Predicting Absenteeism time of employees

Name: Deepanshu Tambi

All the data, graphs, charts, and results generated in the project are coded either using python jupyter or R Studio.

Contents

Chapter 1: Introduction

1.1 PROBLEM STATEMENT:

XYZ is a courier company. As we appreciate that human capital plays an important role in collection, transportation and delivery. The company is passing through genuine issue of Absenteeism. The company has shared its dataset and has requested to have an answer on the following areas:

1. What changes company should bring to reduce the number of absenteeism?
2. How much loss every month can we project in 2011 if same trend of absenteeism continues?

1.2 OVERVIEW OF THE DATASET:

	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
1	11	26.0	7.0	3	1	289.0	36.0	13.0	33.0	239554.0	97.0
2	36	0.0	7.0	3	1	118.0	13.0	18.0	50.0	239554.0	97.0
3	3	23.0	7.0	4	1	179.0	51.0	18.0	38.0	239554.0	97.0
4	7	7.0	7.0	5	1	279.0	5.0	14.0	39.0	239554.0	97.0
5	11	23.0	7.0	5	1	289.0	36.0	13.0	33.0	239554.0	97.0

Fig: 1.1 Columns 1to 11 of the dataset

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

Fig 1.2 Columns 11 to 21 of the dataset

As we can see, there are 8 categorical variables and 12 numeric variables. The dependent variable or the target variable is Absenteeism time in hours, i.e. the number of hours employees remaining absent.

The details of data attributes in the dataset are as follows -

1. Individual identification (ID)

2. Reason for absence (ICD).

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

I Certain infectious and parasitic diseases

II Neoplasms

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

IV Endocrine, nutritional and metabolic diseases

V Mental and behavioural disorders

VI Diseases of the nervous system

VII Diseases of the eye and adnexa

VIII Diseases of the ear and mastoid process

IX Diseases of the circulatory system

X Diseases of the respiratory system

XI Diseases of the digestive system

XII Diseases of the skin and subcutaneous tissue

XIII Diseases of the musculoskeletal system and connective tissue

XIV Diseases of the genitourinary system

XV Pregnancy, childbirth and the puerperium

XVI Certain conditions originating in the perinatal period

XVII Congenital malformations, deformations and chromosomal abnormalities

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

XIX Injury, poisoning and certain other consequences of external causes

XX External causes of morbidity and mortality

XXI Factors influencing health status and contact with health services.

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

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

Chapter 2: Exploratory Data Analysis (EDA)

2.1 Missing Value Analysis:

ID	0
Reason for absence	3
Month of absence	1
Day of the week	0
Seasons	0
Transportation expense	7
Distance from Residence to Work	3
Service time	3
Age	3
Work load Average/day	10
Hit target	6
Disciplinary failure	6
Education	10
Son	6
Social drinker	3
Social smoker	4
Pet	2
Weight	1
Height	14
Body mass index	31
Absenteeism time in hours	22

Fig: 2.1 Missing values in variables of the raw dataset

Fig 2.1 shows the missing values in variables of the raw dataset, mostly the missing values are less than 10 in all variables except Height, Body mass index, and Absenteeism time.

Few insights on the missing values:

1] Taking all the rows containing one or more than one NA came out to be 101. Thus, although there were 145 missing values in total, number of rows came out to be 101 because some rows had more than 1 NA.

2] Dropping more than 100 rows in an 800+ rows dataset was not feasible, so performed

correlation analysis beforehand to check multi collinearity. The result showed Body mass index was redundant variable which had the most missing values, so dropped it before any imputation task. Now, it was fine to drop all NAs since BMI had gone, and there were no missing values in absenteeism time in hours as this was our target variable and all NAs were deleted from it, because it is not good to impute values in target variable as advised, this ultimately made us left with only few missing values in 20s. Although this was an option, but all the proceedings are done by imputing other variables and dropping BMI and missing rows of target variable.

***NOTE**: The imputation of missing values was done through KNN in Python, and through MICE in R.*

2.2 Visualization of the data:

We cannot undermine the fact that, though simple, barplots present the best and swiftest results in getting us the feel of dataset. Thus, will be seeing important variables' plots, mostly with our target variable. The result of graphs can change to an extent that it can lead to changes in answer if we use imputed/processed dataset for plotting graphs.

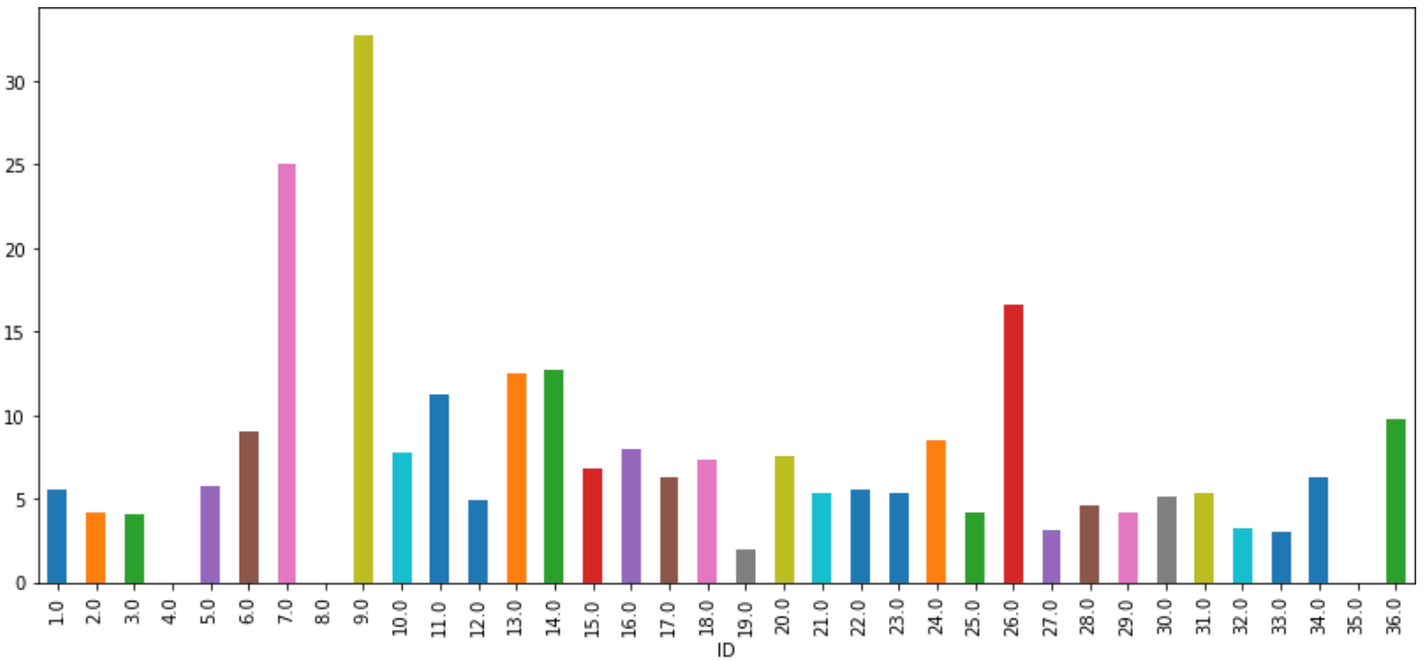

Fig: 2.2a Mean absenteeism hours for each ID

The figure 2.2a shows the mean of hours that persons with ID following from 1 to 36

were absent for the whole period. It shows that person with ID: 9 and 7 had the most number of hours remaining as absent, while ID:4, 8, 35 had the least or Zero number of hours of absenteeism.

