**PROJECT REPORT ON EMPLOYEE ABSENTEEISM**

**BY USHA RAJU**

**INDEX**

1. Introduction

1.1 Problem Statement

1.2 Data

2. Methodology

2.1 Data Exploration

* Identification of variables and their datatypes
* Descriptive statistics
* Conversion of data types into required ones
* Univariate, Bivariate and Multivariate analysis
* Missing Value Analysis
* Outlier Analysis
* Feature Engineering
* Data visualization

2.2 Modelling

* + - Model Development
      * + Linear Regression
        + Decision Tree
        + Random Forest
        + Gradient Boosting
* Model Evaluation
* Model Selection

2.3 Conclusion

**1. Introduction**

**1.1 Problem Statement**

XYZ is a courier company. As we appreciate that human capital plays an important role in collection, transportation and delivery. The company is passing through genuine issue of Absenteeism. The company has shared 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**

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case our company has provided a data set with following features, we need to go through each and every variable of it to understand and for better functioning.

Dataset Characteristics: Timeseries Multivariant

Number of Attributes: 21

Missing Values: Yes

**Attribute Information:**

1. Individual identification (ID)

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

I Certain infectious and parasitic diseases II Neoplasms III Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism IV Endocrine, nutritional and metabolic diseases V Mental and 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 (kilometres)

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)

**2 .Methodology**

**2.1 Data Exploration**

When required to build a predictive model, we need to understand the raw data before we start developing a model. This includes multiple pre-processing steps such as:

1. Identification of variables and their datatypes
2. Descriptive statistics
3. Conversion of data types into required ones
4. Univariate, Bivariate and Multivariate analysis
5. Missing Value Analysis
6. Outlier Analysis
7. Feature Engineering

**1. Identification of variables and their datatypes:**

* To get the dimensions of the dataset.
* To identify the target and predictor variables present in the dataset.
* To get the datatypes of the variables present.

**2. Descriptive statistics:**

* To get the not null entries present in the dataset.
* To get the summary of variables from dataset. This includes count, mean, standard deviation, minimum and maximum values and the quantiles of the data.
* To get the unique values and count of unique values from the dataset.

**3. Conversion of data types of variables into required ones:**

* To Convert the datatype of a variable into its required type.
* The following features are converted to categorical:
  + Reason for absence can only be one of the 21 categories stratified as per International Code of Diseases (ICD)
  + Month of absence can only be 1-12 months.
  + Day of the week can only be 1-7 days.
  + Seasons can only be 1-4 seasons.
  + Disciplinary failure can only be either (yes=1/ no=0)
  + Education can be one of the four(high school (1), graduate (2), postgraduate (3), master and doctor (4))
  + Social drinker can only be either (yes=1/ no=0)
  + Social smoker can only be either (yes=1/ no=0)
  + Pet denotes the number of pets which can be 0 - <10 (maximum value for example)
  + Son denotes the number of children which can be 0 - <10 (maximum value for example)

**4. Univariate, Bivariate and Multivariate analysis:**

Univariate analysis provides summary statistics for each field in the raw dataset (or) summary only on one variable. Bivariate analysis is performed to find the relationship between each variable in the dataset and the target variable of interest (or) using two variables and finding relationship between them. Multivariate analysis is used to visualize more than two variables at once.

**5. Missing value Analysis:**

Missing or null values have to be eliminated from the datasets as it will impact in the performance of the model developed.

* To get the sum of null values from the dataset.
* To create dataframe to analyze missing values present.
* Since the percentage of missing values of missing values in the dataset is less, we can impute the missing values statistically.
* Imputation of missing values in the dataset:

Imputation is a method to fill in the missing values with estimated ones. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable.

* Since, the attributes are numerical and categorical, we can opt for Mean/Median/Mode and KNN methods for imputation. To find the apt method out of all,
  + Remove any random value from the column missing and replace with NA.
  + Impute the null value with methods one by one.
  + Select the method with output nearly equal to the original value.

**6. Outlier Analysis:**

Outliers are the values that are not in the desired range. Based on the basic understanding of the data from Descriptive statistics, it is obvious that the outliers present in the data are due to incorrectly entered or measured data. Hence, we can remove those values.

**Extreme value Analysis:**

* Absenteeism time in hours:
  + Absenteeism hours can’t be zero and hence any value less than or equal to zero is an outlier.
* Month of absence:
  + Maximum and minimum value for the month of absence can be 12 and 1 and any value less than or equal to that is an outlier.
* Reason for absence:
  + Reason for absence can't be zero and hence zero values can be replaced by category - 26 (unjustified absence).

**Boxplot analysis:**

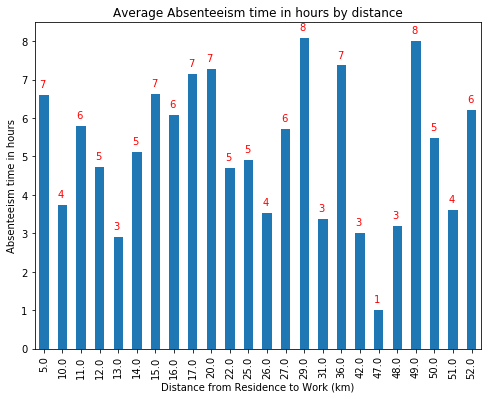
Boxplot analysis is carried out to check if there are any values that are not in the desired minimum and maximum range.

**7. Feature Engineering:**

**Data Visualization:**

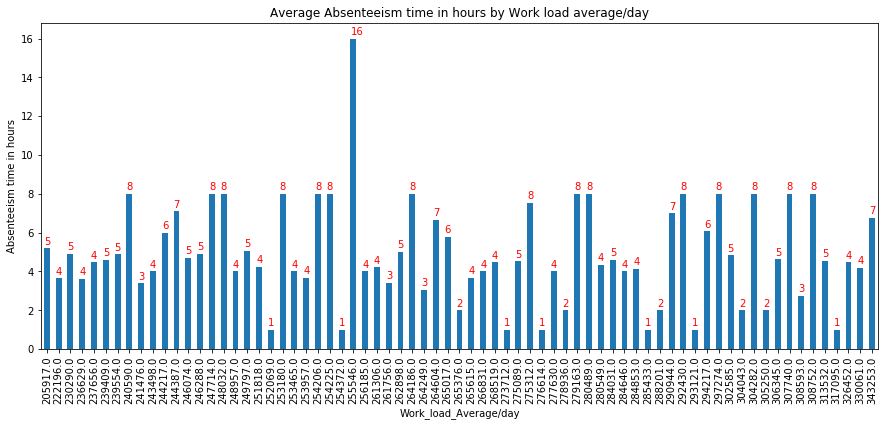
Data visualization helps us to understand the relationship between features. Here, to understand how each independent feature is related to the target feature, we need to perform visualization on data.

***1. Distance from Residence to Work Vs Absenteeism time in hours:***



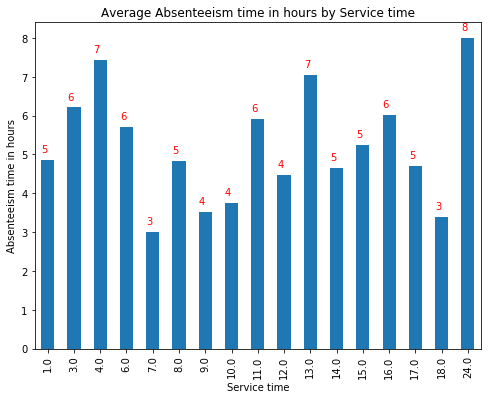
General hypothesis is that distance from residence to work will have an impact on absenteeism. Here, average hours of absent remains the same irrespective of the distance from residence to work of the employees. There is concentration of more leaves where the distance of residence from work is between 10-30 kms.

***2. Work load Average/day Vs Absenteeism time in hours***



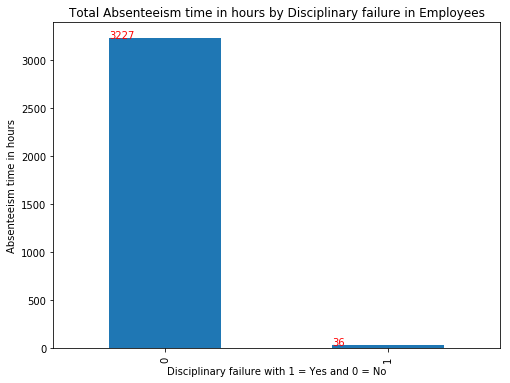
Thus, average hours of absent remains same irrespective of the work load average/day.

***3. Service time Vs Absenteeism time in hours***



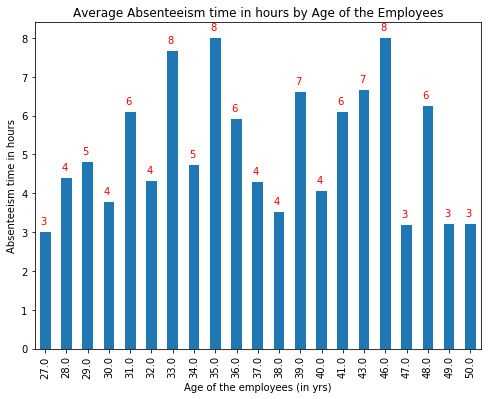
Thus, the employees with service years > 8 tend to take more leaves.

***4. Disciplinary failure Vs Absenteeism time in hours:***



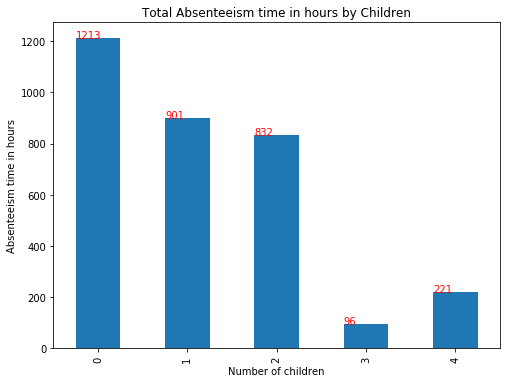
Thus, it is observed that employees with no disciplinary failure have the highest absent hours in total.

###### ***5. Age of the employees Vs Absenteeism time in hours:***



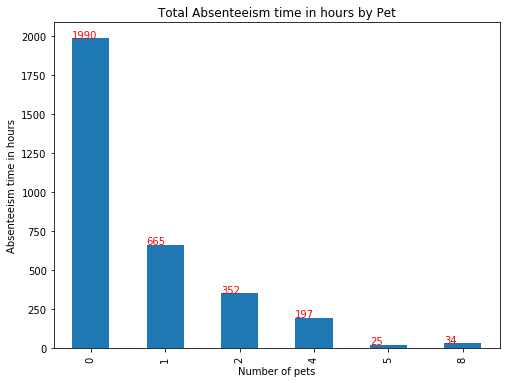
Hypothesis is that as the age increases, employees tend to take more leave compared to others due to health issues. Here, people over 45+ years of age tends to take less leaves compared to others.

***6. Number of children Vs Absenteeism time in hours***



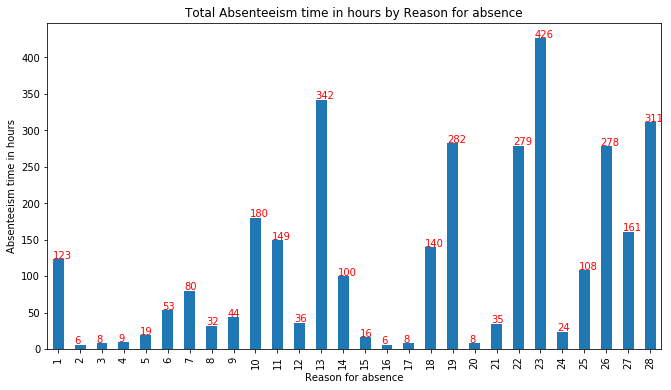
It is interesting to note that employees with no issues have highest absenteeism. This is followed by employees with 1 child and it might be due to the fact that there would be noone else to take care of the child if it falls sick and one parent has to stay back with the kid. In case of 3 or 4 children, there is a high chance that older siblings take care of younger ones.

***7. Number of pets Vs Absenteeism time in hours:***



It is to be noted that the employees with no pet or 1 pet are frequent absentees.

***8. Reason for absence Vs Absenteeism time in hours:***



###### ***Reasons and Remedies/Suggestions to reduce absenteeism:***[***¶***](http://localhost:8888/notebooks/Employee%20absenteeism.ipynb#Reasons-and-Remedies/Suggestions-to-reduce-absenteeism:)

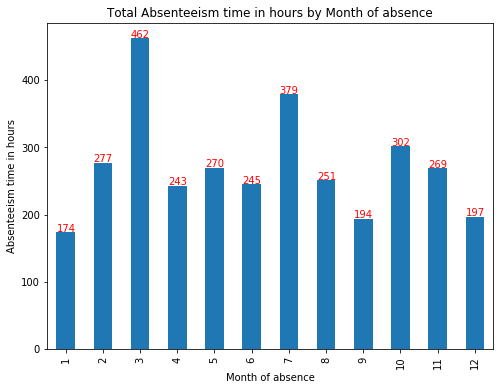
The top four reasons for absence are:

* Diseases of the musculoskeletal system and connective tissue
* Injury, poisoning and certain other consequences of external causes
* Medical consultation
* Unjustified absence

***Remedies/Suggestions to reduce absenteeism:***

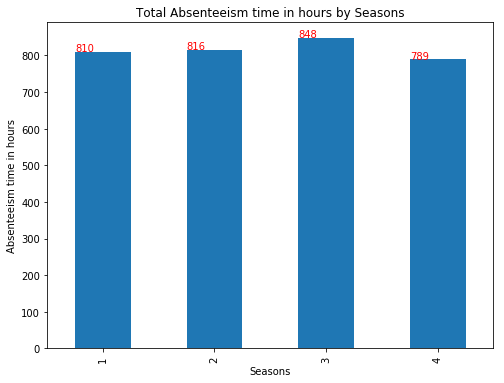
1. Musculoskeletal system disease is the major reason of absenteeism. Bad working posture and high workload are possible reasons for the high incidence of musculoskeletal disease. Company should conduct a study on the working postures of people and come up with ergonomic workplace design. Company should try to optimize workload keeping in mind occupational health of working people.
2. Injury, poisoning and certain other consequences of external causes can be the consequences of the bad and unsafe working environment. Safety of the employees should be the major concern of the company.
3. Medical and dental consultation can be brought down by optimizing workloads and adapting to employee friendly workplace and conducting more programs and medical camps to create awareness on importance of physical and mental wellness among employees.
4. Unjustified absence is too high.Company should try to reduce high workloads and set up measures so that employees don't feel work stress and they get to discuss work related problems and pressures.
5. Counselling sessions are to be conducted on regular basis to employees to get their feedbacks and help them in their work and personal issues if any.

