Age', '4. Hit target', '5. Weight', '6. Body mass index'])

plt.title("BoxPlot of rest of the Variables")

plt.ylabel('Values')

# list of variables which doesn't have outlier

neglect = ['Distance from Residence to Work', 'Weight', 'Body mass index']

# Looping over all continuou variables to detect and remove Outliers

for i in continuous\_vars:

# Avoiding the variables which doesn't have outlier

if i in neglect:

continue

# Getting 75 and 25 percentile of variable "i"

q75, q25 = np.percentile(df[i], [75,25])

# Calculating Interquartile range

iqr = q75 - q25

# Calculating upper extream and lower extream

minimum = q25 - (iqr\*1.5)

maximum = q75 + (iqr\*1.5)

# Replacing all the outliers value to NA

df.loc[df[i]< minimum,i] = np.nan

df.loc[df[i]> maximum,i] = np.nan

# Imputing missing values with KNN

df = pd.DataFrame(KNN(k = 3).fit\_transform(df), columns = df.columns)

# Checking if there is any missing value

df.isnull().sum().sum()

# =============================================================================

# Correlation analysis for continuous variables

# =============================================================================

df.info()

df\_corr = df.loc[:,continuous\_vars]

#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()

# =============================================================================

# loop for ANOVA test Since the target variable is continuous

# =============================================================================

for i in categorical\_vars:

f, p = stats.f\_oneway(df[i], df["Absenteeism time in hours"])

print("P value for variable "+str(i)+" is "+str(p))

# Droping the variables which has redundant information

to\_drop = ['Weight','Age','Service time','ID']

df = df.drop(to\_drop, axis = 1)

# Updating the Continuous Variables and Categorical Variables after droping some variables

continuous\_vars = [i for i in continuous\_vars if i not in to\_drop]

categorical\_vars = [i for i in categorical\_vars if i not in to\_drop]

final\_data = df.copy()

figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

def check\_relation(x,y):

df = final\_data.groupby(x)[y].sum()

df = df.reset\_index()

df[y] = (df[y]\*100)/sum(final\_data[y])

df = df.sort\_values(by=['Absenteeism time in hours'])

print(df)

return df.plot.barh(x=x,y=y, figsize = (12,10))

# =============================================================================

# Checking the relationship between categorical independent variable and target variable

# =============================================================================

#Checking relationship between reason for absence vs Absenteeism time in hours

check\_relation('Reason for absence','Absenteeism time in hours')

#Checking relationship between Month of absence vs Absenteeism time in hours

check\_relation('Month of absence','Absenteeism time in hours')

#Checking relationship between Day of the week vs Absenteeism time in hours

check\_relation('Day of the week','Absenteeism time in hours')

#Checking relationship between Education vs Absenteeism time in hours

check\_relation('Education','Absenteeism time in hours')

#Checking relationship between Son vs Absenteeism time in hours

check\_relation('Son','Absenteeism time in hours')

#Forecasting for 2011

# =============================================================================

# As per data, there is no seperation for year wise. However, as per primary

# investigation, months are repated three times. Therefore, we can assume that given

# data is for three years

# =============================================================================

#Aggreate month of absence and depdnent variable

absence\_record\_monthly= final\_data.groupby('Month of absence')['Absenteeism time in hours'].sum()

absence\_record\_monthly

absence\_record\_monthly=absence\_record\_monthly.reset\_index()

absence\_record\_monthly

#Dividing dataset by 3

absence\_record\_monthly['Absenteeism hours per month'] = absence\_record\_monthly['Absenteeism time in hours']/3

absence\_record\_monthly['Month of absence'] = absence\_record\_monthly['Month of absence'].astype('int')

absence\_record\_monthly

#calling final data

final\_data

#dropping categorical data from

final\_data=final\_data.drop(categorical\_vars,axis=1)

# Importing the dataset

X = final\_data.iloc[:, :-1].values

y = final\_data.iloc[:, -1].values

#==============================================================================

# Feature ranking with recursive feature elimination

#==============================================================================

from sklearn.feature\_selection import RFE

from sklearn.svm import SVR

# feature extraction

estimator = SVR(kernel="linear")

selector = RFE(estimator, 2, step=1)

selector = selector.fit(X, y)

selector.support\_

selector.ranking\_

X=X[:,[4,5]]

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# =============================================================================