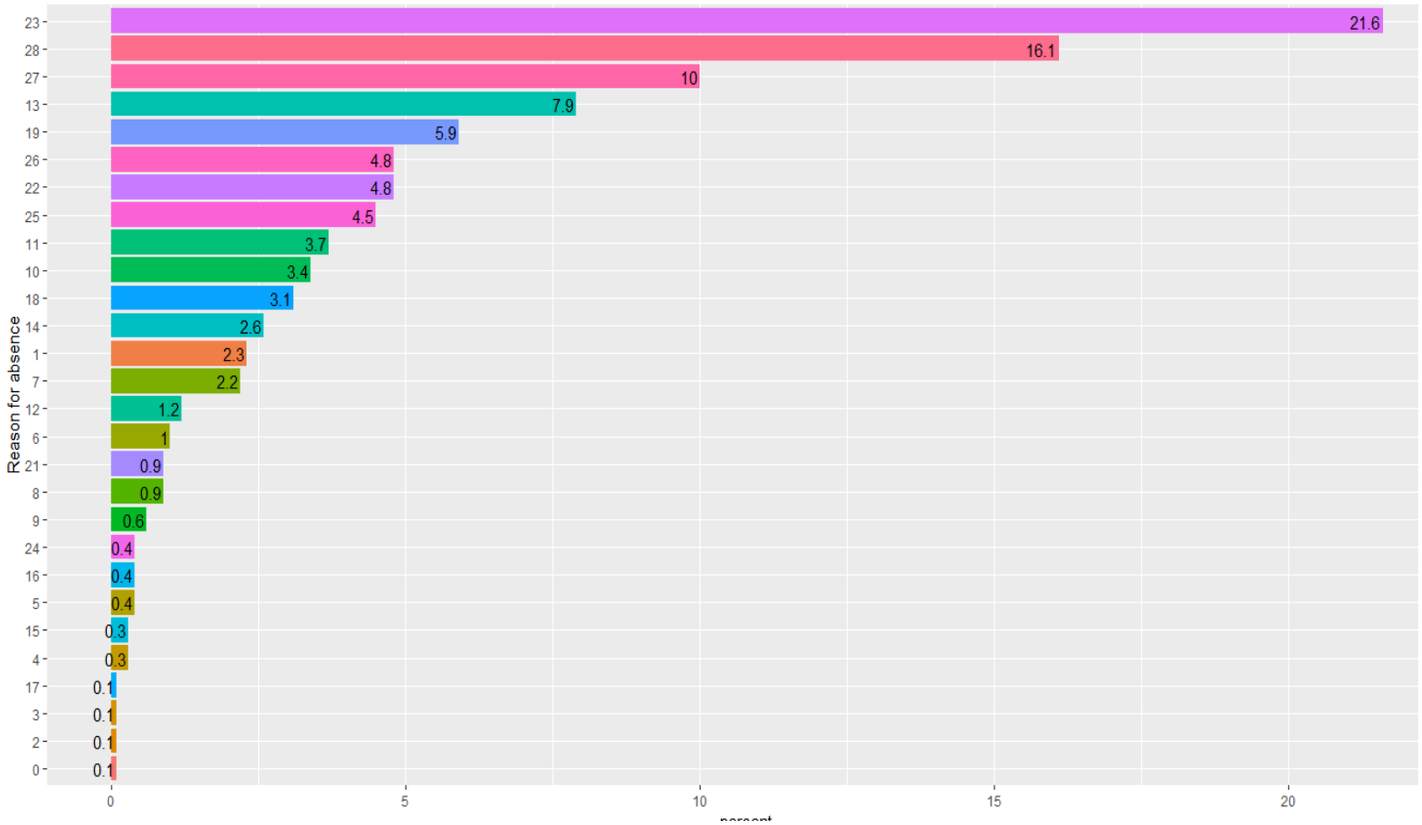

Fig: 2.2b(i) Percentage of the reason of absence used by employees

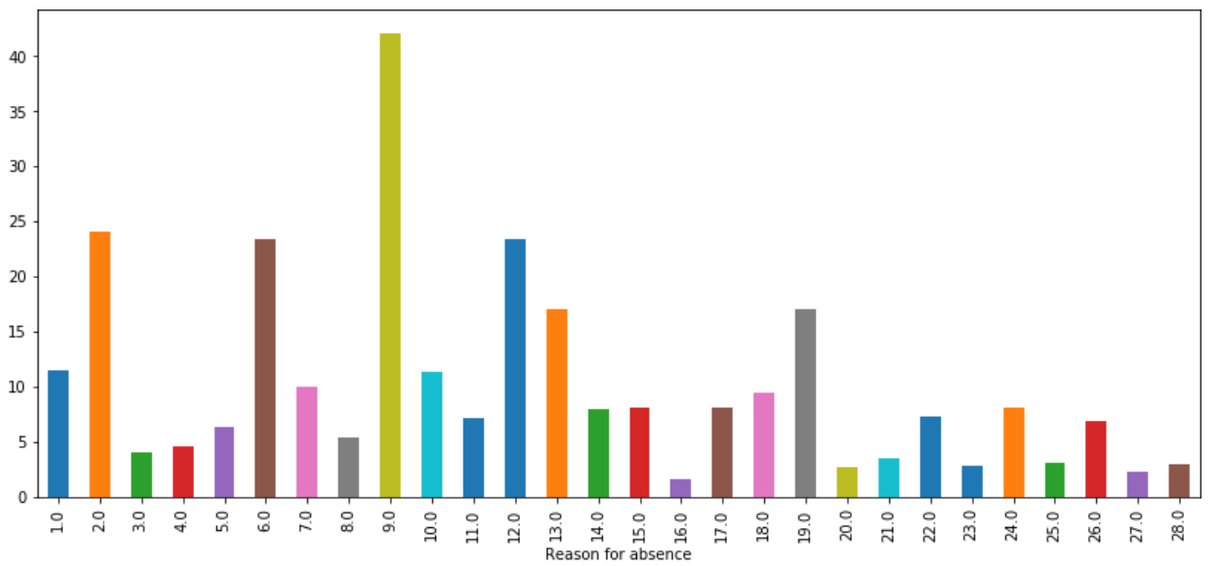

Fig: 2.2b(ii) Mean absenteeism hours for each reason of absence

Fig 2.2a(i) shows the distribution of reason of absence in percentages used by employees over the whole period. It shows that reason 23(medical consultation) is the most prominent reason amongst all, as it is the most evident reason and the most generic one, apart from this 4.8 percent of employees haven't stated their reason of absenteeism which is reason 26(unjustified absence). The least reason stated by employees for being absent are 3 (Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism), 2(Neoplasms) etc. these reasons are also anyhow very uncommon to see in our daily life. Note that, there is 0.1 percent of reason of absence number 0, which should not have been there was removed later, thus by careful scrutinization of each and every variable, all the rows of all variables were imputed which had absurd values.

Fig 2.2b(ii) shows the mean of number of hours that employees have been absent for particular reason. It shows that reason 9(Diseases of the circulatory system) has the most longing effect on the number of hours of absenteeism on employees, and the least ones were 16 (Certain conditions originating in the perinatal period), 21(External causes of morbidity and mortality) etc.

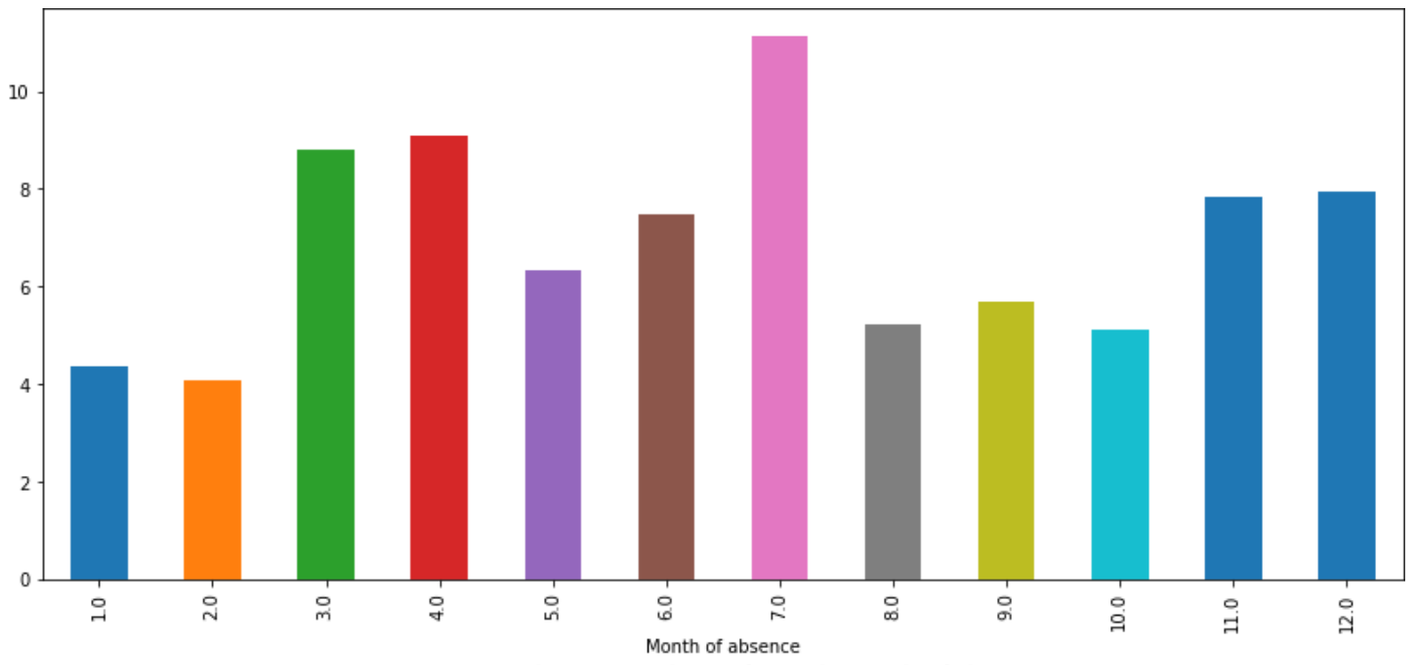

Fig: 2.2c Mean absenteeism hours for each Month of absence

Fig 2.2c shows the mean of absenteeism hours spread over months. It suggests that most number of skipping of hours is seen in the month of July followed by march and so on, and the least being in January.

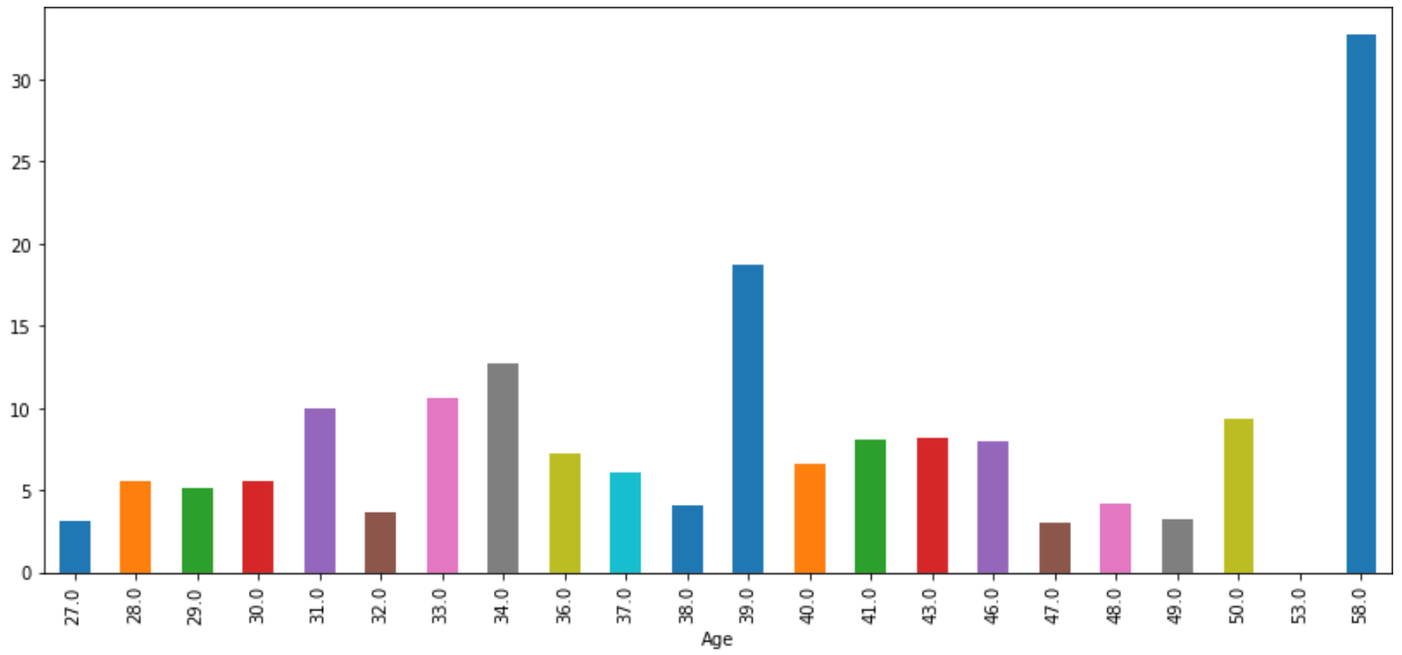

Fig: 2.2d Mean absenteeism hours for each Age Value

Fig 2.2d shows the mean of absenteeism hours spread over different Aged people. It suggests that most number of skipping of hours is seen in elderly people, while the youngest lot seems good in remaining present.