###### ***9. Month of absence Vs Absenteeism time in hours:***



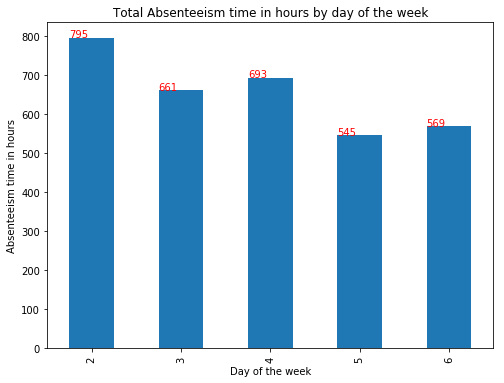
We can infer that maximum hours of absent are recorded in the month of March followed by July. we can check the reason for absence in these months to analyse further.

***10.*** ***Seasons Vs Absenteeism time in hours:***



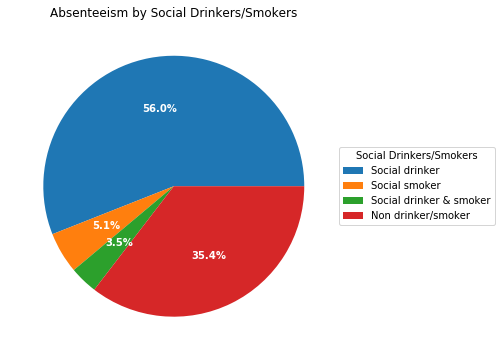
Winter season has highest absenteeism followed by Summer. It is also to be noted that March and July month has highest absenteeism.

***11.*** ***Day of the week Vs Absenteeism time in hours:***



Highest percentage of absenteeism is recorded on start of the week, Mondays followed by Tuesdays which can be due to the fact that most people travel over weekend and they tend to extend their holiday plan till monday/tuesday.

***12***. ***Effect on Absenteeism by Social drinking/smoking in employees:***



Looks like 57% of Social drinkers and interestingly 35% of Non drinkers/smokers are regular absentees.

**Feature selection:**

Dependent variable:

Absenteeism time in hours (target)

Independent variables:

1. Individual identification (ID)

2. Reason for absence

3. Month of absence

4. Day of the week

5. Seasons

6. Transportation expense

7. Distance from Residence to Work (kilometres)

8. Service time

9. Age

10. Work load Average/day

11. Hit target

12. Disciplinary failure

13. Education

14. Son (number of children)

15. Social drinker

16. Social smoker

17. Pet

18. Weight

19. Height

20. Body mass index

**Correlation Matrix:**

We can find the correlation between Absenteeism time in hours (target continuous variable) and independent continuous variables in the dataset using correlation matrix.

And if features are correlated with each other that could introduce bias into our models. Hence there should be no correlation between independent features and there should be high correlation between target and independent variables.

Observation:

None of the independent attributes are highly correlated with the target and with other each other.

**Chi-Square Test:**

This test is used to derive the statistical significance of relationship between the categorical variables in the dataset. It returns probability for the computed chi-square distribution with the degree of freedom.

Hypothesis of Chi-Square test:

1. Null Hypothesis: The null hypothesis of the Chi-Square test is that no relationship exists on the categorical variables in the population; they are independent.
2. Alternate Hypothesis: The alternate hypothesis of the Chi-Square test is that there exists relationship between the categorical variables in the population; they are not independent.

If p-value is less than 0.05 then we reject the null hypothesis. And if p-value is greater than 0.05 then we accept the null hypothesis.

Observation:

We can observe that p value of Seasons vs Month\_of\_absence is 0.000000e+00 which means that it is highly dependent on Month of absence.

**ANOVA test:**

We can find the correlation between Absenteeism time in hours (target continuous variable) and independent categorical variables present in training dataset using ANOVA test of independence. It is compares the mean between each groups in a categorical variable.

Hypothesis of ANOVA testing:

-> Null Hypothesis: Mean of all categories in a variable are same and numerical variable doesn't depend on it.

-> Alternate Hypothesis: Mean of at least one category in a variable is different and numerical variable depends on it.

If p-value is less than 0.05 then we reject the null hypothesis. And if p-value is greater than 0.05 then we accept the null hypothesis.

Observation:

All the attributes except Social smoker are significant to the target.

**Feature Scaling:**

The distributions of the variables present in the dataset are not gaussian and hence we have to normalize the data to bring them in a common range.

**3. Modelling:**

**3.1 Model Development:**

Once all the pre-processing has been done on the dataset, we will now be able to develop a model. The choice of machine learning algorithm depends upon the problem category. As per our problem statement we can develop following models:

• Linear regression

• Decision Tree

• Random forest

• Gradient Boosting

**Linear Regression:**

It is the simplest and powerful statistical model for prediction. It uses weights or coefficients of each independent variable in training data to develop model to be tested on test data.

Since, our training dataset has categorical and continuous variables, we have to encode values of categorical variables into numeric values and then pass as input.

**Decision tree:**

Decision tree is a supervised model for classification and Regression. It is used to create training model which is used to predict/classify values of target variable by learning decision rules based on historical data.

**Random Forest:**

Random Forest is an ensemble of decision trees where n number of random variables are used to construct n decision trees.

**Gradient Boosting:**

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.

**3.2 Model Evaluation:**

After having built a model, we have to measure how accurately the model predicts the response. RMSE (Root Mean Square Error)is one of the good measures and criterion for fit for prediction models.

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model’s predicted values. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors. The RMSE is directly interpretable in terms of measurement units, and so is a better measure of goodness of fit.

**3.4 Model Selection:**

After evaluating regression models, we can select the model which yields low RMSE value and high R-Squared value.

**4. Conclusion:**

Below are the train and test data results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | RMSE | | R-Squared | |
|  | Training dataset | Test dataset | Training dataset | Test dataset |
| Linear regression | 0.156 | 0.206 | 0.46 | 0.352 |
| Decision tree | 0.190 | 0.240 | 0.177 | 0.125 |
| Random Forest | 0.070 | 0.197 | 0.888 | 0.399 |
| Gradient Boosting | 0.128 | 0.637 | 0.826 | 0.312 |

We can conclude that although Random Forest Model yields maximum R^2 value, there is very much difference in the R2 values of training and test dataset, thus there is a chance that it might not generalize well. Hence, we can predict test dataset using GBM Model.

#### **Prediction of absent hours for the year 2011**:

#### 

Thus, from the data given it is clear that it is past data. Now, we are expected to make project losses every month in 2011 if same trend of absenteeism continues. Thus, assuming that the given data is from year 2010, we can frame the sample dataset for 2011 by increasing the age and service time of the employees and keeping the rest of the data constant assuming there will not be much change to it.

**Code in Python:**

In [1]:

#import required libraries

import os as os #to get access to input files

import pandas as pd #to build structured data format and data pre-processing

import numpy as np #for arrays,fourier transforms and linear algebric operations

import matplotlib.pyplot as plt #to create visualization of data

import seaborn as sbn #for visualization of data

from fancyimpute import KNN #for KNN imputation

from scipy.stats import chi2\_contingency #for Chi-Squared test

import statsmodels.api as sm #for statistical tests

from statsmodels.formula.api import ols #for ANOVA test

from sklearn.model\_selection import train\_test\_split #to split dataset into training and test

from sklearn.linear\_model import LinearRegression #to implement Linear Regression model

from sklearn.tree import DecisionTreeRegressor #to build Decision tree regression model

from sklearn.ensemble import RandomForestRegressor #to build Random Forest regression model

from sklearn.ensemble import GradientBoostingRegressor #to build GB model

from sklearn.metrics import mean\_squared\_error, r2\_score #import metrics to evaluate regression model

from sklearn.model\_selection import GridSearchCV #for hyperparameters tuning

from sklearn.model\_selection import RandomizedSearchCV #for hyperparameters tuning

import sys #for handling warnings

if not sys.warnoptions:

import warnings

warnings.simplefilter("ignore")

Using TensorFlow backend.

In [2]:

#Set working directory

os.chdir("C:/Users/Usha/Edwisor/Project - Employee Absenteeism")

#To check if the working directory is set right

os.getcwd()

Out[2]:

'C:\\Users\\Usha\\Edwisor\\Project - Employee Absenteeism'

In [3]:

#Load the company dataset

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

## Data Exploration[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Data-Exploration)

Data Exploration or preparation includes,

1. Identification of variables and their datatypes
2. Descriptive statistics
3. Conversion of data types into required ones
4. Univariate, Bivariate and Multivariate analysis
5. Missing Value Analysis
6. Outlier Analysis
7. Feature Engineering

#### Identification of variables and their datatypes[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Identification-of-variables-and-their-datatypes)

In [4]:

#fetch first five observations from the dataset

df.head()

Out[4]:

|  | **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** | **...** | **Disciplinary failure** | **Education** | **Son** | **Social drinker** | **Social smoker** | **Pet** | **Weight** | **Height** | **Body mass index** | **Absenteeism time in hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 11 | 26.0 | 7.0 | 3 | 1 | 289.0 | 36.0 | 13.0 | 33.0 | 239554.0 | ... | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 1.0 | 90.0 | 172.0 | 30.0 | 4.0 |
| **1** | 36 | 0.0 | 7.0 | 3 | 1 | 118.0 | 13.0 | 18.0 | 50.0 | 239554.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | 31.0 | 0.0 |
| **2** | 3 | 23.0 | 7.0 | 4 | 1 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | ... | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 89.0 | 170.0 | 31.0 | 2.0 |
| **3** | 7 | 7.0 | 7.0 | 5 | 1 | 279.0 | 5.0 | 14.0 | 39.0 | 239554.0 | ... | 0.0 | 1.0 | 2.0 | 1.0 | 1.0 | 0.0 | 68.0 | 168.0 | 24.0 | 4.0 |
| **4** | 11 | 23.0 | 7.0 | 5 | 1 | 289.0 | 36.0 | 13.0 | 33.0 | 239554.0 | ... | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 1.0 | 90.0 | 172.0 | 30.0 | 2.0 |

5 rows × 21 columns

First five rows are displayed from the dataset that contains 21 attributes.

In [5]:

#fetch last five observations from the dataset

df.tail()

Out[5]:

|  | **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** | **...** | **Disciplinary failure** | **Education** | **Son** | **Social drinker** | **Social smoker** | **Pet** | **Weight** | **Height** | **Body mass index** | **Absenteeism time in hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **735** | 11 | 14.0 | 7.0 | 3 | 1 | 289.0 | 36.0 | 13.0 | 33.0 | 264604.0 | ... | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 1.0 | 90.0 | 172.0 | 30.0 | 8.0 |
| **736** | 1 | 11.0 | 7.0 | 3 | 1 | 235.0 | 11.0 | 14.0 | 37.0 | 264604.0 | ... | 0.0 | 3.0 | 1.0 | 0.0 | 0.0 | 1.0 | 88.0 | 172.0 | 29.0 | 4.0 |
| **737** | 4 | 0.0 | 0.0 | 3 | 1 | 118.0 | 14.0 | 13.0 | 40.0 | 271219.0 | ... | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 8.0 | 98.0 | 170.0 | 34.0 | 0.0 |
| **738** | 8 | 0.0 | 0.0 | 4 | 2 | 231.0 | 35.0 | 14.0 | 39.0 | 271219.0 | ... | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 2.0 | 100.0 | 170.0 | 35.0 | 0.0 |
| **739** | 35 | 0.0 | 0.0 | 6 | 3 | 179.0 | 45.0 | 14.0 | 53.0 | 271219.0 | ... | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 77.0 | 175.0 | 25.0 | 0.0 |

5 rows × 21 columns

Last row says that there are 740 rows (range index being 0 to 739 rows) in the dataset.

In [6]:

#to get the dimensions of the loaded dataset

df.shape

Out[6]:

(740, 21)

Dataset comprises of 740 observations and 21 attributes, out of which one is target variable and rest are independent variables.

In [7]:

#to identify target and predictor variables and their datatypes

df.dtypes

Out[7]:

ID int64

Reason for absence float64

Month of absence float64

Day of the week int64

Seasons int64

Transportation expense float64

Distance from Residence to Work float64

Service time float64

Age float64

Work load Average/day float64

Hit target float64

Disciplinary failure float64

Education float64

Son float64

Social drinker float64

Social smoker float64

Pet float64

Weight float64

Height float64

Body mass index float64

Absenteeism time in hours float64

dtype: object

Observation:

Most of the attributes are in float datatype including the target attribute (Absenteeism time in hours). Hence, datatype conversion is required for few attributes according to Attribute Information.

