Employee Absenteeism

SWAROOP H

August 2019

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DESCRIPTION

Absenteeism is the absence of an employee from work. It's a major problem faced by many companies today. We know that human capital plays an important role in any business. Being able to predict the employee absenteeism can prevent company from severe loss. It is very important to know the cause of absenteeism among the employee. And the changes company should bring to reduce the absenteeism. One of the best ways to make the predictions is with the help of machine learning techniques.

1.2 PROBLEM STATEMENT

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

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

The problem statement is to analyse the cause of absenteeism and predict the every month losses in 2011 due to absenteeism. Our task is to build a regression model which will predict the absenteeism in hours based on the employee attributes and information. Although, the problem statement is a Multivariate Time-Series Problem. We will approach it as a Regression Problem.

Our first objective is to find the patterns which leads to number of absenteeism and how the changes can help the company reduce that. The aim of this project is to predict the factors to reduce the number of absenteeism and the work loss company is going to face next year if same trend of absenteeism continues. To make the predictions we used R and Python codes and algorithms.

1.3 DATA OVERVIEW

We are provided with the data of employee absenteeism for three years (July 2007 to July 2010), as we need to forecast the absentee's hours for 2011. Given below is a sample of the data set that we are using to find the trend in the absenteeism.

Table 1.1: Sample of Employee Absenteeism Dataset (Column 1 to 10)

ID	Reason for absence	Month of absence	Day of the week	Seasons	Transpo rtation expense	Distance from Residence to Work	Service time	Age	Work load Average
11	26	7	3	1	289	36	13	33	2,39,554
36	0	7	3	1	118	13	18	50	2,39,554
3	23	7	4	1	179	51	18	38	2,39,554
7	7	7	5	1	279	5	14	39	2,39,554
11	23	7	5	1	289	36	13	33	2,39,554
3	23	7	6	1	179	51	18	38	2,39,554

Table 1.2: Sample of Employee Absenteeism Dataset (Column 11 to 21)

Hit target	Disciplinary failure	Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
97	0	1	2	1	0	1	90	172	30	4
97	1	1	1	1	0	0	98	178	31	0
97	0	1	0	1	0	0	89	170	31	2
97	0	1	2	1	1	0	68	168	24	4
97	0	1	2	1	0	1	90	172	30	2
97	0	1	0	1	0	0	89	170	31	NA

As we look at the data, 740 observations of 21 variables, In which 20 are independent variables and 1 (Absenteeism time in hours) is dependent variable. Since our target variable is continuous in nature, this is a regression problem.

Raw structure:

```
> str(data)
'data.frame':  740 obs. of  21 variables:
 $ ID                              : num  11 36 3 7 11 3 10 20 14 1 ...
 $ Reason.for.absence              : num  26 0 23 7 23 23 22 23 19 22 ...
 $ Month.of.absence                : num  7 7 7 7 7 7 7 7 7 7 ...
 $ Day.of.the.week                 : num  3 3 4 5 5 6 6 6 2 2 ...
 $ Seasons                         : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Transportation.expense          : num  289 118 179 279 289 179 NA 260
 $ Distance.from.Residence.to.Work: num  36 13 51 5 36 51 52 50 12 11 ...
 $ Service.time                    : num  13 18 18 14 13 18 3 11 14 14 ...
 $ Age                             : num  33 50 38 39 33 38 28 36 34 37 ...
 $ Work.load.Average.day.          : num  239554 239554 239554 239554…
 $ Hit.target                      : num  97 97 97 97 97 97 97 97 97 97 ...
 $ Disciplinary.failure            : num  0 1 0 0 0 0 0 0 0 0 ...
 $ Education                       : num  1 1 1 1 1 1 1 1 1 3 ...
 $ Son                             : num  2 1 0 2 2 0 1 4 2 1 ...
 $ Social.drinker                  : num  1 1 1 1 1 1 1 1 1 0 ...
 $ Social.smoker                   : num  0 0 0 1 0 0 0 0 0 0 ...
 $ Pet                             : num  1 0 0 0 1 0 4 0 0 1 ...
 $ Weight                          : num  90 98 89 68 90 89 80 65 95 88 ...
 $ Height                          : num  172 178 170 168 172 170 172 168
 $ Body.mass.index                 : num  30 31 31 24 30 31 27 23 25 29 ...
 $ Absenteeism.time.in.hours       : num  4 0 2 4 2 NA 8 4 40 8 ...
```

Attribute Information:

1. Individual identification (ID): 1 to 36 employee ID
2. Reason for absence (ICD). ***Refer****:* ***Appendix A:***
3. Month of absence: 1 to 12(Jan-Dec)
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: (years of work experience or service)
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

METHODOLOGY

2.0 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is the first step in our data analysis process. Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

We do EDA by taking a broad look at patterns, trends, outliers, unexpected results and so on in our existing data, using visual and quantitative methods to get a sense of the story this tells. To start with this process, we will first have to identify the variable, then we look at Data Visualization, Data preparation and cleaning, etc.

2.1 Variable Identification

First we need to identify *Predictor* and *Target* variables. Next, identify the data type and category of the variables.

In our dataset we have 21 variables out of which *'Absenteeism in hours'* is our target variable, rest of variables are predictor variables. Coming to data category we have *Continuous* and *Categorical variables.* We have *10 Continuous* variable and *11 Categorical* variable as shown in Table 2.1

Table 2.1

Variable Category	
Continuous Variables	***Categorical Variables***
Transportation_expense	*ID*
Distance_from_Residence_to_Work	*Reason_for_absence*
Service_time'	*Month_of_absence*
Age	*Day_of_the_week*
Average_workload	*Season*
Hit_target	*Disciplinary_failure*
Weight	*Education*
Height	*Son*
Body_mass_index	*Social_driker*
Absenteeism_time_in_hours	*Social_smoker*
	Pet

2.2 DATA VISUALISATION

Data visualisation helps us to get better insights of the data. By visualising data, we can identify areas that need attention or improvement and also clarifies which factors influence feature behaviour.

2.2.1 UNIVARIATE ANALYSIS

Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. Since it's a single variable it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.

So, let's have a look at histogram plot, to identify the characteristic of the features and the data.

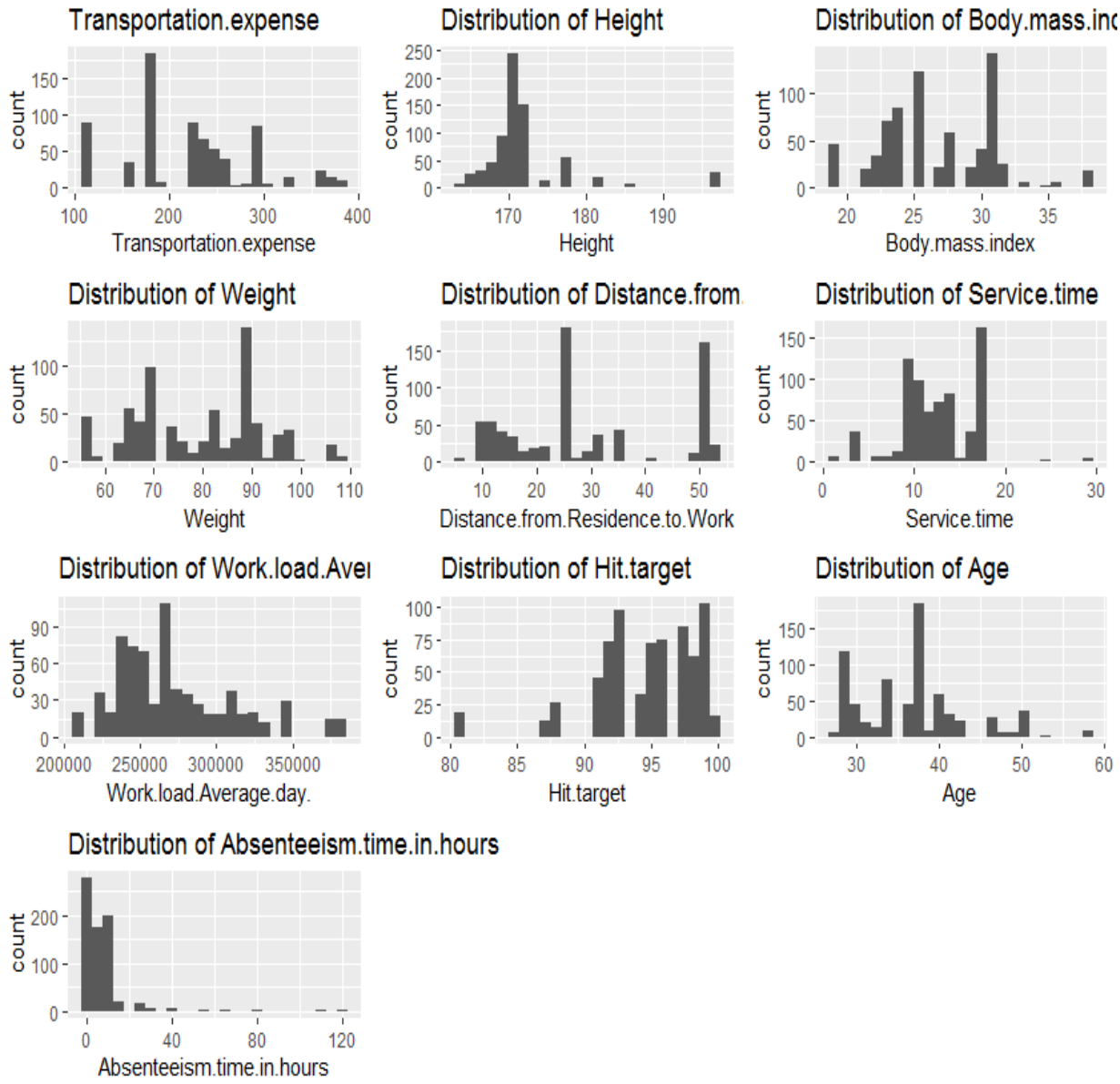


Figure 2.1 : Histogram plot for distribution of continuous features

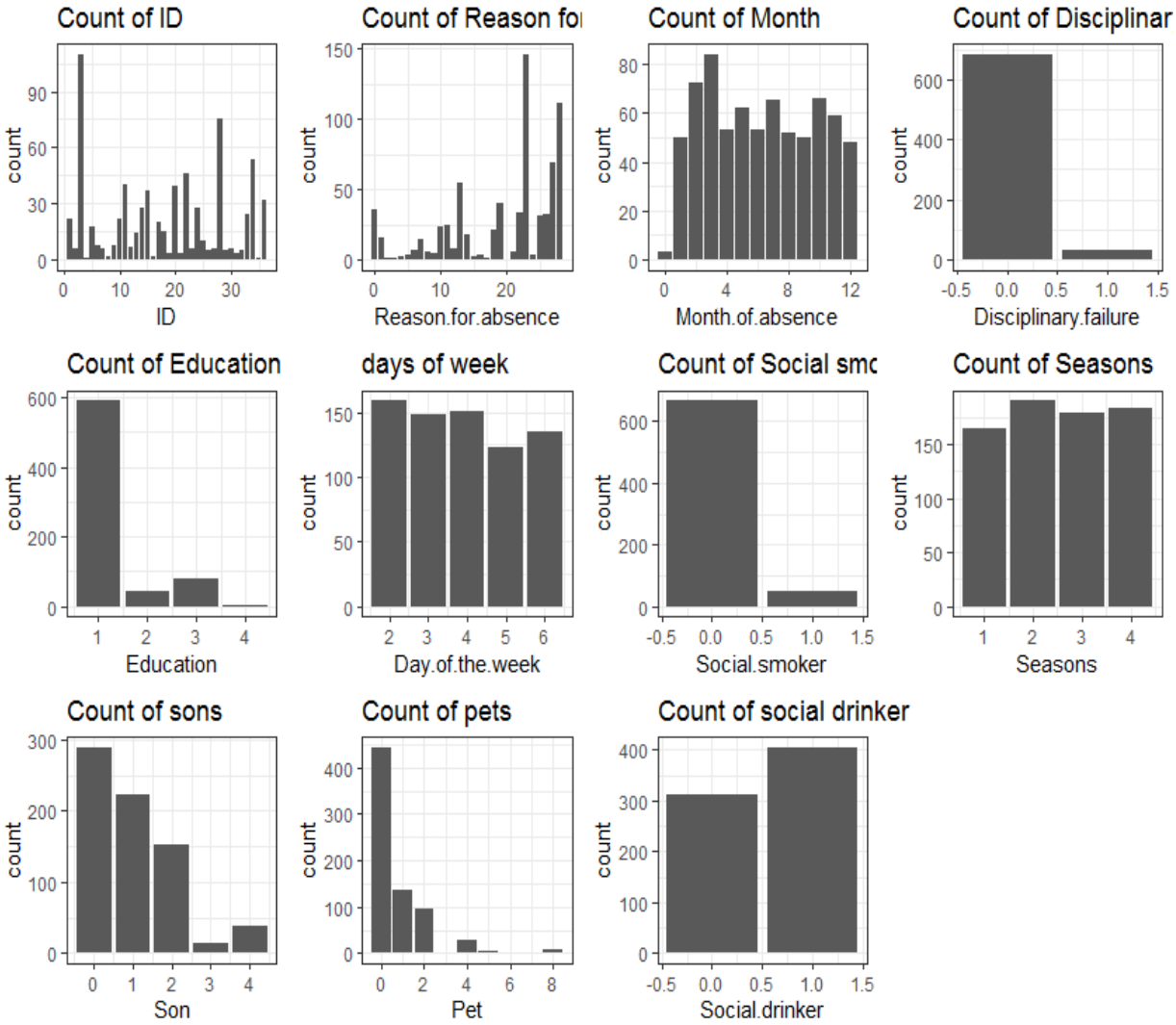


Figure 2.2: Bar graph for distribution of categorical features

Histograms are constructed by binning the data and counting the number of observations in each bin. The objective of plotting Histogram plot is usually to visualise the shape of the distribution. The number of bins needs to be large enough to reveal interesting features and small enough not to be too noisy.