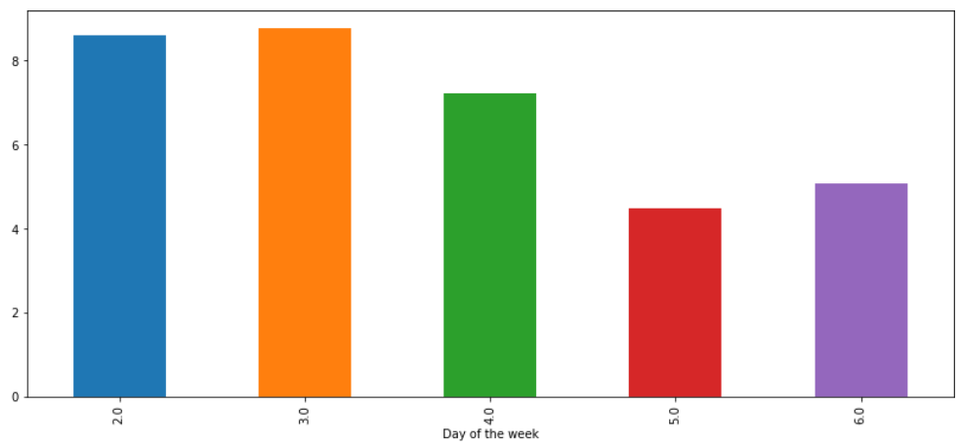

Fig: 2.2e Mean absenteeism hours for each reason of absence

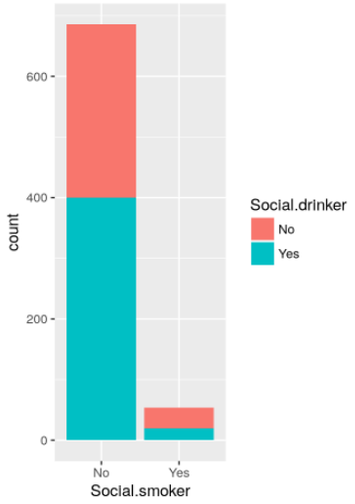

Fig: 2.2f shows the count of instances of smoking and drinking

Fig 2.2e shows that Monday and Tuesday had the most amount of absenteeism hours, while

Thursday, middle day of the week, had the least hours of absenteeism.

Fig 2.2f shows that around 50% of people are social drinkers, while very less percentage of people ~10% are social smoker.

2.3 Feature Selection

For selecting the feature and understanding the inter-relationships between the variables, we performed three tests.

2.3.1 Correlation analysis:

Fig: 2.3a shows the correlation analysis table where we can deduce that Body Mass Index and Weight are highly correlated, also service time and age are somewhat positively correlated. The variable that gets dropped is Body Mass index, since it had the most number of missing values, thus helping us in reducing the pain of filling artificial data.

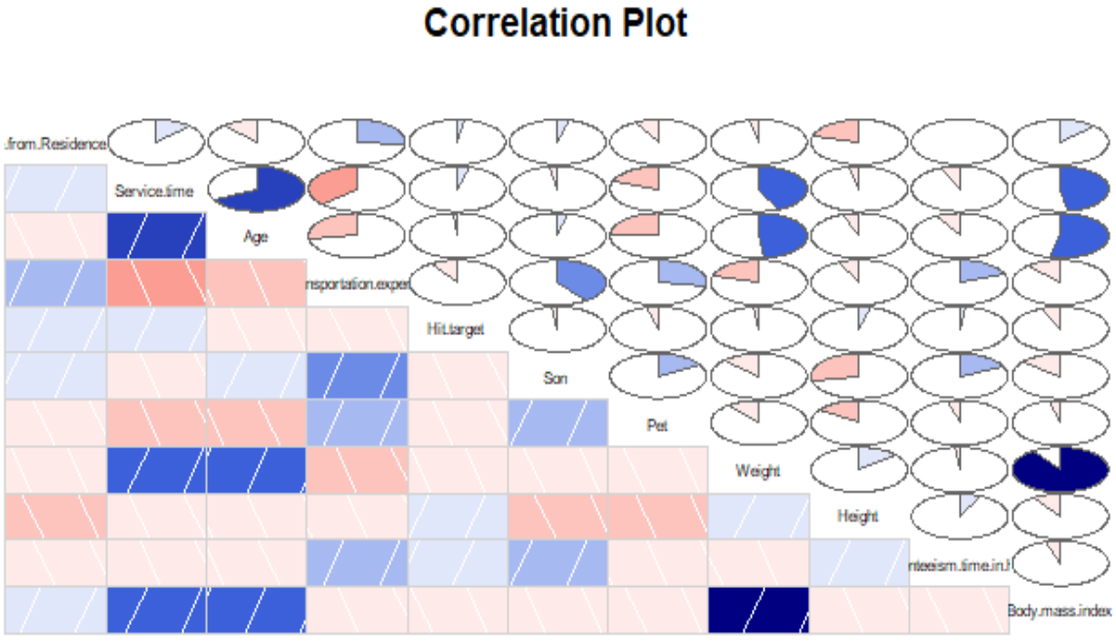

Fig: 2.3a Correlation analysis table heatmap

2.3.2 VIF:

VIF helps in quantifying the extent of correlation between one predictor and the other predictors in a model. It helps not only in determining collinearity with one variable, but helps in making out correlation of one variable with more than one variable.

Distance from Residence to Work	1.681130
Service time	3.371883
Age	2.424660
Work load Average/day	1.049937
Transportation expense	1.597151
Hit target	1.043096
Son	1.255561
Pet	1.580640
Weight	157.814366
Height	28.786233
Body mass index	147.832865
Absenteeism time in hours	1.048464

Fig 2.3b(i) VIF before variable selection

Distance from Residence to Work	1.600629
Service time	3.240114
Age	2.306970
Work load Average/day	1.048436
Transportation expense	1.591382
Hit target	1.042957
Son	1.251408
Pet	1.509672
Weight	1.645911
Height	1.484519
Absenteeism time in hours	1.046549

Fig 2.3b(ii) VIF after variable selection

Fig 2.3b(i) shows the VIF values of different variables before removing Body Mass Index, notice that Weight, Height and BMI are above 10, suggesting that these three have some connection between them which cannot be seen by correlation plot. Also, BMI and weight have the most prominent connection, which can also be concluded by r value of correlation analysis between BMI and Weight. In Fig 2.3b(ii), it can be seen that when BMI is removed all the inflated variation settles down.

2.3.3 Using Extra-tree Regressor:

The Extra-Tree method (extremely randomized trees method), its main objective is to further randomizing tree building in the context of input features. This method is quite similar to random forest, but unlike random forests, in Extra tree regressor splits are selected in random instead of using some criteria like in Random Forest.

Fig 2.3c shows that reason for absence, day of the week and work load are amongst the main features in deciding absenteeism, while variables like education and social smoker doesn't really helps in determining our target variable. However, we would include all the variables except BMI since the variables are less and are carrying at least some importance in determining our target variable

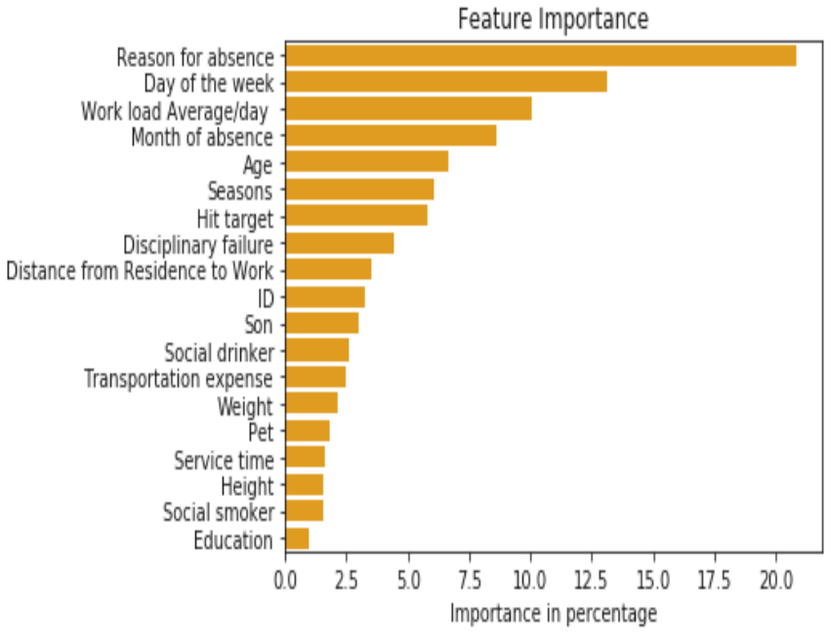

Fig 2.3c Importance of variable in percentage using Extra trees regresssor

2.4 Outlier Analysis

Outlier detection and treatment is always a tricky part especially when our dataset is small. The box plot method detects outlier if any value is present greater than (Q3 + (1.5 * IQR)) or less than (Q1 – (1.5 * IQR)).