#### Descriptive statistics[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Descriptive-statistics)

In [8]:

#to get a concise summary of the dataset

df.info()

<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

Observation:

There are null values present in most of the attributes in the dataset.

In [9]:

#to get the summary statistics of the dataset

df.describe()

Out[9]:

|  | **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** | **...** | **Disciplinary failure** | **Education** | **Son** | **Social drinker** | **Social smoker** | **Pet** | **Weight** | **Height** | **Body mass index** | **Absenteeism time in hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 740.000000 | 737.000000 | 739.000000 | 740.000000 | 740.000000 | 733.000000 | 737.000000 | 737.000000 | 737.000000 | 730.000000 | ... | 734.000000 | 730.000000 | 734.000000 | 737.000000 | 736.000000 | 738.000000 | 739.000000 | 726.000000 | 709.000000 | 718.000000 |
| **mean** | 18.017568 | 19.188602 | 6.319350 | 3.914865 | 2.544595 | 221.035471 | 29.667571 | 12.565807 | 36.449118 | 271188.860274 | ... | 0.053134 | 1.295890 | 1.017711 | 0.567164 | 0.073370 | 0.746612 | 79.063599 | 172.152893 | 26.684062 | 6.977716 |
| **std** | 11.021247 | 8.437493 | 3.435948 | 1.421675 | 1.111831 | 66.954179 | 14.848124 | 4.389813 | 6.480148 | 38981.880873 | ... | 0.224453 | 0.676965 | 1.094928 | 0.495805 | 0.260919 | 1.319726 | 12.868630 | 6.081065 | 4.292819 | 13.476962 |
| **min** | 1.000000 | 0.000000 | 0.000000 | 2.000000 | 1.000000 | 118.000000 | 5.000000 | 1.000000 | 27.000000 | 205917.000000 | ... | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 56.000000 | 163.000000 | 19.000000 | 0.000000 |
| **25%** | 9.000000 | 13.000000 | 3.000000 | 3.000000 | 2.000000 | 179.000000 | 16.000000 | 9.000000 | 31.000000 | 244387.000000 | ... | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 69.000000 | 169.000000 | 24.000000 | 2.000000 |
| **50%** | 18.000000 | 23.000000 | 6.000000 | 4.000000 | 3.000000 | 225.000000 | 26.000000 | 13.000000 | 37.000000 | 264249.000000 | ... | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 83.000000 | 170.000000 | 25.000000 | 3.000000 |
| **75%** | 28.000000 | 26.000000 | 9.000000 | 5.000000 | 4.000000 | 260.000000 | 50.000000 | 16.000000 | 40.000000 | 284853.000000 | ... | 0.000000 | 1.000000 | 2.000000 | 1.000000 | 0.000000 | 1.000000 | 89.000000 | 172.000000 | 31.000000 | 8.000000 |
| **max** | 36.000000 | 28.000000 | 12.000000 | 6.000000 | 4.000000 | 388.000000 | 52.000000 | 29.000000 | 58.000000 | 378884.000000 | ... | 1.000000 | 4.000000 | 4.000000 | 1.000000 | 1.000000 | 8.000000 | 108.000000 | 196.000000 | 38.000000 | 120.000000 |

8 rows × 21 columns

Initial observations:

1. The dataset contains 36 Individual identification (ID)s of the regular absentees.
2. Average value for Reason for absence is 23 (Medical consultation) and minimum value is 0 (no such reason).
3. Average value for Month of absence is 6 (June) and minimum value is 0 (no such month).
4. Average Day of the week for absence is 4 (Wednesday) and the average season of absence is 3 (Winter).
5. Minimum distance from residence to work is 5 kms.
6. Most of the employees are middle aged (35 to 40) and have completed their high school only.
7. Although there are employees who have joined recently but majority of the employees are experienced.
8. Maximum hours of absent is 120 hours, which is equivalent to 5 full days and 15 working days (assuming 8 hours of work time per day).

In [10]:

#to get the count of unique values in the dataset

df.nunique()

Out[10]:

ID 36

Reason for absence 28

Month of absence 13

Day of the week 5

Seasons 4

Transportation expense 24

Distance from Residence to Work 25

Service time 18

Age 22

Work load Average/day 38

Hit target 13

Disciplinary failure 2

Education 4

Son 5

Social drinker 2

Social smoker 2

Pet 6

Weight 26

Height 14

Body mass index 17

Absenteeism time in hours 19

dtype: int64

In [11]:

#to get the unique values of Service\_time from the dataset

df['Service time'].unique()

Out[11]:

array([13., 18., 14., 3., 11., 16., 4., 6., 12., 7., 10., 9., 17.,

nan, 29., 8., 1., 15., 24.])

Minimum and maximum years of experience of employees are 1 and 29 respectively.

#### Conversion of datatypes into required ones[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Conversion-of-datatypes-into-required-ones)

In [12]:

#Rename attributes of the dataset for convenience

df.rename(columns = {'Work load Average/day ':'Work load Average/day'}, inplace = True)

df.columns = df.columns.str.replace(' ','\_')

In [13]:

#Datatype conversion

df['Reason\_for\_absence'] = df['Reason\_for\_absence'].astype("category") #Reason for absence can only be one of the 21 categories stratified as per International Code of Diseases (ICD)

df['Month\_of\_absence'] = df['Month\_of\_absence'].astype("category") #Month of absence can only be 1-12 months.

df['Day\_of\_the\_week'] = df['Day\_of\_the\_week'].astype("category") #Day of the week can only be 1-7 days.

df['Seasons'] = df['Seasons'].astype("category") #Seasons can only be 1-4 seasons.

df['Disciplinary\_failure'] = df['Disciplinary\_failure'].astype("category") # Disciplinary failure can only be either (yes=1/ no=0)

df['Education'] = df['Education'].astype("category") #Education can be one of the four(high school (1), graduate (2), postgraduate (3), master and doctor (4))

df['Social\_drinker'] = df['Social\_drinker'].astype("category") #Social drinker can only be either (yes=1/ no=0)

df['Social\_smoker'] = df['Social\_smoker'].astype("category") #Social smoker can only be either (yes=1/ no=0)

df['Pet'] = df['Pet'].astype("category") #Pet denotes the number of pets which can be 0 - <10 (maximum value for example)

df['Son'] = df['Son'].astype("category") #Son denotes the number of children which can be 0 - <10 (maximum value for example)

In [14]:

#Check datatypes of attributes after datatype conversion

df.dtypes

Out[14]:

ID int64

Reason\_for\_absence category

Month\_of\_absence category

Day\_of\_the\_week category

Seasons category

Transportation\_expense float64

Distance\_from\_Residence\_to\_Work float64

Service\_time float64

Age float64

Work\_load\_Average/day float64

Hit\_target float64

Disciplinary\_failure category

Education category

Son category

Social\_drinker category

Social\_smoker category

Pet category

Weight float64

Height float64

Body\_mass\_index float64

Absenteeism\_time\_in\_hours float64

dtype: object

In [15]:

#Get numeric and categorical attributes from dataset

num\_cols = df.\_get\_numeric\_data().columns

cat\_cols = list(set(df.columns) - set(num\_cols))

#num\_cols = num\_cols.delete(0)

print("Numeric cols are:")

print(num\_cols)

print("Categorical cols are:")

print(cat\_cols)

Numeric cols are:

Index(['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'],

dtype='object')

Categorical cols are:

['Month\_of\_absence', 'Disciplinary\_failure', 'Reason\_for\_absence', 'Social\_drinker', 'Social\_smoker', 'Seasons', 'Pet', 'Day\_of\_the\_week', 'Education', 'Son']

In [16]:

df\_data = df.copy()

##### Extreme value Analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Extreme-value-Analysis)

Based on the basic understanding of the data from Descriptive statistics, it is obvious that there are outliers present in the dataset which are incorrectly entered or measured data. Hence, we can remove those values.

In [17]:

#fetch rows where absenteeism is zero

df[df['Absenteeism\_time\_in\_hours']==0]

Out[17]:

|  | **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** | **...** | **Disciplinary\_failure** | **Education** | **Son** | **Social\_drinker** | **Social\_smoker** | **Pet** | **Weight** | **Height** | **Body\_mass\_index** | **Absenteeism\_time\_in\_hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 36 | 0.0 | 7.0 | 3 | 1 | 118.0 | 13.0 | 18.0 | 50.0 | 239554.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | 31.0 | 0.0 |
| **51** | 29 | 0.0 | 9.0 | 2 | 4 | 225.0 | 26.0 | 9.0 | 28.0 | 241476.0 | ... | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 2.0 | 69.0 | 169.0 | 24.0 | 0.0 |
| **54** | 11 | 0.0 | 9.0 | 3 | 4 | 289.0 | 36.0 | 13.0 | 33.0 | 241476.0 | ... | 1.0 | 1.0 | 2.0 | 1.0 | 0.0 | 1.0 | 90.0 | 172.0 | 30.0 | 0.0 |
| **58** | 13 | 0.0 | 9.0 | 4 | 4 | 369.0 | 17.0 | 12.0 | 31.0 | 241476.0 | ... | 1.0 | 1.0 | 3.0 | 1.0 | 0.0 | 0.0 | 70.0 | 169.0 | NaN | 0.0 |
| **134** | 34 | 27.0 | 1.0 | 2 | 2 | 118.0 | 10.0 | 10.0 | 37.0 | 308593.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 83.0 | 172.0 | 28.0 | 0.0 |
| **203** | 2 | 0.0 | 4.0 | 2 | 3 | 235.0 | 29.0 | 12.0 | 48.0 | 326452.0 | ... | 1.0 | 1.0 | 1.0 | 0.0 | 1.0 | 5.0 | 88.0 | 163.0 | 33.0 | 0.0 |
| **213** | 7 | 0.0 | 5.0 | 4 | 3 | 279.0 | 5.0 | 14.0 | 39.0 | 378884.0 | ... | 1.0 | 1.0 | 2.0 | 1.0 | 1.0 | 0.0 | 68.0 | 168.0 | 24.0 | 0.0 |
| **214** | 18 | 0.0 | 5.0 | 4 | 3 | 330.0 | 16.0 | 4.0 | 28.0 | 378884.0 | ... | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 84.0 | 182.0 | 25.0 | 0.0 |
| **216** | 31 | 0.0 | 5.0 | 4 | 3 | 388.0 | 15.0 | 9.0 | 50.0 | 378884.0 | ... | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 76.0 | 178.0 | 24.0 | 0.0 |
| **251** | 20 | 0.0 | 7.0 | 5 | 1 | 260.0 | 50.0 | 11.0 | 36.0 | 275312.0 | ... | 1.0 | 1.0 | 4.0 | 1.0 | 0.0 | 0.0 | 65.0 | 168.0 | 23.0 | 0.0 |
| **273** | 5 | 0.0 | 9.0 | 5 | 1 | 235.0 | 20.0 | 13.0 | 43.0 | 294217.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 106.0 | NaN | NaN | 0.0 |
| **276** | 8 | 0.0 | 9.0 | 3 | 1 | 231.0 | 35.0 | 14.0 | 39.0 | 294217.0 | ... | 1.0 | 1.0 | 2.0 | 1.0 | 0.0 | 2.0 | 100.0 | 170.0 | 35.0 | 0.0 |
| **277** | 19 | 0.0 | 9.0 | 3 | 1 | 291.0 | 50.0 | 12.0 | 32.0 | 294217.0 | ... | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 65.0 | 169.0 | 23.0 | 0.0 |
| **285** | 5 | 0.0 | 9.0 | 5 | 4 | 235.0 | 20.0 | 13.0 | 43.0 | 294217.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 106.0 | 167.0 | 38.0 | 0.0 |
| **293** | 36 | 0.0 | 10.0 | 6 | 4 | 118.0 | 13.0 | 18.0 | 50.0 | 265017.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | 31.0 | 0.0 |
| **294** | 33 | 0.0 | 10.0 | 6 | 4 | 248.0 | 25.0 | 14.0 | 47.0 | 265017.0 | ... | 1.0 | 1.0 | 2.0 | 0.0 | 0.0 | 1.0 | 86.0 | 165.0 | 32.0 | 0.0 |
| **300** | 5 | 0.0 | 10.0 | 4 | 4 | 235.0 | 20.0 | 13.0 | 43.0 | 265017.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 106.0 | 167.0 | 38.0 | 0.0 |
| **303** | 5 | 0.0 | 10.0 | 6 | 4 | 235.0 | 20.0 | 13.0 | 43.0 | 265017.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 106.0 | 167.0 | 38.0 | 0.0 |
| **311** | 20 | 0.0 | 10.0 | 3 | 4 | 260.0 | 50.0 | 11.0 | 36.0 | 265017.0 | ... | 1.0 | 1.0 | 4.0 | 1.0 | 0.0 | 0.0 | 65.0 | 168.0 | 23.0 | 0.0 |
| **312** | 15 | 0.0 | 10.0 | 3 | 4 | 291.0 | 31.0 | 12.0 | 40.0 | 265017.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 1.0 | 73.0 | 171.0 | 25.0 | 0.0 |
| **325** | 18 | 0.0 | 11.0 | 3 | 4 | 330.0 | 16.0 | 4.0 | 28.0 | 284031.0 | ... | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 84.0 | 182.0 | 25.0 | 0.0 |
| **336** | 23 | 0.0 | 11.0 | 6 | 4 | 378.0 | 49.0 | 11.0 | 36.0 | 284031.0 | ... | 1.0 | 1.0 | 2.0 | 0.0 | 1.0 | 4.0 | 65.0 | 174.0 | 21.0 | 0.0 |
| **400** | 13 | 0.0 | 3.0 | 4 | 2 | 369.0 | 17.0 | 12.0 | 31.0 | 244387.0 | ... | 1.0 | 1.0 | 3.0 | 1.0 | 0.0 | 0.0 | 70.0 | 169.0 | 25.0 | 0.0 |
| **406** | 24 | 0.0 | 3.0 | 5 | 3 | 246.0 | 25.0 | 16.0 | 41.0 | 244387.0 | ... | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 67.0 | 170.0 | 23.0 | 0.0 |
| **407** | 36 | 0.0 | 3.0 | 5 | 3 | 118.0 | 13.0 | 18.0 | 50.0 | 244387.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | 31.0 | 0.0 |
| **446** | 3 | 0.0 | 6.0 | 6 | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 253957.0 | ... | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 89.0 | 170.0 | 31.0 | 0.0 |
| **530** | 28 | 0.0 | 10.0 | 2 | 4 | 225.0 | 26.0 | 9.0 | 28.0 | 284853.0 | ... | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 2.0 | 69.0 | 169.0 | 24.0 | 0.0 |
| **548** | 15 | 0.0 | 11.0 | 3 | 4 | 291.0 | 31.0 | 12.0 | 40.0 | 268519.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 1.0 | 73.0 | 171.0 | 25.0 | 0.0 |
| **549** | 11 | 0.0 | 11.0 | 4 | 4 | 289.0 | 36.0 | 13.0 | 33.0 | 268519.0 | ... | 1.0 | 1.0 | 2.0 | 1.0 | 0.0 | 1.0 | 90.0 | 172.0 | 30.0 | 0.0 |
| **688** | 36 | 0.0 | 5.0 | 3 | 3 | 118.0 | 13.0 | 18.0 | 50.0 | 237656.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | 31.0 | 0.0 |
| **690** | 23 | 0.0 | 5.0 | 4 | 3 | 378.0 | 49.0 | 11.0 | 36.0 | 237656.0 | ... | 1.0 | 1.0 | 2.0 | 0.0 | 1.0 | 4.0 | 65.0 | 174.0 | 21.0 | 0.0 |
| **714** | 2 | 0.0 | 6.0 | 2 | 3 | 235.0 | 29.0 | 12.0 | 48.0 | 275089.0 | ... | 1.0 | 1.0 | 1.0 | 0.0 | 1.0 | 5.0 | 88.0 | 163.0 | 33.0 | 0.0 |
| **715** | 21 | 0.0 | 6.0 | 2 | 3 | 268.0 | 11.0 | 8.0 | 33.0 | 275089.0 | ... | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 79.0 | 178.0 | 25.0 | 0.0 |
| **737** | 4 | 0.0 | 0.0 | 3 | 1 | 118.0 | 14.0 | 13.0 | 40.0 | 271219.0 | ... | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 8.0 | 98.0 | 170.0 | 34.0 | 0.0 |
| **738** | 8 | 0.0 | 0.0 | 4 | 2 | 231.0 | 35.0 | 14.0 | 39.0 | 271219.0 | ... | 0.0 | 1.0 | 2.0 | 1.0 | 0.0 | 2.0 | 100.0 | 170.0 | 35.0 | 0.0 |
| **739** | 35 | 0.0 | 0.0 | 6 | 3 | 179.0 | 45.0 | 14.0 | 53.0 | 271219.0 | ... | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 77.0 | 175.0 | 25.0 | 0.0 |