A bar graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

From the above histogram plot/bar graph, we can observe the following points:

1. People over 40+ years of age tends to take less leaves compare to others.
2. Majority of the employees working in the company have age below 40 years.
3. A very large portion of the population have only passed ‘High School’.
4. More than half of the employees in the company are ‘social drinker’.
5. Only a very few portion of the employees in the company are ‘social smoker’

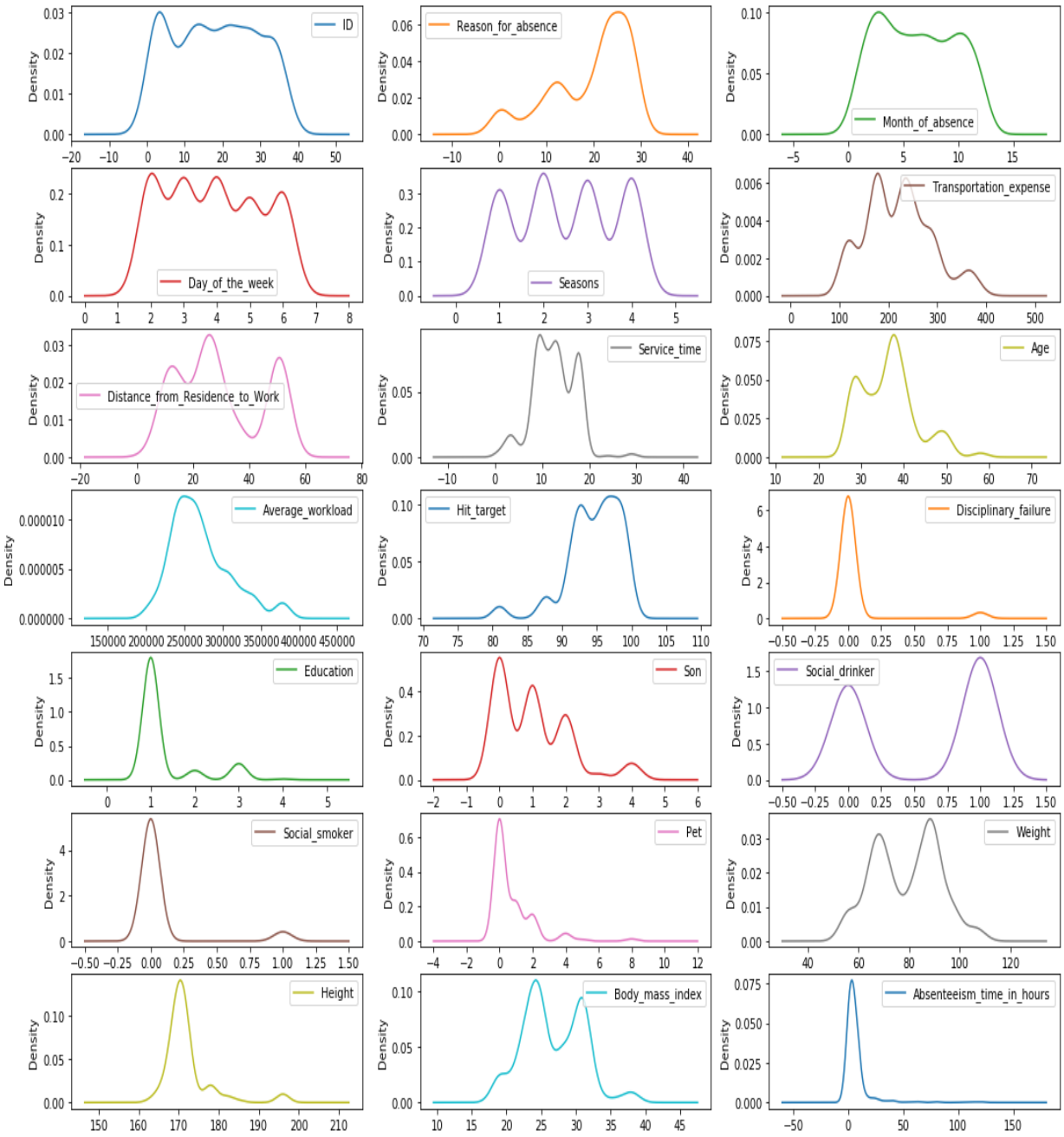


Figure 2.3: KDE plot for distribution of features

A Density Plot visualises the distribution of data over a continuous interval or time period. Density plots can be thought of as plots of smoothed histograms. An advantage Density Plots have over Histograms is that they're better at determining the distribution shape because they're not affected by the number of bins used.

So, looking at the above density plot, we can observe that none of the features follow Gaussian distribution. Few of the features like 'Disciplinary failure', 'Social smoker', *'Average_Workload'* seems to follow Gaussian distribution at first sight but they either have long tail at the left or right.

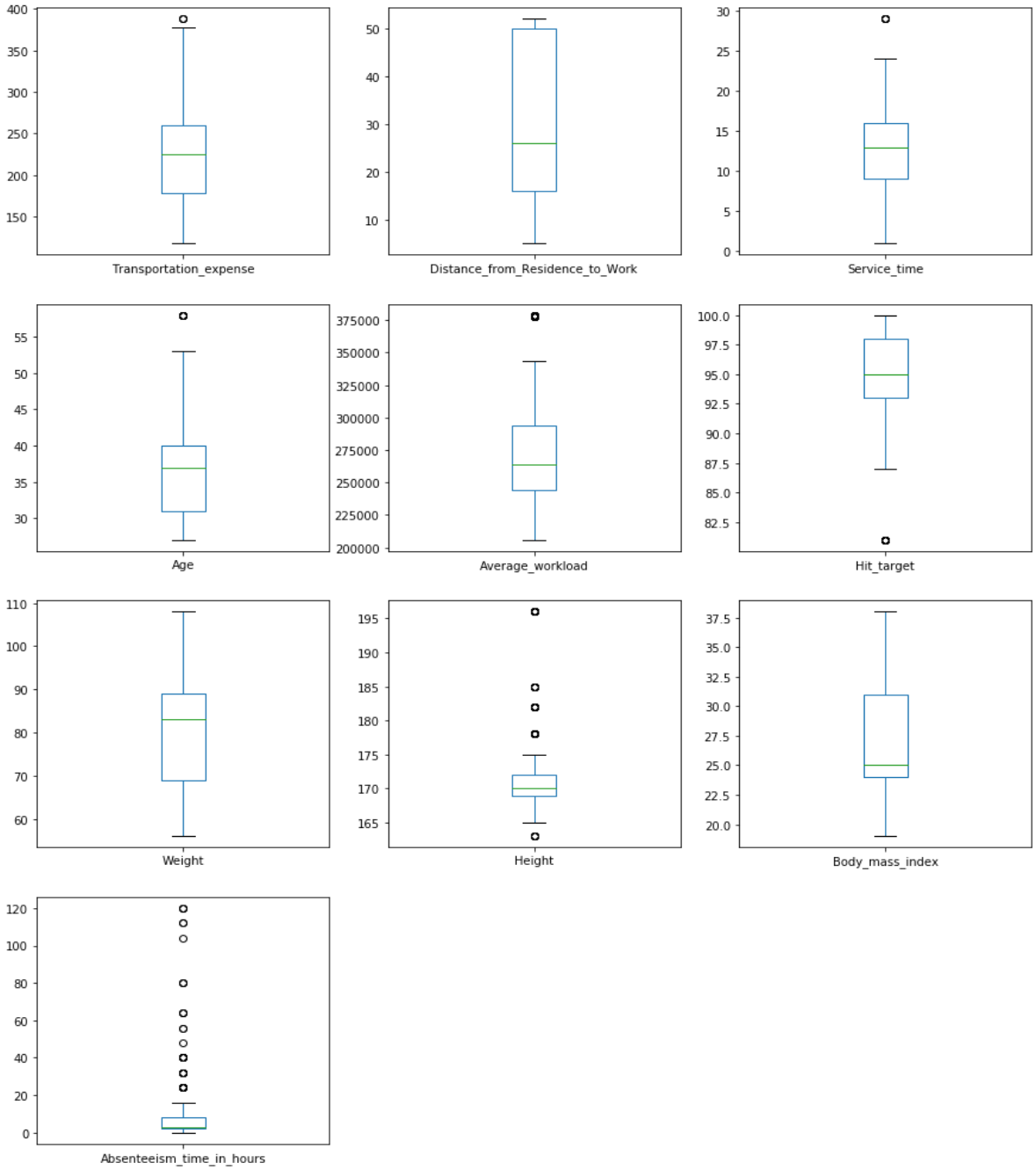


Figure 2.4: Box and Whiskers plot of features in the data

From the above Box and whisker plots, we can observe that not all the features contains outliers. Continuous features like 'Weight', 'Distance from residence to work' does not contain any outliers at all. Few features like 'Average_workload', 'Hit_target' 'Age', 'Service_time' and 'Height', have a very few outliers.

It is also evident from the above plot that none of the features are symmetric to the median and it can easily be interpreted that none of the features follow symmetric distribution. Also, it can also be observed that Median of the feature 'Body mass index' is very close to 25th percentile value which means median of this feature is almost equal to 25th percentile.

2.2.2 BIVARIATE ANALYSIS

Bivariate analysis refers to the analysis of bivariate data. It is one of the simplest forms of statistical analysis, used to find out if there is a relationship between two sets of values. It usually involves one predictor variable and one target variable.
So, Let's have a look at the Scatter plot and Bar Plots to understand the Employee behaviour better.

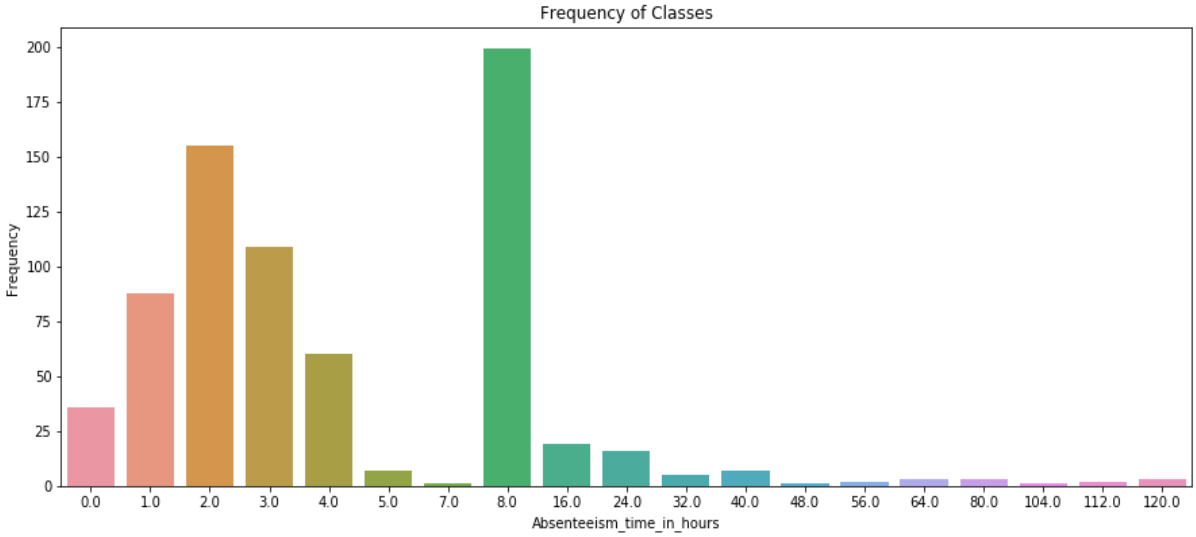


Figure 2.5: Bar plot for Distribution of Target class

From the above plot following observation are made:

- It is obvious that maximum number (i.e. around 200) of employees are absent for 8 hours.
- Around 420 employees are absent for 1 to 4 hours.
- Only 36 employees doesn't take any leave.
- There are only 27 employees, who take leave for 40-120 hours.
- One observation that is worth noting from the above plot is that, after class 8, every other class is a multiple of 8.

Also, as we can see, 'Absenteeism_time_in_hours' are 0 in 36 places. This could be result of cancelled or withdrawn leaves. Let's drop these observations for further analysis.

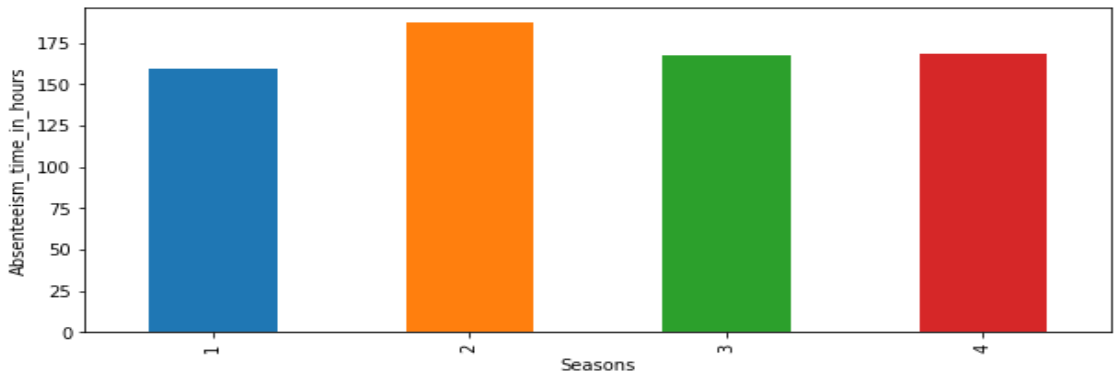


Figure 2.6: Bar plot for relationship btwn Seasons with Target variable

From the above Bar plot , it can be observed that the 'Absenteeism rate' is maximum in Season 2

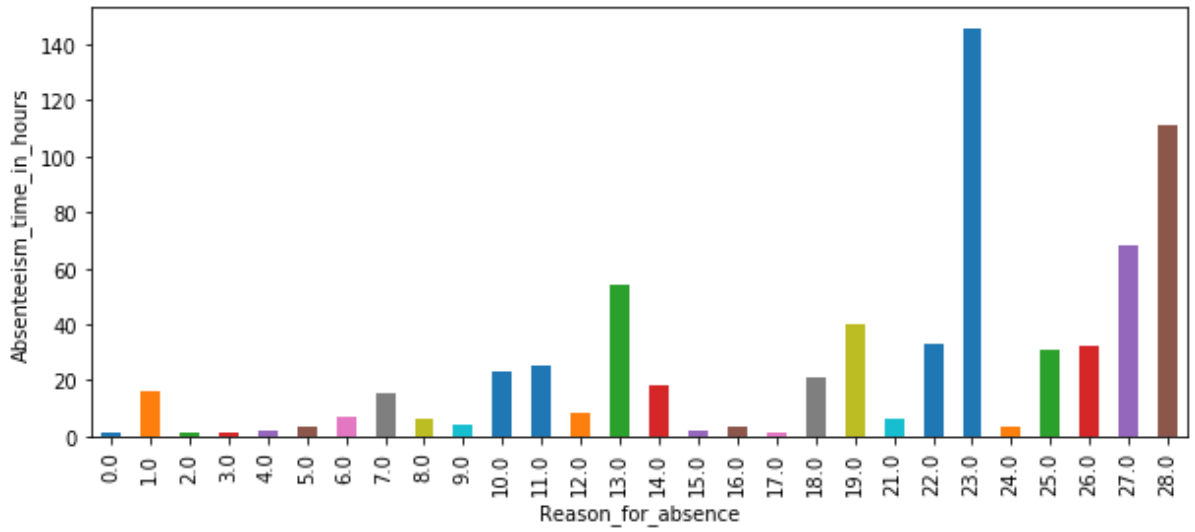


Figure 2.6: Relation btwn Reason_for_absence with Absentee hours

Longest hours of Absence for the Reasons as follows 23,28,13,27,19..

