EMPLOYEE ABSENTEEISM

DATA SCIENCE PROJECT

Abstract

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.

Shivam Baslas

Shivam72baslas@gmail.com

Chapter 1
Introduction

1.1. Problem Statement and Project Description

XYZ is a courier company. As we appreciate that human capital plays an important role in collection, transportation and delivery. The company is passing through genuine issue of Absenteeism. The company has shared it dataset and requested to have an answer on the following areas:
1. What changes company should bring to reduce the number of absenteeism?
2. How much losses every month can we project in 2011 if same trend of absenteeism continues?

1.2 Data

We have 21 variables in given data set. There are 20 independent variable and 1 dependent variables (Absenteeism time in hours). As target variable is continuous it is a regression problem.

Table 1.1: Sample Data

	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
0	11	26.0	7.0	3	1	289.0	36.0	13.0	33.0	239554.0
1	36	0.0	7.0	3	1	118.0	13.0	18.0	50.0	239554.0
2	3	23.0	7.0	4	1	179.0	51.0	18.0	38.0	239554.0
3	7	7.0	7.0	5	1	279.0	5.0	14.0	39.0	239554.0
4	11	23.0	7.0	5	1	289.0	36.0	13.0	33.0	239554.0

	Hit target	Disciplinary failure	Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
0	97.0	0.0	1.0	2.0	1.0	0.0	1.0	90.0	172.0	30.0	4.0
1	97.0	1.0	1.0	1.0	1.0	0.0	0.0	98.0	178.0	31.0	0.0
2	97.0	0.0	1.0	0.0	1.0	0.0	0.0	89.0	170.0	31.0	2.0
3	97.0	0.0	1.0	2.0	1.0	1.0	0.0	68.0	168.0	24.0	4.0
4	97.0	0.0	1.0	2.0	1.0	0.0	1.0	90.0	172.0	30.0	2.0

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)

1.3 Expletory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analysing data sets and summarize their main characteristics.

Our data consist 740 observation and 21 variables. Data type of all variables are either int64 or float64. As per the data analysis we have to find which variables are the categorical variables, continuous variables and target variable. Data types need to be change accordingly.

We have distributed the variables on the basis of continuous and categorical variables. Target variable is continuous.

We have total 740 observation, but as per above summary tables total observation is <740 in some variables. Its means there is missing values present in our dataset. Missing value analysis is required to further understand the data.

Table 1.2: Variables and its Data type

```
(740, 21)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 740 entries, 0 to 739
Data columns (total 21 columns):
ID                                 740 non-null int64
Reason for absence                 737 non-null float64
Month of absence                   739 non-null float64
Day of the week                    740 non-null int64
Seasons                            740 non-null int64
Transportation expense             733 non-null float64
Distance from Residence to Work    737 non-null float64
Service time                       737 non-null float64
Age                                737 non-null float64
Work load Average/day              730 non-null float64
Hit target                         734 non-null float64
Disciplinary failure               734 non-null float64
Education                          730 non-null float64
Son                                734 non-null float64
Social drinker                     737 non-null float64
Social smoker                      736 non-null float64
Pet                                738 non-null float64
Weight                             739 non-null float64
Height                             726 non-null float64
Body mass index                    709 non-null float64
Absenteeism time in hours          718 non-null float64
dtypes: float64(18), int64(3)
memory usage: 121.5 KB
None
```

continuous variable

- ID
- Transportation expense
- Distance from Residence to Work
- Service time
- Age
- Work load Average/day
- Hit target
- Weight
- Height
- Body mass index
- Absenteeism time in hours

categorical variable

- Reason for absence
- Month of absence
- Day of the week
- Seasons
- Disciplinary failure
- Education
- Son
- Social drinker
- Social smoker
- Pet

target variable

- Absenteeism time in hours

Target Variable (Absenteeism time in hours) having missing values due to which we cannot conclude any statement so we can drop those observation.

For “Month of absence” variable month 0 is not possible this is create due to human error we drop the same for our further analysis.

Table 1.3: Variables and its Data types.

```
(715, 21)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 715 entries, 0 to 736
Data columns (total 21 columns):
ID                                 715 non-null int64
Reason for absence                 712 non-null float64
Month of absence                   714 non-null float64
Day of the week                    715 non-null int64
Seasons                            715 non-null int64
Transportation expense             709 non-null float64
Distance from Residence to Work    712 non-null float64
Service time                       712 non-null float64
Age                                713 non-null float64
Work load Average/day              707 non-null float64
Hit target                         709 non-null float64
Disciplinary failure               710 non-null float64
Education                          705 non-null float64
Son                                709 non-null float64
Social drinker                     712 non-null float64
Social smoker                      711 non-null float64
Pet                                713 non-null float64
Weight                             714 non-null float64
Height                             701 non-null float64
Body mass index                    686 non-null float64
Absenteeism time in hours          715 non-null float64
dtypes: float64(18), int64(3)
memory usage: 122.9 KB
None
```

Now after removal the missing value observation for Absenteeism time in hours and Month of absence value 0. There is 715 observation and 21 variables.

There are 11 categorical variables and 10 continuous variables out of one is target variable (Absentieeism time in hours).

All the 11 categorical variables will change to datatype “object” after missing value and Outlier analysis.

Chapter 2

Methodology

2.1 Data Preprocessing:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

2.1.1. Missing Value Analysis

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. If a columns has more than 30% of data as missing value either we ignore the entire column or we ignore those observations.

Table 2.1: Missing values in dataset

	variable	missing_count	missing_Per
0	Body mass index	29	4.055944
1	Height	14	1.958042
2	Education	10	1.398601
3	Work load Average/day	8	1.118881
4	Hit target	6	0.839161
5	Son	6	0.839161
6	Transportation expense	6	0.839161
7	Disciplinary failure	5	0.699301
8	Social smoker	4	0.559441
9	Distance from Residence to Work	3	0.419580
10	Service time	3	0.419580
11	Social drinker	3	0.419580
12	Reason for absence	3	0.419580
13	Age	2	0.279720
14	Pet	2	0.279720
15	Weight	1	0.139860
16	Month of absence	1	0.139860
17	ID	0	0.000000
18	Seasons	0	0.000000
19	Day of the week	0	0.000000
20	Absenteeism time in hours	0	0.000000

We can treat these missing values through statistic methods that is mean median mode and KNN imputation.
To choose best fit method me impute any known value with NA and test all statistic method and apply the method which is close to the actual value.

Fir this data set KNN imputation is provided nearest value to the missing data point.

Table 2.1: Missing values in dataset

```
ID                                 0
Reason for absence                 0
Month of absence                   0
Day of the week                    0
Seasons                            0
Transportation expense             0
Distance from Residence to Work    0
Service time                       0
Age                                0
Work load Average/day              0
Hit target                         0
Disciplinary failure               0
Education                          0
Son                                0
Social drinker                     0
Social smoker                      0
Pet                                0
Weight                             0
Height                             0
Body mass index                    0
Absenteeism time in hours          0
dtype: int64
```

2.1.2. Outlier Analysis

Outliers are the abnormal values, which inconsistent with rest of dataset, or observation which lies abnormal distance from other values in dataset.

There are different-different causes of outlier like- Poor Data Quality, Manual Error, Malfunctioning Equipment, Low Quality Measurement and sometimes correct but exceptional data.

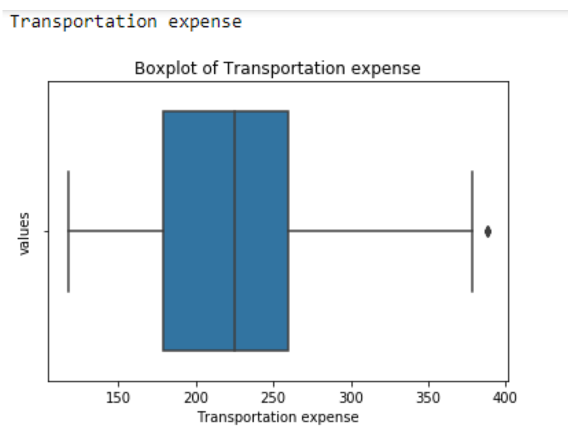
So, to detect the outlier and remove the outlier there are different-different method like Box Plot method, Grubb's test for outlier, R Package Outlier etc.

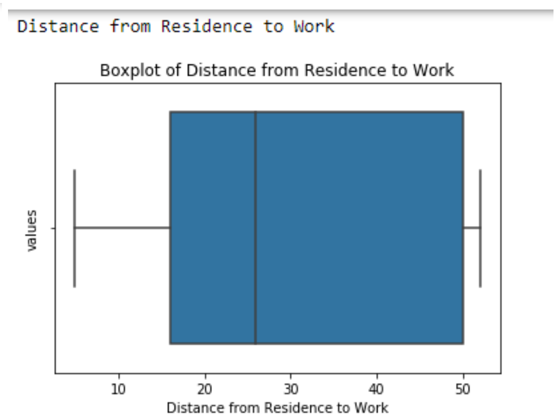
But from all of them prefer the Box Plot Method to detect and remove the outliers.

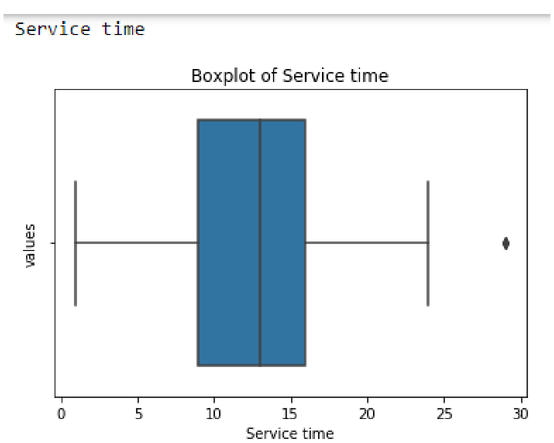
Box Plot method is a simplest way for detecting the outliers. A box plot is a graphical display for describing the distribution of data. It shows the data in its graphical representation with the median, 1st quartile, 3rd quartiles, inner fence and outer fence. If the values of a particular variable are outside from inner and outer fence then these values can be considered as outliers.