36 rows × 21 columns

It is obvious that Absenteeism\_time\_in\_hours can't be zero. we can remove the observations as they are incorrectly entered or recorded.

In [18]:

#Drop rows where Absenteeism\_time\_in\_hours is zero

df = df.drop(df[df['Absenteeism\_time\_in\_hours']==0].index,axis=0)

In [19]:

#Fetch rows where Month\_of\_absence is zero

df[df['Month\_of\_absence']==0]

Out[19]:

|  | **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** | **...** | **Disciplinary\_failure** | **Education** | **Son** | **Social\_drinker** | **Social\_smoker** | **Pet** | **Weight** | **Height** | **Body\_mass\_index** | **Absenteeism\_time\_in\_hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

0 rows × 21 columns

There are no observations where Month\_of\_absence is zero.

In [20]:

#fetch rows where Reason\_for\_absence is zero

df[df['Reason\_for\_absence']==0]

Out[20]:

|  | **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** | **...** | **Disciplinary\_failure** | **Education** | **Son** | **Social\_drinker** | **Social\_smoker** | **Pet** | **Weight** | **Height** | **Body\_mass\_index** | **Absenteeism\_time\_in\_hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **50** | 20 | 0.0 | 9.0 | 2 | 4 | NaN | 50.0 | 11.0 | 36.0 | 241476.0 | ... | NaN | 1.0 | 4.0 | 1.0 | 0.0 | 0.0 | 65.0 | 168.0 | 23.0 | NaN |
| **55** | 36 | 0.0 | 9.0 | 3 | 4 | 118.0 | 13.0 | 18.0 | 50.0 | NaN | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | 31.0 | NaN |
| **64** | 36 | 0.0 | 10.0 | 4 | 4 | 118.0 | 13.0 | 18.0 | 50.0 | 253465.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 98.0 | 178.0 | NaN | NaN |
| **215** | 23 | 0.0 | 5.0 | 4 | 3 | 378.0 | 49.0 | 11.0 | 36.0 | 378884.0 | ... | 1.0 | 1.0 | 2.0 | 0.0 | 1.0 | 4.0 | 65.0 | 174.0 | 21.0 | NaN |
| **313** | 30 | 0.0 | 10.0 | 3 | 4 | 157.0 | 27.0 | 6.0 | 29.0 | 265017.0 | ... | 1.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 | 75.0 | 185.0 | 22.0 | NaN |
| **337** | 7 | 0.0 | 11.0 | 3 | 4 | 279.0 | 5.0 | 14.0 | 39.0 | 284031.0 | ... | 1.0 | 1.0 | 2.0 | 1.0 | 1.0 | 0.0 | 68.0 | 168.0 | 24.0 | 120.0 |
| **405** | 1 | 0.0 | 3.0 | 5 | 3 | 235.0 | 11.0 | 14.0 | 37.0 | 244387.0 | ... | 1.0 | 3.0 | 1.0 | 0.0 | 0.0 | 1.0 | 88.0 | 172.0 | 29.0 | NaN |
| **551** | 5 | 0.0 | 11.0 | 5 | 4 | 235.0 | 20.0 | 13.0 | 43.0 | 268519.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 106.0 | 167.0 | 38.0 | NaN |

8 rows × 21 columns

Reason for absence can't be zero and hence zero values can be replaced by category - 26 (unjustified absence).

In [21]:

#Replace zero values of Reason\_for\_absence by category - 26

df.loc[df['Reason\_for\_absence']==0,'Reason\_for\_absence'] = 26

#### Univariate, Bivariate and Multivariate analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Univariate,-Bivariate-and-Multivariate-analysis)

Univariate analysis provides summary statistics for each field in the raw dataset (or) summary only on one variable. Bivariate analysis is performed to find the relationship between each variable in the dataset and the target variable of interest (or) using two variables and finding relationship between them. Multivariate analysis is used to visualize more than two variables at once.

###### Univariate analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Univariate-analysis)

In [22]:

#Univariate analysis - Transportation\_expense

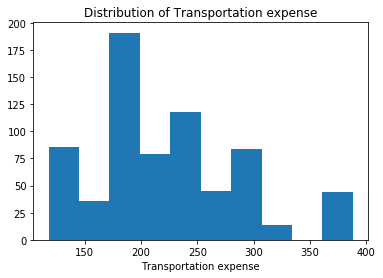
plt.hist(df['Transportation\_expense'])

plt.title('Distribution of Transportation expense')

plt.xlabel('Transportation expense')

Out[22]:

Text(0.5, 0, 'Transportation expense')

****

Most of the employees incur maximum transportation expense in the range of 175 - 200, which is nearly half of the maximum fare.

In [23]:

#Univariate analysis - Distance\_from\_Residence\_to\_Work

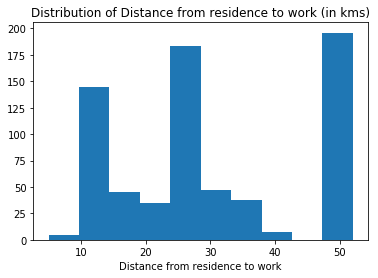
plt.hist(df['Distance\_from\_Residence\_to\_Work'])

plt.title('Distribution of Distance from residence to work (in kms)')

plt.xlabel('Distance from residence to work')

Out[23]:

Text(0.5, 0, 'Distance from residence to work')

****

Most of the employees travel more than 25 kms from their residence to reach their work place.

In [24]:

#Univariate analysis - Reason\_for\_absence

fig, ax= plt.subplots(figsize =(12,8))

ax = sbn.countplot(y = df['Reason\_for\_absence'])

plt.title('Distribution of Reason for absence')

plt.ylabel('Reason for absence')

total = len(df)

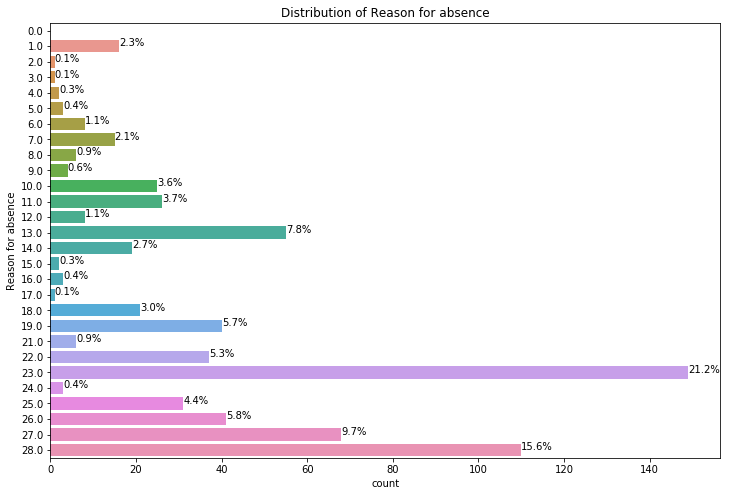
for p in ax.patches:

percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)

x = p.get\_x() + p.get\_width() + 0.02

y = p.get\_y() + p.get\_height()/2

ax.annotate(percentage, (x, y))

****

The top four of them cover 50% of the reasons for absence

* Medical consultation
* Dental consultation
* Physiotherapy
* Diseases of the musculoskeletal system and connective tissue

The unjusitified absence amounts to 4.5% of the total.

In [25]:

#Univariate analysis - Seasons

ax = sbn.countplot(y = df['Seasons'])

plt.title('Distribution of Seasons')

plt.ylabel('Seasons')

total = len(df)

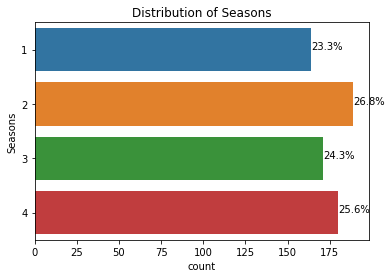
for p in ax.patches:

percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)

x = p.get\_x() + p.get\_width() + 0.02

y = p.get\_y() + p.get\_height()/2

ax.annotate(percentage, (x, y))

****

Highest number of absentees are recored in the Season 4 (Winter) followed by Season 2 (Autumn).

In [26]:

#Univariate analysis - Month\_of\_absence

ax = sbn.countplot(y = df['Month\_of\_absence'])

plt.title('Distribution of Month of absence')

plt.ylabel('Month of absence')

total = len(df)

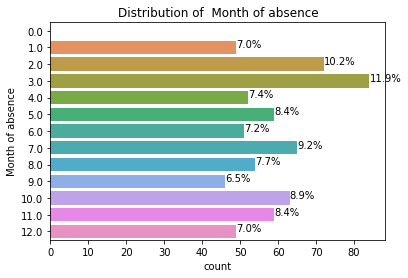
for p in ax.patches:

percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)

x = p.get\_x() + p.get\_width() + 0.02

y = p.get\_y() + p.get\_height()/2

ax.annotate(percentage, (x, y))

****

Highest percentage of absentees are recorded in the month of March followed by February and October, which comes under Winter and Autumn seasons respectively.

In [27]:

#Univariate analysis - Day\_of\_the\_week

ax = sbn.countplot(y = df['Day\_of\_the\_week'])

plt.title('Distribution of Day of the week')

plt.ylabel('Day of the week')

total = len(df)

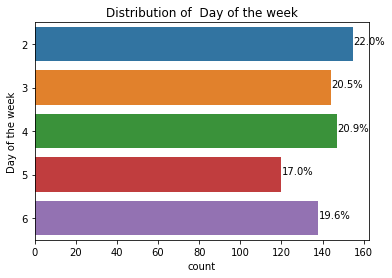
for p in ax.patches:

percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)

x = p.get\_x() + p.get\_width() + 0.02

y = p.get\_y() + p.get\_height()/2

ax.annotate(percentage, (x, y))

****

Highest percentage of absentees are recorded on start of the week, Mondays followed by Wednesdays.