- *#23 - medical consultation (23).*
- *#28 - dental consultation (28).*
- *#27- physiotherapy (27).*
- *#13 - Diseases of the musculoskeletal system and connective tissue.*
- *#19 - Injury, poisoning and certain other consequences of external causes.*

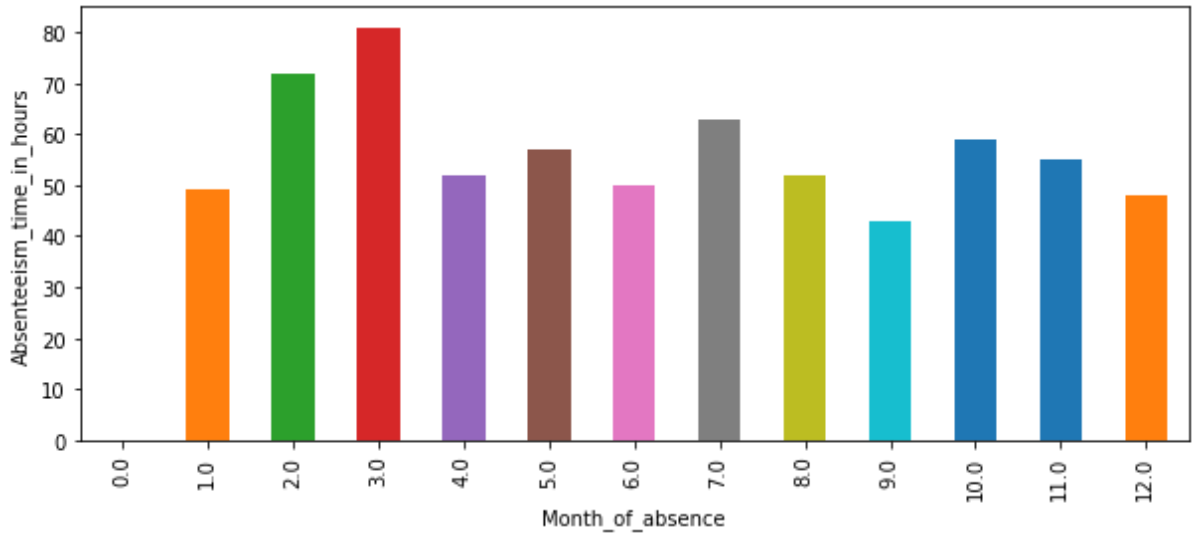


Figure 2.6: Relation btwn Month_of_absence with Absentee hours

From 'Month_of_absent' distribution, we can see that frequency of leaves are more or less uniformly distributed over months, with highest no. of leaves taken in March, followed by Feb and July (holiday season).

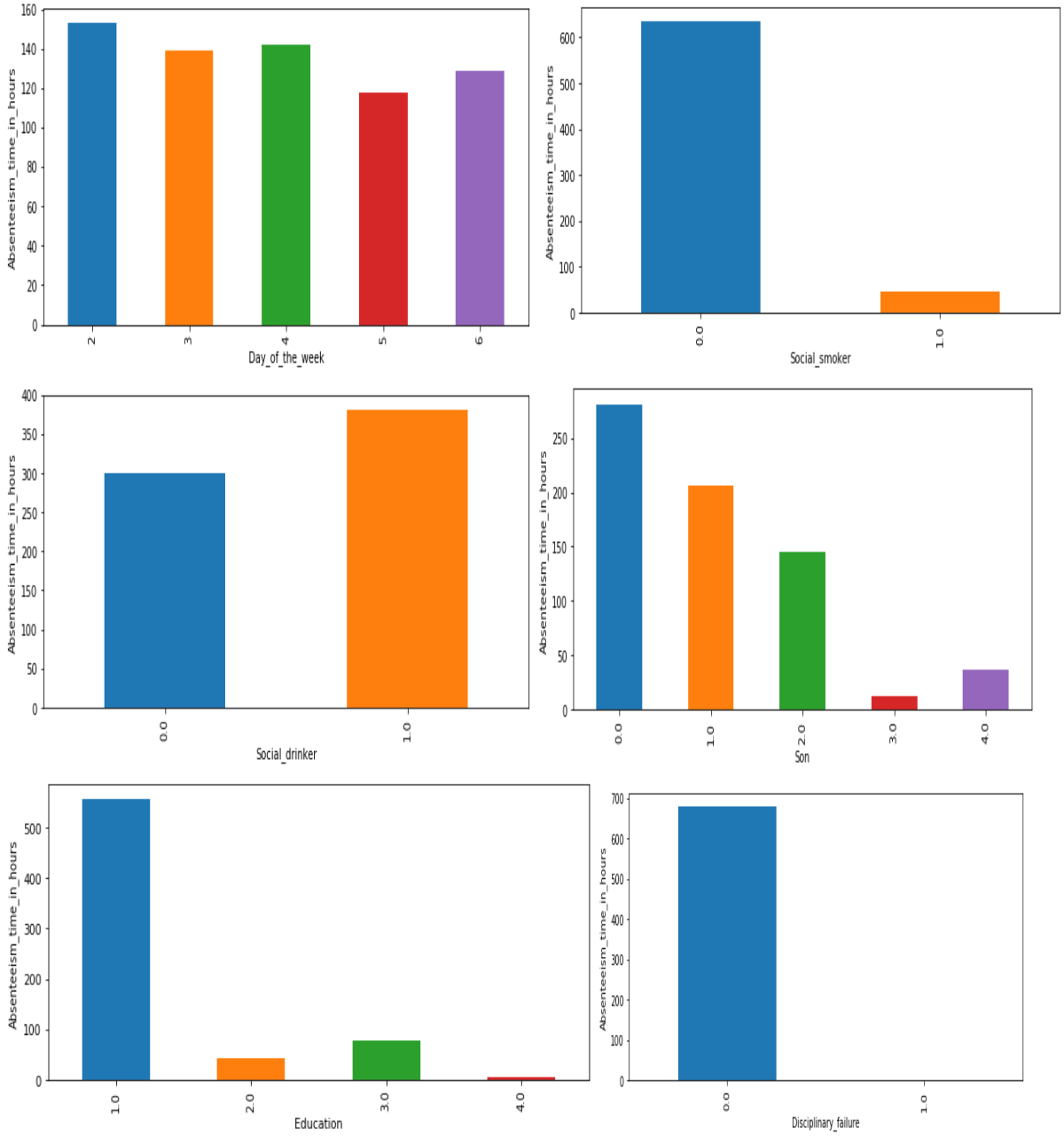


Figure 2.7: Relation between Independent variable with Absentee hours

- From, Absent_Weekday distribution, we can see that frequency of leaves are mostly distributed, with most frequent leaves on 'Monday'.
- Displinary failures are very least.
- Employee with Education ‘High School’ tend to take more hours of absence.
- Employee with 3-4 kids tend to take less hours of absence.
- 'Social Drinker' takes little more leaves than non-drinker.
- From, 'Son' and 'Pet', we can see that people having no kids and no pets(no family responsibilities) tend to take frequent leaves

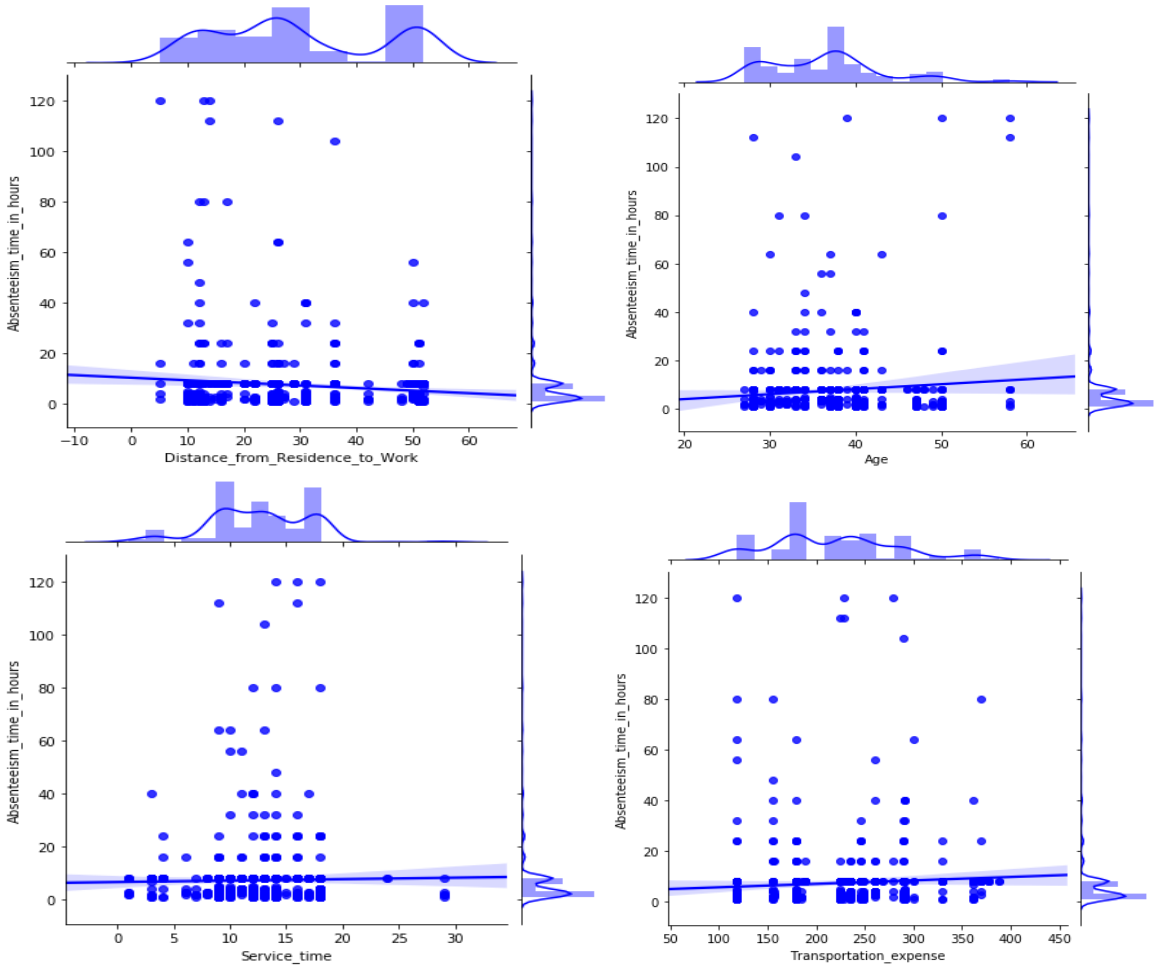


Figure 2.8: Scatter plot b/w Predictor variable & Target variable

From above scatter plot the observations are made as follows:

- This clearly shows concentration of leaves more where the 'Transportation_expense' is between 150-300.
- This clearly shows concentration of leaves more where the 'Distance_from _work' is between 10-30 km.
- Employees with 'service_time' (service years) less than 8 and more than 18 tends to take less leaves

2.3 DATA PREPARATION AND CLEANING

2.3.1 MISSING VALUE ANALYSIS

Missing value analysis helps address several concerns caused by incomplete data. The missing data may reduce the precision of calculated statistics because there is less information than originally planned. Another concern is that the assumptions behind many statistical Procedures are based on complete cases, and missing values can complicate the theory required.

Below table illustrate number of missing values and their percentage present in the each variable in the data.

	Variables	Missing_Values	Missing_percentage
0	Body_mass_index	31	4.189189
1	Absenteeism_time_in_hours	22	2.972973
2	Height	14	1.891892
3	Average_workload	10	1.351351
4	Education	10	1.351351
5	Transportation_expense	7	0.945946
6	Son	6	0.810811
7	Disciplinary_failure	6	0.810811
8	Hit_target	6	0.810811
9	Social_smoker	4	0.540541
10	Age	3	0.405405
11	Reason_for_absence	3	0.405405
12	Service_time	3	0.405405
13	Distance_from_Residence_to_Work	3	0.405405
14	Social_drinker	3	0.405405
15	Pet	2	0.270270
16	Weight	1	0.135135
17	Month_of_absence	1	0.135135
18	Seasons	0	0.000000
19	Day_of_the_week	0	0.000000
20	ID	0	0.000000

Table 2.2: Missing value & missing Percentage of variables

The maximum missing percentage is 4.18% i.e. the highest missing value is in Body mass index where is lowest is in month of absence and weight. Our target variable i.e. Absenteeism time in hours consists of 2.97% of missing values whereas three variables (Day of week, ID, Seasons) does not contain any missing values.

2.3.1.1 IMPUTING MISSING VALUES

In our data, there are missing values in almost all the columns of the dataset, although in small amount. Before thinking for imputing method, first we need to remove the rows of missing values of target variable.

All the missing variables except *"Average_workload"* are the attributes of the same person (i.e.ID).Which means Replace missing of any employee with information of same employee from other instances. For example if 'Age' of employee-2 is missing, then impute it with 'Age' with max value from other instance of employee-2.

Now let's analyse which is the best way to impute missing values for 'Average_Workload. To do so, let's plot scatter plot between 'ID' &'Average_workload'.

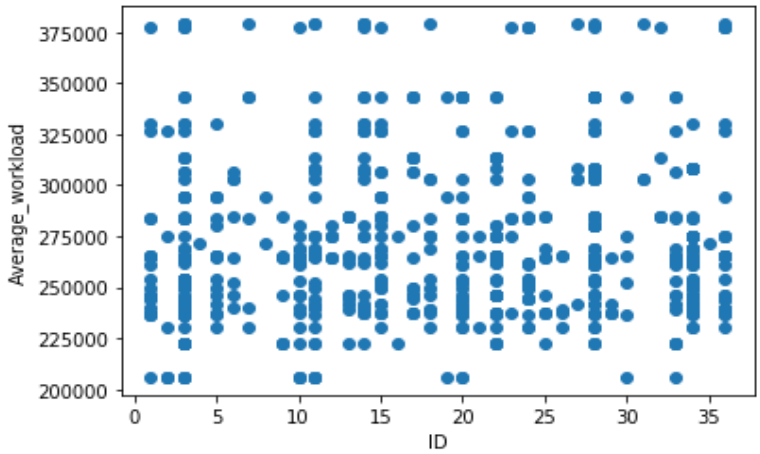


Figure 2.9: relationship b/w 'ID' & 'Average_workload'

The above scatter plot shows different *Average_workload* for same 'ID'. So 'Average workload' is not depends on "ID".

Now, let's plot scatter plot between *'Month of absent'* &*'Average_workload'.*

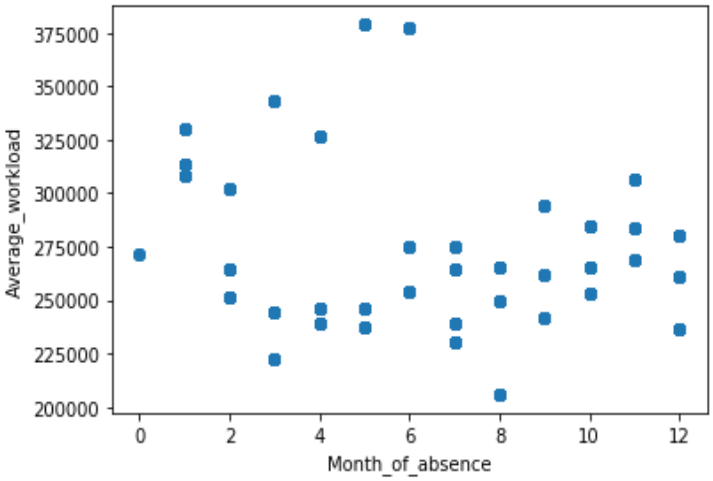


Figure 2.10

From above plot, we can conclude that 'Average_Workload' is distributed mostly by month. So, let's impute missing 'Average_Workload' by mode or mean of that month.

2.3.2 OUTLIERS ANALYSIS

Outlier analysis is one of the pre-processing techniques used to check for abnormal values in the data set clean them and transform the data into a proper shape. Also, Outlier analysis is very important because they affect the mean and median which in turn affects the error (absolute and mean) in any data set. When we plot the error we might get big deviations if outliers are in the data set.

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 greater than (Q3 + (1.5 * IQR)) or less than (Q1 — (1.5 * IQR)). Where

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

In Box plots analysis of individual features, we can clearly observe from these boxplots that, Outliers are found in 'Service_time', 'Age', 'Average_workload', 'Hit_target', 'Height', 'Transportation expense', and 'Absenteeism time in hours'.

While looking at dataset 'Service_time', 'Age' are in proper format, so no need for dropping these variables.

There seems to be many outlier in our target variable Absenteeism_time_in_hours. There are value of 120, 100, 80, 60 which is not possible.
Reason: 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. So, dropping the outliers is probably the best idea.