Fig 2.4a shows the boxplot of all numerical variables, showcasing us the distribution of the data across.

Fig 2.4b shows the number of outliers for each variable. Now, by seeing all the variables individually and checking the nature of outliers, it was observed that most of the outliers like height, number of pets, age etc. were feasible data, so only imputed absenteeism in hours variable by KNN and by MICE method in RStudio.

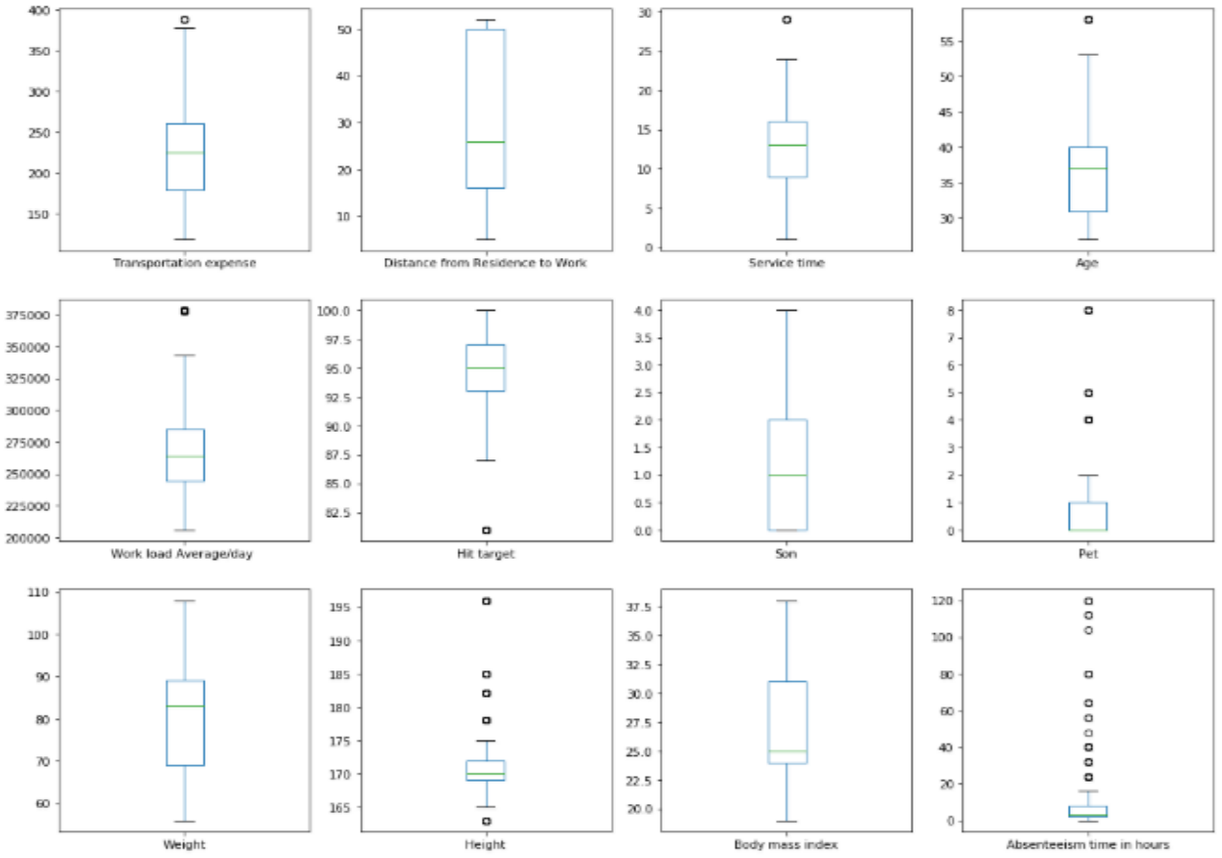

Fig: 2.4a Box Plot of all numerical variables

	0
ID	0
Reason for absence	0
Month of absence	0
Day of the week	0
Seasons	0
Transportation expense	3
Distance from Residence to Work	0
Service time	5
Age	8
Work load Average/day	29
Hit target	19
Disciplinary failure	0
Education	0
Son	0
Social drinker	0
Social smoker	0
Pet	42
Weight	0
Height	114
Absenteeism time in hours	43

Fig: 2.4b Outlier data information for each variable

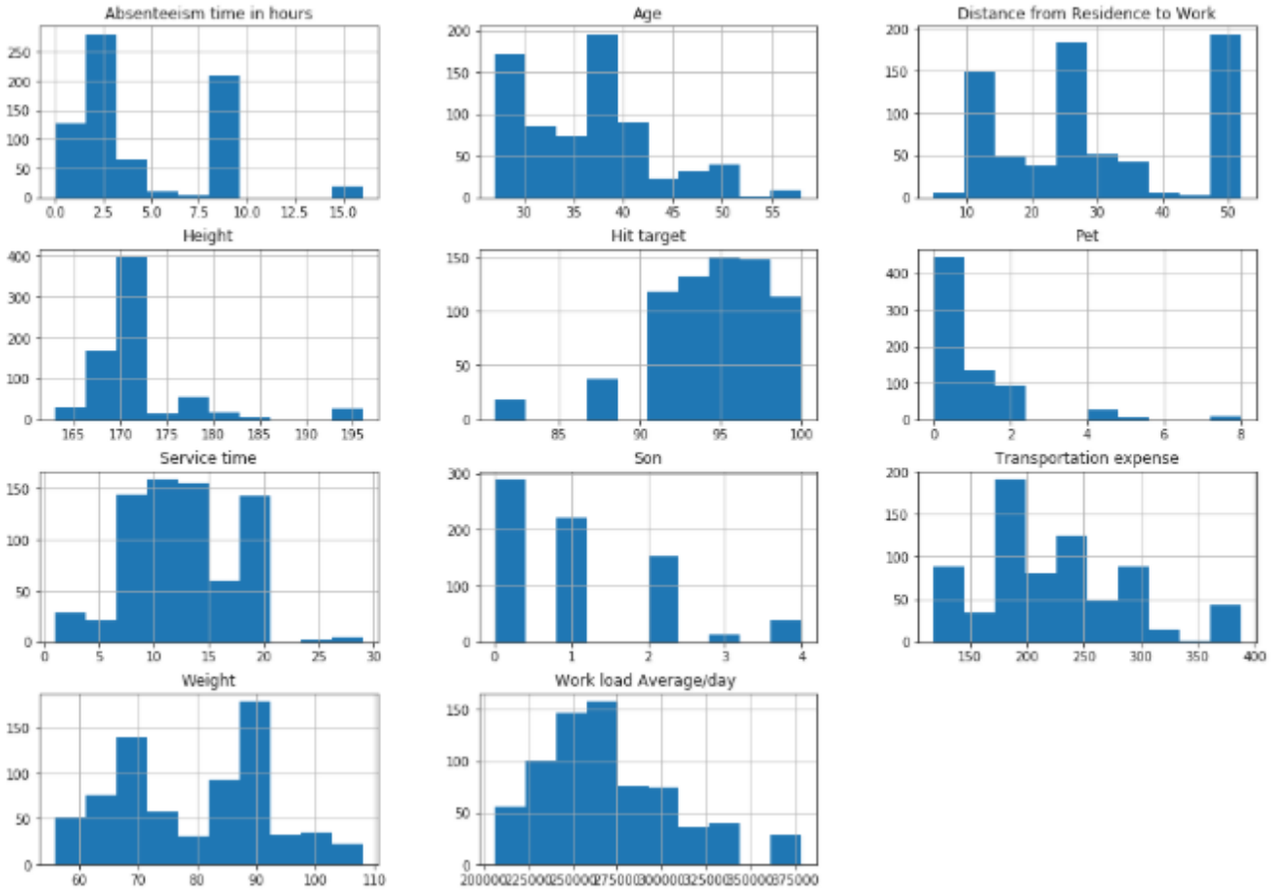

Fig: 2.4d Histograms of all numerical variables after outlier analysis

Fig. 2.4d shows the histograms plot of all numerical features suggesting that none of the variable is following normal distribution, and also by transformation the curve shape didn't change, thus going by original values only and no transformation.

2.5 Feature Scaling:

It was observed that many numerical variables had different ranges of value, some huge and some small, so just to stay assured that large values don't dominate our model, fature scaling was performed and all variables' values were brought in between 0 to 1 by using the below formula.

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

Chapter 3: Model Development

All the models were developed and tested to see their performance on data. The performance metric used was RMSE, as it is the most widely used performance metric, and helps in getting or visualizing deviations easily. Although, MAPE is also a good metric and it provides good a visual enticement to the client as it is very easy to read and work upon.

In all the models, Result-train RMSE means RMSE value obtained when the model is applied to the data from which it was created. Result-test RMSE means RMSE value obtained when the model is applied to different dataset(20%) of the bigdataset.

3.1 Linear Regression:

In Fig 3.1a, the summary of linear regression model is show. The adjusted R-square value(~0.79) is pretty good, and most of our data is explaining the target variable. All other parameters also seem to be pretty good from the OLS Summary table.

```
================================================
================================================
Linear Regression
Result-test
RMSE: 3.480437989373406
Result-train
RMSE: 2.3221722321368317
================================================
================================================
```

The train and test RMSE are 2.32 and 3.48 respectively, which are fine, and the model is overall acceptable with good parameters' value.