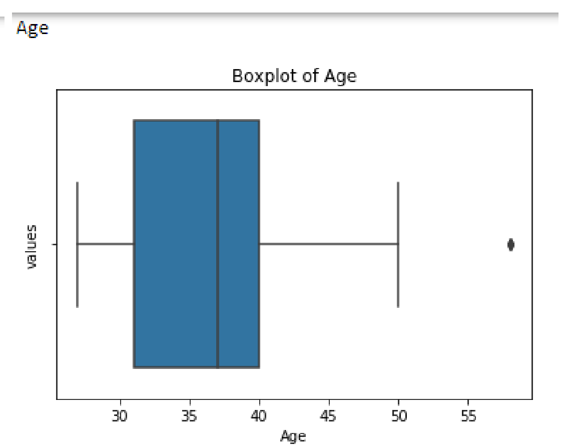
The box plot method detects outlier if any value is present greater than (**Q3 + (1.5 * IQR)**) or less than (**Q1 – (1.5 * IQR)**)
Q1 > 25% of data are less than or equal to this value
Q2 or Median -> 50% of data are less than or equal to this value
Q3 > 75% of data are less than or equal to this value
IQR(Inter Quartile Range) = Q3 – Q1

Table 2.3: Outlier analysis in dataset with Box Plot

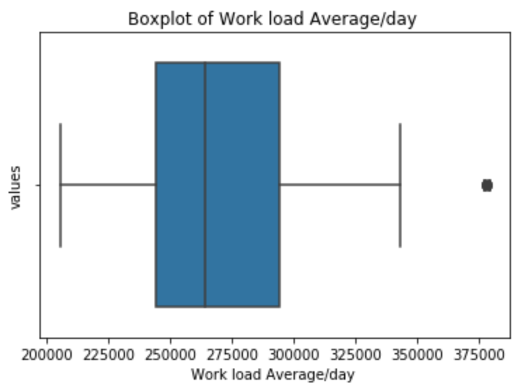




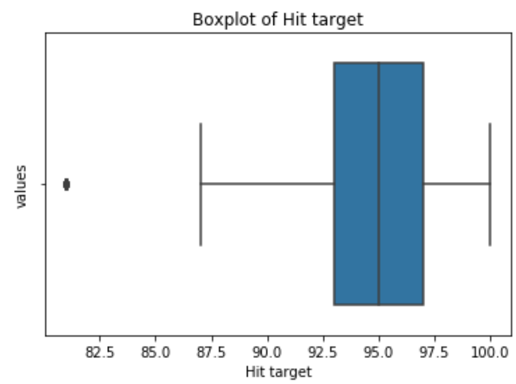




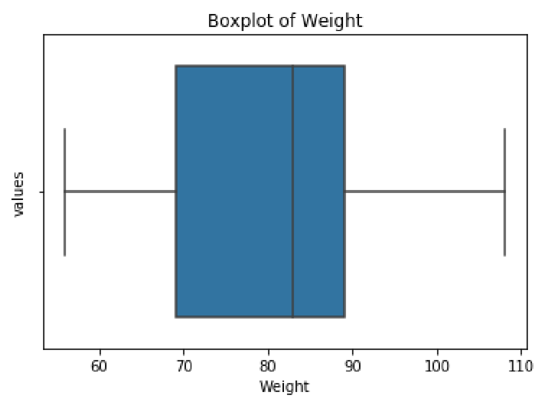
Work load Average/day



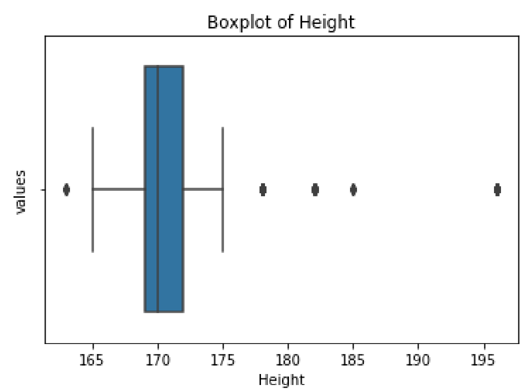
Hit target



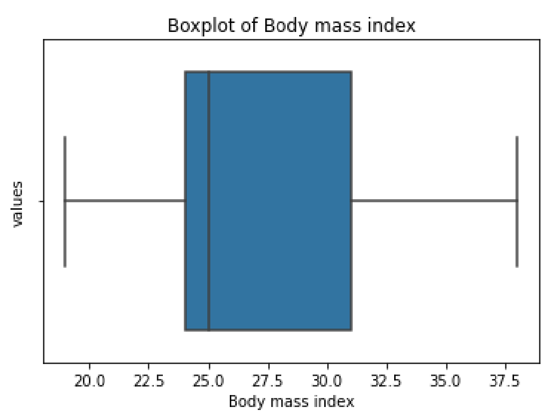
Weight



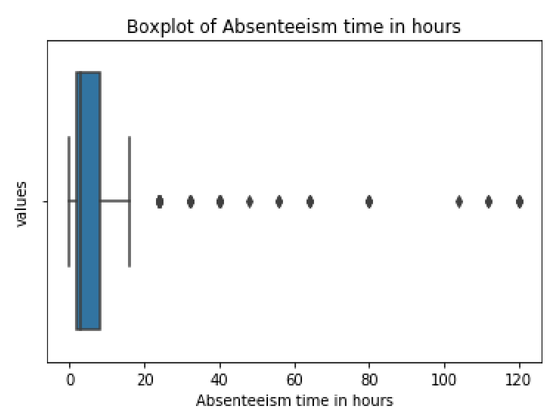
Height



Body mass index



Absenteeism time in hours



Before treating outlier, we should look at nature of outlier. Is it information or outlier(error)?

There seems to be many outlier in our target variable (Absenteeism time in hours) and variable (Service Time). There are value of >24 which is not possible.as the data set is a daily data set with no of absent hour per day. A day has max 24 hours, so all these values seems redundant and we need to eliminate these out.

Logically the absenteeism hours should be less than the service time of that employee. We will use KNN imputation to impute these outliers.

For Height the shown outlier point can be correct data as it is possible to have the height b/w 175cm to 185cm an in exceptional and rare cases is can be 195cm. or it may be due to low measurement quality.

Table 2.4: Summary of Box plot parameters

```
Transportation expense
min= 57.5
max= 381.5
IQR= 81.0
Distance from Residence to Work
min= -35.0
max= 101.0
IQR= 34.0
Service time
min= -1.5
max= 26.5
IQR= 7.0
Age
min= 17.5
max= 53.5
IQR= 9.0
Work load Average/day
min= 169642.0
max= 368962.0
IQR= 49830.0
Hit target
min= 87.0
max= 103.0
IQR= 4.0
Weight
min= 39.0
max= 119.0
IQR= 20.0
Height
min= 164.5
max= 176.5
IQR= 3.0
Body mass index
min= 13.5
max= 41.5
IQR= 7.0
Absenteeism time in hours
min= -7.0
max= 17.0
IQR= 6.0
```

2.1.3. Data observation

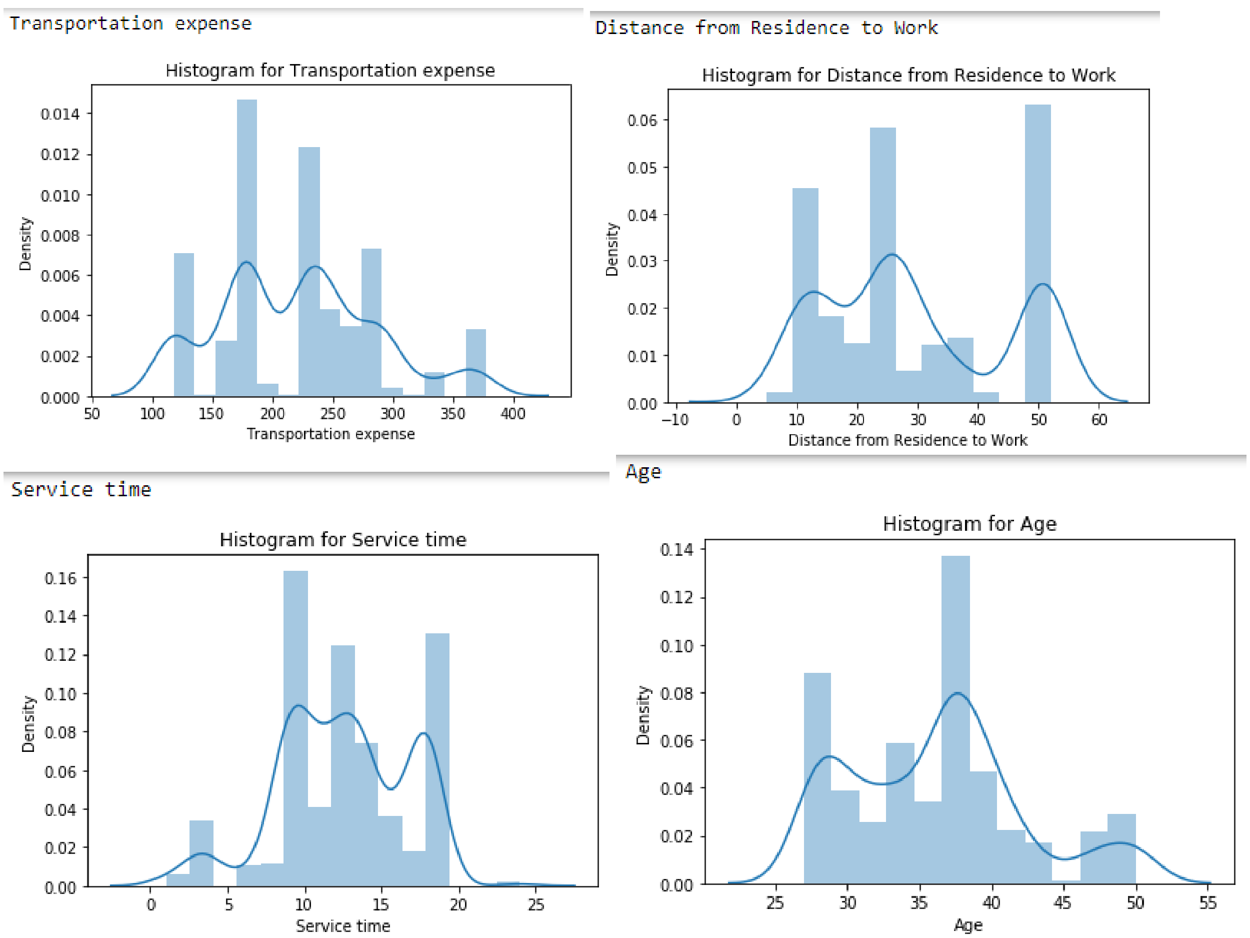
Table 2.5: Categorical Variable summary after Outlier imputation.

```
variable ID having 34 unique values that are:[1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 29 30 31 32 33 34 36]
variable Reason for absence having 28 unique values that are:[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25
26 27 28]
variable Month of absence having 12 unique values that are:[1 2 3 4 5 6 7 8 9 10 11 12]
variable Day of the week having 5 unique values that are:[2 3 4 5 6]
variable Seasons having 4 unique values that are:[1 2 3 4]
variable Disciplinary failure having 2 unique values that are:[0 1]
variable Education having 5 unique values that are:[0 1 2 3 4]
variable Son having 5 unique values that are:[0 1 2 3 4]
variable Social drinker having 2 unique values that are:[0 1]
variable Social smoker having 2 unique values that are:[0 1]
variable Pet having 6 unique values that are:[0 1 2 4 5 8]
```

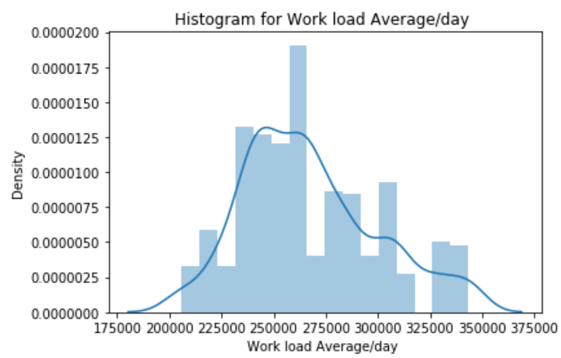
Table 2.6: Continuous Variable summary after Outlier imputation.