2.3.3 FEATURE SELECTION

Selecting subset of relevant features for model construction is known as Feature Selection. When we get raw data we have multiple variables and with the help of variable selection we have to extract relevant data. We cannot use all the features because some features may be carrying the same information or irrelevant information which does not impact the business solution.

To reduce complexity we adopt feature selection technique to extract meaningful features out of it. This in turn helps us to avoid the problem of multi-collinearity. In our project we have used Correlation plot and ANOVA to select important variables and selected the variables according to the values of correlation and ANOVA.

Feature Selection reduces the complexity of a model and makes it easier to interpret. It also reduces over fitting. Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multicollinearity between variables. The highly collinear variables are dropped and then the model is executed.

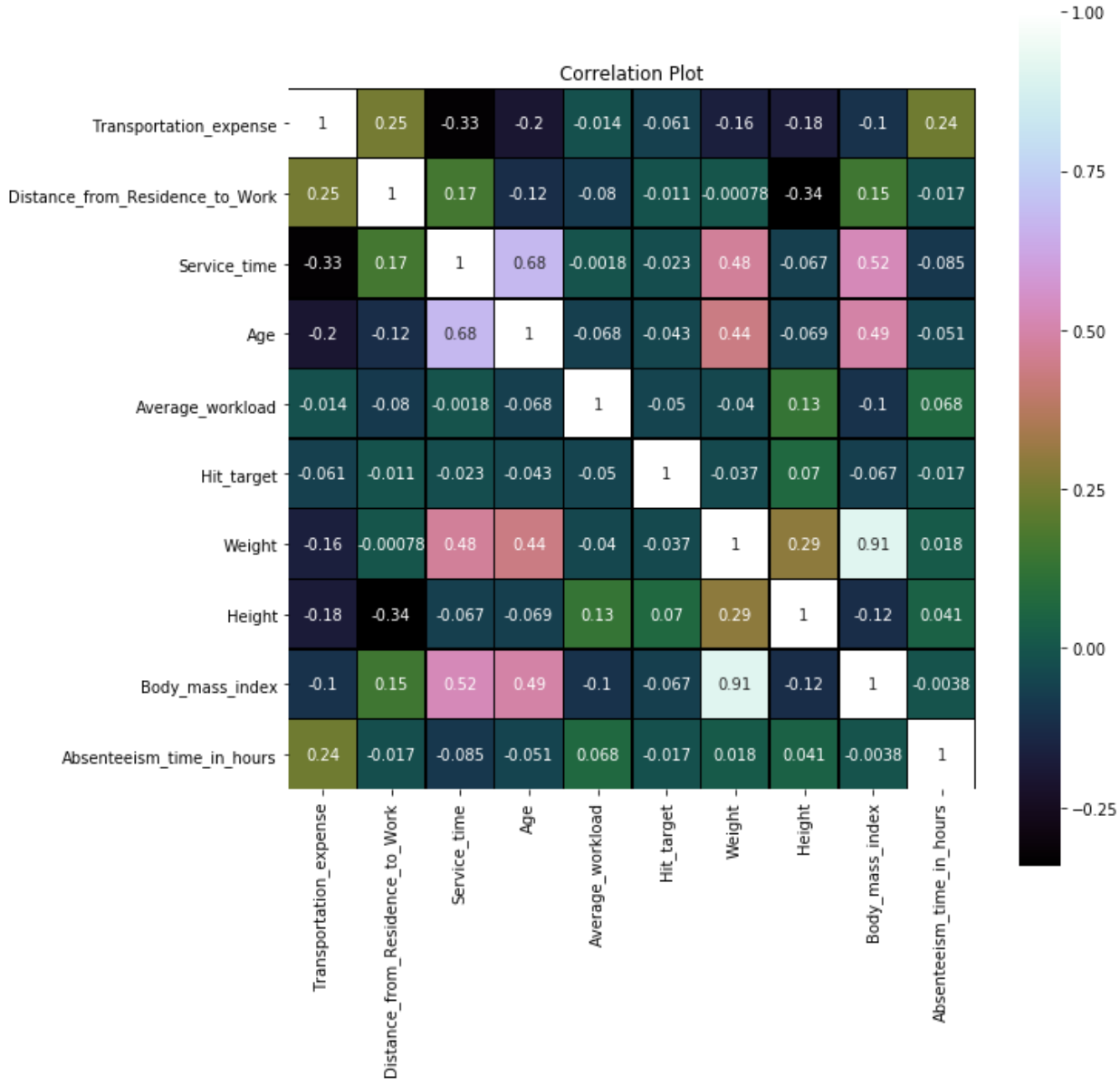


Figure 2.11: Correlation Plot

From correlation analysis we have found that ***Weight*** and ***Body Mass Index*** has high correlation (>0.7), so we have excluded the ***Body Mass Index*** column

2.3.4 FEATURE SCALING

It comes into an action when we are dealing with parameters of different units and scales. It is also known as variable scaling. It is used to limit the range of range of variables so that they can be compared on common basis. Feature scaling is performed only on continuous data.

There are two methods to scale the data Normalisation and Standardisation. *Normalisation* is the process of reducing unwanted variation either within or between variables. Normalisation brings all the variables into proportion with one another. It ranges between 0 and 1 and are sensitive to outliers. Normalisation works on all kind of continuous data whereas *standardisation* works well when data is uniformly distributed.
In our project none of the features follow Gaussian distribution (uniformly distributed), so we have used *Normalisation* for feature scaling.

2.3.5 VARIABLE REDUCTION

2.3.5.1 PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible. With fewer variables, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher dimensional data. After creating dummy variable of categorical variables, the data would have 115 columns and 518 observations, this high number of columns leads to bad accuracy.

By plotting a cumulative distribution function plot to observe how much percentage of variance is explained by how many variables (Principle Components).

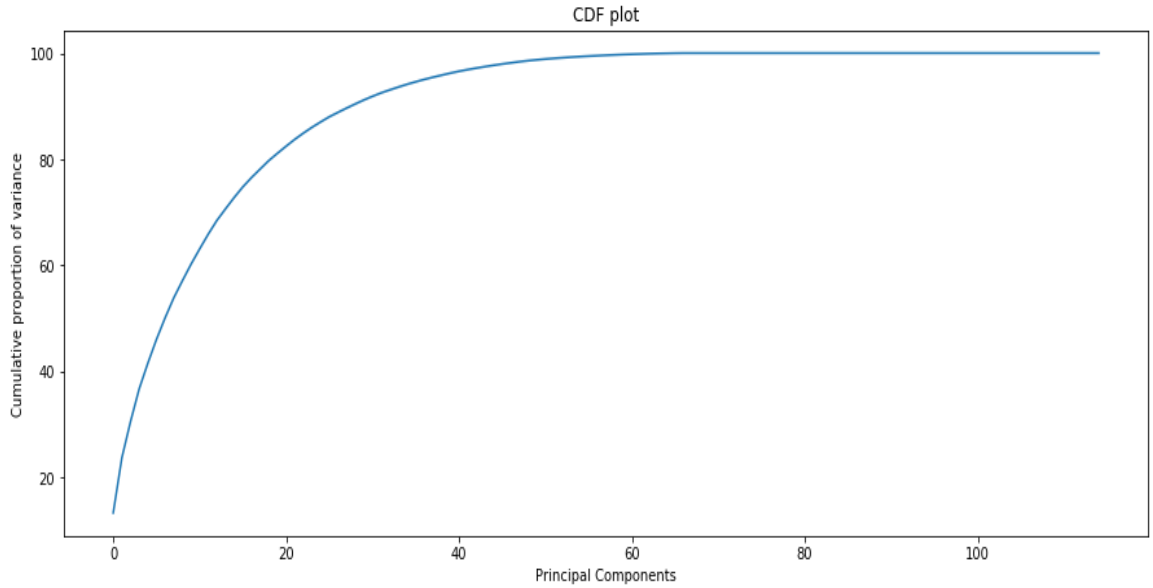


Figure 2.12: CDF Plot for 'Cumulative proportion of variance' Vs 'Principal Components'

After applying PCA algorithm and observing the above a cumulative distribution function plot, we have concluded that 37 variables out of 115 explains more than 95% variance. So we have selected only those 37 variables to feed our models.

CHAPTER 3

3.1 MODELLING

Now, that our dataset is free from any anomaly, colinearity, and is scaled to the same range. We are ready to make machine learning models to predict bike rental count.

Approach

We have divided the dataset into train and test part using random sampling. Where train contains 80% of data set and test contains 20% data. We'll be using various different machine learning algorithms to make different models and comparing amongst themselves using different methodologies and after selecting the machine learning model we'll be tuning some parameter to make model best as possible.

The Models we build here are
1. Decision tree regression
2. Random Forest
3. Linear Regression
4. Ridge Regression
5. KNN Regressor
6. Support Vector Regression
7. Gradient Boosting

3.1.0 Performance Metric

RMSE: 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.

Also, since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. So, RMSE becomes more useful when large errors are particularly undesirable. So, Roost Mean Square value seems like a perfect choice for our problem at hand.

3.1.1 Decision Tree Regression

Decision Tree is a predictive model based on branching series of Boolean tests. It is one of the most powerful and popular algorithm and belongs to family of supervised learning algorithm. Decision tree is a rule and output of decision tree is in form of simple business rules which is extremely easy to understand by business users.

Decision tree builds regression, models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. So, after we implement Decision Tree Regression on our data, we get the following results:

```
depth : 1       : Decision Tree rmse: 2.7542181521089755
depth : 2       : Decision Tree rmse: 2.8339027846412277
depth : 3       : Decision Tree rmse: 2.9904693930866504
depth : 4       : Decision Tree rmse: 3.153150417284761
depth : 5       : Decision Tree rmse: 2.9842130373190683
depth : 10      : Decision Tree rmse: 3.4184279505045905
depth : 15      : Decision Tree rmse: 3.4428670872355895
depth : None    : Decision Tree rmse: 3.826149902568256
```

Freeze the Decisiontree model with max_depth = 1.

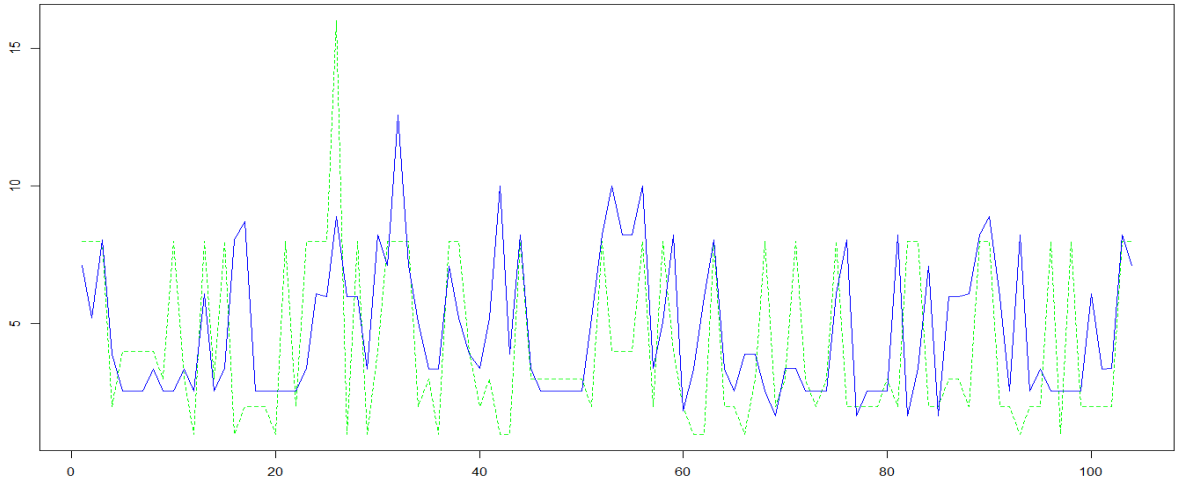


Figure 3.1: Plot of actual values vs. predicted values for Decision Tree

3.1.2 Random Forest Regression

Random Forest is a supervised learning algorithm. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression problems. The method of combining trees is known as an ensemble method.

Random Forest Regression or Regression Trees are known to be very unstable, in other words, a small change in our data may drastically change your model. The Random Forest uses this instability as an advantage through bagging resulting on a very stable model. So, after we implement Random Forest Regression or Regression Trees on our data, we get the following results:

```
depth : 2,      n_estimators : 100     : RF rmse: 2.7058392169553516
depth : 5,      n_estimators : 100     : RF rmse: 2.402107145452076
depth : 10,     n_estimators : 100     : RF rmse: 2.353047415515257
depth : 20,     n_estimators : 100     : RF rmse: 2.4057863355749918
depth : 2,      n_estimators : 200     : RF rmse: 2.691911078164883
depth : 5,      n_estimators : 200     : RF rmse: 2.4608530315075265
depth : 10,     n_estimators : 200     : RF rmse: 2.322212981142843
depth : 20,     n_estimators : 200     : RF rmse: 2.3500232364449634
depth : 2,      n_estimators : 500     : RF rmse: 2.70458312129482
depth : 5,      n_estimators : 500     : RF rmse: 2.416512793577292
depth : 10,     n_estimators : 500     : RF rmse: 2.305027980673508
depth : 20,     n_estimators : 500     : RF rmse: 2.3243010609344052
depth : 2,      n_estimators : 1000    : RF rmse: 2.694956358456143
depth : 5,      n_estimators : 1000    : RF rmse: 2.412715993931755
depth : 10,     n_estimators : 1000    : RF rmse: 2.3258202047383696
depth : 20,     n_estimators : 1000    : RF rmse: 2.315724432981271
```

So, we can see from the above results that, our model gives a RMSE= 2.30, the number of decision trees used for prediction in the forest is 500.

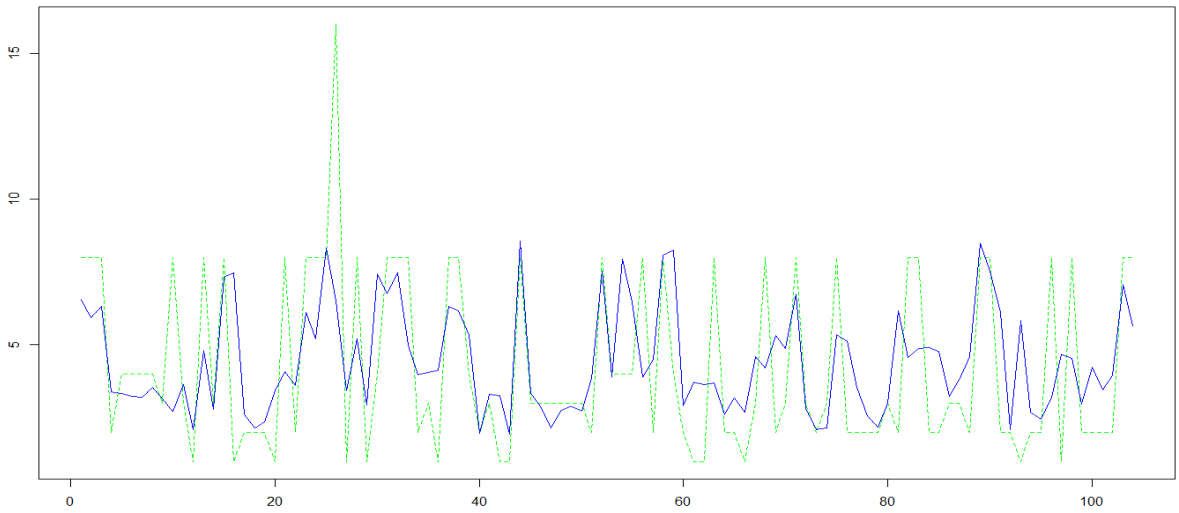


Figure 3.2: Plot of actual values vs. predicted values for Random forest

3.1.3 Linear Regression

Multiple linear regression is the most common form of linear regression analysis. Multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical.