# Feature Scaling

# =============================================================================

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# =============================================================================

# Fitting Random Forest Regressor & SVR to X\_train and y\_train

# =============================================================================

from sklearn.ensemble import RandomForestRegressor

regressor\_RandomForestRegressor = RandomForestRegressor(n\_estimators = 10, random\_state = 0)

regressor\_RandomForestRegressor.fit(X\_train, y\_train)

#SVR

from sklearn.svm import SVR

regressor\_SVR = SVR(kernel = 'rbf')

regressor\_SVR.fit(X\_train, y\_train)

#Linear Regression

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 3)

X\_poly = poly\_reg.fit\_transform(X\_train)

regressor\_L\_R = LinearRegression()

regressor\_L\_R.fit(X\_poly, y\_train)

# =============================================================================

# preidicting test set results For Random Forest and SVR

# =============================================================================

y\_pred=regressor\_RandomForestRegressor.predict(X\_test)

y\_pred=y\_pred.astype(int)

y\_pred\_SVR=regressor\_SVR.predict(X\_test)

y\_pred\_SVR=y\_pred\_SVR.astype(int)

y\_pred\_pl\_r = regressor\_L\_R.predict(poly\_reg.fit\_transform(X\_test))

y\_pred\_pl\_r = y\_pred\_pl\_r.astype(int)

# =============================================================================

# Evaluating the model using R2 Score

# =============================================================================

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

R2\_RF=r2\_score(y\_test, y\_pred)

print(R2\_RF)

RMSE\_RF=mean\_squared\_error(y\_test, y\_pred)

print(RMSE\_RF)

R2\_SVR=r2\_score(y\_test, y\_pred\_SVR)

print(R2\_SVR)

RMSE\_SVR=mean\_squared\_error(y\_test, y\_pred\_SVR)

print(RMSE\_SVR)

R2\_LR=r2\_score(y\_test, y\_pred\_pl\_r)

print(R2\_LR)

RMSE\_LR=mean\_squared\_error(y\_test, y\_pred\_pl\_r)

print(RMSE\_LR)

# =============================================================================

# Fitting 10 fold cross validation for Random Forest & SVR

# =============================================================================

from sklearn.model\_selection import cross\_val\_score

accuracies\_L = cross\_val\_score(estimator = regressor\_RandomForestRegressor,

X = X\_train, y = y\_train, cv = 10)

accuracies\_L.mean()

accuracies\_L.std()

accuracies\_SVR = cross\_val\_score(estimator = regressor\_SVR, X = X\_train, y = y\_train, cv = 10)

accuracies\_SVR.mean()

accuracies\_SVR.std()

accuracies\_LR = cross\_val\_score(estimator = regressor\_L\_R, X = X\_train, y = y\_train, cv = 10)

accuracies\_LR.mean()

accuracies\_LR.std()

## **R Code**

rm(list = ls())

#For project, set working directory

setwd("E:/Subject/edwisor/project 3")

#Load required lib for project

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "e1071",

"DataCombine", "pROC")

library(utils)

install.packages("Rcpp")

library(readxl)

install.packages("readxl")

library(tibble)

#load required packages

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

# To read data from file, we have used readxl package

df = read\_excel("Absenteeism\_at\_work\_Project.xls")

#description of data

str(df)

#Renaming columns

colnames(df) = c("ID","Reason.for.absence","Month.of.absence","Day.of.the.week","Seasons","Transportation.expense"

,"Distance.from.Residence.to.Work","Service.time","Age","Work.load.Average.day","Hit.target","Disciplinary.failure"

,"Education","Son","Social.drinker","Social.smoker","Pet","Weight","Height","Body.mass.index","Absenteeism.time.in.hours")

#Analysis of missing values

missing\_value = (as.data.frame(colSums(is.na(df)))\*100/nrow(df))

colnames(missing\_value) <- c("Missng Value Percentage")

#Reason.for.absence Vs. Absenteeism.time.in.hours -- box plot

ggplot(df,aes\_string(y=df$Absenteeism.time.in.hours,x=as.factor(df$Reason.for.absence)))+geom\_boxplot()+xlab('Reason.for.absence')+ylab('Absenteeism.time.in.hours')