	count	mean	std	min	25%	50%	75%	max
Transportation expense	715.0	220.053147	65.197910	118.0	179.0	225.0	260.0	378.0
Distance from Residence to Work	715.0	29.558042	14.785409	5.0	16.0	26.0	50.0	52.0
Service time	715.0	12.461538	4.157220	1.0	9.0	12.0	16.0	24.0
Age	715.0	36.139860	6.062699	27.0	31.0	37.0	40.0	50.0
Work load Average/day	715.0	267256.806993	32309.365234	205917.0	244387.0	264249.0	284853.0	343253.0
Hit target	715.0	94.920280	3.092517	87.0	93.0	95.0	97.0	100.0
Weight	715.0	79.006993	12.850954	56.0	69.0	83.0	89.0	108.0
Height	715.0	170.078322	1.772658	165.0	169.0	170.0	171.0	175.0
Body mass index	715.0	26.657343	4.192338	19.0	24.0	25.0	31.0	38.0
Absenteeism time in hours	715.0	4.346853	3.388475	0.0	2.0	3.0	8.0	16.0

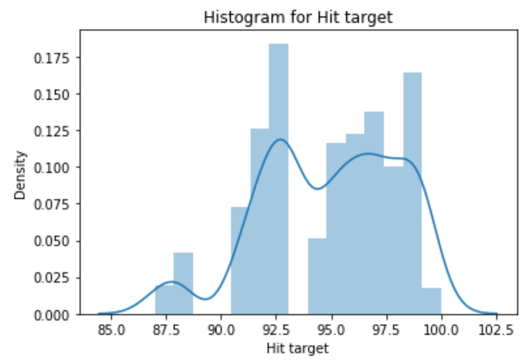
Table 2.7: Histogram of Continuous Variable



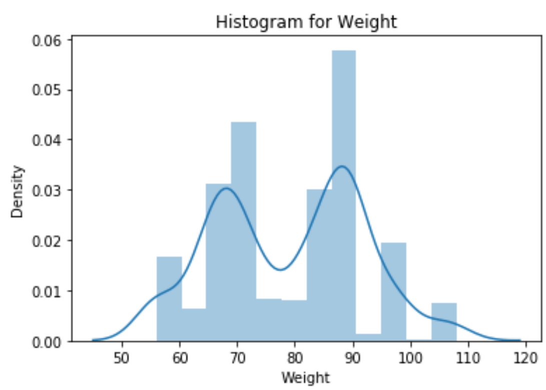
Work load Average/day



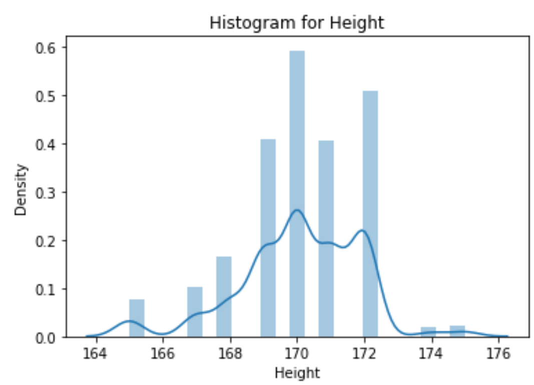
Hit target



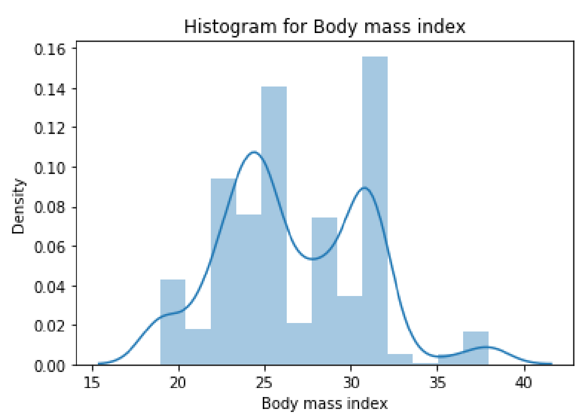
Weight



Height



Body mass index



Absenteeism time in hours

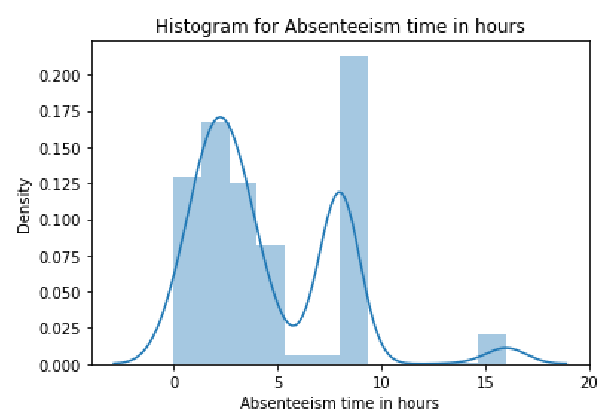
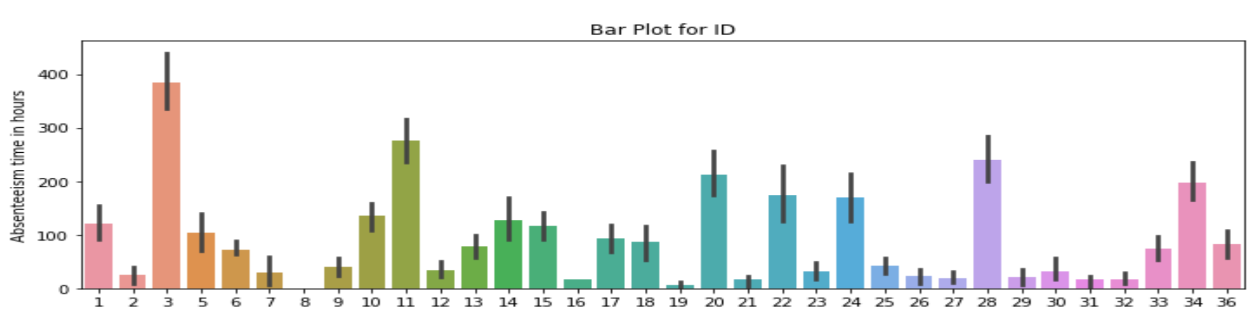
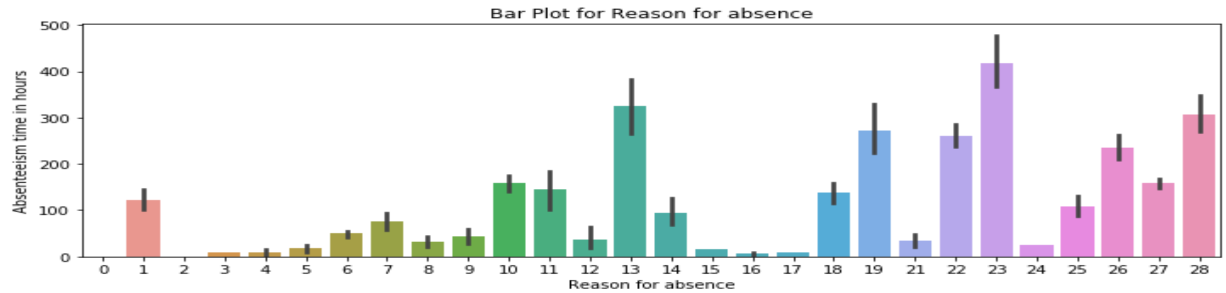


Table 2.8: Bar Plot of Categorical Variable

ID



Reason for absence

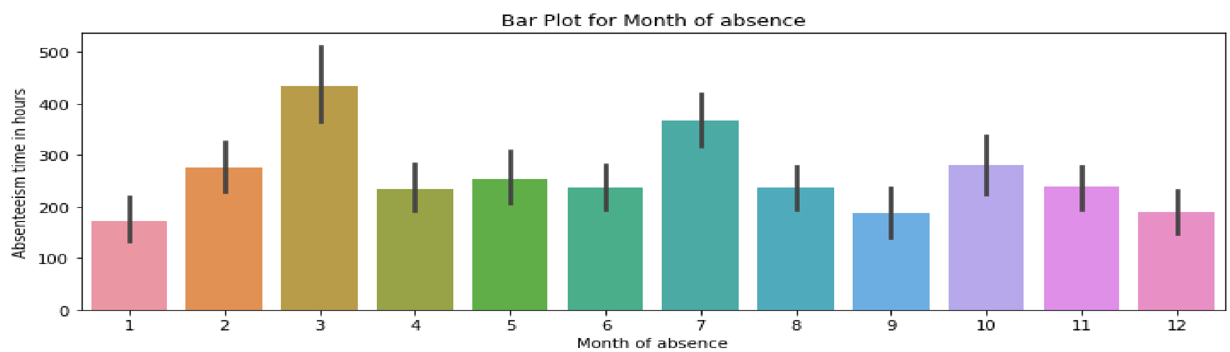


The Reasons for absence for high Absenteeism time in hours is given below-

- X Diseases of the respiratory system
- XI Diseases of the digestive system
- XIII Diseases of the musculoskeletal system and connective tissue
- XVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
- XIX Injury, poisoning and certain other consequences of external causes
- Patient follow-up (22), medical consultation (23), laboratory examination (25), unjustified absence (26), physiotherapy (27), dental consultation (28).

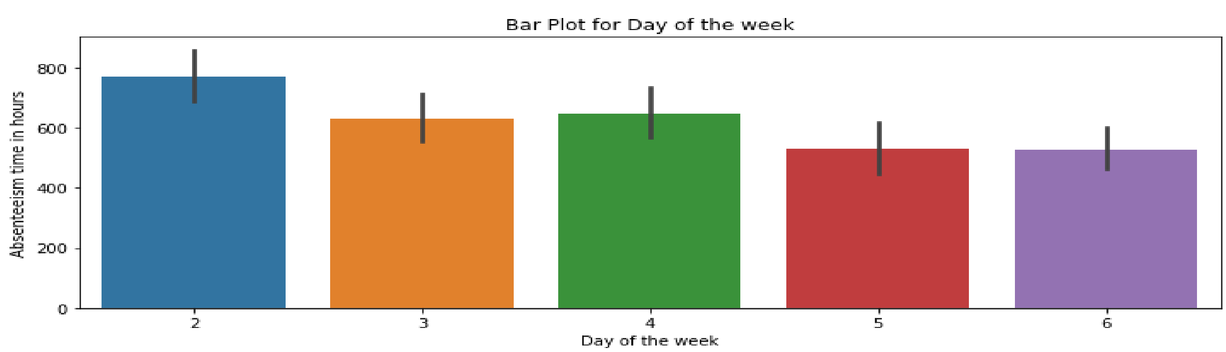
Above reasons are come under common diseases and consultation.

Month of absence



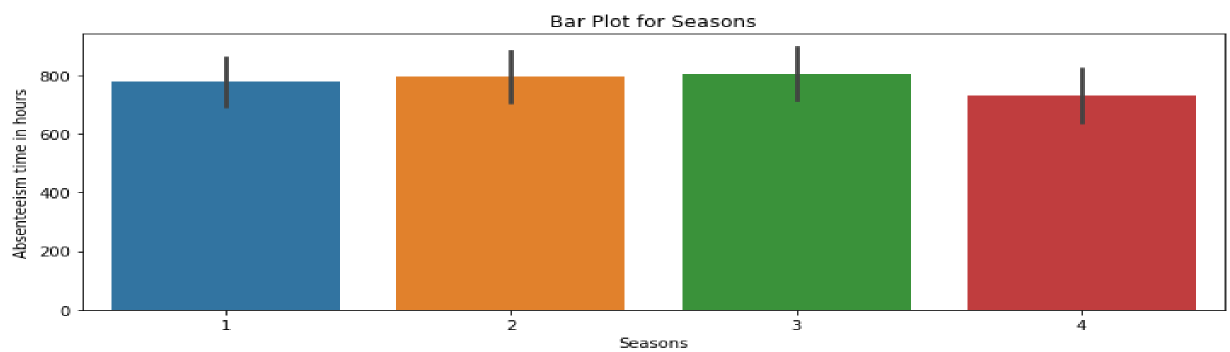
Month 3 and 7 having high absenteeism time in hours. In the data we did not have the date and year so can not conclude any decision.