In [28]:

#Univariate analysis - Social\_drinker

ax = df['Social\_drinker'].value\_counts().head(10).plot.bar(title='Distribution of Social drinking status of Employees')

plt.ylabel('Count')

plt.xlabel('Social drinking with 1 = Yes and 0 = No')

totals = []

total = len(df)

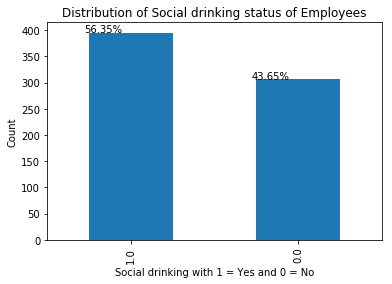
for i in ax.patches:

totals.append(i.get\_height())

total = sum(totals)

for i in ax.patches:

ax.text(i.get\_x()-.03, i.get\_height()+.5, str(round((i.get\_height()/total)\*100, 2))+'%')

****

In [29]:

#Univariate analysis - Social\_smoker

ax = df['Social\_smoker'].value\_counts().head(10).plot.bar(title='Distribution of Social smoking status of Employees')

plt.ylabel('Count')

plt.xlabel('Social smoking with 1 = Yes and 0 = No')

totals = []

total = len(df)

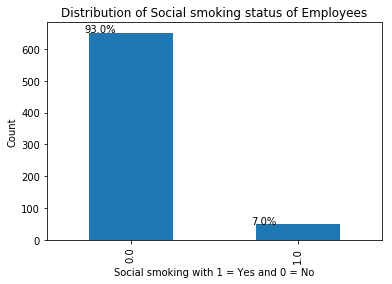
for i in ax.patches:

totals.append(i.get\_height())

total = sum(totals)

for i in ax.patches:

ax.text(i.get\_x()-.03, i.get\_height()+.5, str(round((i.get\_height()/total)\*100, 2))+'%')

****

Thus, most of the employees have the habit of social drinking but very few employees have the habit of social smoking.

In [30]:

#Univariate analysis - Number of children

ax = sbn.countplot(y = df['Son'])

plt.title('Distribution of number of children')

plt.ylabel('Number of children')

total = len(df)

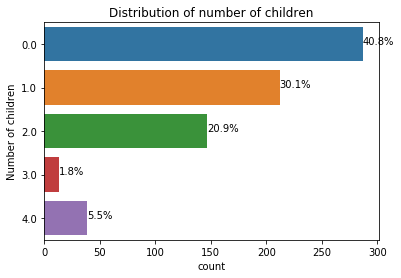
for p in ax.patches:

percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)

x = p.get\_x() + p.get\_width() + 0.02

y = p.get\_y() + p.get\_height()/2

ax.annotate(percentage, (x, y))

****

Most of the employees are married with 0 or 1 child. Thus, they have dependents/children to take care of and their presence might be required in case if their child/dependent fall sick.

###### Bivariate analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Bivariate-analysis)

In [31]:

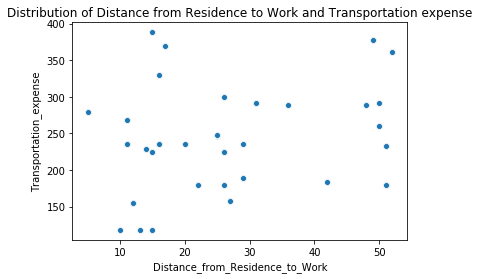
#Bivariate analysis - Distance\_from\_Residence\_to\_Work and Transportation\_expense

sbn.scatterplot(x='Distance\_from\_Residence\_to\_Work',y='Transportation\_expense',data=df)

plt.title('Distribution of Distance from Residence to Work and Transportation expense')

Out[31]:

Text(0.5, 1.0, 'Distribution of Distance from Residence to Work and Transportation expense')

****

This plot shows us that distance from residence to work and transportation expense are weakly correlated and there is no information on the mode of transport, peak hours, traffic factors; But,the general hypothesis is that transportation expense increases with distance from residence to work.

In [32]:

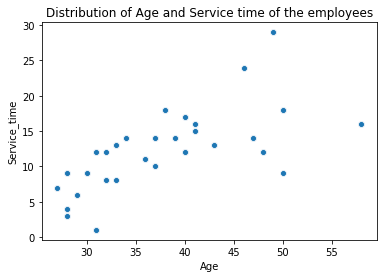
#Bivariate analysis - Age and Service\_time

sbn.scatterplot(x='Age',y='Service\_time',data=df)

plt.title('Distribution of Age and Service time of the employees')

Out[32]:

Text(0.5, 1.0, 'Distribution of Age and Service time of the employees')

****

This plot clearly tells us that Age and Service time are positively correlated and it is obvious that service time of a person increases with his age.

In [33]:

#Bivariate analyis - Age and Work\_load\_Average/day

age\_work = df.groupby('Age', as\_index=False)[['Work\_load\_Average/day']].mean()

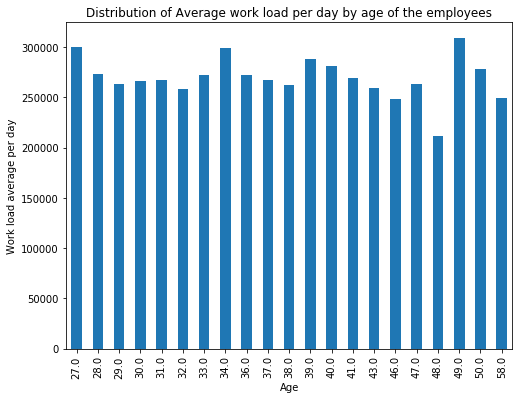
ax = age\_work.plot(kind='bar', x='Age', figsize=(8,6), legend=False)

ax.set\_ylabel('Work load average per day')

ax.set\_title('Distribution of Average work load per day by age of the employees')

Out[33]:

Text(0.5, 1.0, 'Distribution of Average work load per day by age of the employees')

****

The work load seems to be same irrespective of the age.

In [34]:

#Bivariate analysis - Age and Distance\_from\_Residence\_to\_Work

age\_distance = df.groupby('Age')[['Distance\_from\_Residence\_to\_Work']].mean()

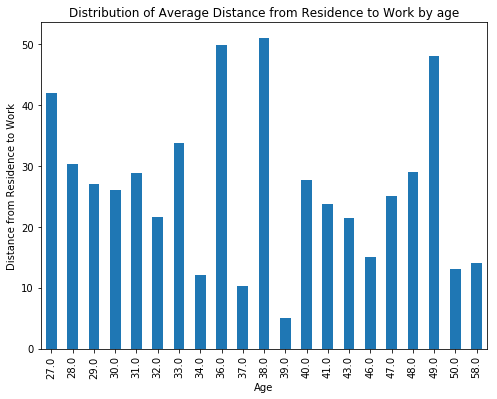
ax = age\_distance.plot(kind='bar', figsize=(8,6), legend=False)

ax.set\_ylabel('Distance from Residence to Work')

ax.set\_title('Distribution of Average Distance from Residence to Work by age')

Out[34]:

Text(0.5, 1.0, 'Distribution of Average Distance from Residence to Work by age')

****

Another hypothesis is that aged employees might stay closer to the office. But, values after age 33 are not significant to compare as hypothesis fails here after age 33.

In [35]:

#Bivariate analysis - Age and Hit\_target

age\_distance = df.groupby('Age')[['Hit\_target']].mean()

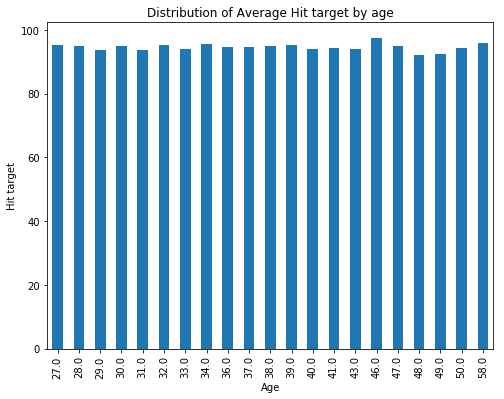
ax = age\_distance.plot(kind='bar', figsize=(8,6), legend=False)

ax.set\_ylabel('Hit target')

ax.set\_title('Distribution of Average Hit target by age')

Out[35]:

Text(0.5, 1.0, 'Distribution of Average Hit target by age')

****

Hit target seems to be same irrespective of the age of the employees.

In [36]:

#Bivariate analysis - Social drinking and Hit\_target

drink\_target = df.groupby('Social\_drinker')[['Hit\_target']].mean()

ax = drink\_target.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(drink\_target.values):

ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

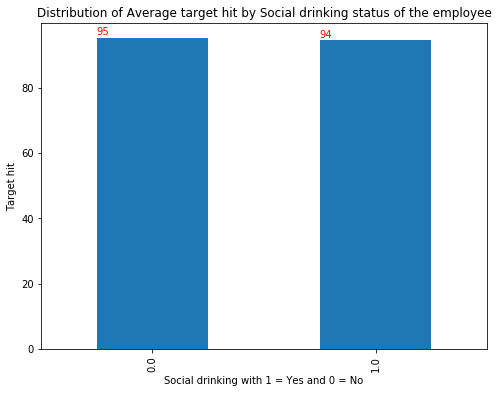
ax.set\_xlabel('Social drinking with 1 = Yes and 0 = No')

ax.set\_ylabel('Target hit')

ax.set\_title('Distribution of Average target hit by Social drinking status of the employee')

Out[36]:

Text(0.5, 1.0, 'Distribution of Average target hit by Social drinking status of the employee')

****

It is interesting to observe that employees with social drinking habit have average target hit value nearly equal to non drinkers. Thus, their drinking status doesn't have much impact on their target hit.

In [37]:

#Bivariate analysis - Social drinking and Disciplinary\_failure

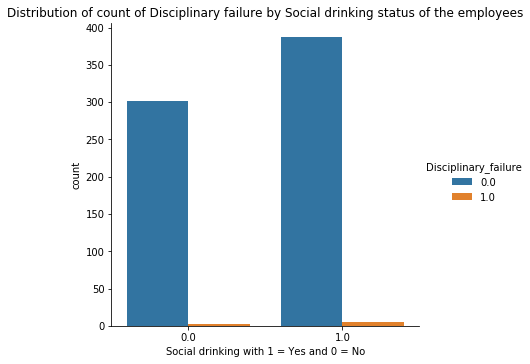
sbn.catplot(x="Social\_drinker", hue="Disciplinary\_failure",data=df, kind="count")

plt.title('Distribution of count of Disciplinary failure by Social drinking status of the employees')

plt.xlabel('Social drinking with 1 = Yes and 0 = No')

Out[37]:

Text(0.5, 21.70625000000002, 'Social drinking with 1 = Yes and 0 = No')

****

Thus, it is clear that social drinking doesn't have any effect on disciplinary failure in employees.

In [38]:

#Bivariate analysis - Work\_load\_Average/day and Month\_of\_absence

work\_month = df.groupby('Month\_of\_absence')[['Work\_load\_Average/day']].mean()

ax = work\_month.plot(kind='bar', figsize=(10,6), legend=False)

#for i, v in enumerate(work\_month.values):

#ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

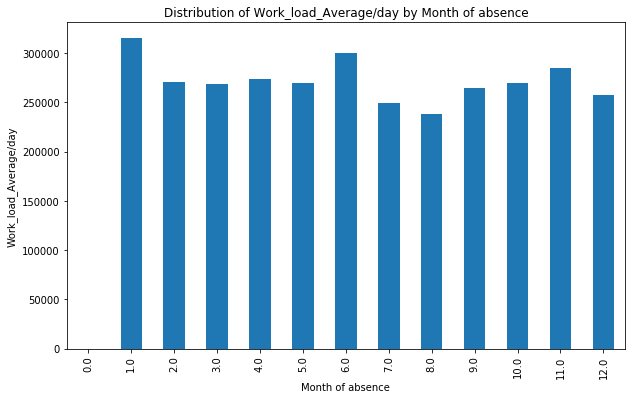
ax.set\_xlabel('Month of absence')

ax.set\_ylabel('Work\_load\_Average/day')

ax.set\_title('Distribution of Work\_load\_Average/day by Month of absence')

Out[38]:

Text(0.5, 1.0, 'Distribution of Work\_load\_Average/day by Month of absence')

****

Work load average/day seems to be higher in the month of January followed by June. It is almost in the same range for rest of the months

In [39]:

#Bivariate analysis - Work\_load\_Average/day and Hit\_target

work\_target = df.groupby('Work\_load\_Average/day')[['Hit\_target']].mean()

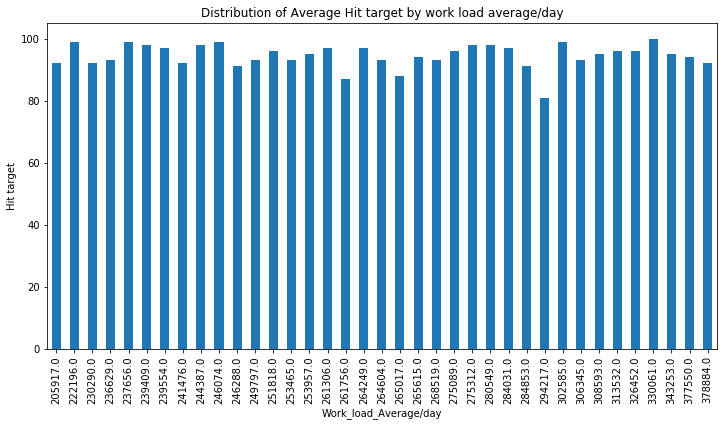
ax = work\_target.plot(kind='bar', figsize=(12,6), legend=False)

ax.set\_ylabel('Hit target')

ax.set\_title('Distribution of Average Hit target by work load average/day')

Out[39]:

Text(0.5, 1.0, 'Distribution of Average Hit target by work load average/day')

****

Thus, average target hit is same for almost all the work load average/day.