#Impute the values in dataset columns

df$Reason.for.absence[is.na(df$Reason.for.absence)] = 27

#Zero category of 'Reason for absence' value has equal to 26

df$Reason.for.absence[df$Reason.for.absence==0] = 26

#Putting Month.of.absence null value equal to 10.

df$Month.of.absence[is.na(df$Month.of.absence)] = 10

#Finding ID column values for which there are missing values in Transportation.expense

df$ID[is.na(df$Transportation.expense)]

for (i in c(1,3,10,15,20,22)){

df$Transportation.expense[is.na(df$Transportation.expense) & df$ID==i] = mean(df$Transportation.expense[df$ID==i],na.rm = T)

}

#to find rows for missing values

df$ID[is.na(df$Distance.from.Residence.to.Work)]

#As a return, we got 34,22 abd 28

for (i in c(34,22,28)){

df$Distance.from.Residence.to.Work[is.na(df$Distance.from.Residence.to.Work) & df$ID==i] = mean(df$Distance.from.Residence.to.Work[df$ID==i],na.rm = T)

}

#Find missing values value in ID column

df$ID[is.na(df$Service.time)]

# As return we got 28, 34

for (i in c(34,28)){

df$Service.time[is.na(df$Service.time) & df$ID==i] = mean(df$Service.time[df$ID==i],na.rm = T)

}

#Find missing values value in age

df$ID[is.na(df$Age)]

# [1] 28 24 24

for (i in c(24,28)){

df$Age[is.na(df$Age) & df$ID==i] = mean(df$Age[df$ID==i],na.rm = T)

}

#Converting into numeric values

df$Work.load.Average.day = as.numeric(df$Work.load.Average.day)

#Work.load.Average.day missing values are imputed using Month.of.absence and Hit.target

df$Month.of.absence[is.na(df$Work.load.Average.day)]

df$Hit.target[is.na(df$Work.load.Average.day)]

df = data.frame(m=c(9,10,11,11,12,12,1,1,1,5),h=c(92,93,93,93,97,97,95,95,95,92))

for (i in 1:10){

df$Work.load.Average.day[(is.na(df$Work.load.Average.day) &

df$Month.of.absence==df[i,1]) & df$Hit.target==df[i,2]] =

mean(df$Work.load.Average.day[df$Month.of.absence==df[i,1] & df$Hit.target==df[i,2]],na.rm = T)

}

#Impute value of hit target by month of absence and work load average day

df$Month.of.absence[is.na(df$Hit.target)]

df$Work.load.Average.day[is.na(df$Hit.target)]

final\_data = data.frame(m1=c(11,12,1),w1=c(306345,261306,308593))

for (i in 1:3){

df$Hit.target[(is.na(df$Hit.target) & df$Month.of.absence==final\_data[i,1]) & df$Work.load.Average.day==final\_data[i,2]] = mean(df$Hit.target[df$Month.of.absence==final\_data[i,1] & df$Work.load.Average.day==final\_data[i,2]],na.rm = T)

}

df$Disciplinary.failure[is.na(df$Disciplinary.failure)] = 0

#For education missing value for impute, we used id value again

df$ID[is.na(df$Education)]

# [1] 11 10 34 34 14 34 34 34 10 24

for (i in c(10,11,14,24,34)){

df$Education[is.na(df$Education) & df$ID==i] = mean(df$Education[df$ID==i],na.rm=T)

}

#For son missing value, use id value

df$ID[is.na(df$Son)]

for (i in c(1,14,20,27,34)){

df$Son[is.na(df$Son) & df$ID==i] = mean(df$Son[df$ID==i],na.rm=T)

}

df$ID[is.na(df$Social.drinker)]

for (i in c(10,14,17)){

df$Social.drinker[is.na(df$Social.drinker) & df$ID==i] = mean(df$Social.drinker[df$ID==i],na.rm=T)

}

#For social smoker the id column has used

df$ID[is.na(df$Social.smoker)]

for (i in c(34,1,11,15)){

df$Social.smoker[is.na(df$Social.smoker) & df$ID==i] = mean(df$Social.smoker[df$ID==i],na.rm=T)

}

#For fill pet missing value id column has used

df$ID[is.na(df$Pet)]

for (i in c(1,13)){

df$Pet[is.na(df$Pet) & df$ID==i] = mean(df$Pet[df$ID==i],na.rm=T)