Day of the week



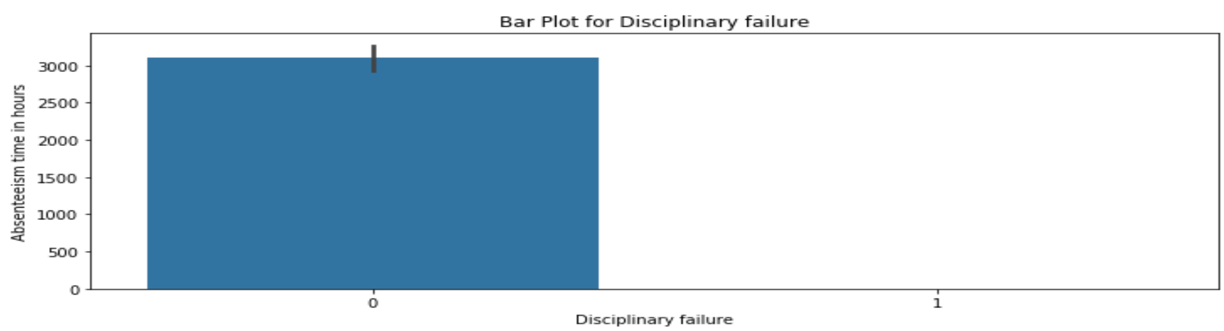
Day Monday has more absenteeism than other weekdays it show after a weekend employees tends to more absents.

Seasons



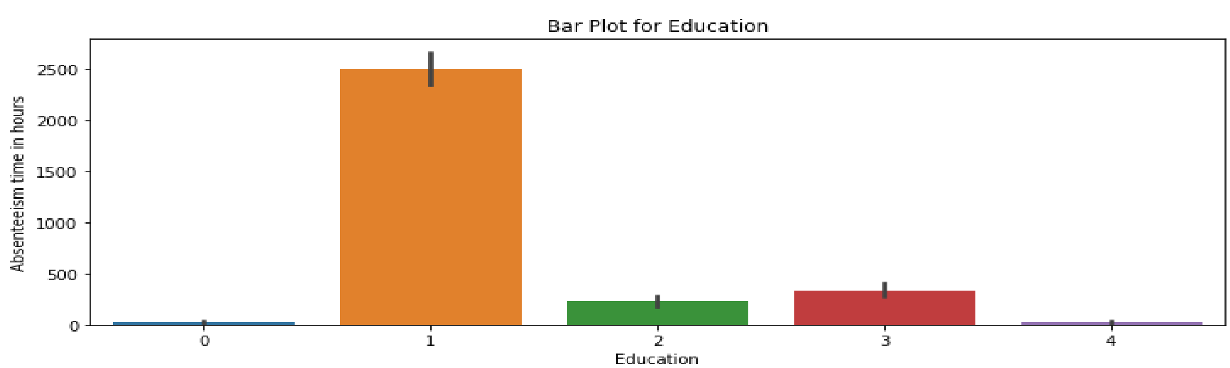
There is no impact seen on Absenteeism time in hours by season. All the season having arround same absenteeism.

Disciplinary failure



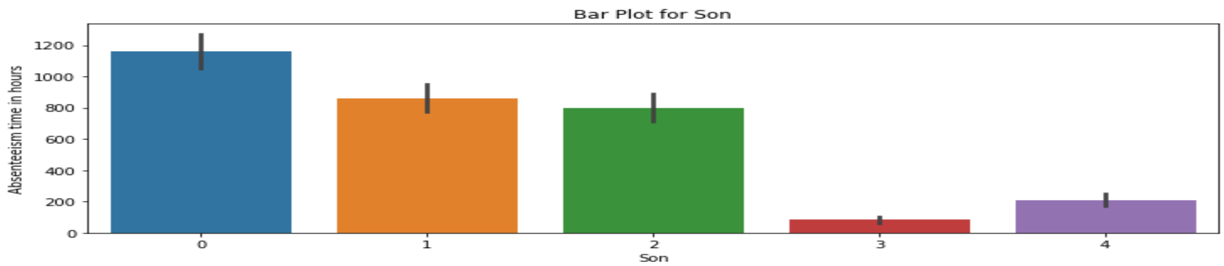
Disciplinary failure is the major role as Absenteeism time in hours is high.

Education



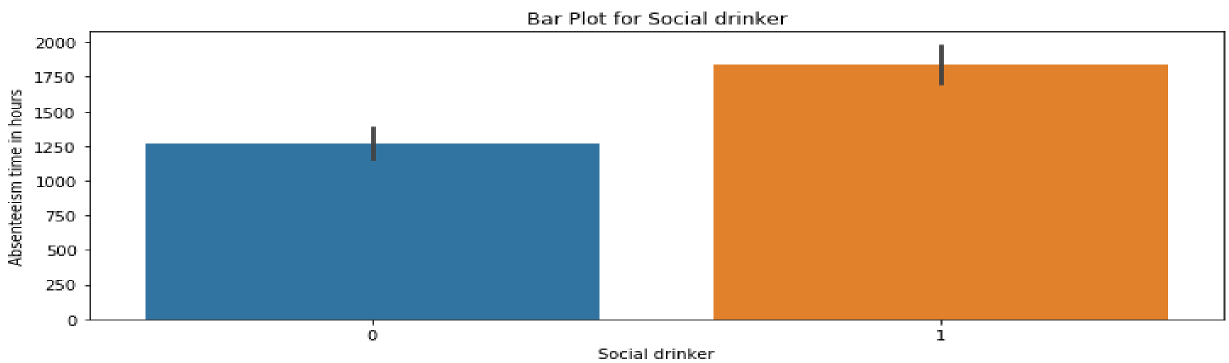
Employee having High School education is more tend to Absent from work.

Son

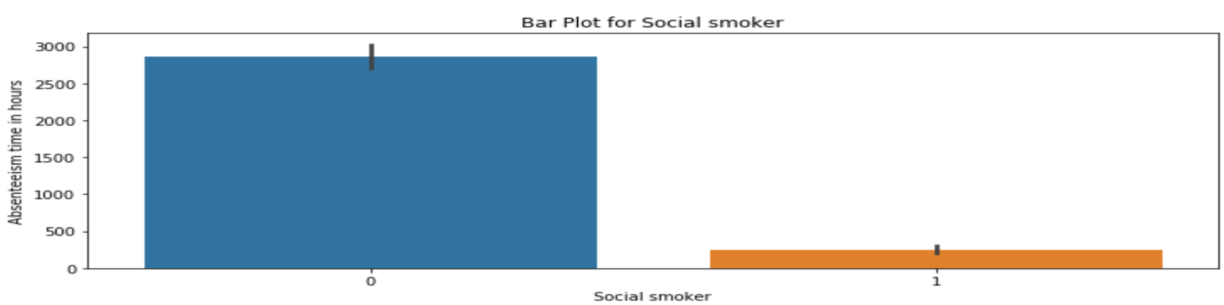


Employee having the no child or 1 or 2 Child are tend to more absent. It sees that the less family responsibilities on the employee having less work motivation.

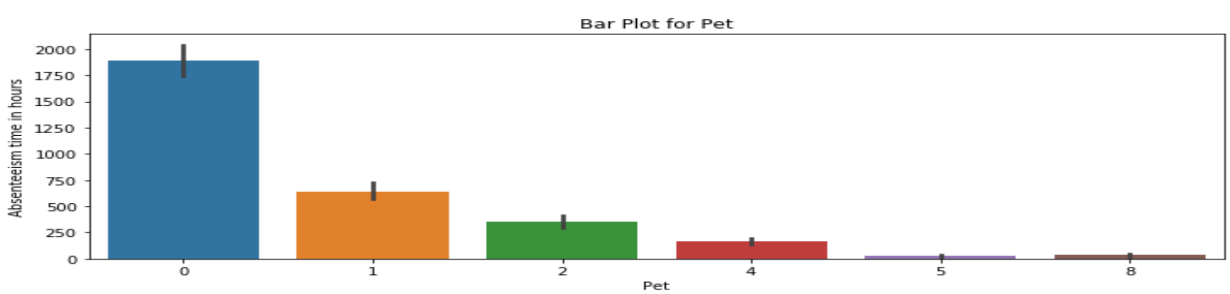
Social drinker



Social smoker



Pet

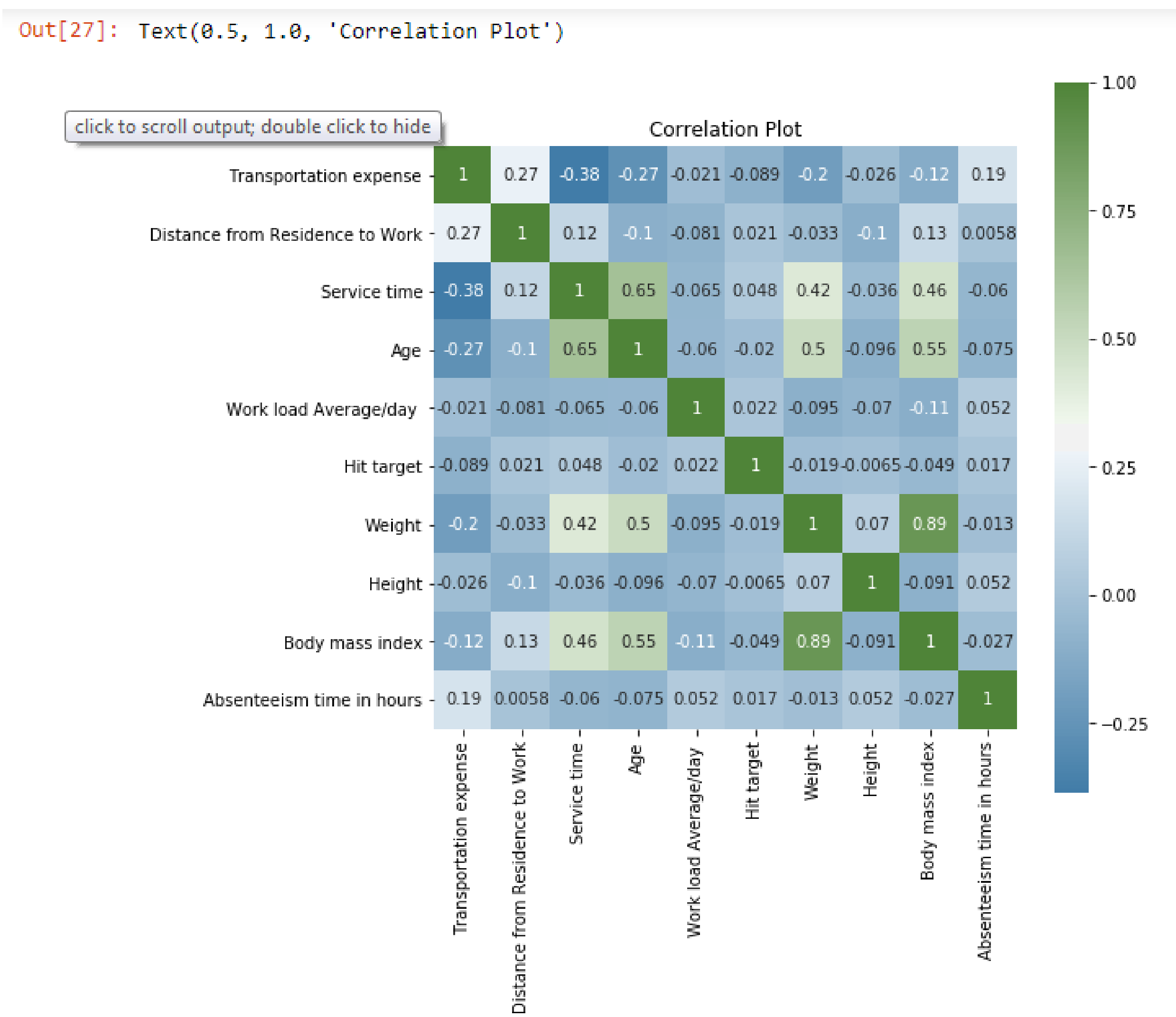


As per the trend of son, the employee having no pet or 1 and 2 pet are more absent.