Now we will try to implement Multiple Linear Regression algorithm using Ordinary Least Squares, the simplest of all. Ordinary least squares (OLS) minimises the squared Distances between the observed and the predicted dependent variable. So, we get the following results after implementing the model:

```
Ordinary Least Squares rmse: 2.4903341296815635
```

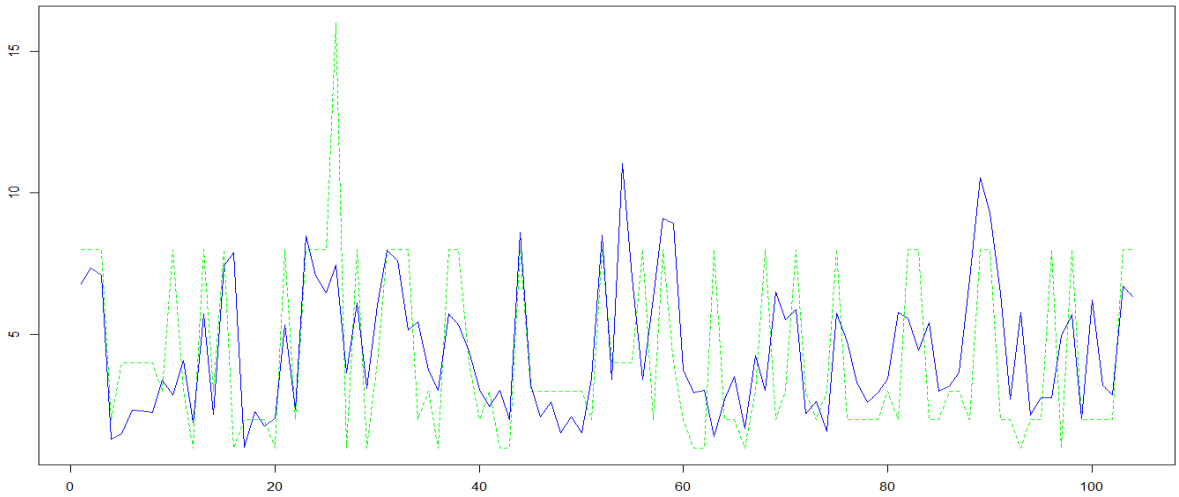


Figure 3.3: Plot of actual values vs. predicted values for LR

3.1.4 Ridge Regression

Ridge Regression essentially is an instance of Linear Regression with regularisation. Ridge regression is that it enforces the β coefficients to be lower, but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimise their impact on the trained model. So, after we implement Ridge Regression on our data, we get the following results:

```
alpha : 0.1     :Ridge rmse: 2.1604453639020718
alpha : 0.5     :Ridge rmse: 2.161240465238058
alpha : 1.0     :Ridge rmse: 2.1626117379720537
alpha : 3.0     :Ridge rmse: 2.171181115588791
alpha : 7.0     :Ridge rmse: 2.1960846782423515
alpha : 10.0    :Ridge rmse: 2.217229817212254
```

We can see from the above results that, our model gives a RMSE=2.16, we can claim that Ridge Regression performs better than all the algorithm implemented before.

3.1.5 KNN Regression

KNN regression is one of the simplest algorithm in the whole of Machine learning. It gives a weighted average of the regression function in a local space (k nearest points to a given point). So, we first try to implement and fit KNN regression to our Data and got following results after tuning the hyper parameter k:

```
n_neighbours : 3        : KNN rmse: 2.6317830555123747
n_neighbours : 5        : KNN rmse: 2.6275902970651206
n_neighbours : 7        : KNN rmse: 2.5893757089324483
n_neighbours : 10       : KNN rmse: 2.5655071473447357
n_neighbours : 15       : KNN rmse: 2.5458865767230208
n_neighbours : 20       : KNN rmse: 2.5551936175321277
n_neighbours : 25       : KNN rmse: 2.616659643955953
n_neighbours : 30       : KNN rmse: 2.679498374409199
```

For k=15, the model performance is better. I.e. RMSE=2.54

3.1.6 Support Vector Regression

Support Vector Machine can be applied not only to classification problems but also to the case of regression. Still it contains all the main features that characterise maximum margin algorithm: a non-linear function is leaned by linear learning machine mapping into high dimensional kernel induced feature space. So, after we implement Support Vector Regression on our data, we get the following results:

```
C : 1 ,::       gamma : 0.001 ::       :SVR rmse: 3.1502166030146106
C : 1 ,::       gamma : 0.0001 ::      :SVR rmse: 3.208344955083732
C : 10 ,::      gamma : 0.001 ::       :SVR rmse: 2.8391076005765292
C : 10 ,::      gamma : 0.0001 ::      :SVR rmse: 3.149582761139894
C : 100 ,::     gamma : 0.001 ::       :SVR rmse: 2.2661312872999164
C : 100 ,::     gamma : 0.0001 ::      :SVR rmse: 2.8374148798117487
C : 1000 ,::    gamma : 0.001 ::       :SVR rmse: 2.1764484982897456
C : 1000 ,::    gamma : 0.0001 ::      :SVR rmse: 2.261896605573216
```

We can see from the above results that, our model gives a RMSE (Root Mean Square Value) of 2.17. for C=1000 & gama=0.01.

3.1.7 Gradient Boosting Decision Tree Regression

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. So, after we implement Gradient Boosting Decision Trees on our data, we get the following results:

```
depth: 1,       :learning_rate: 0.001   : rmse: 2.980309425993696
depth: 1,       :learning_rate: 0.01    : rmse: 2.86119371466673
depth: 1,       :learning_rate: 0.1     : rmse: 2.5207350825424943
depth: 2,       :learning_rate: 0.001   : rmse: 2.9659256781568115
depth: 2,       :learning_rate: 0.01    : rmse: 2.7224583633729056
depth: 2,       :learning_rate: 0.1     : rmse: 2.3916282784497747
depth: 5,       :learning_rate: 0.001   : rmse: 2.8974924408012495
depth: 5,       :learning_rate: 0.01    : rmse: 2.538287171502763
depth: 5,       :learning_rate: 0.1     : rmse: 2.3820745646560795
depth: None,    :learning_rate: 0.001   : rmse: 2.9022342352016803
depth: None,    :learning_rate: 0.01    : rmse: 2.9285159378350327
depth: None,    :learning_rate: 0.1     : rmse: 3.4889704030407165
```

Gradient Boosting Decision Tree Regression does not give impressive results and our other linear algorithms gives much lower error. Also, it can be noticed that, the result of Decision Trees and GBDT are almost same.

CHAPTER 4

CONCLUSION

4.1 MODEL EVALUATION

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Employee Absenteeism, the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models.
Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

4.1.1 Root Mean Square Error (RMSE)

RMSE: 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.

Also, since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. So, RMSE becomes more useful when large errors are particularly undesirable. So, Roost Mean Square value seems like a perfect choice for our problem at hand.

4.2 Model Selection

Looking at models performance, we can say that 'Ridge Regression' is the best model based on RMSE value, followed by 'Random Forest' and 'Support Vector Regression' so, we can pick any of those and perform modelling. All these model will give almost same results.

Model comparison table is given below.

Model	RMSE VALUE	
	PYTHON	R-Program
Decision Trees	2.754	2.929
Random Forest	2.305	2.351
Linear Regression	2.490	2.498
Ridge Regression	2.160	2.358
KNN Regression	2.545	2.500
SVR	2.176	2.404
Gradient Boosting	2.382	2.703

Table 3.1: Model comparison

We saw that the models Ridge Regression along with SVR and Random Forest perform comparatively on RMSE (Root Mean Square Error) , Although Random Forest gives the best results on the test data but it is unstable and it over fits on the train data. So, it will be a wise decision to use either SVR or Ridge Regression for deployment.

So, it is obvious from above model performance comparison table, models Ridge Regression, SVR and Random Forest perform comparatively on RMSE (Root Mean Square Error) and can be used for deployment.

4.3. ANSWER TO THE ASKED QUESTIONS

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

Looking at the EDA of the features, we observe and have understood the behaviour of Employee attributes against their Absenteeism rate in the company, we can introduce following changes to reduce the number of Absenteeism:

A). Primary Changes: changes by cause/reason.

Longest hours of Absence for the Reasons as follows 23,28,13,27,19.

a) *#23 - medical consultation.*
b) *#28 - dental consultation.*
c) *#27- physiotherapy.*
d) *#13 - Diseases of the musculoskeletal system and connective tissue.*
e) *#19 - Injury, poisoning and certain other consequences of external causes.*

Overall,

- *Seems like employee takes most absences for medical consolations/dental consultation/physiotherapy. These hours can be reduced by setting up a medical consultation/dental consultation/ physiotherapy booth (with visiting doctors may be) at office/facility.*
- *In long term, introducing exercise/yoga sessions in office once/twice a week will help reduce physiotherapy issues.*
- *A Health care facility can be introduced in the company, so that the employees can have regular Medical check-ups to keep them fit and Working. Also, it would help with the company's reputation to have taken the responsibility of their Employees.*

B). Secondary Changes:

- *It is observed that employees having education only till high school tend to be absent more than others. So, the company can either hire employees who have at least graduated from college.*
- *Strict action could be taken towards the employee with high absence rate in the workplace without any valid reason for the absence and Employees with no absence or a minimum absence can be rewarded*
- *As Absenteeism rate is maximum in Season of 'winter' and in month of July, April and March respectively, we can issue special notices regarding the Absenteeism scenario around the company.*
- *People who tend to be social drinkers tend to be more absent than those who don't drink. Company should inform those employees to reduce the intake of alcohol during working days.*
- *We can start by Increasing the employee morale, engagement, and commitment to the organisation. By apply performance policies to act at the root of the problem. In some cases, absence rate might be reduced by clear specification of employees' responsibilities and targets.*

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

To calculate Loss per month, we introduce the following formula:

- In a month excluding weekend 22 days are working days.
- there are 36 employee in the xyz company
- 8 hours of work per day

I.e. Total Monthly hours = 22*8*36

Monthly losses % = (total absent hours / Total Monthly hours)*100

	Month_of_absence	Absent_hours_2011	monthly_loss_percentage
1	1.0	150.929537	2.382095
2	2.0	213.564963	3.370659
3	3.0	342.810677	5.410522
4	4.0	183.369302	2.894086
5	5.0	130.928843	2.066427
6	6.0	105.855510	1.670699
7	7.0	272.304962	4.297742
8	8.0	167.423159	2.642411
9	9.0	87.816583	1.385994
10	10.0	246.458907	3.889819
11	11.0	211.360047	3.335859
12	12.0	169.989426	2.682914

Table 4.1

Looking at the above results, we can observe that most likely, the company would incurred most of loss in the month of 'March', followed by 'October' and 'July'.

Appendix A

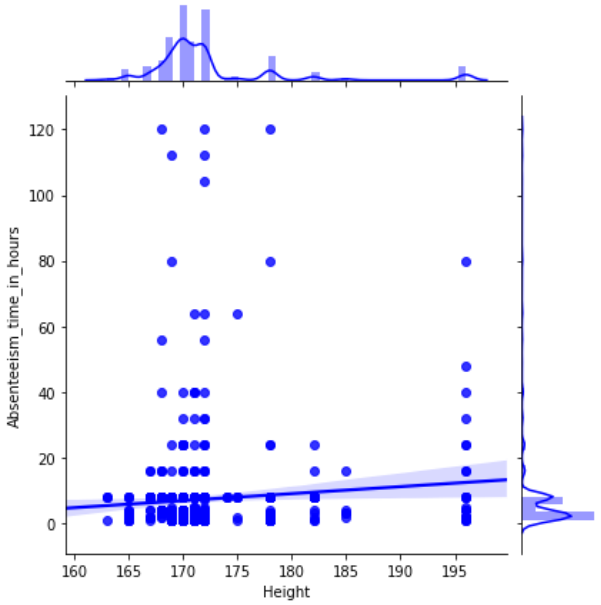
Reason for absence in the data attribute:

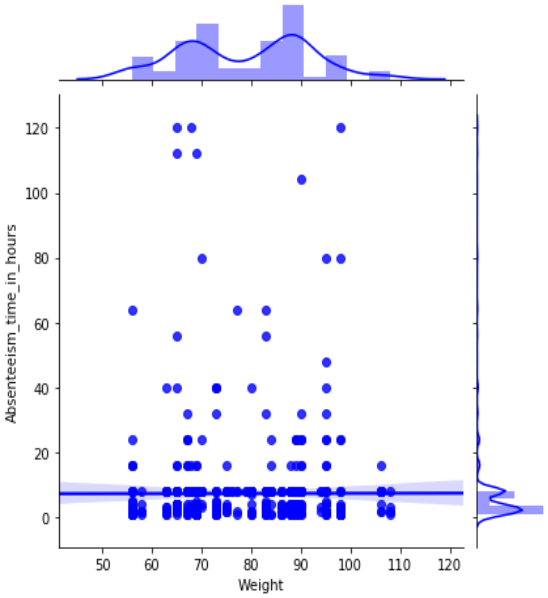
Absences attested by the International Code of Diseases (ICD) stratified into 21 categories (I to 21) as follows:

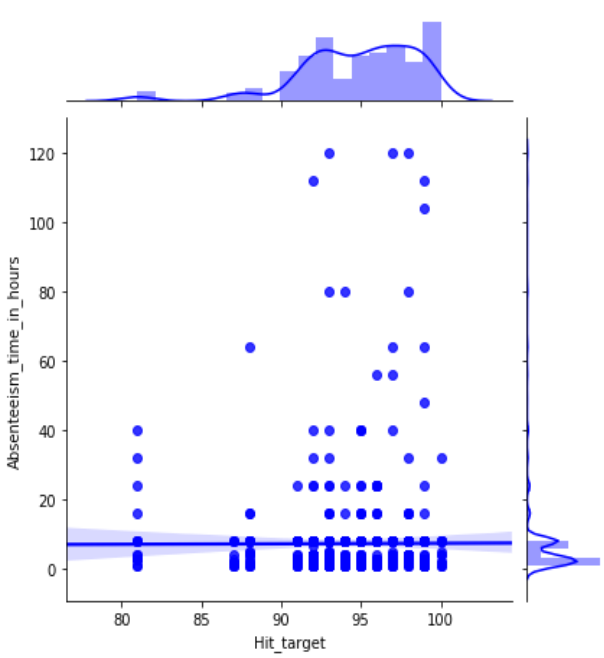
(1.) Certain infectious and parasitic diseases
(2.) Neoplasms
(3.) Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
(4.) Endocrine, nutritional and metabolic diseases
(5.) Mental and behavioural disorders
(6.) Diseases of the nervous system
(7.) Diseases of the eye and adnexa
(8.) Diseases of the ear and mastoid process
(9.) Diseases of the circulatory system
(10.) Diseases of the respiratory system
(11) Diseases of the digestive system
(12) Diseases of the skin and subcutaneous tissue
(13)Diseases of the musculoskeletal system and connective tissue
(14) Diseases of the genitourinary system
(15) Pregnancy, childbirth and the puerperium
(16) Certain conditions originating in the perinatal period
(17) Congenital malformations, deformations and chromosomal abnormalities
(18) Symptoms, signs and abnormal clinical and laboratory findings
(19) Injury, poisoning and certain other consequences of external causes
(20) External causes of morbidity and mortality
(21) Factors influencing health status and contact with health services.