###### Multivariate analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Multivariate-analysis)

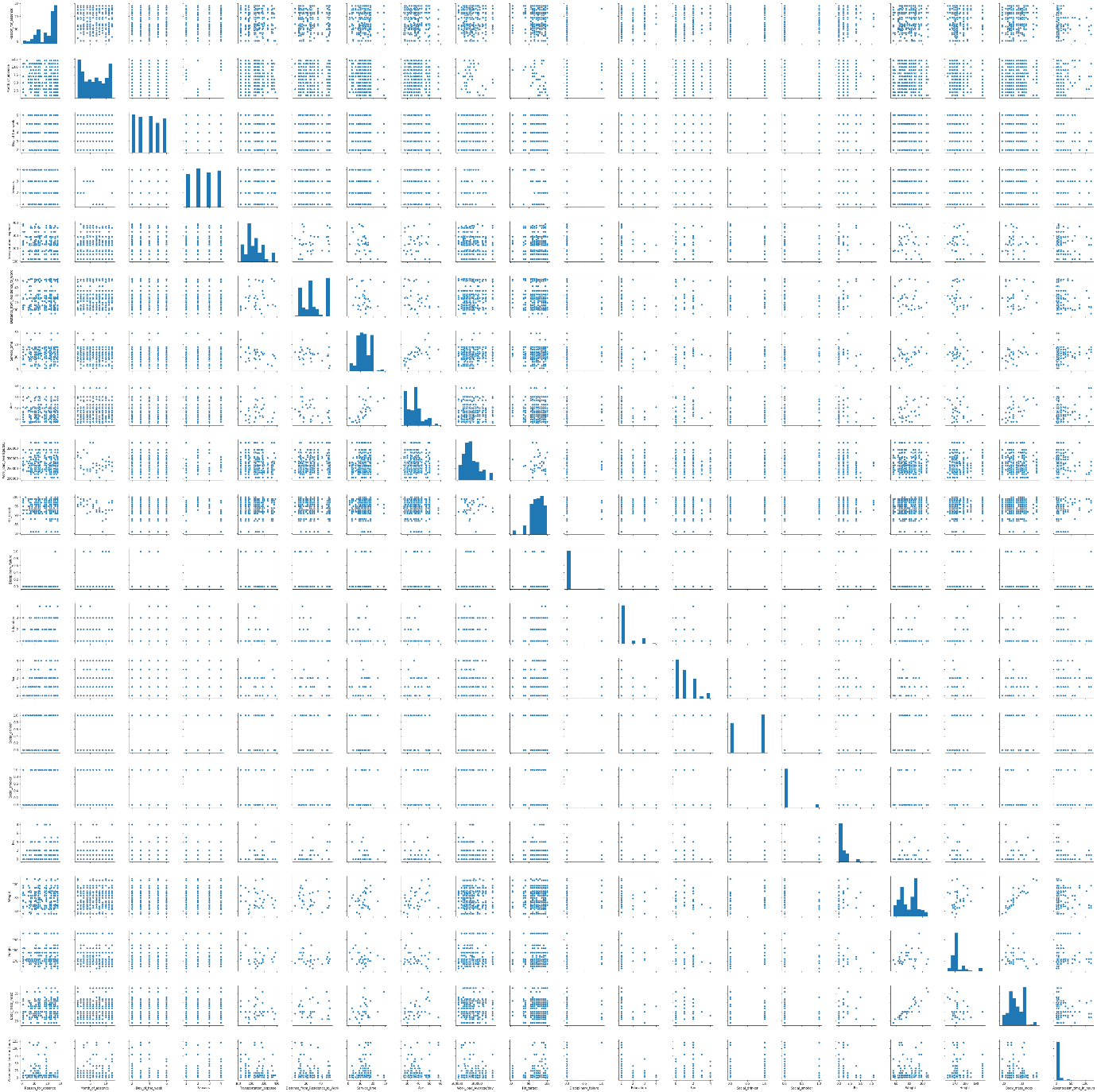
In [40]:

#Multivariate analysis

sbn.pairplot(df.drop('ID', axis=1))

Out[40]:

<seaborn.axisgrid.PairGrid at 0x235c6db10f0>

****

There seems to be not much relationship between attributes except for Age and service time, weight and BMI (Body Mass Index).

#### Missing value Analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Missing-value-Analysis)

In [41]:

#to get the sum of missing values in the dataset

df.isnull().sum()

Out[41]:

ID 0

Reason\_for\_absence 3

Month\_of\_absence 1

Day\_of\_the\_week 0

Seasons 0

Transportation\_expense 7

Distance\_from\_Residence\_to\_Work 3

Service\_time 3

Age 3

Work\_load\_Average/day 10

Hit\_target 6

Disciplinary\_failure 6

Education 10

Son 6

Social\_drinker 3

Social\_smoker 4

Pet 2

Weight 1

Height 13

Body\_mass\_index 29

Absenteeism\_time\_in\_hours 22

dtype: int64

In [42]:

#create dataframe to analyze missing values in training dataset

miss\_val = pd.DataFrame(df.isnull().sum())

miss\_val = miss\_val.reset\_index()

miss\_val = miss\_val.rename(columns = {"index":"Attributes",0:"Missing values"})

miss\_val['Missing Percentage'] = (miss\_val['Missing values']/len(df))\*100

miss\_val = miss\_val.sort\_values("Missing Percentage",ascending=False).reset\_index(drop=True)

miss\_val

Out[42]:

|  | **Attributes** | **Missing values** | **Missing Percentage** |
| --- | --- | --- | --- |
| **0** | Body\_mass\_index | 29 | 4.119318 |
| **1** | Absenteeism\_time\_in\_hours | 22 | 3.125000 |
| **2** | Height | 13 | 1.846591 |
| **3** | Work\_load\_Average/day | 10 | 1.420455 |
| **4** | Education | 10 | 1.420455 |
| **5** | Transportation\_expense | 7 | 0.994318 |
| **6** | Son | 6 | 0.852273 |
| **7** | Disciplinary\_failure | 6 | 0.852273 |
| **8** | Hit\_target | 6 | 0.852273 |
| **9** | Social\_smoker | 4 | 0.568182 |
| **10** | Age | 3 | 0.426136 |
| **11** | Reason\_for\_absence | 3 | 0.426136 |
| **12** | Service\_time | 3 | 0.426136 |
| **13** | Distance\_from\_Residence\_to\_Work | 3 | 0.426136 |
| **14** | Social\_drinker | 3 | 0.426136 |
| **15** | Pet | 2 | 0.284091 |
| **16** | Weight | 1 | 0.142045 |
| **17** | Month\_of\_absence | 1 | 0.142045 |
| **18** | Seasons | 0 | 0.000000 |
| **19** | Day\_of\_the\_week | 0 | 0.000000 |
| **20** | ID | 0 | 0.000000 |

Since the percentage of missing values of the attributes in the dataset is less, we can impute the missing values statistically.

Imputation of missing values in the dataset:

Imputation is a method to fill in the missing values with estimated ones. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable.

Since, the attributes are numerical and categorical, we can opt for Mean/Median/Mode and KNN methods for imputation.

To find the apt method out of all,

1. Remove any random value from the column missing and replace with NA.
2. Impute the null value with methods one by one.
3. Select the method with output nearly equal to the original value

In [43]:

#Imputing missing values - continuous attributes

#Choosing a random value from Weight attribute to replace it as NA

act\_val = df['Weight'].iloc[20]

#Replace actual value with NaN

df['Weight'].iloc[20] = np.nan

#Mean imputation

mean\_val = np.nanmean(df['Weight'])

df['Weight'].iloc[20] = np.nan

#Median imputation

median\_val = np.nanmedian(df['Weight'])

df['Weight'].iloc[20] = np.nan

#KNN imputation using k = 3, 5, 7 values

df\_knn3 = pd.DataFrame(KNN(k=3).fit\_transform(df),columns = df.columns,index=df.index)

knn3\_val = df\_knn3['Weight'].iloc[20]

df\_knn5 = pd.DataFrame(KNN(k=5).fit\_transform(df),columns = df.columns,index=df.index)

knn5\_val = df\_knn5['Weight'].iloc[20]

df\_knn7 = pd.DataFrame(KNN(k=7).fit\_transform(df),columns = df.columns,index=df.index)

knn7\_val = df\_knn7['Weight'].iloc[20]

#Substitute with actual value

df['Weight'].iloc[20] = act\_val

#Imputing missing values - categorical attributes

#Choosing a random value from Education attribute to replace it as NA

act\_cat\_val = df['Education'].iloc[83]

#Replace actual value with NaN

df['Education'].iloc[83] = np.nan

#Mode inputation

mode\_val = df['Education'].mode()

df['Education'].iloc[83] = np.nan

#Substitute with actual value

df['Education'].iloc[83] = act\_cat\_val

Imputing row 1/704 with 0 missing, elapsed time: 0.382

Imputing row 101/704 with 1 missing, elapsed time: 0.392

Imputing row 201/704 with 0 missing, elapsed time: 0.393

Imputing row 301/704 with 0 missing, elapsed time: 0.394

Imputing row 401/704 with 0 missing, elapsed time: 0.395

Imputing row 501/704 with 0 missing, elapsed time: 0.395

Imputing row 601/704 with 0 missing, elapsed time: 0.396

Imputing row 701/704 with 0 missing, elapsed time: 0.396

Imputing row 1/704 with 0 missing, elapsed time: 0.104

Imputing row 101/704 with 1 missing, elapsed time: 0.106

Imputing row 201/704 with 0 missing, elapsed time: 0.109

Imputing row 301/704 with 0 missing, elapsed time: 0.109

Imputing row 401/704 with 0 missing, elapsed time: 0.110

Imputing row 501/704 with 0 missing, elapsed time: 0.112

Imputing row 601/704 with 0 missing, elapsed time: 0.112

Imputing row 701/704 with 0 missing, elapsed time: 0.113

Imputing row 1/704 with 0 missing, elapsed time: 0.183

Imputing row 101/704 with 1 missing, elapsed time: 0.187

Imputing row 201/704 with 0 missing, elapsed time: 0.189

Imputing row 301/704 with 0 missing, elapsed time: 0.191

Imputing row 401/704 with 0 missing, elapsed time: 0.193

Imputing row 501/704 with 0 missing, elapsed time: 0.194

Imputing row 601/704 with 0 missing, elapsed time: 0.195

Imputing row 701/704 with 0 missing, elapsed time: 0.195

In [44]:

#Results of Imputation

print("###### Imputing missing values for continuous attribute - example ######")

print("Actual value is: %f" % (act\_val))

print("Mean value is:%i" % int(mean\_val))

print("Median value is:%i" % int(median\_val))

print("KNN value for k = 3 is:%i" % int(knn3\_val))

print("KNN value for k = 5 is:%i" % int(knn5\_val))

print("KNN value for k = 7 is:%i" % int(knn7\_val))

print("###### Imputing missing values for categorical attribute - example ######")

print("Actual value is: %f" % (act\_cat\_val))

print("Mode value is:%f" % (mode\_val))

print("KNN value for k = 3 is:%f" % (df\_knn5['Education'].iloc[83]))

print("KNN value for k = 5 is:%f" % (df\_knn5['Education'].iloc[83]))

print("KNN value for k = 7 is:%f" % (df\_knn7['Education'].iloc[83]))

###### Imputing missing values for continuous attribute - example ######

Actual value is: 89.000000

Mean value is:78

Median value is:80

KNN value for k = 3 is:89

KNN value for k = 5 is:88

KNN value for k = 7 is:87

###### Imputing missing values for categorical attribute - example ######

Actual value is: 1.000000

Mode value is:1.000000

KNN value for k = 3 is:1.000000

KNN value for k = 5 is:1.000000

KNN value for k = 7 is:1.000000

Thus, from the above methods of imputation, we can choose KNN method for imputation with k = 3.

In [45]:

#Impute missing values with k = 3

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

for i in num\_cols:

df[i]= round(df[i])

df['ID'] = df['ID'].astype('int64')

for i in cat\_cols:

df[i] = df[i].astype('int64')

df[i] = df[i].astype('category')

Imputing row 1/704 with 0 missing, elapsed time: 0.091

Imputing row 101/704 with 1 missing, elapsed time: 0.093

Imputing row 201/704 with 0 missing, elapsed time: 0.094

Imputing row 301/704 with 0 missing, elapsed time: 0.095

Imputing row 401/704 with 0 missing, elapsed time: 0.096

Imputing row 501/704 with 0 missing, elapsed time: 0.096

Imputing row 601/704 with 0 missing, elapsed time: 0.097

Imputing row 701/704 with 0 missing, elapsed time: 0.097

In [46]:

#to check for missing values after imputation

df.isnull().sum()

Out[46]:

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

Thus, there are no missing values present in the dataset.