OLS Regression Results

Dep. Variable:	Absenteeism time in hours	**R-squared:**	0.807
Model:	OLS	**Adj. R-squared:**	0.784
Method:	Least Squares	**F-statistic:**	35.06
Date:	Fri, 28 Sep 2018	**Prob (F-statistic):**	2.46e-146
Time:	20:02:15	**Log-Likelihood:**	-1302.1
No. Observations:	574	**AIC:**	2726.
Df Residuals:	513	**BIC:**	2992.
Df Model:	61		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Transportation expense	1.3097	0.621	2.108	0.036	0.089	2.531
Distance from Residence to Work	0.0701	0.562	0.125	0.901	-1.034	1.174
Service time	0.1185	1.296	0.091	0.927	-2.427	2.664
Age	-1.3471	0.797	-1.690	0.092	-2.913	0.219
Work load Average/day	1.6122	0.573	2.815	0.005	0.487	2.737
Hit target	1.8350	0.839	2.188	0.029	0.188	3.482
Son	0.8702	0.491	1.773	0.077	-0.094	1.834
Pet	-1.3783	0.899	-1.533	0.126	-3.145	0.388

Weight	1.1267	0.665	1.693	0.091	-0.181	2.434
Height	-1.8893	0.859	-2.199	0.028	-3.577	-0.202
Reason for absence_2.0	-0.7065	2.666	-0.265	0.791	-5.945	4.532
Reason for absence_3.0	2.1236	2.626	0.809	0.419	-3.035	7.283
.....						
Education_4.0	1.1559	1.351	0.855	0.393	-1.499	3.811
Social drinker_1.0	0.4152	0.364	1.140	0.255	-0.300	1.131
Social smoker_1.0	0.2872	0.554	0.518	0.604	-0.802	1.376

Omnibus:	149.241	**Durbin-Watson:**	1.916
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	594.890
Skew:	1.135	**Prob(JB):**	6.62e-130
Kurtosis:	7.441	**Cond. No.**	62.0

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Fig 3.1a Summary of linear regression model

3.2 Ridge and Lasso Regression:

Ridge or lasso are forms of regularized linear regressions. Lasso calculates its value by reducing the power of unnecessary variables or least important variables, while ridge on the other hand tends to bring all variables' contribution in building models. Both of these are generally used when we want to use a more complex or when we have to overcome overfitting issues from linear model. Anyways, since we did not have much issue with all the above stated problems in linear model, it is ok to pick linear model as a better choice compared to these two. Nonetheless, ridge has definitely performed well here and is almost equivalent to linear model in terms of performance.

```
==========================================================
==========================================================
Ridge
Result-test
RMSE: 3.4796849319660326
Result-train
RMSE: 2.334055816414068
==========================================================
==========================================================
Lasso
Result-test
RMSE: 4.058357539349357
Result-train
RMSE: 3.232807836883631
==========================================================
==========================================================
```

3.3 SVR:

Support Vector Regressor(Gausian Kernel) has a test value of ~4, hence the model is not apt for building in our dataset.

```
==========================================================
==========================================================
SVR
Result-test
RMSE: 4.040058113370883
Result-train
RMSE: 2.993197635737354
==========================================================
==========================================================
```

3.4 KNN:

The test value RMSE for this model is ~3.56, which is fine as compared to other models, so to c heck the variation of result by varying the number of neighbors, a graph is plotted between RMS E value and number of neighbors in Fig.3.4b. According to figure, k = 5(which luckily is also th e default value) seems the best, with RMSE of ~3.56

```
================================================
================================================
KNeighborsRegressor
Result-test
RMSE: 3.566121453655529
Result-train
RMSE: 2.4403261334564816
================================================
================================================
```

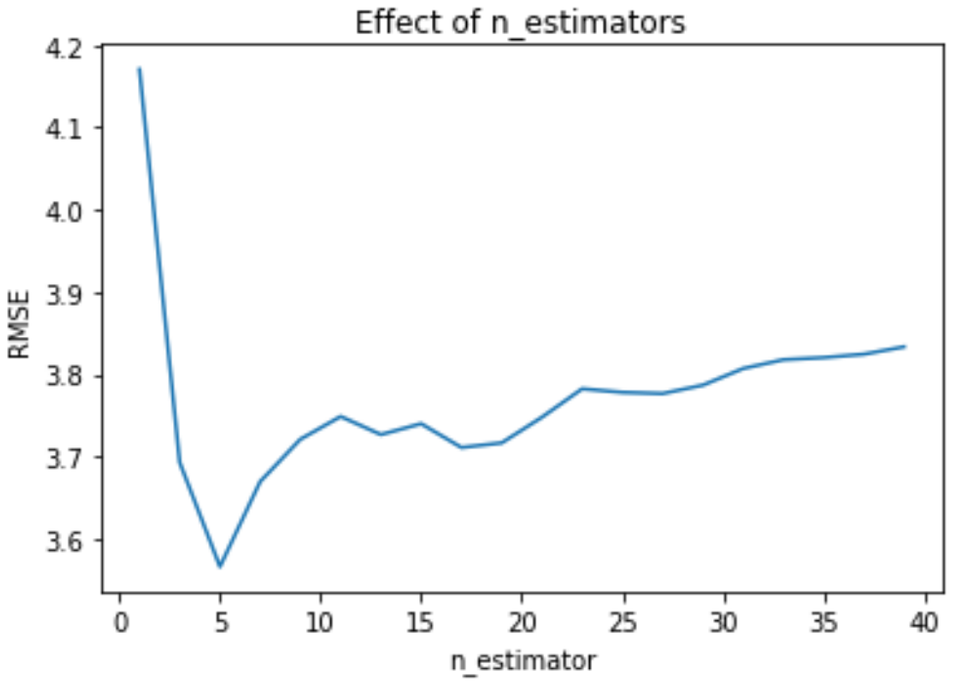

Fig 3.4b Number of nearest neighbors in KNN v/s RMSE values

3.5 Decision Tree Regressor:

The values of train RMSE is close to 0 which suggests that there is over-fitting problem in this model, thus a regularized model of Decision tree is needed to treat this.

```
================================================
================================================
DecisionTreeRegressor
Result-test
RMSE: 4.346134936801766
Result-train
RMSE: 0.26807482443947966
================================================
================================================
```

3.5 Random Forest Regressor:

The solution to the previous model of decision tree gets exactly served by the regularized method, Random Forest. Notice that regularized method helped when there was over-fitting in traditional method, like in the earlier where linear model did not have over-fitting problem,, regularized ridge and lasso didn't come much of handy in improving score. But, here in random forest the score has increased considerably from 4.36 to 3.45 with train value also getting better than decision tree's model.

```
================================================
================================================
RandomForestRegressor
Result-test
RMSE: 3.454816155486407
Result-train
RMSE: 1.1326756427399138
================================================
================================================
```

The Fig.3.4c depicts the RMSE values with number of trees, which is the least in 20-25 and 160-180 range with RMSE of around 3.36, this makes the model of choice to be higher.

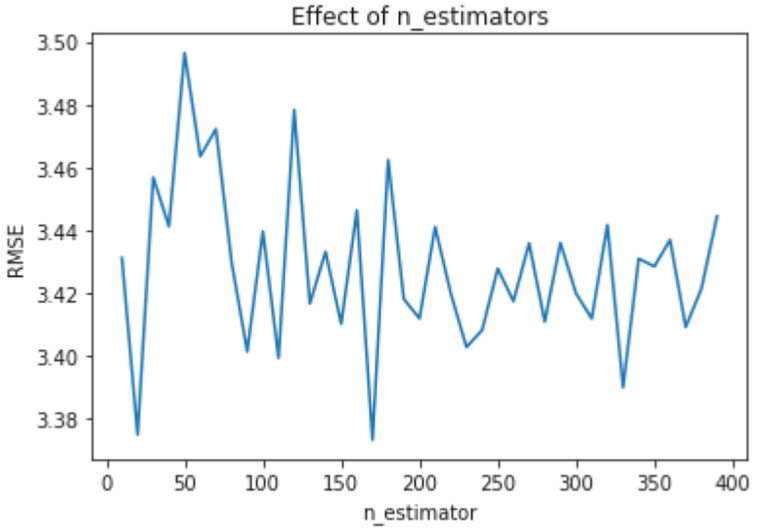

Fig 3.4c Number of trees in Random Forest v/s RMSE values

3.6 ADA Boost Regressor and Gradient Boost Regressor:

Both ADA boost and Gradient boost are boosting algorithms which means that they convert a set of weak learners into a single strong learner. They both initialize a strong learner (usually a decision tree) and iteratively create a weak learner that is added to the strong learner. They just differ on how they create the weak learners during the iterative process.

```
================================================
================================================
AdaBoostRegressor
Result-test
RMSE: 3.7049464047044403
Result-train
RMSE: 2.7963536152480746
================================================
================================================
GradientBoostingRegressor
Result-test
RMSE: 3.4747885667678045
Result-train
RMSE: 1.9479853430273673
================================================
================================================
```

3.7 XGBoost Regressor:

XGBoost uses a few computational tricks to speed up gradient descent and line search components, as well as a penalty function.

```
================================================
================================================
XGBRegressor
Result-test
RMSE: 3.4722148845178946
Result-train
RMSE: 2.0042449748723015
================================================
================================================
```

3.8 Hyperparameter tuning:

Here we will perform parameter tuning of XGBoost Regressor using GridSearchCV and will then see how our model performs on tuned parameters.

The value comes out to be:

```
{'gamma': 0, 'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 4
00, 'random_state': 1, 'subsample': 0.7}
```

After building the model on tuned parameters, it was seen that not significant improvement was there in the score, moreover it decreased suggesting that default parameters were better in model building. This might have occurred because the range of values input for GridSearchCV was not vast as it would have taken a long computational time, thus it missed on the best parameters. Nevertheless, in actual condition, by increasing the range, this problem hardly occurs.