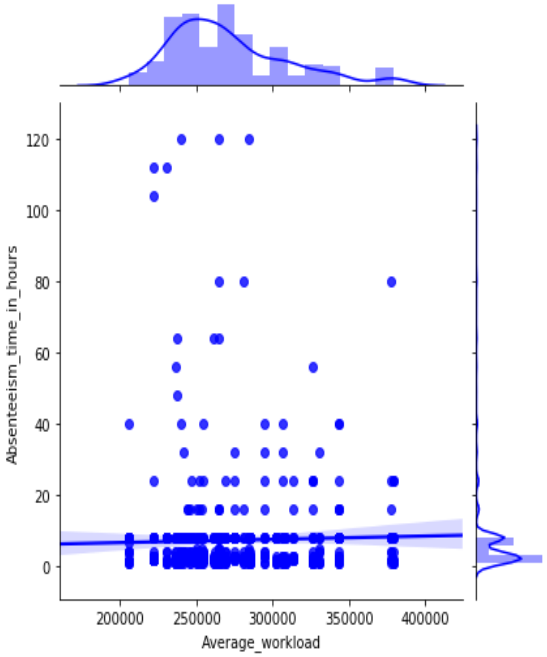
And 7 categories without (CID)
(22) patient follow-up
(23) medical consultation
(24) blood donation
(25) laboratory examination
(26) unjustified absence
(27) physiotherapy
(28).dental consultation

Bivarient Analysis: Extra Scatter plots









Basic Output Terms

Residual standard error: It is also called standard deviation error. It measures the average amount that the coefficient estimates vary from actual average value of our response variable. It helps in calculation of p-value.

t- value: It measures how many standard deviation our coefficients are away from 0. Coefficients should be far away from zero because if coefficient of any variable is near to 0 it means that variable is not able to explain the target variable i.e. that variable is an irrelevant variable. With help of t-value we calculate p-value.

P-value: It helps us to decide whether to accept or reject the variable i.e. a variable is contributing much information or not.

F-statistics: It is a good indicator of whether there is a relationship between our predictor and the response variable. F-statistics should be greater than 1.

Degrees of Freedom: Number of observation (training data) – Total number of variable

R Square: It is numerical value which tells us the amount of variance of the dependent variable is explained by all independent variable. It tells us how much our model is robust and what the strength of model on training data is.

Adjusted R Square: It is derived from R-Square values. Adjusted R Square will penalize the effect of additional variables which are not carrying much information. It should be always less than R Square.

AIC (Alkaline Information Criteria): It adjusts the loc likelihood based on the number of observation and complexity of the model.

BIC (Baisen Information Criteria): It is similar to AIC but has high penalty for models.

Omnibus: Provides combined statistical test for the presence of skewness and kurtosis. Basically it is breakdown of skewness and kurtosis.

Skew and Kurtosis: These tests are basically for time series dataset.

Null Deviance: It tells us how well the response variable is predicted by the model with intercept only.

Residual Deviance: It tells us how well the response variable is predicted by using null deviance and all other independent variables.

COMPLETE R-CODE:

```
###################################Employee Absenteeism##############################
#Remove all the objects stored
rm(list=ls())
#set & check working directory
setwd("F:/ed_project_2/R")
getwd()

#Load Libaries
x=c("ggplot2","corrgram","DMwR","caret","randomForest","unbalanced","C50","MASS","ridge",
"dummies","e1071","xlsx","gbm","rpart","data.table","Information","ROSE","inTrees",
"DataCombine","sampling","reshape","dplyr","plyr")

#Install pakages(x)
lapply(x,require,character.only=T)
rm(x)
#Reading Employee Absenteeism DATA
library(xlsx)
data=read.xlsx("Absenteeism_at_work_Project.xls",sheetIndex = 1,header=T)

########################Exploratory Data Analysis [EDA]#################################
#Observe top 5 rows
head(data)

#Getting the dimensions of data
dim(data)

#fetching Structure Of data
str(data)

#Retrieving Column names of train and test data.
colnames(data)

#get the length of unique value counts of columns:
for (i in 1:ncol(data)) {
print(colnames(data[i]))
print(length(unique(data[i])[,1]))
}
#Summary
summary(data)

###################################### Missing Value Analysis ####################
#Get number of missing values & their missing percentage
missing_val = data.frame(apply(data,2,function(x){sum(is.na(x))}))
missing_val$Columns = row.names(missing_val)
names(missing_val)[1] = "Missing_cnt"
row.names(missing_val) = NULL
missing_val$Missing_perc = (missing_val$Missing_cnt/nrow(data)) * 100
missing_val = missing_val[order(-missing_val$Missing_perc),]
missing_val = missing_val[,c(2,1,3)]
missing_val
```

```
#There are missing values in almost all the columns of the dataset, although in small amount.
#We'll drop all the missing value rows for target variable and We will imputenull values for all other features.
# Droping observation in which "Absenteeism time in hours" has missing value
data=data[!(is.na(data$Absenteeism.time.in.hours)),]

# Droping observation in which "Month_of_absence" has missing value
data = data[!is.na(data$Month.of.absence),]


#####Imputing Missing Values####
#Note:
#Impute missing values for all the independent featues(exept Average_Workload).
#Replace missing of any employee with information of same employee from other instances.
#for example if 'Age' of employee-x is missing, then impute it with 'Age' from other instance of employee-X.
col=colnames(data)
col=col[col!=c("ID","Work.load.Average.day.")]

#Loop to impute missing values for all the independent featues(exept Average_Workload)
for (i in unique(data$ID)) {
for (j in col) {
data[(data$ID==i) & (is.na(data[j])),j] = max(data[data["ID"]==i,j],na.rm = T)
}
}

##### Now let's analyze which is the best way to impute missing values for 'Average_Workload'####
##lets plot scatter plot between 'ID' &'Average_workload'
ggplot(data,aes(x=ID, y=Work.load.Average.day.))+ geom_point()
#The above scatter plot shows 'Average_workload' is not depends on "ID".

##lets plot scatter plot between 'Month_of_absence' &'Average_workload'
ggplot(data,aes(x=Month.of.absence, y=Work.load.Average.day.))+geom_point()
#From this plot, we can conclude that 'Average_Workload' is distributed mostly by month.
#So,let's impute missing 'Average_Workload' by mode of that month

##Impute "Average_Workload" based on corresponding "month's workload"
for (i in unique(data$Month.of.absence)) {
mod=mean(data[(data["Month.of.absence"]==i),"Work.load.Average.day."],na.rm = T)
data[(data["Month.of.absence"]==i) &
(is.na(data["Work.load.Average.day."])),"Work.load.Average.day."]=mod
}

#check for missing values
sum(is.na(data))
anyNA(data)

emp=data #stage-1 backup
#data=emp

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

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

#Converting data to proper formats (continuous & catagorical)
for (i in catagorical_var) {
data[,i]=as.factor(data[,i])
}
#Confirming the changes type
str(data)

############################ Data Visualisation ###################################
####Univariate Analysis####
###Distribution of catagorical features using bar plot
bar1 = ggplot(data = emp, aes(x = ID)) + geom_bar() + ggtitle("Count of ID") + theme_bw()
bar2 = ggplot(data = emp, aes(x = Reason.for.absence)) + geom_bar() +
ggtitle("Count of Reason for absence") + theme_bw()
bar3 = ggplot(data = emp, aes(x = Month.of.absence)) + geom_bar() + ggtitle("Count of Month") +
theme_bw()
bar4 = ggplot(data = emp, aes(x = Disciplinary.failure)) + geom_bar() +
ggtitle("Count of Disciplinary failure") + theme_bw()
bar5 = ggplot(data = emp, aes(x = Education)) + geom_bar() + ggtitle("Count of Education") +
theme_bw()
bar6 = ggplot(data = emp, aes(x = Day.of.the.week)) + geom_bar() + ggtitle("days of week") +
theme_bw()
bar7 = ggplot(data = emp, aes(x = Social.smoker)) + geom_bar() +
ggtitle("Count of Social smoker") + theme_bw()
bar8 = ggplot(data = emp, aes(x = Seasons)) + geom_bar() + ggtitle("Count of Seasons") +
theme_bw()
bar9 = ggplot(data = emp, aes(x = Son)) + geom_bar() + ggtitle("Count of sons") + theme_bw()
bar10= ggplot(data = emp, aes(x = Pet)) + geom_bar() + ggtitle("Count of pets") + theme_bw()
bar11= ggplot(data = emp, aes(x = Social.drinker)) + geom_bar() + ggtitle("Count of social drinker") +
theme_bw()

gridExtra::grid.arrange(bar1,bar2,bar3,bar4,bar5,bar6,bar7,bar8,bar9,ncol=3)
gridExtra::grid.arrange(bar10,bar11,ncol=1)

###Check the distribution of numerical data using histogram
hist1 = ggplot(data = data, aes(x =Transportation.expense)) +
ggtitle("Transportation.expense") + geom_histogram(bins = 25)
hist2 = ggplot(data = data, aes(x =Height)) +
ggtitle("Distribution of Height") + geom_histogram(bins = 25)
hist3 = ggplot(data = emp, aes(x =Body.mass.index)) +
ggtitle("Distribution of Body.mass.index") + geom_histogram(bins = 25)
hist4 = ggplot(data = data, aes(x =Weight)) +
ggtitle("Distribution of Weight") + geom_histogram(bins = 25)
hist5 = ggplot(data = data, aes(x =Distance.from.Residence.to.Work)) +
ggtitle("Distribution of Distance.from.Residence.to.Work") + geom_histogram(bins = 25)
hist6 = ggplot(data = data, aes(x =Service.time)) +
ggtitle("Distribution of Service.time") + geom_histogram(bins = 25)
hist7 = ggplot(data = data, aes(x =Work.load.Average.day.)) +
ggtitle("Distribution of Work.load.Average.day.") + geom_histogram(bins = 25)
```

```
hist8 = ggplot(data = data, aes(x =Hit.target)) +
ggtitle("Distribution of Hit.target") + geom_histogram(bins = 25)
hist9 = ggplot(data = data, aes(x =Age)) +
ggtitle("Distribution of Age") + geom_histogram(bins = 25)
hist10= ggplot(data = data, aes(x =Absenteeism.time.in.hours)) +
ggtitle("Distribution of Absenteeism.time.in.hours") + geom_histogram(bins = 25)

gridExtra::grid.arrange(hist1,hist2,hist3,hist4,hist5,hist6,hist7,hist8,hist9,ncol=3)
hist10


#Infer:
#People over 40+ years of age tends to take less leaves compare to others.
#A very large portion of the population have only passed â High Schoolâ .
#More then half of the employees in the company are â social drinkerâ .
#Only a very few portion of the employees in the company are â social smoker


#### KDE plot ( Normality Check)
library("kdensity")
par(mfrow=c(5,4))
par(mar=rep(2,4))
plot(density(emp$ID))
plot(density(emp$Reason.for.absence))
plot(density(emp$Month.of.absence))
plot(density(emp$Day.of.the.week))
plot(density(emp$Seasons))
plot(density(emp$Transportation.expense))
plot(density(emp$Distance.from.Residence.to.Work))
plot(density(emp$Service.time))
plot(density(emp$Age))
plot(density(emp$Work.load.Average.day.))
plot(density(emp$Hit.target))
plot(density(emp$Absenteeism.time.in.hours))
plot(density(emp$Body.mass.index))
plot(density(emp$Height))
plot(density(emp$Weight))
plot(density(emp$Social.smoker))
plot(density(emp$Social.drinker))
plot(density(emp$Son))
plot(density(emp$Education))
plot(density(emp$Disciplinary.failure))
par(mfrow=c(1,1))
#we can observe that none of the features follow Gaussian distribution.

####Bivariate Analysis####
#The relationship btwn independent continuous variables and Target variable.
g1=ggplot(emp, aes(x=Distance.from.Residence.to.Work,y=Absenteeism.time.in.hours))
+geom_point()+geom_smooth()
g2=ggplot(emp, aes(x=Service.time ,y=Absenteeism.time.in.hours)) +geom_point()+geom_smooth()
g3=ggplot(emp, aes(x=Age ,y=Absenteeism.time.in.hours)) +geom_point()+geom_smooth()
g4=ggplot(emp, aes(x=Height ,y=Absenteeism.time.in.hours)) +geom_point()+geom_smooth()
```

```
g5=ggplot(emp, aes(x=Work.load.Average.day. ,y=Absenteeism.time.in.hours))
+geom_point()+geom_smooth()
g6=ggplot(emp, aes(x=Hit.target ,y=Absenteeism.time.in.hours)) +geom_point()+geom_smooth()
g7=ggplot(emp, aes(x=Weight ,y=Absenteeism.time.in.hours)) +geom_point()+geom_smooth()
g8=ggplot(emp, aes(x=Body.mass.index ,y=Absenteeism.time.in.hours))
+geom_point()+geom_smooth()
g9=ggplot(emp, aes(x=Transportation.expense ,y=Absenteeism.time.in.hours))
+geom_point()+geom_smooth()
gridExtra::grid.arrange(g1,g2,g3,g4,g5,g6,g7,g8,g9,ncol=3)
#ALso, as we can see, 'Absenteeism_time_in_hours' are 0 in 36 places.
#This could be result of cancelled or withdrwan leaves. Lets drop these observations

#removing the observation in whic absent hour is zero
data=data[data$Absenteeism.time.in.hours>0,]


############################## Outlier Analysis #########################################
#Check for outliers using boxplots
emp_cnt=data[continuous_var]
for(i in 1:ncol(emp_cnt)) {
assign(paste0("box",i), ggplot(data = emp_cnt, aes_string(y = emp_cnt[,i])) +
stat_boxplot(geom = "errorbar", width = 0.5) +
geom_boxplot(outlier.colour = "red", fill = "grey", outlier.size = 1) +
labs(y = colnames(emp_cnt[i])) +
ggtitle(colnames(emp_cnt[i])))
}

#Arrange the plots in grids
gridExtra::grid.arrange(box1,box2,box3,box4,box5,box6,ncol=3)
gridExtra::grid.arrange(box7,box8,box9,box10,ncol=2)

#From boxplot Outliers are found in 'Service_time', 'Age', 'Average_workload',
#'Transportation_expense', 'Hit_target', 'Absenteeism_time_in_hours', 'Height'.

##Loop to Remove outliers using boxplot method
c_out=continuous_var[-c(2,3)] #excluding 'Service_time' & 'Age'
for(i in c_out)
{
print(i)
val = data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
#print(length(val))
data = data[which(!data[,i] %in% val),]
}
rownames(data)=NULL #resetting row index


############################## Feature Selection#####################################
#Correlation Plot
library(corrgram)
corrgram(data[,continuous_var] , order =F, upper.panel = panel.pie ,
text.panel = panel.txt , main = "Correlation Plot")
```

```
#correlation matrix
cor(data[continuous_var])
#This shows that there is multicollinearity in the dataset. "Body_mass_index" and "Weight" are highly
correlated
#droping corelated variable
data = subset(data,select=-c(Body.mass.index))
# Updating the Continous Variables and Categorical Variables after droping some variables
continuous_var=continuous_var[-9]


########## ANOVA Test###################
a_var=c(catagorical_var, "Absenteeism.time.in.hours")
absent=data[a_var]
summary(aov(formula=Absenteeism.time.in.hours~ID, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Reason.for.absence, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Month.of.absence, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Day.of.the.week, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Seasons, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Education, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Social.smoker, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Social.drinker, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Son, data = absent))
summary(aov(formula=Absenteeism.time.in.hours~Pet, data = absent))

################################## Feature Scaling ###################################
#since none of the features are normally distributed,so we are Normalizing the data
x_data=data #stage-zero backup

#Nomalisation
cnames=continuous_var[-9] #removing 'Absenteeism.time.in.hours'
for (i in cnames) {
print(i)
x_data[,i] = (x_data[,i] - min(x_data[,i]))/(max(x_data[,i])-min(x_data[,i]))
}
###################### CREATING DUMMIES FOR CATEGORICAL VARIABLES ###########
df1=x_data #stage-2 backup
#x_data=df1
library(fastDummies)
d1 = fastDummies::dummy_cols(x_data[catagorical_var] )
d1=subset(d1,select=-c(ID,Reason.for.absence,Month.of.absence,Day.of.the.week,Seasons,
Disciplinary.failure,Education,Social.drinker,Social.smoker,Son,Pet ))
d2=cbind(x_data,d1)
x_data=subset(d2,select=-c(ID,Reason.for.absence,Month.of.absence,Day.of.the.week,Seasons,
Disciplinary.failure,Education,Social.drinker,Social.smoker,Son,Pet))


######## Splitting the data into train and test#####
rmExcept(c("x_data","data","emp","df1","catagorical_var","continuous_var","cnames"))
set.seed(542)
train_index = sample(1:nrow(x_data), 0.8 * nrow(x_data))
train = x_data[train_index,]
test = x_data[-train_index,]
```

```
x_train = subset(train,select = -c(Absenteeism.time.in.hours))
x_test = subset(test,select = -c(Absenteeism.time.in.hours))
y_train = subset(train,select = c(Absenteeism.time.in.hours))
y_test = subset(test,select = c(Absenteeism.time.in.hours))


###################### DIMENSION REDUCTION USING PCA ##########################
#principal component analysis
prin_comp = prcomp(x_train)
#compute standard deviation of each principal component
std_dev = prin_comp$sdev
#compute variance
pr_var = std_dev^2
#proportion of variance explained
prop_varex = pr_var/sum(pr_var)
#cdf plot for principle components
plot(cumsum(prop_varex), xlab = "Principal Component",
ylab = "Cumulative Proportion of Variance Explained",
type = "b")
#add a training set with principal components
x_train.data = data.frame( prin_comp$x)
# From the above plot selecting 37 components since it explains almost 95+ % data variance
x_train.data =x_train.data[,1:37]
#transform test into PCA
x_test.data = predict(prin_comp, newdata = x_test)
x_test.data = as.data.frame(x_test.data)
#select the first 37 components
x_test.data=x_test.data[,1:37]


################################### Machine learning model##############################
x_train.data$Absenteeism.time.in.hours = y_train$Absenteeism.time.in.hours
x_test.data$Absenteeism.time.in.hours = y_test$Absenteeism.time.in.hours

########################1.Decision Tree#################################
#Build decsion tree using rpart
fit_DT = rpart(Absenteeism.time.in.hours ~., data = x_train.data, method = 'anova')
#Predict the test cases
pred_DT = predict(fit_DT,x_test.data[,-38])
#Calcuate MAE, RMSE, R-sqaured for testing data
print(postResample(pred = pred_DT, obs = x_test.data[,38]))
#Plot a graph for actual vs predicted values
plot(x_test.data$Absenteeism.time.in.hours,type="l",lty=2,col="green")
lines(pred_DT,col="blue")
#rmse=2.929


############################2.Random Forest#################################
#Train the model using training data
fit_RF = randomForest(Absenteeism.time.in.hours ~., data = x_train.data, ntree = 500)
#Predict the test cases
pred_RF = predict(fit_RF,x_test.data[,-38])
```

```
#Calcuate MAE, RMSE, R-sqaured for testing data
print(postResample(pred = pred_RF, obs = x_test.data[,38]))
#Plot a graph for actual vs predicted values
plot(x_test.data$Absenteeism.time.in.hours,type="l",lty=2,col="green")
lines(pred_RF,col="blue")
#rmse=2.351


##############################3.Linear Regression#################################
set.seed(100)
#Develop Model on training data
fit_LR = lm(Absenteeism.time.in.hours ~ ., data = x_train.data)
#Lets predict for testing data
pred_LR = predict(fit_LR,x_test.data[,-38])
# Results
print(postResample(pred = pred_LR, obs =x_test.data[,38]))
#Plot a graph for actual vs predicted values
plot(x_test.data$Absenteeism.time.in.hours,type="l",lty=2,col="green")
lines(pred_LR,col="blue")
#rmse=2.498


##########################4.Ridge regression###############################################
library(ridge)
Ridge = linearRidge(Absenteeism.time.in.hours~ ., data = x_train.data)
#Lets predict for testing data
pred_Ridge = predict(Ridge,x_test.data[,-38])
# Results
print(postResample(pred = pred_Ridge, obs =x_test.data[,38]))
#rmse=2.358


##########################5.KNN Regressor#########################################
#Develop Model on training data
KNN = knnreg(Absenteeism.time.in.hours~ ., data = x_train.data)
#Lets predict for testing data
pred_KNN = predict(KNN,x_test.data[,-38])
# Results
print(postResample(pred = pred_KNN, obs =x_test.data[,38]))
#Plot a graph for actual vs predicted values
plot(x_test.data$Absenteeism.time.in.hours,type="l",lty=2,col="red")
lines(pred_KNN,col="blue")
#rmse=2.500

#######################6.Support Vector Regression ###################################
#Develop Model on training data
fit_SVM = svm(Absenteeism.time.in.hours ~ ., data = x_train.data)
#Lets predict for testing data
pred_SVM = predict(fit_SVM,x_test.data[,-38])
# Results
print(postResample(pred = pred_SVM, obs =x_test.data[,38]))
#rmse=2.404
```

```
#######################7.Gradient Boosting#####################################
fit_GBDT = gbm(Absenteeism.time.in.hours~., data = x_train.data, n.trees = 500, interaction.depth =
2)
#Lets predict for testing data
pred_GBDT = predict(fit_GBDT,x_test.data[,-38], n.trees = 500)
# For testing data
print(postResample(pred = pred_GBDT, obs = x_test.data[,38]))
#rmse=2.703


#########################################2010#####################################
####Looking at models performance, we can say that 'Random forest' is the best model based on
RMSE value
##TEST DATA PREDICTIONS FOR YEAR 2010
Absent_prediction=df1[-train_index,]
Absent_prediction$Predicted_Absent_hours=pred_RF
head(Absent_prediction) #Sample output(with actual counts and predicted counts)
#Predicted absence hours of 2010
sum(Absent_prediction$Predicted_Absent_hours)
#Actual absence hours of 2010
sum(Absent_prediction$Absenteeism.time.in.hours)
#Predicted absence hours per month[2010]
aggre.months = ddply(Absent_prediction, c("Month.of.absence"), function(x)
colSums(x[c("Absenteeism.time.in.hours","Predicted_Absent_hours")]))
aggre.months


######################### PREDICTIONS FOR YEAR 2011 ##############################
#data for 2011
#Service and Age will be added by 1
data_2011 = data
data_2011$Service.time = data$Service.time + 1
data_2011$Age = data$Age + 1
data_2011=select(data_2011,-c("Absenteeism.time.in.hours"))

#Nomalisation
for (i in cnames) {
print(i)
data_2011[,i] = (data_2011[,i] - min(data_2011[,i]))/(max(data_2011[,i])-min(data_2011[,i]))
}

#get dummies
library(fastDummies)
d1 = fastDummies::dummy_cols(data_2011[catagorical_var] )
d1=select(d1,-catagorical_var)
d2=cbind(data_2011,d1)
emp_2011=select(d2,-catagorical_var)
#Using PCA
prin_comp = prcomp(emp_2011)
emp_2011 = data.frame( prin_comp$x)
emp_2011 =emp_2011[,1:37] # selecting 37 components since it explains almost 95+ % data variance
```

```
#predicting the 2011 model
predict_2011_absence = predict(fit_RF,emp_2011)

#Absent prediction 2011
Predit_2011=data_2011
Predit_2011$Absent_hours_2011=predict_2011_absence
head(Predit_2011)

#Predicted absence hours per month of 2011
monthly_absence = ddply(Predit_2011, c("Month.of.absence"), function(x)
colSums(x[c("Absent_hours_2011")]))
monthly_absence


#########*******MONTHLY LOSSES PREDICTED FOR YEAR 2011 PER MONTH****########
#In a month excluding weekend 22 days are working days.
#there are 36 employee in the xyz company
#8 hoursof work per day
Total_Monthly_hours = 22*8*36
# total losses % = (absent_hours / Total_Monthly_hours)*100
monthly_absence$monthly_loss_percentage = (monthly_absence$Absent_hours_2011
/Total_Monthly_hours) * 100
print(monthly_absence)

#Looking at the above results, we can observe that most likely,
#the company would incur most of loss in the month of 'March', followed by 'October' and 'July'.*

#saving output results
write.csv(Predit_2011,"R_Employee Absenteeism_2011.csv",row.names = FALSE)
write.csv(monthly_absence,"R_Monthly_loss.csv",row.names = FALSE)

#############################################################################################
#Hereby, concluding the project with above predictions.
#Regards
#SWAROOP H
#############################################################################################
```