#### Outlier analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Outlier-analysis)

##### Boxplot analysis[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Boxplot-analysis)

In [47]:

#Plot boxplot to identify outliers in the dataset

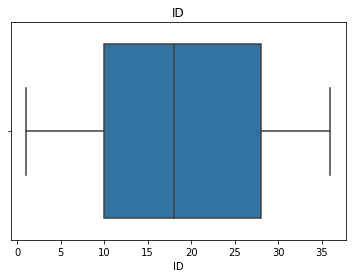
for i in num\_cols:

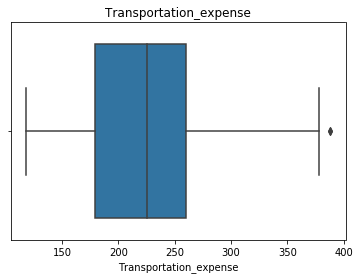
plt.figure()

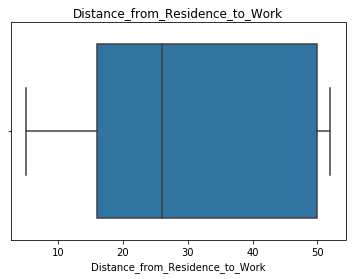
plt.clf() #clearing the figure

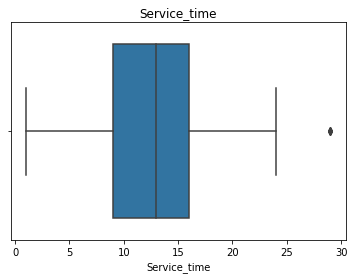
sbn.boxplot(df[i])

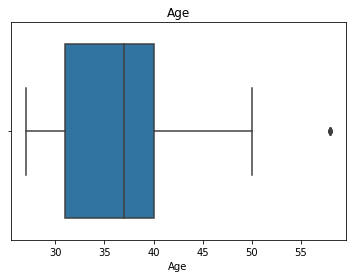
plt.title(i)

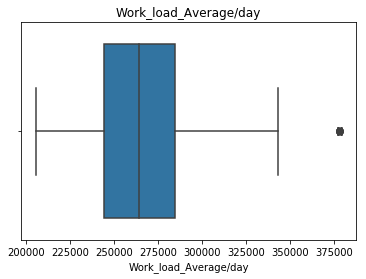
****

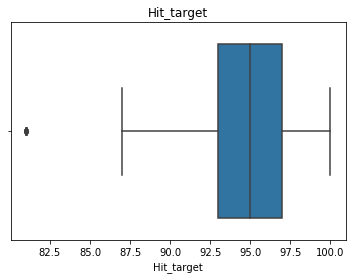
****

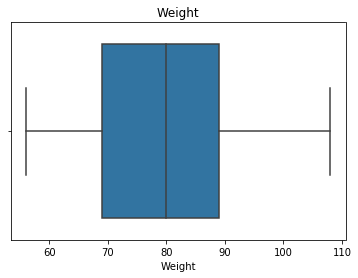
****

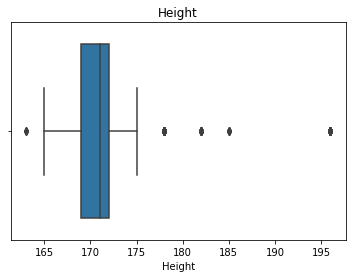
****

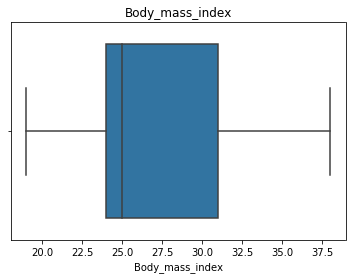
****

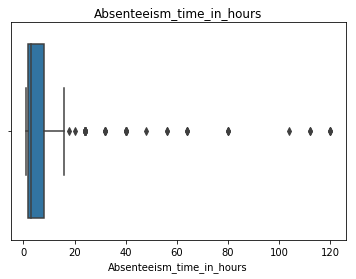
****

****

****

****

****

****

In [48]:

#Outlier analysis using boxplot method

print("\*\*\*\*\*\*\*\* Outlier Analysis \*\*\*\*\*\*\*\*")

for i in num\_cols:

print(" ")

print("\*\*\*\*\* " + i + " \*\*\*\*\*")

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

iqr = q75 - q25

min\_val = q25 - (iqr\*1.5)

max\_val = q75 + (iqr\*1.5)

print("Minimum and Maximum values are:")

print(min\_val)

print(max\_val)

print('{var} -----> {X} Outliers'.format(var = i, X = (df.loc[df.loc[:,i] < min\_val,i].count() +

df.loc[df.loc[:,i] > max\_val,i].count())))

\*\*\*\*\*\*\*\* Outlier Analysis \*\*\*\*\*\*\*\*

\*\*\*\*\* ID \*\*\*\*\*

Minimum and Maximum values are:

-17.0

55.0

ID -----> 0 Outliers

\*\*\*\*\* Transportation\_expense \*\*\*\*\*

Minimum and Maximum values are:

57.5

381.5

Transportation\_expense -----> 2 Outliers

\*\*\*\*\* Distance\_from\_Residence\_to\_Work \*\*\*\*\*

Minimum and Maximum values are:

-35.0

101.0

Distance\_from\_Residence\_to\_Work -----> 0 Outliers

\*\*\*\*\* Service\_time \*\*\*\*\*

Minimum and Maximum values are:

-1.5

26.5

Service\_time -----> 5 Outliers

\*\*\*\*\* Age \*\*\*\*\*

Minimum and Maximum values are:

17.5

53.5

Age -----> 8 Outliers

\*\*\*\*\* Work\_load\_Average/day \*\*\*\*\*

Minimum and Maximum values are:

182813.625

346076.625

Work\_load\_Average/day -----> 28 Outliers

\*\*\*\*\* Hit\_target \*\*\*\*\*

Minimum and Maximum values are:

87.0

103.0

Hit\_target -----> 15 Outliers

\*\*\*\*\* Weight \*\*\*\*\*

Minimum and Maximum values are:

39.0

119.0

Weight -----> 0 Outliers

\*\*\*\*\* Height \*\*\*\*\*

Minimum and Maximum values are:

164.5

176.5

Height -----> 109 Outliers

\*\*\*\*\* Body\_mass\_index \*\*\*\*\*

Minimum and Maximum values are:

13.5

41.5

Body\_mass\_index -----> 0 Outliers

\*\*\*\*\* Absenteeism\_time\_in\_hours \*\*\*\*\*

Minimum and Maximum values are:

-7.0

17.0

Absenteeism\_time\_in\_hours -----> 45 Outliers

In [49]:

#to replace outlier values with NaN

print("\*\*\*\*\* After Replacing Outlier values with NaN \*\*\*\*\*")

print(" ")

for i in num\_cols:

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

iqr = q75 - q25

min\_val = q25 - (iqr\*1.5)

max\_val = q75 + (iqr\*1.5)

df.loc[df.loc[:,i]<min\_val, i] = np.nan

df.loc[df.loc[:,i]>max\_val, i] = np.nan

print('{var} -----> {X} Outliers'.format(var = i, X = (df.loc[df.loc[:,i] < min\_val,i].count() +

df.loc[df.loc[:,i] > max\_val,i].count())))

\*\*\*\*\* After Replacing Outlier values with NaN \*\*\*\*\*

ID -----> 0 Outliers

Transportation\_expense -----> 0 Outliers

Distance\_from\_Residence\_to\_Work -----> 0 Outliers

Service\_time -----> 0 Outliers

Age -----> 0 Outliers

Work\_load\_Average/day -----> 0 Outliers

Hit\_target -----> 0 Outliers

Weight -----> 0 Outliers

Height -----> 0 Outliers

Body\_mass\_index -----> 0 Outliers

Absenteeism\_time\_in\_hours -----> 0 Outliers

In [50]:

#to impute NaN values with KNN method using k = 3

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

for i in num\_cols:

df[i]= round(df[i])

df['ID'] = df['ID'].astype('int64')

for i in cat\_cols:

df[i] = df[i].astype('int64')

df[i] = df[i].astype('category')

Imputing row 1/704 with 0 missing, elapsed time: 0.098

Imputing row 101/704 with 1 missing, elapsed time: 0.099

Imputing row 201/704 with 2 missing, elapsed time: 0.100

Imputing row 301/704 with 0 missing, elapsed time: 0.103

Imputing row 401/704 with 0 missing, elapsed time: 0.105

Imputing row 501/704 with 0 missing, elapsed time: 0.107

Imputing row 601/704 with 0 missing, elapsed time: 0.109

Imputing row 701/704 with 0 missing, elapsed time: 0.111

In [51]:

#to check if the outliers have been imputed

for i in num\_cols:

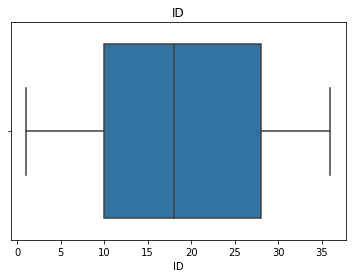
plt.figure()

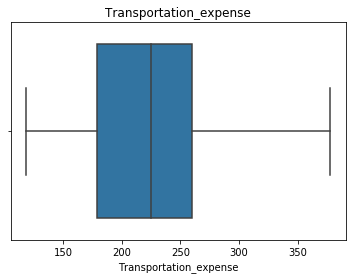
plt.clf() #clearing the figure

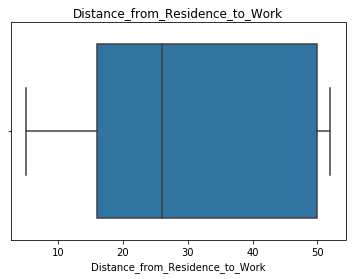
sbn.boxplot(df[i])

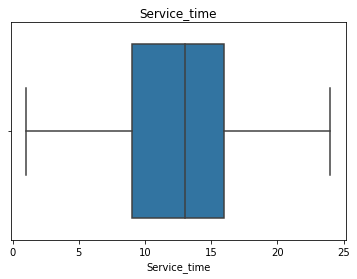
plt.title(i)

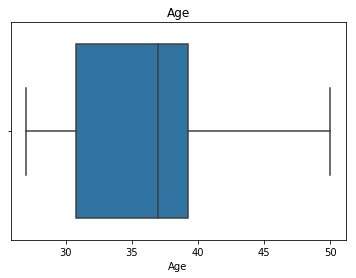
plt.show()

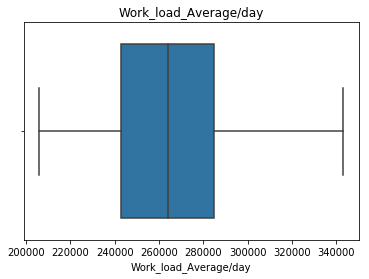
****

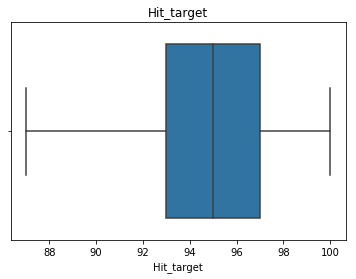
****

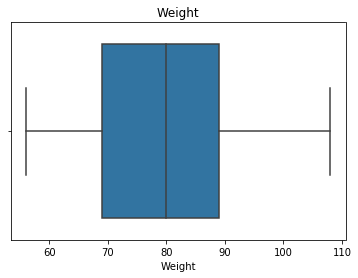
****

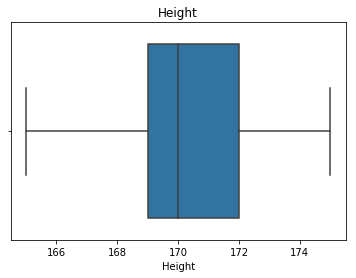
****

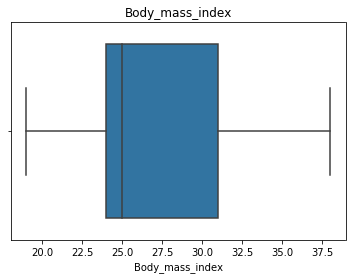
****

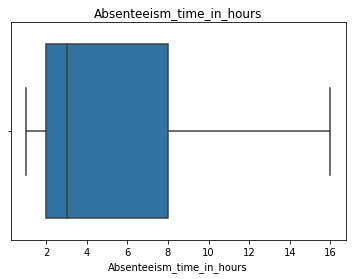
****

****

****

****

****

****

Thus, there are no outliers present in the dataset.

##### Data Visualization[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Data-Visualization)

Data visualization helps us to understand the relationship between features. Here, to understand how each independent feature is related to the target feature, we need to perform visualization on data.

###### Distance from Residence to Work Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Distance-from-Residence-to-Work-Vs-Absenteeism-time-in-hours)

In [52]:

dist = df.groupby('Distance\_from\_Residence\_to\_Work')[['Absenteeism\_time\_in\_hours']].mean()

ax = dist.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(dist.values):

ax.text(i-.25, v + 0.2, str(np.int(np.round(v))), color='red')

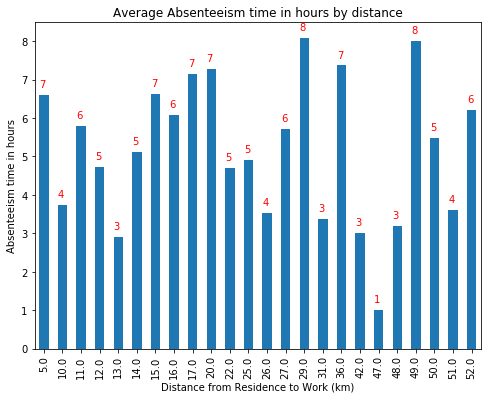
plt.xlabel('Distance from Residence to Work (km)')

plt.ylabel('Absenteeism time in hours')

plt.title('Average Absenteeism time in hours by distance')

Out[52]:

Text(0.5, 1.0, 'Average Absenteeism time in hours by distance')

****

General hypothesis is that distance from residence to work will have an impact on absenteeism. Here, average hours of absent remains the same irrespective of the distance from residence to work of the employees. There is concentration of more leaves where the distance of residence from work is between 10-30 kms.