Chapter 4: Final Inference and answers

4.1 Inference:

From all the above models, it was seen that Random Forest, Linear regression and all boosting m ethods performed well, so selection of any one of these would just be satisfactory, but the main f ocus lies in the preprocessing steps involved, the reason being, there were about two big imputat ions/changes in the dataset, one while dealing with missing values and another while dealing wit h outliers. Also, many ways of feature selection and binning of categories were available. Juggli ng with all these parameters and applying various permutations and combinations on the same, p lus the domain knowledge would have helped near the best model. Only after this, the tuning an d model building should have been done and overseen. Thus, there are several other ways and m ixed approaches we could employ till we can eliminate previous models and reach to an optial m odel.

The Fig 4.1a below shows and sums up the test MSE in a graphical plot of all the models used.

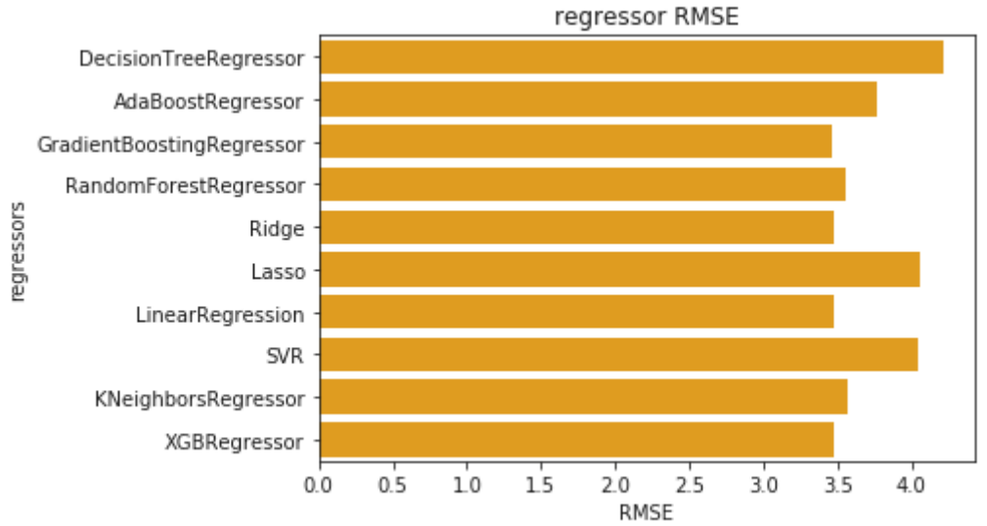

Fig. 4.1a Model v/s the test RMSE score

4.2 Answers:

Q1. What changes company should bring to reduce the number of absenteeism?

-The changes company can bring to reduce absenteeism hours is by cutting or improving in the areas where the absenteeism hours gets affected. For e.g. by observing graphs and plots in visualizations section, it was seen that oldest aged group of ~58 had tremendous shoot in absenteeism, thus employing younger batch can be helpful. The most number of instances(~21%) for the reason for absence was for medical consultation, thus having a half-yearly or quarterly checkup, or a doctor solely for company employees can also be appointed. It was also observed that work load was one of the important factors during the model building phase, so by bringing the work load to an optimized point which makes the employees avert absenteeism and the profits of the company also doesn't degrade much can be put in work. There was about 5% of instances where employees didn't justify the reason of absence, so stringent rules can be brought up against such employees. Like these, many other small changes can be brought up by the company, but broadly the above stated reasons would suffice. If absenteeism is a serious concern faced by the company, then it can definitely invest in the areas stated while compromising little on profits.

Q2. How much loss every month can we project in 2011 if same trend of absenteeism continues?

-If the same trend of absenteeism continues according to the dataset, then the losses incurred can be projected and be seen in Fig. 4.1b which shows that most of the losses were projected in the month of july, followed by march and april. The least was seen in the month of February.

The projection of losses was measured by grouping by means of service time per month, work load average/day, and number of absenteeism hours. The formula used was (absenteeism time in hours) / (Service time), which measured the part out of whole which were rendered absent by the employees, and this factor was multiplied by average work load(or units of work to be done in days), this gets us the final average losses per month.

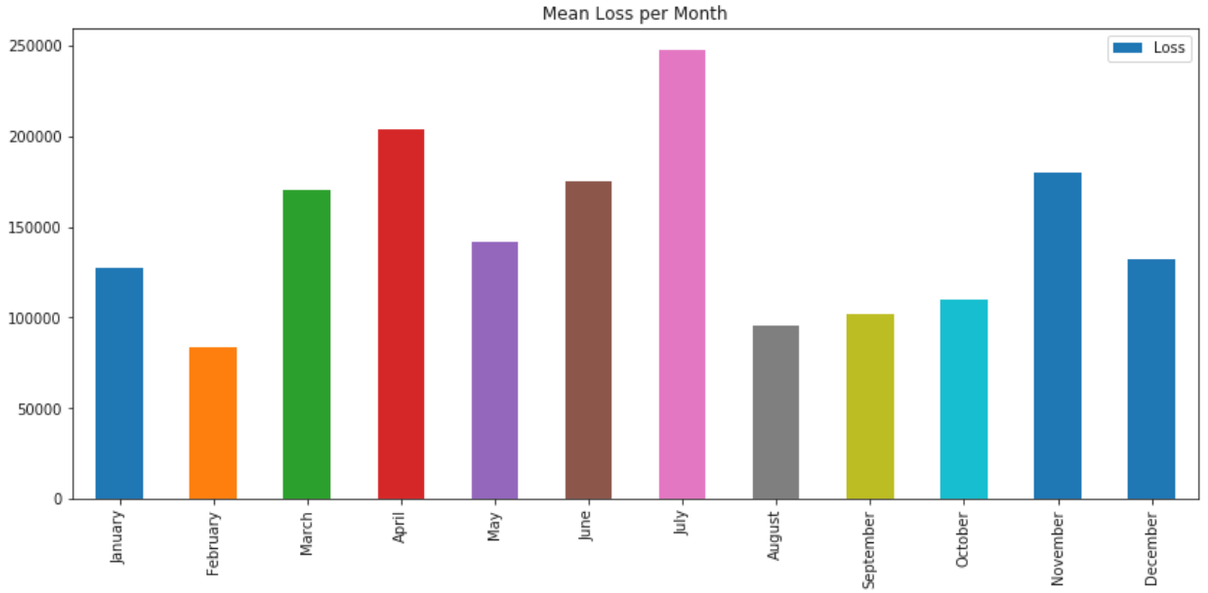

Fig. 4.2b Mean Loss per month due to absenteeism

Complete Python Code:

```
import os
import pandas as pd
import numpy as np
import matplotlib as mlt
import seaborn as sn
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Extracting Data
original_data = pd.read_excel("Absenteeism_at_work_Project.xls")
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
original_data = original_data.reset_index(drop=True)
original_data.index += 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df=original_data
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Converting variables to categoric and numeric
a= list(range(1, 5, 1))
b= list(range(11, 17, 1))
c=list(set().union(a,b))
for i in c:
    df.iloc[:, i] = df.iloc[:, i].astype('category')


df.iloc[:, 0] = df.iloc[:, 0].astype(object)
df.iloc[:, 13] = df.iloc[:, 13].astype('float64')
df.iloc[:, 16] = df.iloc[:, 16].astype('float64')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Checking missing values**
--------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Missing value analysis
missing_val = pd.DataFrame(original_data.isnull().sum())
```

```
missing_val
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df1 = df[df.isnull().any(axis=1)]
df1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Creating a new Dataframe by data types
df_num=df.select_dtypes(['float64'])
df_cat=df.select_dtypes(['category'])
df_num = df_num.reset_index(drop=True)
df_num.index += 1
df_cat = df_cat.reset_index(drop=True)
df_cat.index += 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Feature Selection before imputaion**
--------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Feature Selection using Correlation, VIF, and Chi-Square test

## 1) Correlation

correlations = df_num.corr()
correlations
correlations.style.background_gradient()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
## 2) Chi-square test

from scipy.stats import chi2_contingency
factors_paired_bin = [(i,j) for i in df_cat.columns.values for j in df_cat.columns.values]

chi2_bin, p_values_bin =[], []

for f in factors_paired_bin:
    if f[0] != f[1]:
             chitest_bin = chi2_contingency(pd.crosstab(df_cat[f[0]], df_cat[f[1]]))
             chi2_bin.append(chitest_bin[0])
             p_values_bin.append(chitest_bin[1])

    else:
```

```
            chi2_bin.append(0)
            p_values_bin.append(0)

chi2_bin = np.array(chi2_bin).reshape((8,8))
chi2_df_bin = pd.DataFrame(chi2_bin, index=df_cat.columns.values,
columns=df_cat.columns.values)
p_values_bin = np.array(p_values_bin).reshape((8,8)) # shaping it as a matrix
p_values_bin = pd.DataFrame(p_values_bin, index=df_cat.columns.values,
columns=df_cat.columns.values)
p_values_bin
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#VIF before dropping variable any variable
from statsmodels.stats.outliers_influence import variance_inflation_factor as vf
from statsmodels.tools.tools import add_constant
original_data=original_data.dropna()
numeric_df_v = add_constant(original_data[cnames_v])
vif_v = pd.Series([vf(numeric_df_v.values, i) for i in range(numeric_df_v.shape[1])],
                  index = numeric_df_v.columns)
vif_v
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
cnames = ['Distance from Residence to Work', 'Service time', 'Age', 'Work load Average/day
', 'Transportation expense',
       'Hit target', 'Son', 'Pet', 'Weight', 'Height','Absenteeism time in hours']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
catnames = list(set(df.columns) - set(cnames_v))
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#VIF after dropping Body mass index variable
numeric_df = add_constant(original_data[cnames])
vif = pd.Series([vf(numeric_df.values, i) for i in range(numeric_df.shape[1])],
                  index = numeric_df.columns)
vif
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Dropping redundant Features, and NAs from Target variable
df = df.drop(['Body mass index'], axis=1)
df = df.drop(df[df['Absenteeism time in hours'].isnull()].index, axis=0)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Marking cells as NA containing months of absence and reason of absence equal to zero
df.loc[df['Reason for absence'] == 0,'Reason for absence'] = np.nan
df.loc[df['Month of absence'] == 0,'Month of absence'] = np.nan
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
missing_val1 = pd.DataFrame(df.isnull().sum())
missing_val1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df2 = df[df.isnull().any(axis=1)]
df2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Imputing missing values after feature selection**
---------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
## Imputing Missing Values using KNN imputation
from fancyimpute import KNN
df = pd.DataFrame(KNN(k = 3).complete(df), columns = df.columns).round(0)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Visualization of data**
-------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Visualizations using barplots
import matplotlib.pyplot as plt
df.groupby('ID')['Absenteeism time in hours'].mean().plot(kind='bar',title='Mean Absentism in hours per ID',figsize=(14,6))
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
```

```
df.groupby('Month of absence')['Absenteeism time in
hours'].mean().plot(kind='bar',title='Mean Absentism in hours per month ',figsize=(14,6))
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df.groupby('Age')['Absenteeism time in hours'].mean().plot(kind='bar',title='Mean
Absentism in hours by age',figsize=(14,6))
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df.groupby('Disciplinary failure')['Absenteeism time in
hours'].mean().plot(kind='bar',title='Mean Absentism in hours wrt disciplinary
failure',figsize=(14,6))
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df.groupby('Reason for absence')['Absenteeism time in
hours'].mean().plot(kind='bar',title='Mean of Hours of Absentism for reason for absence
',figsize=(14,6))
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df.groupby('Day of the week')['Absenteeism time in
hours'].mean().plot(kind='bar',title='Mean of Hours of Absentism by day of the
week',figsize=(14,6))
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Boxplots of numeric varaibles before outlier removal
df_num.plot(kind='box', subplots=True,layout=(8,4), figsize=(17,39), fontsize=10)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Checking the variable importance with target variable using Extra Tree Regressor to get a
feel of importance of variables
from sklearn.ensemble import ExtraTreesRegressor
reg = ExtraTreesRegressor(n_estimators=70)
X = df.drop(columns=('Absenteeism time in hours'))
y = df['Absenteeism time in hours']
reg.fit(X, y)
imp_feat = pd.DataFrame({'Feature': df.drop(columns='Absenteeism time in hours').columns,
```

```
                          'importance':reg.feature_importances_*100})
imp_feat=imp_feat.sort_values(by = 'importance', ascending=False).reset_index(drop = True)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
sn.set_color_codes("muted")
sn.barplot(x='importance', y='Feature', data=imp_feat, color="orange")

plt.xlabel('Importance in percentage')
plt.title('Feature Importance')
plt.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df_o=df.copy()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Outlier Analysis**
--------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Checking for outliers range and replacing it with NAs
for i in cnames:
    print(i)
    q75, q25 = np.percentile(df_o.loc[:,i], [75 ,25])
    iqr = q75 - q25

    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)
    df_o.loc[df_o.loc[:,i] < min, i] = np.nan
    df_o.loc[df_o.loc[:,i] > max, i] = np.nan
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
missing_val2 = pd.DataFrame(df_o.isnull().sum())
missing_val2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
cnames_o = ['Absenteeism time in hours']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Creating outliers only for Absenteeism per hour, and leaving other variables untouched
```