2.1.4. Feature Selection

Selecting subset of relevant columns for the model construction is known as Feature Selection. We cannot use all the features because some features may be carrying the same information or irrelevant information which can increase overhead. To reduce overhead we adopt feature selection technique to extract meaningful features out of data. This in turn helps us to avoid the problem of multi collinearity.
In this project we have selected Correlation Analysis for numerical variable and ANOVA (Analysis of variance) for categorical variables.

Table 2.9: Correlation analysis for continuous variable with Heatmap and Correlation Plot



Weight and Body mass index are high correlation (>=.9) so we can drop Weight for model development. As weight is less dependent on target variable

Table 2.10: ANOVA Test for Categorical Variables

	sum_sq	df	F	PR(>F)
ID	1274.897076	33.0	3.80022	1.951747e-11
Residual	6923.083343	681.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Reason_for_absence	3467.035149	27.0	18.646756	8.085474e-65
Residual	4730.945271	687.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Month_of_absence	270.456504	11.0	2.180331	0.013903
Residual	7927.523915	703.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Day_of_the_week	67.287858	4.0	1.468952	0.209862
Residual	8130.692562	710.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Seasons	62.512894	3.0	1.821107	0.141888
Residual	8135.467525	711.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Disciplinary_failure	653.710625	1.0	61.781417	1.418044e-14
Residual	7544.269795	713.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Education	74.740434	4.0	1.633145	0.164045
Residual	8123.239986	710.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Son	279.745255	4.0	6.270941	0.000058
Residual	7918.235165	710.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Social_drinker	52.236841	1.0	4.57231	0.032832
Residual	8145.743579	713.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Social_smoker	15.748049	1.0	1.372286	0.241811
Residual	8182.232371	713.0	NaN	NaN

	sum_sq	df	F	PR(>F)
Pet	140.921514	5.0	2.480145	0.030663
Residual	8057.058906	709.0	NaN	NaN

As per the above result P value is greater than .05 for Day of the week, Season, Education, Social smoker day variables. So we can drop these variables as our target variable (Absenteeism time in hours) is not much dependent on these variables.

2.1.5. Feature Scaling:

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

Since our data is not uniformly distributed we will use Normalization as Feature Scaling Method for all continuous variables.

Table 2.11: Data Set after Feature Scaling

ID	Reason for absence	Month of absence	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target	Disciplinary failure	Son	Social drinker	Pet	Height	Body mass index	Absenteeism time in hours
11	26	7	0.657692	0.659574	0.521739	0.260870	0.244925	0.769231	0	2	1	1	0.7	0.526316	0.250
36	0	7	0.000000	0.170213	0.739130	1.000000	0.244925	0.769231	1	1	1	0	0.5	0.631579	0.000
3	23	7	0.234615	0.978723	0.739130	0.478261	0.244925	0.769231	0	0	1	0	0.5	0.631579	0.125
7	7	7	0.619231	0.000000	0.565217	0.521739	0.244925	0.769231	0	2	1	0	0.3	0.263158	0.250
11	23	7	0.657692	0.659574	0.521739	0.260870	0.244925	0.769231	0	2	1	1	0.7	0.578947	0.125

2.1.6. Data after EDA and preprocessing

We remove "Weight","Day of the week","Social smoker","Education","Seasons” .
Rest of the variables for further data analysis are-

continuous variable
ID
Transportation expense
Distance from Residence to Work
Service time
Age
Work load Average/day
Hit target
Height
Body mass index
Absenteeism time in hours

categorical variable
Reason for absence
Month of absence
Disciplinary failure
Son
Social drinker
Pet

target variable
Absenteeism time in hours

Table 2.12: Dataset after EDA and Preprocessing

ID	Reason for absence	Month of absence	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target	Disciplinary failure	Son	Social drinker	Pet	Height	Body mass index	Absenteeism time in hours
11	26	7	0.657692	0.659574	0.521739	0.260870	0.244925	0.769231	0	2	1	1	0.7	0.526316	0.250
36	0	7	0.000000	0.170213	0.739130	1.000000	0.244925	0.769231	1	1	1	0	0.5	0.631579	0.000
3	23	7	0.234615	0.978723	0.739130	0.478261	0.244925	0.769231	0	0	1	0	0.5	0.631579	0.125
7	7	7	0.619231	0.000000	0.565217	0.521739	0.244925	0.769231	0	2	1	0	0.3	0.263158	0.250
11	23	7	0.657692	0.659574	0.521739	0.260870	0.244925	0.769231	0	2	1	1	0.7	0.578947	0.125

```
(715, 16)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 715 entries, 0 to 714
Data columns (total 16 columns):
ID                                 object
Reason for absence                 object
Month of absence                   object
Transportation expense             float64
Distance from Residence to Work    float64
Service time                       float64
Age                                float64
Work load Average/day              float64
Hit target                         float64
Disciplinary failure               object
Son                                object
Social drinker                     object
Pet                                object
Height                             float64
Body mass index                    float64
Absenteeism time in hours          float64
dtypes: float64(9), object(7)
memory usage: 89.5+ KB
None
```

2.2. Model Development

After Data pre-processing the next step is to develop a model using a train or historical data, Which can perform to predict accurate result on test data or new data. Here we have tried with different model and will choose the model Which will provide the most accurate values.

2.2.1. Model Building

2.2.2.1. Decision Tree

Decision Tree is a supervised machine learning algorithm, which is used to predict the data for classification and regression. It accepts both continuous and categorical variables. A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each branch connects nodes with "and" and multiple branches are connected by "or". Extremely easy to understand by the business users. It provides its output in the form of rule, which can easily understood by a non-technical person also.

2.2.2.2. Random Forest

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations. It means to build each decision tree on random forest we are not going to use the same data. The higher no of trees in the random forest will give higher no of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important.

2.2.2.3. Liner Regression

Linear Regression is one of the statistical method of prediction. It is most common predictive analysis algorithm. It uses only for regression, means if the target variable is continuous than we can use linear regression machine learning algorithm.

2.2.2.4. Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, It produces a prediction model in the form of an ensemble of weak learner models and produce a strong learner with less misclassification.

2.2.2.5. Results

we have applied four algorithms on our dataset and calculate the Root Mean Square Error (RMSE) and R-Squared Value for all the models.

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. RMSE is an absolute measure of fit. RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. R-squared is a relative measure of fit. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. Value of R-squared between 0-1, where 0 means independent variable unable to explain the target variable and 1 means target variable is completely explained by the independent variable. So, Lower values of RMSE and higher value of R-Squared Value indicate better fit of model.

Table 2.7: PYTHON result after model development

Model Name	R-squared_Test	R-squared_Train	RMSE_Test	RMSE_Train
Decision Tree	0.080366	0.163039	0.227971	0.186789
Random Forest	0.414173	0.863474	0.181952	0.075441
Linear Regression	0.343839	0.520301	0.192565	0.141411
Gradient Boosting	0.381762	0.602112	0.186918	0.128789

Table 2.6: R result after model development

Model	RMSE_Train	RMSE_Test	R.Squared_Train	R.Squared_Test
Decision Tree for Regression	0.10640701	0.07343983	0.2549407	0.10595240
Random Forest	0.07445446	0.07309151	0.6908082	0.27922900
Linear Regression	0.10230492	0.10469365	0.3112788	0.09828519
Gradient Boosting	0.10483332	0.05844292	0.2909610	0.09155062

Now, we will tune our best models i.e. Random Forest with the help of hyperparameter tuning we would find optimum values for parameter used in function and would increase our accuracy.

2.2.2. Hyperparameter Tuning

In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed. In any machine learning algorithm, these parameters need to be initialized before training a model. Choosing appropriate hyperparameters plays a crucial role in the success of good model.
Since it makes a huge impact on the learned model. For example, if the learning rate is too low, the model will miss the important patterns in the data. If it is high, it may have collisions. we used two techniques of Hyperparameter in our model-

- Random Search
- Grid Search

2.2.2.1. Random Search Hyperparameter Tuning

Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model.
In this search pattern, random combinations of parameters are considered in every iteration. The chances of finding the optimal parameter are comparatively higher in random search because of the random search pattern where the model might end up being trained on the optimised parameters without any aliasing.

2.2.2.2. Grid Search Hyperparameter Tuning

Grid search is a technique which tends to find the right set of hyperparameters for the particular model. Hyperparameters are not the model parameters and it is not possible to find the best set from the training data. Model parameters are learned during training when we
optimise a loss function using something like a gradient descent. In this tuning technique, we simply build a model for every combination of various hyperparameters and evaluate each model. The model which gives the highest accuracy wins. The pattern followed here is similar to the grid, where all the values are placed in the form of a matrix. Each set of parameters is taken into consideration and the accuracy is noted. Once all the combinations are evaluated, the model with the set of parameters which give the top accuracy is considered to be the best.

Chapter 3

Conclusion

3.1. Model Evaluation

In the previous chapter we have applied four algorithms on our dataset and calculate the Root Mean Square Error (RMSE) and R-Squared Value for all the models.

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. RMSE is an absolute measure of fit. RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. R-squared is a relative measure of fit. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. Value of R-squared between 0-1, where 0 means independent variable unable to explain the target variable and 1 means target variable is completely explained by the independent variable. So, Lower values of RMSE and higher value of R-Squared Value indicate better fit of model.

Here, the result of each model in R and Python as-

Table 3.1: Python Final result after Hyperparameter Tuning

Model Name	R-squared_Test	R-squared_Train	RMSE_Test	RMSE_Train
Decision Tree	0.080366	0.163039	0.227971	0.186789
Random Forest	0.414173	0.863474	0.181952	0.075441
Linear Regression	0.343839	0.520301	0.192565	0.141411
Gradient Boosting	0.381762	0.602112	0.186918	0.128789
Random Search CV in Random Forest	0.412058	0.739898	0.182280	0.104128
Grid Search CV in Random Forest	0.330296	0.467146	0.194542	0.149040

Table 3.2: R Final result after Hyperparameter Tuning

Model	RMSE_Train	RMSE_Test	R.Squared_Train	R.Squared_Test
Decision Tree for Regression	0.10640701	0.07343983	0.2549407	0.10595240
Random Forest	0.07438751	0.07986267	0.6841084	0.25980668
Linear Regression	0.10230492	0.10469365	0.3112788	0.09828519
Gradient Boosting	0.10483332	0.06209095	0.2909610	0.09015433
Random Search CV in Random Forest	0.07512401	0.06175003	0.7180639	0.28211166
Grid Search CV in Random Forest	0.08111481	0.05444408	0.6910952	0.24477765

3.2. Model Selection

From the observation of all **RMSE Value** and **R-Squared** Value we have concluded that **Random Forest** has minimum value of RMSE and it's **R-Squared** Value is also maximum .Means, By Random forest algorithm predictor are explain 68% to the target variable on the test data.