}

#For weight missing value, id has used

df$ID[is.na(df$Weight)]

for (i in c(27)){

df$Weight[is.na(df$Weight) & df$ID==i] = mean(df$Weight[df$ID==i],na.rm=T)

}

df$ID[is.na(df$Height)]

for (i in c(20,10,28,34,27,11,5,22,13,24,32)){

df$Height[is.na(df$Height) & df$ID==i] = mean(df$Height[df$ID==i],na.rm=T)

}

df$ID[is.na(df$Body.mass.index)]

for (i in c(3,24,11,30,2,19,34,28,13,36,14,20,18,17,15,22,5)){

df$Body.mass.index[is.na(df$Body.mass.index) & df$ID==i] = mean(df$Body.mass.index[df$ID==i],na.rm=T)

}

df$Reason.for.absence[is.na(df$Absenteeism.time.in.hours)]

for (i in c(23,14,10,22,26,6,28,11,13)){

df$Absenteeism.time.in.hours[is.na(df$Absenteeism.time.in.hours) & df$Reason.for.absence==i] = mean(df$Absenteeism.time.in.hours[df$Reason.for.absence==i],na.rm=T)

}

# Distribution for Continuous Variables

#First ,Transportation expense

hist(df$Transportation.expense,prob = TRUE,xlab = 'Transportation.expense')

lines(density(df$Transportation.expense))

# second, Distance from Residence to Work

hist(df$Distance.from.Residence.to.Work,prob = TRUE,xlab = 'Distance.from.Residence.to.Work')

lines(density(df$Distance.from.Residence.to.Work))

#third, Service time

hist(df$Service.time,prob = TRUE,xlab = 'Service.time')

lines(density(df$Service.time))

#Age

hist(df$Age,prob = TRUE,xlab = 'Age')

lines(density(df$Age))

#For, Work load Average day

hist(df$Work.load.Average.day,prob = TRUE,xlab = 'Work.load.Average.day')

lines(density(df$Work.load.Average.day))

#next, Hit target

hist(df$Hit.target,prob = TRUE,xlab = 'Hit.target')

lines(density(df$Hit.target))

#Also,Weight

hist(df$Weight,prob = TRUE,xlab = 'Weight')

lines(density(df$Weight))

#Height

hist(df$Height,prob = TRUE,xlab = 'Height')

lines(density(df$Height))

#Body.mass.index

hist(df$Body.mass.index,prob = TRUE,xlab = 'Body.mass.index')

lines(density(df$Body.mass.index))

#Independent variable's Outlier analsis

num\_col =c('Weight', 'Height', 'Body.mass.index','Absenteeism.time.in.hours','Transportation.expense',

'Distance.from.Residence.to.Work', 'Service.time', 'Age','Hit.target','Work.load.Average.day')

cat\_col = c('')

for (i in 1:length(num\_col))

{

assign(paste0("gn",i),ggplot(aes\_string(y = (num\_col[i]), x = 'Absenteeism.time.in.hours'),data = df) +

stat\_boxplot(geom = "errorbar", width = 0.5) +geom\_boxplot(outlier.colour="blue", fill = "skyblue",

outlier.shape=18,outlier.size=1, notch=FALSE) +labs(y=num\_col[i],x="Absenteeism in Hours")+

ggtitle(paste("Box plot of responded for",num\_col[i])))

}

# Plotting

gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)

gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)

gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)

#converting into a dataframe

df = as.data.frame(df)

for (i in c('Transportation.expense','Service.time','Age','Work.load.Average.day','Hit.target','Height','Absenteeism.time.in.hours')){

qnt = quantile(df[,i], probs=c(.25, .75), na.rm = T)

iqr1 = qnt[2]-qnt[1]

min1 = qnt[1]-1.5\*iqr1

max1 = qnt[2]+1.5\*iqr1

df[,i][df[,i]<min1] = min1

df[,i][df[,i]>max1] = max1

}

#Correlation Analysis of variables and Convert categorical into catcols

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

for (i in catcols){

df[,i] = as.factor(df[,i])

}

str(df)

#Measur of Chi-square test

pval = c()

#Calculating & storing p-values in vector pval from chisquare test

for(i in catcols){

for(j in catcols){

chi2 = chisq.test(df[,i],df[,j])

pval = c(pval,chi2$p.value)

}

}

length(pval)#100

#convertiino matrix m1

m1 = matrix(pval,ncol=10)