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

    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)
    df.loc[df.loc[:,i] < min, i] = np.nan
    df.loc[df.loc[:,i] > max, i] = np.nan
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
missing_val3 = pd.DataFrame(df.isnull().sum())
missing_val3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
## Imputing Outliers using KNN imputation
from fancyimpute import KNN
df = pd.DataFrame(KNN(k = 3).complete(df), columns = df.columns).round(0)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
for i in catnames:
    df.loc[:, i] = df.loc[:, i].astype('category')


df.iloc[:, 0] = df.iloc[:, 0].astype(object)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df_num=df.select_dtypes(['float64'])
df_cat=df.select_dtypes(['category'])
df_num = df_num.reset_index(drop=True)
df_num.index += 1
df_cat = df_cat.reset_index(drop=True)
df_cat.index += 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Histograms after outlier removal
from matplotlib import pyplot
df_num.hist(figsize=(17,12))
pyplot.show()
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Boxplots of numeric varaibles after outlier removal
df_num.plot(kind='box', subplots=True,layout=(8,3), figsize=(17,39), fontsize=10)
pyplot.show()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
cnames = ['Distance from Residence to Work', 'Service time', 'Age', 'Work load Average/day
', 'Transportation expense',
       'Hit target', 'Son', 'Pet', 'Weight', 'Height']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Feature Scaling**
------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Feature Scaling of all numeric varaibles except the target varaible
for i in cnames:

    df[i] = (df[i] - df[i].min())/(df[i].max() - df[i].min())
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
from sklearn.cross_validation import train_test_split
from random import *
from numpy import *
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
df1=df.copy()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Creating dummies for all categorical variables

mylist = list(df.select_dtypes(include=['category']).columns)
```

```
dummies = pd.get_dummies(df1[mylist], prefix= mylist,drop_first=True)
df1.drop(mylist, axis=1, inplace = True)
df1 = pd.concat([df1,dummies], axis =1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Creating train and test data
random.seed(2)
X = df1.drop(columns=['Absenteeism time in hours','ID'])
y = df1['Absenteeism time in hours']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
2)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
X_train = X_train.reset_index(drop=True)
X_train.index += 1
X_test = X_test.reset_index(drop=True)
X_test.index += 1
y_train = y_train.reset_index(drop=True)
y_train.index += 1
y_test = y_test.reset_index(drop=True)
y_test.index += 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Model Development**
=====================

### **Performing Linear regression and checking its summary**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
#Linear Regression
from sklearn.metrics import mean_squared_error
from math import sqrt
lr = LinearRegression()
lr_model = lr.fit(X_train, y_train)
y_pred_train = lr_model.predict(X_train)
y_pred = lr_model.predict(X_test).round(0)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
model_lr_dummies = sm.OLS(y_train,X_train).fit()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
model_lr_dummies.summary()
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
train_rmse_lr=sqrt(mean_squared_error(y_train,y_pred_train))
print("train_rmse_lr:")
train_rmse_lr
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
test_rmse_lr=sqrt(mean_squared_error(y_test,y_pred))
print("test_rmse_lr:")
test_rmse_lr
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Performing all the regressions in one go with their default parameters**
-------------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
from sklearn.tree import DecisionTreeRegressor, ExtraTreeRegressor
from sklearn.linear_model import Ridge, Lasso, LinearRegression, ElasticNet, Lars
from sklearn.ensemble import AdaBoostRegressor, GradientBoostingRegressor,
ExtraTreesRegressor, RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor

regressors = [DecisionTreeRegressor(),  AdaBoostRegressor(), GradientBoostingRegressor(),
RandomForestRegressor(),
Ridge(), Lasso(), LinearRegression(), SVR(), KNeighborsRegressor(),XGBRegressor()]


imp_cols=["regressors", "MSE"]
imp = pd.DataFrame(columns=imp_cols)
for rgr in regressors:
    rgr.fit(X_train, y_train)
    name = rgr.__class__.__name__

    print("="*50)
    print(name)

    print('Result-test')
    train_predictions = rgr.predict(X_test)
    mse = np.sqrt(mean_squared_error(y_test, train_predictions))
    print("RMSE: {}".format(mse))
    print('Result-train')
```

```
    ori_predictions = rgr.predict(X_train)
    mse_t = np.sqrt(mean_squared_error(y_train, ori_predictions))
    print("RMSE: {}".format(mse_t))

    print("="*50)
    imp_entry = pd.DataFrame([[name, mse]], columns=imp_cols)
    imp = imp.append(imp_entry)
```

~~~~~~

~~~~~~ python

```
#A plot indicating all regressions' RMSE values for easy comparison
sn.set_color_codes("muted")
sn.barplot(x='MSE', y='regressors', data=imp, color="orange")

plt.xlabel('RMSE')
plt.title('regressor RMSE')
plt.show()
```

~~~~~~

~~~~~~ python

```
#Checking the number of trees v/s RMSE values

rf_model = RandomForestRegressor().fit(X_train, y_train)
estimators = np.arange(10, 400, 10)
scores = []

for n in estimators:
    rf model.set params(n estimators=n)
    forest=rf_model.fit(X_train,y_train)
    train_predictions_y = forest.predict(X_test)
    scores.append(np.sqrt(mean_squared_error(y_test, train_predictions_y)))

plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("RMSE")
plt.plot(estimators, scores)
```

~~~~~~

~~~~~~ python

```
#Checking the number of nearest neighbours v/s RMSE for KNN

KNN_model=KNeighborsRegressor().fit(X_train, y_train)
estimators = np.arange(1, 41, 2)
scores = []
```

```
for n in estimators:
    KNN_model.set_params(n_neighbors=n)
    KNN=KNN_model.fit(X_train,y_train)
    train_predictions_y = KNN.predict(X_test)
    scores.append(np.sqrt(mean_squared_error(y_test, train_predictions_y)))

plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("RMSE")
plt.plot(estimators, scores)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