3.3. Answers of asked questions

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

Monday has the highest absenteeism time in hours followed by Tuesday and Wednesday. **Looks like people don't want to go to work on time after a good weekend.** As Monday have the highest hours, may be company can extend the service hours for Friday and Thursday and decrease a bit on Monday by opening the office 1 or 2 hours later than usual.

Some employee with ID 3, 11, 28 are often absent from work, company should take action against them. Or even a warning to them might help.

When there is a disciplinary action, the absentees hours are very low almost negligible. As people take disciplinary actions seriously, they can implement a rule where a person being absent for more than 15 hours quarterly will be given a warning. After three warnings employer has the right to fire that employee based on professional ethics.

Company can also introduce a policy where in the Top 5 disciplined employees, holding the least Absentee's hours will be rewarded. This Reward can be in form of some Reward points that they can redeem later or some gift voucher. This action will encourage employees to strive for excellence in Discipline.

The maximum people taking the absent hours are from below mentioned medical conditions.

- X Diseases of the respiratory system
- XI Diseases of the digestive system
- XIII Diseases of the musculoskeletal system and connective tissue
- XVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
- XIX Injury, poisoning and certain other consequences of external causes
- patient follow-up (22), medical consultation (23), laboratory examination (25), unjustified absence (26), physiotherapy (27), dental consultation (28).

All these reasons are common health issues or consultation which people might give as an excuse as they don't have a medical certificate to show.

As the majority of the reason are consultation, company can organize a free health checkup once in 6 months to keep the track of the medical history of employee. This will also keep a good company environment for the employees and an added perk which can help the company loses in the important business hours.

we also observed that that people with no children or no pets tend to be absent more than people who have children or pets, it shows that these people are far from their home town or unmarried.

Note:- all the above observation graphs are shown in 2.3 Data observation

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

Assuming work load average per day is the target workload for that day we shall calculate its loss due to absenteeism time in hours by the formula given below.

Work loss = (workload per day* Absenteeism_in_hours) /24

By using above formula we can calculate monthly work loss in time (hrs.). Here, in below index we have calculated the work loss monthly by assuming that the trend will be same for the year 2011.

Table 3.2: PYTHON Output

Month of absence	Work load Average/day	Absenteeism time/month(hrs.)	Work loss per month
1	15707306	173	2277330
2	19454925	276	3132251
3	22596746	435	5113523
4	14567117	235	2668552
5	15391481	253	2592842
6	14249336	236	2653156
7	16258458	367	3871547
8	12380750	237	2331578
9	13572704	188	2127681
10	17812436	281	3151564
11	16743861	238	2824523
12	12353497	189	2034626

NOTE: ALL THE ABOVE GRAPHS AND TABLES ARE FROM JUPYTER NOTEBOOK IN THE REPORT

Chapter 4

Codes

4.1 R Code:

```
### clear environment ###
rm(list=ls())

### load the libraries  ###

library(ggplot2)
library(corrgram)
library(corrplot)


###set working directory###

setwd("D:/DATA SCIENCE STUDY METERIAL/Projects/Employee Absenteeism_Project")
getwd()


### Load Bike renting Data CSV file ###
library(xlsx)
df=read.xlsx("DATA set.xls",sheetIndex = 1)

head(df)

###########################################################################
####################   1.3 Expletory Data Analysis   #####################
###########################################################################


dim(df)      # checking the dimension of data frame.

str(df)      # checking datatypes of all columns.

# drop the observation where Absenteeism time in hour is NAN

df= df[(!df$Absenteeism.time.in.hours %in% NA),]

#store categorical and continuous Variable column names
col=colnames(df)
print(col)

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

con_var=c('Transportation.expense','Distance.from.Residence.to.Work','Service.time','Age',
          'Work.load.Average.day.','Hit.target','Weight','Height','Body.mass.index','Absenteeism.time.in.hours')


###########################################################################
####################   2.1. Data Preprocessing   ##########################
###########################################################################


###################  2.1.1. Missing Value Analysis  #######################

#Calculate missing values

miss_val = data.frame(apply(df,2,function(x){sum(is.na(x))}))
miss_val
miss_val$Columns = row.names(miss_val)
row.names(miss_val)=NULL
names(miss_val)[1]="Missing_percentage"
miss_val= miss_val[,c(2,1)]
miss_val$Missing_percentage = (miss_val$Missing_percentage/nrow(df)) * 100
miss_val = miss_val[order(-miss_val$Missing_percentage),]

#Missing value imputation
data=df
data$Body.mass.index[11]                 #Lets take one sample data for referance

#Actual value= 23
#Mean= 26.71
#Median= 25
#KNN= 23
```

```
#Mean method-
data$Body.mass.index[11]=NA
data$Body.mass.index[is.na(data$Body.mass.index)] = mean(data$Body.mass.index,
                                                   na.rm=TRUE)
data$Body.mass.index[11]
#Mean = 26.71

#Median Method-
data$Body.mass.index[11]=NA
data$Body.mass.index[is.na(data$Body.mass.index)]= median(data$Body.mass.index,na.rm=TRUE)
data$Body.mass.index[11]
#Median= 25

#KNN Imputation- #reload the data first
data$Body.mass.index[11]=NA
library(DMwR)  #Library for KNN
data= knnImputation (data,k = 5)
data$Body.mass.index[11]
#KNN=23

###################  2.1.2. outlier analysis #############################


#create Box plot for outlier analysis
cnames= con_var
for(i in 1:length(cnames)){
  assign(paste0("AB",i),ggplot(aes_string(x="Absenteeism.time.in.hours",y=(cnames[i])),data=subset(df))+
           geom_boxplot(outlier.color = "Red",outlier.shape = 18,outlier.size = 2,
                        fill="Purple")+theme_get()+
           stat_boxplot(geom = "errorbar",width=0.5)+
           labs(x="Absenteeism.time.in.hours",y=cnames[i])+
           ggtitle("Boxplot o f",cnames[i]))
}

gridExtra::grid.arrange(AB1,AB2,AB3,AB4,AB5,ncol=5)   # plot all graph
gridExtra::grid.arrange(AB6,AB7,AB8,AB9,AB10,ncol=5)
#Replace outliers with NA

for(i in con_var){
  print(i)
  outlier= df[,i][df[,i] %in% boxplot.stats(df[,i])$out]
  print(length(outlier))
  df[,i][df[,i] %in% outlier]=NA
}

sum(is.na(df))

#Impute outliers by KNN method

data= knnImputation (data,k = 5)


df=data

########################  2.1.3. Data Observation  ############################

#convert data into proper data type after KNN imputation

for (i in col){
  df[,i]=as.integer(df[,i])
}

for (i in cat_var){
  df[,i]=as.factor(df[,i])

}

str(df)                                #checking datatypes of all columns

summary(df[,con_var])                  # checking numerical variables
summary(df[,cat_var])                  # checking categorical variables
```

```
# final data after missing value and outlier analysis

write.csv(df,"Data after missing value and outlier.csv", row.names=FALSE )



##############################################################################
########################## VISULAIZATION ####################################
##############################################################################

# bar graph for categorical variables

plot_bar <- function(cat, y, fun){
  gp = aggregate(x = df[, y], by=list(cat=df[, cat]), FUN=fun)
  ggplot(gp, aes_string(x = 'cat', y = 'x'))+
    geom_bar(stat = 'identity',fill = "aquamarine3")+
    labs(y = y, x = cat)+theme(panel.background = element_rect("antiquewhite"))+
    theme(plot.title = element_text(size = 9))+
    ggtitle(paste("Bar plot for",y,"wrt to",cat))
}

for(i in 1:length(cat_var)) {
  assign(paste0("PB",i),plot_bar(cat_var[i],'Absenteeism.time.in.hours','sum'))
}

gridExtra::grid.arrange(PB1,PB2,PB3,ncol=3)
gridExtra::grid.arrange(PB4,PB5,PB6,ncol=3)
gridExtra::grid.arrange(PB7,PB8,PB9,ncol=3)



# histogram for continuous variables

hist_plot <- function(column, dataset){
  hist(x=dataset[,column],col="Green",xlab=column,ylab="density",
       main=paste("Histogram of ", column))
}

for(i in 1:length(con_var)) {
  assign(paste0("PH",i),hist_plot(con_var[i],dataset= df))
}


# Chacking VIF for skewness

cnames=con_var
library(propagate)
for(i in cnames){
  print(i)
  skew= skewness(df[,i])
  print(skew)
}

#########################  2.1.4.Feature Selection  ###############################


####  for continuous variables ####

# correlation plot for numerical feature

corrgram(df[,con_var], order = FALSE,
         upper.panel = panel.cor, text.panel = panel.txt,
         main = "Correlation Plot")



####  for categorical Variable  ####

#Anova analysis for categorical variable with target numeric variable-
for(i in cat_var){
  print(i)
  Anova_result= summary(aov(formula = Absenteeism.time.in.hours~df[,i],df))
  print(Anova_result)
}
```

```
#########################  2.1.5. Feature Scaling  ###############################

#as the variables are not unifomaly distributed we use normalization for Feature scaling

#Normalization-

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


######################  2.1.5.  Data after EDA and preprocessing  ################


df= subset(df,select= -c (Weight,Day.of.the.week,Social.smoker,Education,Seasons,Pet))


head(df)
dim(df)
str(df)

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

# change categorical to numeric making bin for regression model

cat_index=sapply(df,is.factor)
cat_data=df[,cat_index]
cat_var=colnames(cat_data)
library(dummies)
df= dummy.data.frame(df,cat_var)


##########################################################################
####################   2.2. Model Development   ##########################
##########################################################################


#######################  2.2.1 Model building  #######################

#clear all the data except final data set.

data=df
df=data

library(DataCombine)
rmExcept("data")


#Function for Error metrics to calculate the performance of model-

rmse= function(y,y1){
  sqrt(mean(abs(y-y1)^2))
}

#Function for r2 to calculate the goodness of fit of model-
rsquare=function(y,y1){
  cor(y,y1)^2
}


# devide the data in train and test

set.seed(123)
train_index= sample(1:nrow(data),0.8*nrow(data))
train= data[train_index,]
test= data[-train_index,]
```

```r
################# 2.2.1.desision tree for regression  #########################

library(rpart)

fit=rpart(Absenteeism.time.in.hours~.,data=train,method = "anova") #model development on train data

DT_test=predict(fit,test[,-96])             #predict test data
DT_train= predict(fit,train[,-96])          #predict train data

DT_RMSE_Test = rmse(test[,96],DT_test)      # RMSE calculation for test data
DT_RMSE_Train = rmse(train[,96],DT_train)   # RMSE calculation for train data

DT_r2_test=rsquare(test[,96],DT_test)       # r2 calculation for test data
DT_r2_train= rsquare(train[,96],DT_train)   # r2 calculation for train data


### 2.2.2. Random forest for regression ###

library(randomForest)

RF_model= randomForest(Absenteeism.time.in.hours~.,train,ntree=100,method="anova") #Model development on train data

RF_test= predict(RF_model,test[-96])       #Prediction on test data
RF_train= predict(RF_model,train[-96])     #Prediction on train data

RF_RMSE_Test=rmse(test[,96],RF_test)       #RMSE calculation of test data-
RF_RMSE_Train=rmse(train[,96],RF_train)    #RMSE calculation of train data

RF_r2_test=rsquare(test[,96],RF_test)      #r2 calculation for test data-
RF_r2_train=rsquare(train[,96],RF_train)   #r2 calculation for train data-



###  2.2.3. Linear Regression ###


LR_model= lm(Absenteeism.time.in.hours~.,train)              #Model devlopment on train data
summary(LR_model)

LR_test= predict(LR_model,test[-96])       #prediction on test data
LR_train= predict(LR_model,train[-96])     #prediction on train data

LR_RMSE_Test=rmse(test[,96],LR_test)       #RMSE calculation of test data
LR_RMSE_Train=rmse(train[,96],LR_train)    #RMSE calculation of train data

LR_r2_test=rsquare(test[,96],LR_test)      #r2 calculation for test data
LR_r2_train=rsquare(train[,96],LR_train)   #r2 calculation for train data


### 2.2.4. Gradient Boosting ###

library(gbm)

GB_model = gbm(Absenteeism.time.in.hours~., data = train, n.trees = 100, interaction.depth = 2) #Model devlopment on train da

GB_test = predict(GB_model, test[-96], n.trees = 100)      #prediction on test data
GB_train = predict(GB_model, train[-96], n.trees = 100)    #prediction on train data

GB_RMSE_Test=rmse(test[,96],GB_test)
GB_RMSE_Train=rmse(train[,96],GB_train)                    #Mape calculation of train data

GB_r2_test=rsquare(test[,96],GB_test)                      #r2 calculation for test data-
GB_r2_train=rsquare(train[,96],GB_train)                   #r2 calculation for train data-
```

```r
Result= data.frame('Model'=c('Decision Tree for Regression','Random Forest',
                             'Linear Regression','Gradient Boosting'),
                   'RMSE_Train'=c(DT_RMSE_Train,RF_RMSE_Train,LR_RMSE_Train,GB_RMSE_Train),
                   'RMSE_Test'=c(DT_RMSE_Test,RF_RMSE_Test,LR_RMSE_Test,GB_RMSE_Test),
                   'R-Squared_Train'=c(DT_r2_train,RF_r2_train,LR_r2_train,GB_r2_train),
                   'R-Squared_Test'=c(DT_r2_test,RF_r2_test,LR_r2_test,GB_r2_test))

Result          #Random forest and Gradient Bosting have best fit model for the data.


########################  2.2.2. Hyperparameter Tuning  ########################

#Random Search CV in Random Forest

library(caret)

control = trainControl(method="repeatedcv", number=3, repeats=1,search='random')


RRF_model = caret::train(Absenteeism.time.in.hours~., data=train, method="rf",trControl=control,tuneLength=1)   #model 
best_parameter = RRF_model$bestTune                              #Best fit parameters
print(best_parameter)
#mtry=17 As per the result of best_parameter

RRF_model = randomForest(Absenteeism.time.in.hours ~ .,train, method = "rf", mtry=17,importance=TRUE)       #build mod

RRF_test= predict(RRF_model,test[-96])                           #Prediction on test data
RRF_train= predict(RRF_model,train[-96])                         #Prediction on train data


RRF_RMSE_Test = rmse(test[,96],RRF_test)                         #Mape calculation of test data
RRF_RMSE_Train = rmse(train[,96],RRF_train)                      #Mape calculation of train data


RRF_r2_test=rsquare(test[,96],RRF_test)                           #r2 calculation for test data
RRF_r2_train= rsquare(train[,96],RRF_train)                       #r2 calculation for train data


# Grid Search CV in Random Forest

control = trainControl(method="repeatedcv", number=3, repeats=3, search="grid")
tunegrid = expand.grid(.mtry=c(6:18))


GRF_model= caret::train(Absenteeism.time.in.hours~.,train, method="rf", tuneGrid=tunegrid, trControl=control)     #mo
best_parameter = GRF_model$bestTune                              #Best fit parameters
print(best_parameter)
#mtry=8 As per the result of best_parameter

GRF_model = randomForest(Absenteeism.time.in.hours ~ .,train, method = "anova", mtry=9)               #build mod

GRF_test= predict(GRF_model,test[-96])                           #Prediction on test data
GRF_train= predict(GRF_model,train[-96])                         #Prediction on train data


GRF_RMSE_Test = rmse(test[,96],GRF_test)                         #Mape calculation of test data
GRF_RMSE_Train = rmse(train[,96],GRF_train)                      #Mape calculation of train data


GRF_r2_test=rsquare(test[,96],GRF_test)                          #r2 calculation for test data
GRF_r2_train= rsquare(train[,96],GRF_train)                      #r2 calculation for train data
```

```
final_result= data.frame('Model'=c('Decision Tree for Regression','Random Forest','Linear Regression',
                                   'Gradient Boosting','Random Search CV in Random Forest',
                                   'Grid Search CV in Random Forest'),
                         'RMSE_Train'=c(DT_RMSE_Train,RF_RMSE_Train,LR_RMSE_Train,GB_RMSE_Train,
                                        RRF_RMSE_Train,GRF_RMSE_Train),
                         'RMSE_Test'=c(DT_RMSE_Test,RF_RMSE_Test,LR_RMSE_Test,GB_RMSE_Test,
                                       RRF_RMSE_Test,GRF_RMSE_Test),
                         'R-Squared_Train'=c(DT_r2_train,RF_r2_train,LR_r2_train,GB_r2_train,
                                             RRF_r2_train,GRF_r2_train),
                         'R-Squared_Test'=c(DT_r2_test,RF_r2_test,LR_r2_test,GB_r2_test,
                                            RRF_r2_test,GRF_r2_test))

print(final_result)

###############################################################################################
#2. How much losses every month can we project in 2011 if same trend of absenteeism continues?
###############################################################################################
df1= read.csv('Data after missing value and outlier.csv')
colnames(df1)
data_work_loss=subset(df1,select= c(Month.of.absence, Work.load.Average.day., Absenteeism.time.in.hours))
data_work_loss$Work_loss_per_day=
  (data_work_loss$Work.load.Average.day./24)*data_work_loss$Absenteeism.time.in.hours
Monthly_loss= aggregate(data_work_loss$Work_loss_per_day, by= list(data_work_loss$Month.of.absence), FUN=sum)
Monthly_loss
```