#Convert m1 to dataframe chi\_df

chi\_df = data.frame(m1)

#catcols to row names

row.names(chi\_df) = catcols

#catcols - column names

colnames(chi\_df) = catcols

#As per values <0.05. Many column doesn't fit into the criteria. SO all will drop out from that

df[,c('Month.of.absence','Seasons','Disciplinary.failure','Education','Son','Social.drinker','Social.smoker','Pet')] = list(NULL)

#continuous independent variables correlation

cor(df[,4:13])

#As per values, there is no relation between independent and dependent variable(i<0.95 and d ,0.2)

#Therefore, aggreate dependent variablen and reason for absence

Reasons = aggregate(df$Absenteeism.time.in.hours, by=list(Category=df$Reason.for.absence), FUN=sum)

Reasons$Absence = (Reasons$x/sum(df$Absenteeism.time.in.hours))\*100

Reasons = Reasons[order(Reasons$Absence),]

Reasons

barplot(Reasons$Absence,names.arg=Reasons$Category,xlab="Reason.for.absence",ylab="Absence",col="blue")

#Again read data file

final\_data = read\_excel('Absenteeism.xls')

#Same, rename to column names

colnames(final\_data) = c("ID","Reason.for.absence","Month.of.absence","Day.of.the.week","Seasons","Transportation.expense"

,"Distance.from.Residence.to.Work","Service.time","Age","Work.load.Average.day","Hit.target","Disciplinary.failure"

,"Education","Son","Social.drinker","Social.smoker","Pet","Weight","Height","Body.mass.index","Absenteeism.time.in.hours")

#Impute missing values as required

final\_data$Month.of.absence[is.na(final\_data$Month.of.absence)] = 10

final\_data$Reason.for.absence[is.na(final\_data$Reason.for.absence)] = 27

final\_data$Reason.for.absence[final\_data$Reason.for.absence==0] = 26

for (i in c(23,14,10,22,26,6,28,11,13)){

final\_data$Absenteeism.time.in.hours[is.na(final\_data$Absenteeism.time.in.hours) & final\_data$Reason.for.absence==i] = median(final\_data$Absenteeism.time.in.hours[final\_data$Reason.for.absence==i],na.rm=T)

}

#Converting Month.of.absence to factor

final\_data$Month.of.absence = as.factor(final\_data$Month.of.absence)

#Making a timeseries aggregating Absenteeism.time.in.hours by Month.of.absence

monthly\_absence = aggregate(final\_data$Absenteeism.time.in.hours,by=list(Category=final\_data$Month.of.absence),FUN=sum)

monthly\_absence = monthly\_absence[2:13,]

monthly\_absence

#Calculating dfeeism time as percent of total time in column dfhours

monthly\_absence$dfhours = monthly\_absence$x/3

row.names(monthly\_absence) = monthly\_absence$Category

monthly\_absence

# Modelling time series using arima

tsdata = ts(monthly\_absence$dfhours)

class(tsdata)

#Display timeseries data

plot(tsdata)

#Checking stationarity - Augmented Dickey-Fuller Test

library(tseries)

adf.test(tsdata, alternative="stationary", k=0)

# Augmented Dickey-Fuller Test

#Subtract shifted time series from original time series.

tsdata2 = tsdata - stats::lag((tsdata),1)

plot(tsdata2)

#Dickey-Fuller Test again

adf.test(tsdata2, alternative="stationary", k=0)

#plot for ACF

acf(tsdata2)

#plot for PACF

pacf(tsdata2)

library(forecast)

model = arima(tsdata2,c(4,0,9))

fit1 = fitted(model)

residuals1 = tsdata2 - fit1

sum(residuals1\*\*2)

plot(tsdata2)

lines(fit1)

df2011 = predict(model,n.ahead = 12)

#Scaling df2011 back to original

absence\_2011 = cumsum(df2011$pred)

absence\_2011\_2 = absence\_2011 + rep(tsdata[4],12)

as.data.frame(absence\_2011\_2)

ts\_2011 = ts(absence\_2011\_2)

final\_data = as.data.frame(absence\_2011\_2)

row.names(final\_data) = c(13:24)

ts\_2011 = ts(final\_data$absence\_2011\_2,start=13)

#Display original timeseries & predicted values

plot(tsdata,xlim=c(1,24))

lines(ts\_2011)