### **Parameter tuning using GridSearchCV**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
from sklearn.model_selection import GridSearchCV
regressor = XGBRegressor(random_state=1)
params = [{'n_estimators' : [250, 300,400], 'max_depth':[2, 3],
          'learning_rate':[0.01,0.05, 0.1, 0.3],'gamma':[0, 0.001, 0.01],
           'subsample':[1, 0.7, 0.8],'random_state' :[1]}]
grid_search = GridSearchCV(estimator=regressor, param_grid=params,cv = 9,
                           scoring = 'explained_variance', n_jobs=-1)
grid_search = grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
# Building XGBRegressor on tuned parameter
regressor = XGBRegressor(random_state=1, learning_rate=0.01, max_depth=3,
n_estimators=400,
                         gamma = 0, subsample=0.7)
XGB=regressor.fit(X_train, y_train)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
test_predictions_XGB = regressor.predict(X_test)
mse_XGB = np.sqrt(mean_squared_error(y_test, test_predictions_XGB))
mse_XGB
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
q2=original_data.copy()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Answer for projection of losses**
-----------------------------------
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
q2.loc[q2['Reason for absence'] == 0,'Reason for absence'] = np.nan
q2.loc[q2['Month of absence'] == 0,'Month of absence'] = np.nan
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
a1=q2.groupby('Month of absence')['Service time'].mean().to_frame()
a2=q2.groupby('Month of absence')['Absenteeism time in hours'].mean().to_frame()
a3=q2.groupby('Month of absence')['Work load Average/day '].mean().to_frame()
h=pd.merge(a1, a2, on='Month of absence')
f2=pd.merge(h, a3, on='Month of absence')
f2['Loss']=(f2['Absenteeism time in hours']/f2['Service time'])*f2['Work load Average/day
']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
f2t=f2.T
f2t.columns = ["Month of
Absence","January","February","March","April","May","June","July","August","September","Oc
tober","November","December"]
f2=f2t.T
f2=f2.dropna()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
f2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ python
f2.plot(kind='bar', y='Loss', title='Mean Loss per Month',figsize=(14,6))
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Complete R Code:

```
rm(list=ls())
library(rpart)
library(MASS)
library(ggplot2)
library("scales")
library("psych")
library("gplots")
library(corrgram)
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees',"dplyr","plyr","reshape","data.table")
getwd()
#devtools::install_github("hadley/dplyr")
.libPaths()
library(githubinstall)
#githubinstall("dplyr")
#install.packages(x)
#rm(x)



#Extracting Data
marketing_train= read.csv("C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Project
2\\Absenteeism_at_work_Project.csv", header = T)


#Missing Values Analysis
missing_val = data.frame(apply(marketing_train,2,function(x){sum(is.na(x))}))
sum(missing_val)

new_DF <- marketing_train[rowSums(is.na(marketing_train)) > 0,]

#Changing variable types
cnames <- c('Distance.from.Residence.to.Work', 'Service.time', 'Age',
'Work.load.Average.day', 'Transportation.expense',
          'Hit.target', 'Son', 'Pet', 'Weight',
'Height','Absenteeism.time.in.hours','Body.mass.index')
cat_names <- c('Social.smoker',
```

```
            'Month.of.absence',
            'Social.drinker',
            'Reason.for.absence',
            'Disciplinary.failure',
            'ID',
            'Education',
            'Seasons',
            'Day.of.the.week')
#Checking the summary
summary(marketing_train[,cnames])
lapply(marketing_train[,cnames], function(feat) length(unique(feat)))
str(marketing_train)


for(i in cat_names){
  marketing_train[,i] = as.factor(marketing_train[,i])
  }
str(marketing_train)
library(DMwR)
marketing_train=marketing_train[!is.na(marketing_train$Absenteeism.time.in.hours), ]

#Imputing all the missing values using MICE
library(mice)
miceMod <- mice(marketing_train, method="rf", seed=1)  # perform mice imputation, based
on random forests.
miceOutput <- complete(miceMod)  # generate the completed data.
anyNA(miceOutput)

write.csv(miceOutput, "C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Project
2\\mice_output_pro_2.csv", row.names = F)
miceOutput=read.csv ("C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Project
2\\mice_output_pro_2.csv")

da=miceOutput

#######Data exploration and visualizing data########

library(ggthemes)
library(grid)
library(gridExtra)
  p <- ggplot(da, aes(x = Pet, fill = Pet)) + geom_bar()
s <- ggplot(da, aes(x = Son, fill = Son)) + geom_bar()
```

```
 82
 83 SS <- ggplot(da, aes(x =  Social.smoker, fill =  Social.drinker)) + geom_bar()
 84
 85 S <- ggplot(da, aes(x =   Seasons,fill = Seasons)) + geom_bar()
 86
 87 grid.arrange(p,s, nrow = 1)
 88 grid.arrange(SS,S, nrow = 1)
 89
 90 library(dplyr)
 91
 92 absent <- as.data.frame( da %>% select(everything()) %>%
 93 filter(Absenteeism.time.in.hours > 0))
 94 Reason <-  as.data.frame(absent %>% group_by(Reason.for.absence) %>% summarise(count=
 95 n(), percent = round(count*100/nrow(absent),1))%>% arrange(desc(count)))
 96 ggplot(Reason,aes(x = reorder(Reason.for.absence,percent), y= percent, pos=3, xpd=NA,
 97 fill= Reason.for.absence)) + geom_bar(stat = 'identity') + coord_flip() +
 98 theme(legend.position='none') +
 99   geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) + xlab('Reason for
100 absence')
101
102
103 a <- ggplot(da, aes(x = Age, fill = Son)) + geom_bar()
104 grid.arrange(a, nrow = 1)
105
106 for (i in cnames)
107 {
108   print(i)
109   value=da[,i][da[,i] %in% boxplot.stats(da[,i],coef=1.5)$out]
110   print(value)
111 }
112
113
114 for (i in 1:length(cnames))
115 {
116   assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i])), data =
117 subset(marketing_train))+
118            stat_boxplot(geom = "errorbar", width = 0.1) +
119            geom_boxplot(outlier.colour="red", fill = "white" ,outlier.shape=18,
120                         outlier.size=2, notch=FALSE) +
121            theme(legend.position="bottom")+
122            labs(y=cnames[i]))
123 }
```

```
124
125 # Plotting plots together
126 gridExtra::grid.arrange(gn1,gn5,gn2,gn6,gn3,gn7,gn8,gn9,gn10,gn11,ncol=3, nrow=4)
127
128 marketing_train=miceOutput
129
130 #Replacing outliers with NA using boxplot
131 for(i in cnames){
132      val = marketing_train[,i][marketing_train[,i] %in%
133 boxplot.stats(marketing_train[,i])$out]
134      print(length(val))
135      marketing_train[,i][marketing_train[,i] %in% val] = NA
136    }
137
138 #Imputing outliers using MICE method
139 miceMod_2 <- mice(marketing_train, method="rf", seed=1)  # perform mice imputation,
140 based on random forests.
141 miceOutput_2 <- complete(miceMod_2)  # generate the completed data.
142 anyNA(miceOutput_2)
143
144 write.csv(miceOutput, "C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Project
145 2\\mice_output_2_pro_2.csv", row.names = F)
146
147 miceOutput_2=read.csv("C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Project
148 2\\mice_output_2_pro_2.csv")
149 marketing_train=miceOutput_2
150
151
152 #Feature Selection####
153
154
155 # Correlation Plot
156
157 corrgram(marketing_train[,cnames], order = F,
158          upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
159
160
161 #ANOVA Analysis
162 anova_multi_way <- aov(Absenteeism.time.in.hours~(Social.smoker)+
163                                                  (Month.of.absence)+
164                                                  (Social.drinker)+
165                                                  (Reason.for.absence)+
```

```
                                                    (Disciplinary.failure)+
                                                    (ID)+
                                                    (Education)+
                                                    (Seasons)+
                                                    (Day.of.the.week), data =
marketing_train)
summary(anova_multi_way)

## Dimension Reduction
marketing_train = subset(marketing_train,
                         select = -c(Body.mass.index,ID))

#####Feature Scaling####
cnames <- c('Distance.from.Residence.to.Work', 'Service.time', 'Age',
'Work.load.Average.day', 'Transportation.expense',
            'Hit.target', 'Son', 'Pet', 'Weight', 'Height')
cat_names <- c('Social.smoker',
               'Month.of.absence',
               'Social.drinker',
               'Reason.for.absence',
               'Disciplinary.failure',
               'Education',
               'Seasons',
               'Day.of.the.week')
for (i in cat_names){

  marketing_train[,i] = as.factor(marketing_train[,i])
}

#Scaling all numeric variables leaving our target variable untouched
for(i in cnames){
  print(i)
  marketing_train[,i] = as.numeric(marketing_train[,i])
  marketing_train[,i] = (marketing_train[,i] - min(marketing_train[,i]))/
    (max(marketing_train[,i] - min(marketing_train[,i])))
}
str(marketing_train)

#Removing the rows containing absurd information
marketing_train= marketing_train[!marketing_train$Month.of.absence==0 &
!marketing_train$Reason.for.absence==0, ]

```

```
#Creating dummies for categorical variables
DFdummies <- as.data.frame(model.matrix(~. -1, marketing_train))
dim(DFdummies)

library(DataCombine)
rmExcept(c("marketing_train","DFdummies","miceoutput_2"))

wdf=marketing_train
marketing_train=DFdummies

#Creating train and test data
library(caret)
set.seed(1)
train.index = createDataPartition(marketing_train$Absenteeism.time.in.hours, p = .80,
list = FALSE)
X_train = marketing_train[ train.index,]
y_train  = marketing_train[-train.index,]
y_test=y_train$Absenteeism.time.in.hours
X_test=X_train$Absenteeism.time.in.hours

library(usdm)


#Creating a function to run all types of regression and comparing the values
Show.RMSE <- function(method, train_data, test_data){
  regressor_fit <- caret::train(Absenteeism.time.in.hours~., data = X_train, method =
method)

  y_pred <- predict(regressor_fit, y_train)
  print("RMSE value of test data")
  print(caret::RMSE(y_test, y_pred))
}
require(gbm)
library (ridge)
library(enet)
library(elasticnet)
library(h20)
regressors=c('lm','knn','svmLinear3',
'rpart2','rf','xgbTree','ridge','lasso','gbm_h2o')

#Running all the regressions and checking the performance on test data
for(i in regressors){
```

```
  print(i)
Show.RMSE(i, X_train, y_test)
print(strrep('-',50))
}

#Finding the optimum parameters for XG Boost tree using repeatedCV
control <- trainControl(method="repeatedcv", number=5, repeats=2)
reg_XG <- caret::train(Absenteeism.time.in.hours~., data = X_train, method =
"xgbTree",trControl = control)
reg_XG$bestTune
```