4.2 Python Code (Jupyter Notebook)

```
# Load Libs
import os
import numpy as np
import pandas as pd
from fancyimpute import KNN
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# set working dir
os.chdir('D:\\DATA SCIENCE STUDY METERIAL\\Projects\\Employee Absenteeism_Project')
os.getcwd()
```

```
#load dataset
df=pd.read_excel('DATA set.xls')
df.head()
```

1.3. Exploratory Data Analysis

```
print(df.shape)
print(df.info())
```

```
# drop the observation where Absenteeism time in hour is NAN
df=df.drop(df[df['Absenteeism time in hours'].isnull()].index)
df=df.drop(df[df['Month of absence']==0].index)
print( df.shape)
print(df.info())
```

```
col=df.columns
print(col)
cat_var=['ID','Reason for absence', 'Month of absence', 'Day of the week','Seasons','Disciplinary failure',
         'Education', 'Son', 'Social drinker','Social smoker', 'Pet']
con_var=['Transportation expense', 'Distance from Residence to Work','Service time', 'Age',
         'Work load Average/day ', 'Hit target','Weight', 'Height', 'Body mass index',
       'Absenteeism time in hours']
```

2.1. Data Preprocessing

2.1.1. Missing Value Analysis

```
#missing val

missing_val=pd.DataFrame(df.isnull().sum()).reset_index()
missing_val=missing_val.rename(columns={'index':'variable',0:'missing_count'})
missing_val['missing_Per']=(missing_val['missing_count']*100)/len(df)
missing_val.sort_values(by='missing_Per',ascending=False).reset_index(drop=True)
```

```
data=df.copy()  # make a copy of data
```

```
data['Body mass index'][0]

#actual value= 30.0
#by mean=26.68
#by median=25.0
#by KNN = 29.13
```

```
# Replace NA with mean
data['Body mass index'][0]=np.nan
data['Body mass index']=data['Body mass index'].fillna(data['Body mass index'].mean())
data['Body mass index'][0]
```

```
# Replace NA with mean
data['Body mass index'][0]=np.nan
data['Body mass index']=data['Body mass index'].fillna(data['Body mass index'].median())
data['Body mass index'][0]
```

```
# Replace NA with KNN
data['Body mass index'][0]=np.nan
data=pd.DataFrame(KNN(k=5).fit_transform(data),columns=data.columns)
data['Body mass index'][0]
```

```
#convert data into proper data type after KNN imputation
for i in col:
    data[i]=data[i].astype(int)

for i in cat_var:
    data[i]=data[i].astype(object)
```

```
# data after missing value treatment by KNN
df=data.copy()
df.isnull().sum()
```

2.1.2. Outlier Analysis

```
#create Box plot for outlier analysis
for i in con_var:
    print(i)
    sns.boxplot(data[i])
    plt.xlabel(i)
    plt.ylabel("values")
    plt.title("Boxplot of "+i)
    plt.show()
```

```
#calculate iqr, lower fence and upper fence-
for i in con_var:
    print(i)
    q75,q25= np.percentile(data.loc[:,i],[75,25])
    iqr= q75-q25
    minimum= q25-(iqr*1.5)
    maximum= q75+(iqr*1.5)
    print("min= "+str(minimum))
    print("max= "+str(maximum))
    print("IQR= "+str(iqr))

#replace outliers with NA-

    data.loc[df[i]<minimum,i]=np.nan
    data.loc[df[i]>maximum,i]=np.nan
```

```
#impute outlier with KNN

data=pd.DataFrame(KNN(k=5).fit_transform(data),columns=data.columns)
```

```
data.isnull().sum()
```

```
#convert data into proper data type after KNN imputation
for i in col:
    data[i]=data[i].astype(int)

for i in cat_var:
    data[i]=data[i].astype(object)

data.info()
```

```
# final data after missing value and outlier analysis

df=data.copy()
df.to_csv("Data after missing value and outlier.csv",index=False)
```

2.1.3. Data observation

```
df[con_var].describe().transpose()
```

```
#df[cat_var].describe().transpose()
#unique values
for i in cat_var:
    print('variable {} having {} unique values that are:{}'.format(i,df[i].nunique(),df[i].sort_values().unique()))
    #print(df[i].value_counts())
```

```
pd.set_option('max_info_rows',70)
for i in cat_var:
    print(i)
    print(df[i].sort_values().value_counts())
```

Visualization

```
# histogram for continuous variables-
for i in con_var:
    print(i)
    sns.distplot(df[i],bins='auto')
    plt.ylabel('Density')
    plt.title('Histogram for '+i)
    plt.show()
```

```
for i in con_var:
    print(i)
    sns.jointplot(x=i,y='Absenteeism time in hours',data= df)
    plt.ylabel('Absenteeism time in hours')
    plt.title('scatter plt for '+i)
    plt.show()
    plt.tight_layout
```

```
# Bar plot for categorical variables
for i in cat_var:
    print(i)
    fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize= (12,4), squeeze=False)
    sns.barplot(x=i,y='Absenteeism time in hours',data= df,estimator=np.sum,ax=ax[0][0])
    plt.ylabel('Absenteeism time in hours')
    plt.title('Bar Plot for '+i)
    plt.show()
    plt.tight_layout

plt.tight_layout
```

```
# Bar plot with statistic mean as count is diffrent for levels
for i in cat_var:
    print(i)
    fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize= (12,4), squeeze=False)
    sns.barplot(x=i,y='Absenteeism time in hours',data= df,estimator=np.mean,ax=ax[0][0])
    plt.ylabel('Absenteeism time in hours')
    plt.title('Bar Plot for '+i)
    plt.show()
    plt.tight_layout

plt.tight_layout
```