Complete Python Code:

Employee Absenteeism

1.Problem 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?

Define and categorize the problem statement

The problem statement is to analyze the cause of absenteeism and predict the every month losses in 2011 due to absenteism. Our task is to build a regression model which will predict the absentieesm in hours based on the employee attributes and information. Although, the problem statement is a Multivariate Time-Series Problem. We will approach it as a Regression Problem

.

```
import os
os.chdir("F:\ed_project_2\py")
## Import all the required libraries
import os
import pandas as pd
import numpy as np
from scipy import stats

#for preprocessing
from fancyimpute import KNN
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import StandardScaler

#for model building
import sklearn
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR

#for model selection
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
```

```
#for visualization
import matplotlib.pyplot as plt
import seaborn as sns

#for model evaluation
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import mean_absolute_error

import sys
if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

Reading Data:

```
#Importing dataset
data = pd.read_excel("Absenteeism_at_work_Project.xls")
```

2. Exploratory Data Analysis[EDA]

```
data.head(5)
```

Here the target variable is: 'Absenteeism time in hour' and other 20 columns, which are mix of continous and categorical variables are predictors.

```
print("column names: {}".format(data.columns))
print("*"*99)
print("shape of data: {}".format(data.shape))
print("*"*99)
print("Total number of Features : {}" .format(data.shape[1]))


#Replacing the spaces between the variable names with underscore.
data.columns = data.columns.str.replace(' ', '_')
data = data.rename(columns = {'Work_load_Average/day_': 'Average_workload'})
data.columns


#Summary
data.describe()
# Check the properties of the data
data.info()
data.nunique()
```

Infer:

There are null values in the dataset.

The datatypes are int and float.

2.1 Missing Value Analysis

```
#Creating dataframe with missing values present in each variable
missing_val = pd.DataFrame(data.isnull().sum()).reset_index()

#Calculating percentage of missing value
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_Values'})
missing_val['Missing_percentage'] = (missing_val['Missing_Values']/len(data))*100
missing_val = missing_val.sort_values('Missing_percentage', ascending = False).reset_index(drop = True)
missing_val
```

Infer:

There are missing values in almost all the columns of the dataset, although in small amount. We'll drop all the missing value rows for target variable and We will impute null values for all other features.

```
# Droping observation in which "Absenteeism time in hours" has missing value
data = data.drop(data[data['Absenteeism_time_in_hours'].isnull()].index, axis=0)
# Droping observation in which "Month_of_absence" has missing value
data = data[~data["Month_of_absence"].isnull()]
```

2.1.1 Imputing missing values

Impute missing values for all the independent featues(exept Average_Workload).
Replace missing of any employee with information of same employee from other instances.
for example if 'Age' of employee-x is missing, then impute it with 'Age' from other instance of employee-X.

```
#lets impute missing values for all the independent featues(exept Average_Workload) with refer to "ID"
col=['Reason_for_absence', 'Day_of_the_week','Seasons', 'Transportation_expense',
    'Distance_from_Residence_to_Work','Service_time', 'Age', 'Hit_target','Disciplinary_failure',
    'Education', 'Son', 'Social_drinker','Social_smoker', 'Pet', 'Weight', 'Height', 'Body_mass_index']

for i in data['ID'].unique():
   for j in col :
      data.loc[((data['ID'] == i) & (data[j].isna())), j] = data[(data.ID==i)][j].max()
```

* Now let's analyze which is the best way to impute missing values for 'Average_Workload'

```
#lets plot scatter plot between 'ID' &'Average_workload'
plt.scatter(x='ID', y='Average_workload',data=data)
plt.xlabel('ID')
plt.ylabel('Average_workload')
```

Infer:

The above scatter plot shows 'Average_workload' is not depends on "ID".

```
#lets plot scatter plot between 'Month_of_absence' &'Average_workload'
plt.scatter(x='Month_of_absence', y='Average_workload', data=data)
plt.xlabel('Month_of_absence')
```

```
plt.ylabel('Average_workload')
```

Infer:

From above plot, we can conclude that 'Average_Workload' is distributed mostly by month. So,let's impute missing 'Average_Workload' by mode of that month.

```
#Impute Average_Workload with the mode of corresponding month's workload
for i in data['Month_of_absence'].unique():
    mode= stats.mode(data[data['Month_of_absence']==i]['Average_workload'])[0][0]
    data.loc[((data['Month_of_absence']==i) & pd.isna(data['Average_workload'])),'Average_workload'] = mode

data.isnull().sum()


emp=data.copy() #backup
```

2.2 Variable Identification

```
#Converting data to proper formats(Variable Identification)

categorical_var=['ID','Reason_for_absence', 'Month_of_absence', 'Day_of_the_week','Seasons',
                 'Disciplinary_failure', 'Education', 'Son', 'Social_drinker','Social_smoker', 'Pet']

continous_var=['Transportation_expense', 'Distance_from_Residence_to_Work','Service_time', 'Age',
               'Average_workload', 'Hit_target', 'Weight', 'Height', 'Body_mass_index','Absenteeism_time_in_hours']

for i in categorical_var:
    data[i] = data[i].astype("category")
```

2.3 Data Visualisation

2.3.1 Univariate Analysis

```
#Histogram plot for distribution of features in the data
import matplotlib.pyplot as plt
emp.hist(figsize=(18,15))
plt.show()
```

Infer:

From the above plot following observation are made...

- People over 40+ years of age tends to take less leaves compare to others. Majority of the employees working in the company have age below 40 years.
- A very large portion of the population have only passed ‘High School’.
- More then half of the employees in the company are ‘social drinker’.
- Only a very few portion of the employees in the company are ‘social smoker’

```
# Kernel Density Estimate (KDE) plot for distribution of features in the data
emp.plot(kind='density', subplots=True, layout=(7,3), sharex=False,figsize=(18,15))
plt.show()
```

Infer : None of the features follow Normal distribution

From the above Box and whisker plots, we can observe that not all the features contains outliers. Continuous features like 'Weight', 'Distance from residence to work' does not contain any outliers at all. Few features like 'Average_workload', 'Hit_target' 'Age', 'Service_time' and 'Height', have a very few outliers.

It is also evident from the above plot that none of the features are symmetric to the median and it can easily be interpreted that none of the features follow symmetric distribution. Also, it can also be observed that Median of the feature 'Body mass index' is very close to 25th percentile value which means median of this feature is almost equal to 25th percentile.

```
#Box and Whiskers plot of features in the data
data[continous_var].plot(kind='box', subplots=True, layout=(8,3), sharex=False, sharey=False, fontsize=10,fig
size=(15,10))
plt.subplots_adjust(left=0.125, bottom=0.1, right=0.9, top= 3,wspace=0.2, hspace=0.2)
plt.show()
```

Infer:

The dataset contains both contionus & categorical features. the purpose of boxplot is to visualize the outliers in the continous varibles.

- Features like 'Service_time', 'Average_workload', 'Transportation_expense', 'Hit_target', contains a few outliers.
- It can also be seen that features like 'Absenteeism_time_in_hours', 'Height' contains the most number of outliers.

```
# Checking the distribution of target feature
sns.mpl.rc("figure", figsize=(15,6))
sns.countplot(x="Absenteeism_time_in_hours", data= data)
plt.ylabel('Frequency')
plt.xlabel('Absenteeism_time_in_hours')
plt.title("Frequency of Classes")
plt.show()
```

Infer :

- *From the above plot, it is obvious that maximum number (i.e around 200) of employes are absent for 8 hours.*
- *Around 420 employes are absent for 1 to 4 hours.*
- *Only 36 employes doesnt take any leave.*
- *There are only 27 employes, who take leave for 40-120 hours.*

One observation that is worth noting from the above plot is that, after class 8, every other class is a multiple of 8.

ALso, as we can see, 'Absenteeism_time_in_hours' are 0 in 36 places.This could be result of cancelled or withdrwan leaves.
Lets drop these observations

```
#droping the observations of 'Absenteeism_time_in_hours' with '0'
data = data[(data.Absenteeism_time_in_hours > 0)]
```

2.3.2 Bivarient Analysis

```
#The relationship btwn independent continous variables and Target variable using JOINT PLOTs
for i in continous_var:
    sns.jointplot(i, "Absenteeism_time_in_hours", data=data, kind='reg', color='b', size=6, dropna=True)
```

Infer:

- *This clearly shows concentration of leaves more where the 'Transportation_expense' is between 150-300.*
- *This clearly shows concentration of leaves more where the 'Distance_from_Resistance_to_work' is between 10-30 km.*
- *Employees with 'service_time'(service years) less than 8 and more than 18 tends to take less leaves.*

```
#The relationship btwn independent categorical variables and Target variable.
for i in categorical_var:
 data.groupby(i)['Absenteeism_time_in_hours'].count().plot(kind='bar',figsize=(10,4))
 plt.ylabel('Absenteeism_time_in_hours')
 plt.show()
```

Infer:

- *Longest hours of Absence for the Reasons as follows 23,28,13,27,19...*
  - *#23 - medical consultation (23).*
  - *#28 - dental consultation (28).*
  - *#13 - Diseases of the musculoskeletal system and connective tissue.*
  - *#27- physiotherapy (27).*
  - *#19 - Injury, poisoning and certain other consequences of external causes.*
- *From 'Month_of_absent' distribution, we can see that frquency of leaves are more or less uniformally distributed over months, with highest no. of leaves taken in March, Feb and July(holiday season).*
- *From, Absent_Weekday distribution, we can see that frquency of leaves are mostly distributed, with most frequent leaves on 'Monday'.*
- *Displinary failures are very least.*
- *Employee with Education 'High School' tend to take more hours of absence.*
- *Employee with 3-4 kids tend to take less hours of absence.*
- *'Social Drinker' takes little more leaves than non drinker.*
- *From, 'Son' and 'Pet', we can see that people having no kids and no pets(no family responsibilities) tend to take frequent leaves.*

2.4 Outlier Analysis

From boxplot Outliers are found in 'Service_time', 'Average_workload', 'Transportation_expense', 'Hit_target', 'Absenteeism_time_in_hours', 'Height'.

```
#Create a function to remove outliers
def rm_outlier(data,col_name):
    q75,q25 = np.percentile(data[col_name],[75,25])      # Getting 75th and 25th Percentile
    iqr = q75 - q25                                      # Calculating Interquartile range
    lower_fence = q25 - (iqr*1.5)                        # Lower fence
    upper_fence = q75 + (iqr*1.5)                        # Upper fence
    df = data.loc[(data[col_name] > lower_fence) & (data[col_name]< upper_fence)]
    return df

#remove outliers (excluding  'Service_time' & 'Age')
c_out=continous_var.copy()
c_out.remove('Service_time')
c_out.remove('Age')

for i in c_out:
    data=rm_outlier(data,i)

data.shape
```

2.5 Feature Selection

2.5.1 Correlation Plot

```
#Correlation Plot
corr=data[continous_var].corr()

#Set the width and hieght of the plot
plt.figure(figsize=(10, 10))
#Plot using seaborn library
plt.title('Correlation Plot')
correlation_plot = sns.heatmap(corr, linewidths=0.4,vmax=1.0, square=True, cmap="cubehelix", linecolor='k',
annot=True)
```

Infer:

** This shows that there is multicollinearity in the dataset. "Body_mass_index" and "Weight" are highly correlated.*

r("Body_mass_index")< r("Weight") while comparing with target variable.
So,we are droping "Body_mass_index".

```
#correlation matrix
corr
```

```
#droping corelated variable
data = data.drop(['Body_mass_index'], axis=1)

# Updating the Continous Variables and Categorical Variables after droping some variables
continous_var.remove('Body_mass_index')
```

2.5.2 ANOVA Test

```
#loop for ANOVA test Since the target variable is continuous
for i in categorical_var:
    f, p = stats.f_oneway(data[i], data['Absenteeism_time_in_hours'])
    print("P value for variable "+str(i)+" is "+str(p)  )
```

Spliting the data in to target & predictor

```
data= data.reset_index(drop = True)
#spliting the train data
x_data = data.drop(['Absenteeism_time_in_hours'], axis=1)
y_data = data['Absenteeism_time_in_hours']
```

2.6 Feature Scaling

```
#Nomalisation
cnames=continous_var.copy()
cnames.remove('Absenteeism_time_in_hours')

for i in cnames:
    print(i)
    x_data[i] = (x_data[i] - np.min(x_data[i]))/(np.max(x_data[i]) - np.min(x_data[i]))
```

Get dummies for categorical variables

```
# Copying dataframe
df1 = x_data.copy()

# Get dummy variables for categorical variables
x_data = pd.get_dummies(data = x_data, columns = categorical_var)

x_data.shape
```

2.6 Dimensionality Reduction using Principal Component Analysis

```
from sklearn.decomposition import PCA

# Converting data to numpy array
#X = x_data.values
X=x_data.copy()
```

```
# Data has 115 variables so no of components of PCA:115
pca = PCA(n_components=115)
pca.fit(X)

# The amount of variance that each PC explains
var= pca.explained_variance_ratio_

# Cumulative Variance explains
var1=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)

plt.plot(var1)
plt.xlabel('Principal Components')
plt.ylabel('Cumulative proportion of variance')
plt.title('CDF plot')
plt.show()
```

Infer :

We can see that, almost 98% variance is explained by less than 50 variables.

```
X=x_data.copy()

# From the above plot selecting 95% data variance
pca = PCA(.95)

# Fitting the 95% of selected components to the data
X=pca.fit(X).transform(X)

X=pd.DataFrame(X)
X.shape[1]
```

Almost 95% variance is explained by just 37 variables. So, we will reduce the components to 37

```
#splitting data into test and train
x_train, x_test, y_train, y_test = train_test_split(X, y_data, test_size=0.2)
#x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2)
print(x_train.shape)
print(x_test.shape)
```

3. MODELLING

3.1 Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor

#defining the function for Decision tree with variable depth
def DT_model(depth):
    DT = DecisionTreeRegressor( max_depth = depth).fit(x_train, y_train)
    y_pred = DT.predict(x_test)
    print('depth : {} \t : Decision Tree rmse: {}'.format(depth,(sqrt(mean_squared_error(y_test,y_pred)))))
```

```
for depth in [1,2,3,4,5,10,15,None]:
    DT_model(depth)
```

Freeze the Decisiontree model with max_depth = 1.

3.2 Random Forest

```
from sklearn.ensemble import RandomForestRegressor

#defining the function for Random forest with n_est & depth.
def RF_model(n_est, depth):
 RF = RandomForestRegressor(n_estimators=n_est, max_depth=depth, n_jobs=-1).fit(x_train, y_train)
 y_pred = RF.predict(x_test)
 print('depth : {},\t n_estimators : {}  \t: RF rmse: {}'.format(depth, n_est, (sqrt(mean_squared_error(y_test,y_pr
ed)))))

for n_est in [100, 200, 500, 1000]:
    for depth in [2, 5, 10 , 20]:
        RF_model(n_est, depth)
```

3.3 Linear regression (Ordinary Least Squares)

```
from sklearn.linear_model import LinearRegression
ols = LinearRegression()
model = ols.fit(x_train, y_train)
y_pred = model.predict(x_train)
print('Ordinary Least Squares rmse: {}'.format(sqrt(mean_squared_error(y_train,y_pred))))
```

3.4 Ridge Regression

```
from sklearn.linear_model import Ridge
def RIDGE_model(alpha):
    model = Ridge(alpha= alpha).fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print('alpha : {}  \t:Ridge rmse: {}'.format(alpha, (sqrt(mean_squared_error(y_test,y_pred)))))
for alpha in [0.1,0.5, 1.0,3.0,7.0,10.0]:
    RIDGE_model(alpha)
```

3.5 KNN Regressor

```
from sklearn.neighbors import KNeighborsRegressor
def KNN_model(n_neigh):
    model = KNeighborsRegressor(n_neighbors= n_neigh).fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print('n_neighbours : {}  \t: KNN rmse: {}'.format(n_neigh,(sqrt(mean_squared_error(y_test,y_pred)))))
for n_neigh in [3,5,7,10,15,20,25,30]:
    KNN_model(n_neigh)
```

3.6 Support Vector Regression

```
from sklearn.svm import SVR
def SVR_model(C, gamma):
    model = SVR(C= C, gamma = gamma).fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print('C : {} ,::\t gamma : {} ::\t:SVR rmse: {}'.format(C, gamma ,(sqrt(mean_squared_error(y_test,y_pred))))
)
for C in [1, 10, 100,1000]:
    for gamma in [0.001, 0.0001]:
        SVR_model(C, gamma)
```

3.7 Gradient Boosting

```
from sklearn.ensemble import GradientBoostingRegressor
def GBR_model(depth, learning_rate):
    model = GradientBoostingRegressor( max_depth = depth, learning_rate =learning_rate).fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print('depth: {},\t:learning_rate: {} \t: rmse: {}'.format(depth,learning_rate,(sqrt(mean_squared_error(y_test,y
_pred)))))

for depth in [1,2,5,None]:
    for learning_rate in [0.001,0.01,0.1]:
        GBR_model(depth, learning_rate)
```

4. Model selection

```
ML = pd.DataFrame({"rmse":[2.570,2.143,2.488,2.116,2.273,2.175,2.147],
                   "Model" : ['Decision Trees',"Random Forest" ,'Linear Regression','Ridge Regression','KNN Regressi
on',
                              'Support Vector Regression',"Gradient Boosting"]})
ML
```

Infer:

Looking at models performance, we can say that 'Ridge Regression' is the best model based on RMSE value, followed by 'Random Forest' and 'Support Vector Regression' So, we can pick any of those and perform modelling. All these model will give almost same results.

```
#Ridge Regression with alpha=10
model = Ridge(alpha= 10).fit(x_train, y_train)
y_pred = model.predict(x_test)
print("rmse : {}".format(sqrt(mean_squared_error(y_test,y_pred))))

#RF
#model = RandomForestRegressor(n_estimators=500, max_depth=20, n_jobs=-1).fit(x_train, y_train)
#y_pred = model.predict(x_test)
#print("rmse : {}".format(sqrt(mean_squared_error(y_test,y_pred))))
```

```
TEST DATA PREDICTIONS FOR YEAR 2010--->
Absence_prediction=df1.iloc[x_test.index.values,:]
Absence_prediction["Absent_hours"]=y_test
Absence_prediction["Predicted_Absent_hours"]=y_pred
Absence_prediction.head()              #Sample output(with actual counts and predicted counts)

#Predicted absence hours of 2010
Absence_prediction.Predicted_Absent_hours.sum()

#Actual absence hours of 2010
Absence_prediction.Absent_hours.sum()

#Predicted absence hours per month[2010]
Absence_prediction.groupby('Month_of_absence').sum().reset_index()[['Month_of_absence','Absent_hours','Predicted_Absent_hours']]
```

Since, Ridge Regression/ Random Forest model is our final model to be used for prediction, We'll use this model to predict the losses of 2011. Let's prepare data for 2011.

To prepare data for 2011,assuming that all the employees are retained in 2011 and all other condition remains and same trends continues, we need to add +1 to 'Service_time' and 'Age'(keeping all other features same).

5. PREDICTIONS FOR YEAR 2011

```
#data for 2011
#Service and Age will be added by 1

data_2011 = data
data_2011.Service_time = data.Service_time + 1
data_2011.Age = data.Age + 1
data_2011= data_2011.drop(columns = ['Absenteeism_time_in_hours'])

#Normalizing
for i in cnames:
 data_2011[i] = (data_2011[i] - np.min(data_2011[i]))/(np.max(data_2011[i]) - np.min(data_2011[i]))

# Get dummy variables for categorical variables
emp_2011 = pd.get_dummies(data = data_2011, columns = categorical_var)

#pca
pca = PCA(.95)
emp_2011=pca.fit(emp_2011).transform(emp_2011)

#predicting the 2011 model
predict_2011_absence =model.predict(emp_2011)
```

```
#Absent prediction 2011
Predit_2011=data_2011
Predit_2011["Absent_hours_2011"]=predict_2011_absence
Predit_2011.head()

#Predicted absence hours per month of 2011
monthly_absence= Predit_2011.groupby('Month_of_absence').sum().reset_index()[['Month_of_absence','Absent
_hours_2011']]
monthly_absence=monthly_absence.drop(monthly_absence.index[0])
monthly_absence
```

---------MONTHLY LOSSES PREDICTED FOR YEAR 2011 PER MONTH ----------

```
#In a month excluding weekend 22 days are working days.
#there are 36 employee in the xyz company          : data["ID"].nunique() i.e.36
#8 hoursof work per day

Total_Monthly_hours = 22*8*36

# total losses % = (absent_hours / Total_Monthly_hours)*100
monthly_absence['monthly_loss_percentage'] = (monthly_absence["Absent_hours_2011"]/Total_Monthly_hour
s) * 100


monthly_absence
```

Looking at the above results, we can observe that most likely,the company would incurred most of loss in the month of 'March', followed by 'October' and 'July'.

```
#saving output results
Predit_2011.to_csv("Py_Employee Absenteeism_2011.csv",index=False)
monthly_absence.to_csv("Py_Monthly_loss.csv",index=False)
```

Hereby, concluding the project with above predictions.

Regards
SWAROOP H

References:

https://edwisor.com/career-data-scientist

https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/

https://medium.com/district-data-labs/principal-component-analysis-with-python-4962cd026465

https://www.udemy.com/python-for-data-science-and-machine-learning-bootcamp/

https://seaborn.pydata.org/tutorial/relational.html

MIT 18.05, Introduction to Probability and Statistics, taught by Jeremy Orloff and Jonathan Bloom. Provides intuition for probabilistic reasoning & statistical inference, which is invaluable for understanding how machines think, plan, and make decisions.

"An Introduction to Data Cleaning with R" – by "Edwin de Jonge and Mark van der Loo".