2.1.4. Feature Selection

```
#correlation analysis for numeric variables-

#extract only numeric variables in dataframe for correlation-
df_corr= df.loc[:,con_var]

#generate correlation matrix-
corr_matrix= df_corr.corr()
corr_matrix
```

```
#correlation plot-
f,ax= plt.subplots(figsize=(8,8))

#plot-
sns.heatmap(corr_matrix,mask=np.zeros_like(corr_matrix,dtype=np.bool),cmap=sns.diverging_palette(240,120,as_cmap=True),
            square=True,ax=ax,annot=True)
plt.title("Correlation Plot")
```

```
data=df.copy()
# Replacing the white spaces " " in the feature name with "_"
for i in data.columns:
    data = data.rename(index=str, columns={i: i.replace(" ", "_")})
cat_var1=['ID','Reason_for_absence', 'Month_of_absence', 'Day_of_the_week','Seasons',
         'Disciplinary_failure', 'Education', 'Son', 'Social_drinker','Social_smoker', 'Pet']
data.columns
```

```
#Anova analysis for categorical variable with target numeric variable

import statsmodels.api as sm
from statsmodels.formula.api import ols

label = 'Absenteeism_time_in_hours'
for  i in cat_var1:
    frame = label + ' ~ ' + i
    model = ols(frame,data=data).fit()
    anova = sm.stats.anova_lm(model, typ=2)
    print(anova)
```

2.1.5. Feature Scaling

```
#as the variables are not unifomaly distributed we use normalization for Feature scaling
#Normalization-

for i in con_var:
    print(i)
    df[i]= (df[i]-min(df[i]))/(max(df[i])-min(df[i]))
    print(df[i])
```

2.1.6. Data after EDA and preprocessing

```
df = df.drop(["Weight","Day of the week","Social smoker","Education","Seasons"],axis=1)
```

```
print(df.shape)
print(df.info())
```

```
df.shape
df.to_csv("Absenteeism_Pre_processed_Data.csv",index=False)
df.head()
```

2.2. Model Development

2.2.1. Models building

```
# change categorical to numeric making bin for regression model
df= pd.get_dummies(df,columns=['ID','Reason for absence', 'Month of absence','Disciplinary failure',
          'Son', 'Social drinker', 'Pet'])
```

```
#copy data
data=df.copy()
```

```
#Import libraries-
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
#split data for predictor and target seperatly-
X= df.drop(['Absenteeism time in hours'],axis=1)
y= df['Absenteeism time in hours']
```

```
#divide data into train and test part
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.20,random_state=0)
```

2.2.1.desision tree for regression

```
from sklearn.tree import DecisionTreeRegressor                  #import libraries

DT_model= DecisionTreeRegressor(max_depth=2).fit(X_train,y_train)   #Decision tree for regression

DT_test= DT_model.predict(X_test)                               #Model prediction on test data
DT_train= DT_model.predict(X_train)                             #Model prediction on train data

RMSE_test=np.sqrt(mean_squared_error(y_test, DT_test))          #Model performance on test data
RMSE_train=np.sqrt(mean_squared_error(y_train,DT_train))        #Model performance on train data


r2_test=r2_score(y_test,DT_test)                                #r2 value for test data
r2_train= r2_score(y_train,DT_train)                            #r2 value for train data


print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
# result
df1= {'Model Name': ['Decision Tree'],'RMSE_Train':[RMSE_train],'RMSE_Test':[RMSE_test],'R-squared_Train':[r2_train],
     'R-squared_Test':[r2_test]}
result1= pd.DataFrame(df1)
```

2.2.2. Random forest for regression

```
from sklearn.ensemble import RandomForestRegressor                     #import libraris

RF_model= RandomForestRegressor(n_estimators=100).fit(X_train,y_train)      #Random Forest for regression


RF_test= RF_model.predict(X_test)                                      #model prediction on test data
RF_train= RF_model.predict(X_train)                                    #model prediction on train data
```

```
RMSE_test=np.sqrt(mean_squared_error(y_test, RF_test))          #Model performance on test data
RMSE_train=np.sqrt(mean_squared_error(y_train,RF_train))        #Model performance on train data


r2_test=r2_score(y_test,RF_test)                                #r2 value for test data
r2_train= r2_score(y_train,RF_train)                            #r2 value for train data


print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
#result
df2= {'Model Name': ['Random Forest '],'RMSE_Train':[RMSE_train],'RMSE_Test':[RMSE_test],'R-squared_Train':[r2_train],
      'R-squared_Test':[r2_test]}
result2= pd.DataFrame(df2)
```

```
#append the results
result= result1.append(result2)
result
```

2.2.3. Linear Regression

```
import statsmodels.api as sm                                    #import libraries

LR_model= sm.OLS(y_train,X_train).fit()                         #Linear Regression model for regression
LR_test= LR_model.predict(X_test)                                 #model prediction on test data
LR_train= LR_model.predict(X_train)                               #model prediction on train data
```

```
RMSE_test=np.sqrt(mean_squared_error(y_test, LR_test))          #Model performance on test data
RMSE_train=np.sqrt(mean_squared_error(y_train,LR_train))        #Model performance on train data


r2_test=r2_score(y_test,LR_test)                                #r2 value for test data
r2_train= r2_score(y_train,LR_train)                            #r2 value for train data


print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
df3= {'Model Name': ['Linear Regression '],'RMSE_Train':[RMSE_train],'RMSE_Test':[RMSE_test],'R-squared_Train':[r2_train],
      'R-squared_Test':[r2_test]}
result3= pd.DataFrame(df3)
```

```
result= result.append(result3)
result
```

2.2.4. Gradient Boosting

```
from sklearn.ensemble import GradientBoostingRegressor          #import libraries

GB_model = GradientBoostingRegressor().fit(X_train, y_train)    #Gradient Boosting for regression

GB_test= GB_model.predict(X_test)                               #model prediction on test data
GB_train= GB_model.predict(X_train)                             #model prediction on train data
```

```
RMSE_test=np.sqrt(mean_squared_error(y_test, GB_test))           #Model performance on test data
RMSE_train=np.sqrt(mean_squared_error(y_train,GB_train))         #Model performance on train data


r2_test=r2_score(y_test,GB_test)                                 #r2 value for test data
r2_train= r2_score(y_train,GB_train)                             #r2 value for train data


print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
df4= {'Model Name': ['Gradient Boosting '],'RMSE_Train':[RMSE_train],'RMSE_Test':[RMSE_test],'R-squared_Train':[r2_train],
      'R-squared_Test':[r2_test]}
result4= pd.DataFrame(df4)
```

```
result= result.append(result4)
result
```

2.2.2. Hyperparameter Tuning

Random Search CV in Random Forest

```
from sklearn.model_selection import RandomizedSearchCV                          #import libraries

RandomRandomForest = RandomForestRegressor(random_state = 0)
n_estimator = list(range(1,100,2))
depth = list(range(1,20,2))
random_search = {'n_estimators':n_estimator, 'max_depth': depth}

#Random Grid Random Forest model-
RRF_model= RandomizedSearchCV(RandomRandomForest,param_distributions= random_search,n_iter=3,cv=10)
RRF_model= RRF_model.fit(X_train,y_train)

best_parameters = RRF_model.best_params_                                    #Best parameters for model
best_model = RRF_model.best_estimator_                                      #Best model

RRF_test = best_model.predict(X_test)                                       #Model prediction on test data
RRF_train = best_model.predict(X_train)                                     #Model prediction on train data

RMSE_test=np.sqrt(mean_squared_error(y_test, RRF_test))          #Model performance on test data
RMSE_train=np.sqrt(mean_squared_error(y_train,RRF_train))        #Model performance on train data

r2_test=r2_score(y_test,RRF_test)                                           #r2 value for test data
r2_train= r2_score(y_train,RRF_train)                                       #r2 value for train data

print("Best Parameter="+str(best_parameters))
print("Best Model="+str(best_model))
print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
df5= {'Model Name': ['Random Search CV in Random Forest '],'RMSE_Train':[RMSE_train],
      'RMSE_Test':[RMSE_test],'R-squared_Train':[r2_train],'R-squared_Test':[r2_test]}
result5= pd.DataFrame(df5)
```

```
result= result.append(result5)
result
```

```
from sklearn.model_selection import GridSearchCV                            #import libraries

GridRandomForest= RandomForestRegressor(random_state=0)
n_estimator = list(range(1,20,2))
depth= list(range(1,20,2))
grid_search= {'n_estimators':n_estimator, 'max_depth': depth}
```

```
#Grid Search CV Random Forest model-
GRF_model= GridSearchCV(GridRandomForest,param_grid=grid_search,cv=10)
GRF_model= GRF_model.fit(X_train,y_train)

best_parameters = GRF_model.best_params_                                  #Best parameters for model

best_model = GRF_model.best_estimator_                                    #Best model

GRF_test = best_model.predict(X_test)                                     #Model prediction on test data
GRF_train = best_model.predict(X_train)                                   #Model prediction on train data


RMSE_test=np.sqrt(mean_squared_error(y_test, GRF_test))              #Model performance on test data
RMSE_train=np.sqrt(mean_squared_error(y_train,GRF_train))            #Model performance on train data

r2_test=r2_score(y_test,GRF_test)                                         #r2 value for test data
r2_train= r2_score(y_train,GRF_train)                                     #r2 value for train data

print("Best Parameter="+str(best_parameters))
print("Best Model="+str(best_model))
print("Root Mean Square Rate for train data="+str(RMSE_train))
print("Root Mean Square Rate for test data="+str(RMSE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
```

```
df
```

```
df6= {'Model Name': ['Grid Search CV in Random Forest '],'RMSE_Train':[RMSE_train],'RMSE_Test':[RMSE_test],
      'R-squared_Train':[r2_train],'R-squared_Test':[r2_test]}
result6= pd.DataFrame(df6)
```


```
df6= {'Model Name': ['Grid Search CV in Random Forest '],'RMSE_Train':[RMSE_train],'RMSE_Test':[RMSE_test],
      'R-squared_Train':[r2_train],'R-squared_Test':[r2_test]}
result6= pd.DataFrame(df6)
```


```
result= result.append(result6)
result
```

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

```
df1=pd.read_csv("Data after missing value and outlier.csv")
df1.head()
```

```
#work_loss= (Work load perday/24)*Absenteeism time in hour
```

```
data_work_loss=df1[['Month of absence','Work load Average/day ','Absenteeism time in hours']]
```

```
data_work_loss['Work loss per day']=(data_work_loss['Work load Average/day ']/24)*data_work_loss['Absenteeism time in hours']
```

```
Monthly_loss = data_work_loss.groupby(by='Month of absence').sum().astype(int)
```

```
Monthly_loss= Monthly_loss.rename(columns={'Absenteeism time in hours': 'Absenteeism time/month(hrs.)',
                              'Work loss per day': 'Work loss per month'})
```

```
Monthly_loss
```

References-

1. For Data Cleaning and Model Development -
https://edwisor.com/career-data-scientist
2. For Visualization –
https://www.udemy.com/python-for-data-science-and-machine-learningbootcamp/

THANK YOU