###### Work load Average/day Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Work-load-Average/day-Vs-Absenteeism-time-in-hours)

In [53]:

wl = df.groupby('Work\_load\_Average/day')[['Absenteeism\_time\_in\_hours']].mean()

ax = wl.plot(kind='bar', figsize=(15,6), legend=False)

for i, v in enumerate(wl.values):

ax.text(i-.25, v + 0.2, str(np.int(np.round(v))), color='red')

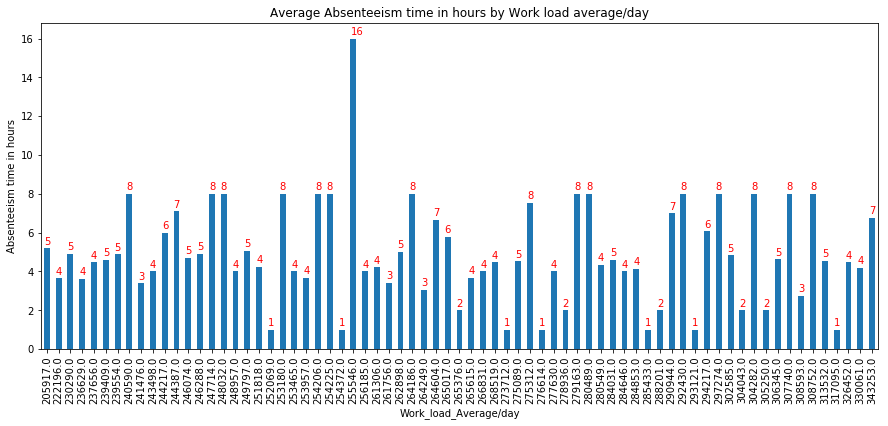
plt.xlabel('Work\_load\_Average/day')

plt.ylabel('Absenteeism time in hours')

plt.title('Average Absenteeism time in hours by Work load average/day')

Out[53]:

Text(0.5, 1.0, 'Average Absenteeism time in hours by Work load average/day')

****

Thus, average hours of absent remains same irrespective of the work load average/day.

###### Service time Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Service-time-Vs-Absenteeism-time-in-hours)

In [54]:

st = df.groupby('Service\_time')[['Absenteeism\_time\_in\_hours']].mean()

ax = st.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(st.values):

ax.text(i-.25, v + 0.20, str(np.int(np.round(v))), color='red')

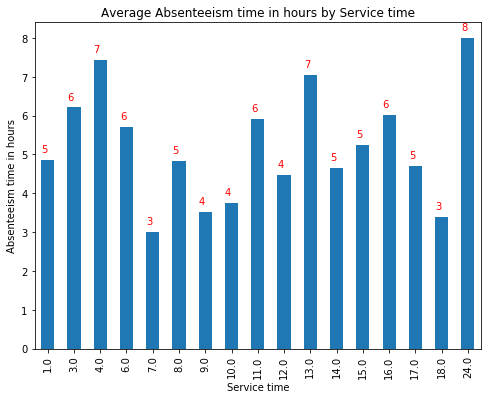
plt.xlabel('Service time')

plt.ylabel('Absenteeism time in hours')

plt.title('Average Absenteeism time in hours by Service time')

Out[54]:

Text(0.5, 1.0, 'Average Absenteeism time in hours by Service time')

****

Thus, the employees with service years > 8 tend to take more leaves.

###### Disciplinary failure Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Disciplinary-failure-Vs-Absenteeism-time-in-hours)

In [55]:

disp\_absent = df.groupby('Disciplinary\_failure')['Absenteeism\_time\_in\_hours'].sum()

ax = disp\_absent.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(disp\_absent.values):

ax.text(i-.25, v + 0.4, str(np.int(np.round(v))), color='red')

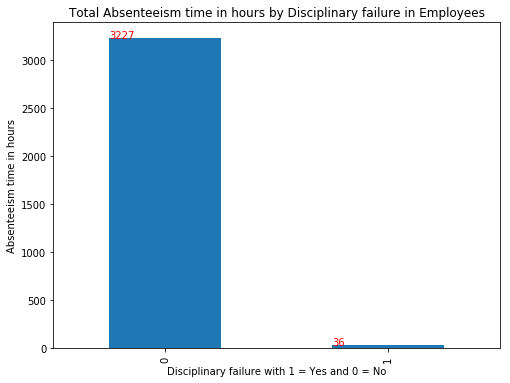
plt.xlabel('Disciplinary failure with 1 = Yes and 0 = No')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by Disciplinary failure in Employees')

Out[55]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by Disciplinary failure in Employees')

****

Thus, it is observed that employees with no disciplinary failure have the highest absent hours in total.

###### Age of the employees Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Age-of-the-employees-Vs-Absenteeism-time-in-hours)

In [56]:

age\_abs = df.groupby('Age')[['Absenteeism\_time\_in\_hours']].mean()

ax = age\_abs.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(age\_abs.values):

ax.text(i-.25, v + 0.2, str(np.int(np.round(v))), color='red')

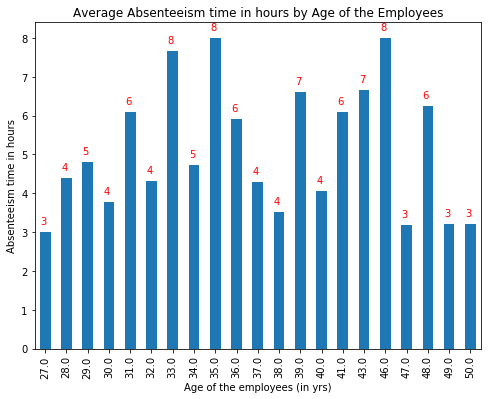
plt.xlabel('Age of the employees (in yrs)')

plt.ylabel('Absenteeism time in hours')

plt.title('Average Absenteeism time in hours by Age of the Employees')

Out[56]:

Text(0.5, 1.0, 'Average Absenteeism time in hours by Age of the Employees')

****

Hypothesis is that as the age increases, employees tend to take more leave compared to others due to health issues. Here, people over 45+ years of age tends to take less leaves compared to others.

###### Number of children Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Number-of-children-Vs-Absenteeism-time-in-hours)

In [57]:

son = df.groupby('Son')['Absenteeism\_time\_in\_hours'].sum()

ax = son.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(son.values):

ax.text(i-.25, v + 0.5, str(np.int(np.round(v))), color='red')

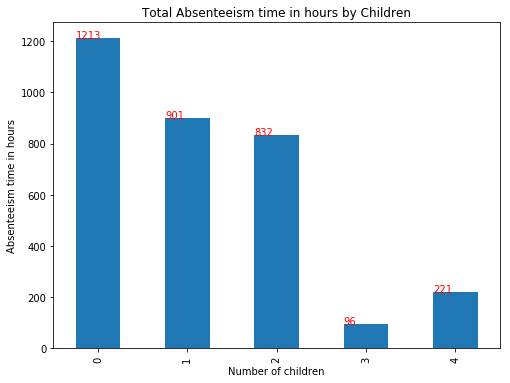
plt.xlabel('Number of children')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by Children')

Out[57]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by Children')

****

It is interesting to note that employees with no issues have highest absenteeism. This is followed by employees with 1 child and it might be due to the fact that there would be noone else to take care of the child if it falls sick and one parent has to stay back with the kid. In case of 3 or 4 children, there is a high chance that older siblings take care of younger ones.

###### Number of pets Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Number-of-pets-Vs-Absenteeism-time-in-hours)

In [58]:

pet = df.groupby('Pet')['Absenteeism\_time\_in\_hours'].sum()

ax = pet.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(pet.values):

ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

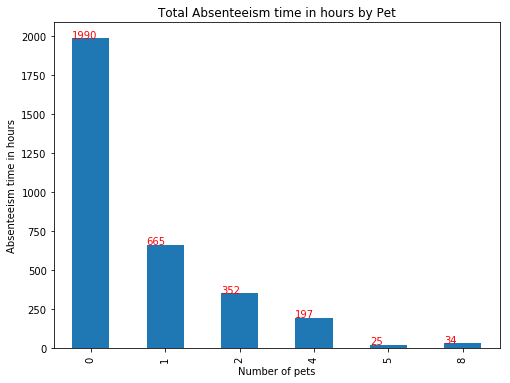
plt.xlabel('Number of pets')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by Pet')

Out[58]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by Pet')

****

It is to be noted that the employees with no pet or 1 pet are frequent absentees.

###### Reason for absence Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Reason-for-absence-Vs-Absenteeism-time-in-hours)

In [59]:

reason\_absent = df.groupby('Reason\_for\_absence')[['Absenteeism\_time\_in\_hours']].sum()

ax = reason\_absent.plot(kind='bar', figsize=(11,6), legend=False)

for i, v in enumerate(reason\_absent.values):

ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

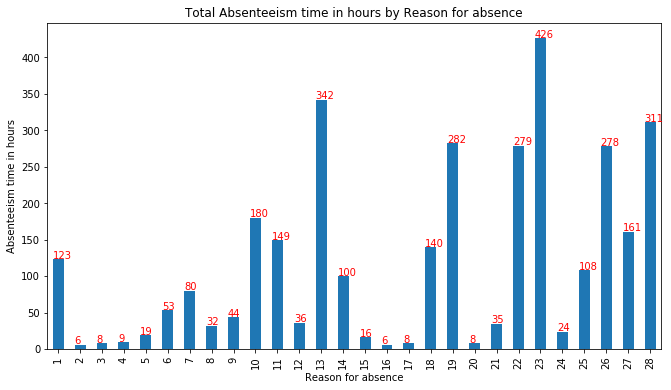
plt.xlabel('Reason for absence')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by Reason for absence')

Out[59]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by Reason for absence')

****

###### Reasons and Remedies/Suggestions to reduce absenteeism:[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Reasons-and-Remedies/Suggestions-to-reduce-absenteeism:)

The top four reasons for absence are:

* Diseases of the musculoskeletal system and connective tissue
* Injury, poisoning and certain other consequences of external causes
* Medical consultation
* Unjustified absence

Remedies/Suggestions to reduce absenteeism:

1. Musculoskeletal system disease is the major reason of absenteeism. Bad working posture and high workload are possible reasons for the high incidence of musculoskeletal disease. Company should conduct a study on the working postures of people and come up with ergonomic workplace design. Company should try to optimize workload keeping in mind occupational health of working people.
2. Injury, poisoning and certain other consequences of external causes can be the consequences of the bad and unsafe working environment. Safety of the employees should be the major concern of the company.
3. Medical and dental consultation can be brought down by optimizing workloads and adapting to employee friendly workplace and conducting more programs and medical camps to create awareness on importance of physical and mental wellness among employees.
4. Unjustified absence is too high.Company should try to reduce high workloads and set up measures so that employees don't feel work stress and they get to discuss work related problems and pressures.
5. Counselling sessions are to be conducted on regular basis to employees to get their feedbacks and help them in their work and personal issues if any.

###### Month of absence Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Month-of-absence-Vs-Absenteeism-time-in-hours)

In [60]:

month = df.groupby('Month\_of\_absence')[['Absenteeism\_time\_in\_hours']].sum()

ax = month.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(month.values):

ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

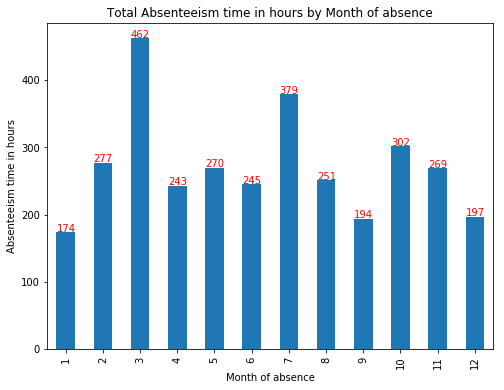
plt.xlabel('Month of absence')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by Month of absence')

Out[60]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by Month of absence')

****

We can infer that maximum hours of absent are recorded in the month of March followed by July. we can check the reason for absence in these months to analyse further.

In [61]:

monthly\_absence = df.groupby('Month\_of\_absence')['Absenteeism\_time\_in\_hours'].sum()

monthly\_absence = monthly\_absence.reset\_index()

monthly\_absence['Percentage\_of\_absent\_hours'] = monthly\_absence['Absenteeism\_time\_in\_hours']/sum(monthly\_absence['Absenteeism\_time\_in\_hours']) \*100

monthly\_absence = monthly\_absence.sort\_values('Absenteeism\_time\_in\_hours',ascending = False).reset\_index(drop=True)

monthly\_absence

Out[61]:

|  | **Month\_of\_absence** | **Absenteeism\_time\_in\_hours** | **Percentage\_of\_absent\_hours** |
| --- | --- | --- | --- |
| **0** | 3 | 462.0 | 14.158750 |
| **1** | 7 | 379.0 | 11.615078 |
| **2** | 10 | 302.0 | 9.255287 |
| **3** | 2 | 277.0 | 8.489120 |
| **4** | 5 | 270.0 | 8.274594 |
| **5** | 11 | 269.0 | 8.243947 |
| **6** | 8 | 251.0 | 7.692308 |
| **7** | 6 | 245.0 | 7.508428 |
| **8** | 4 | 243.0 | 7.447135 |
| **9** | 12 | 197.0 | 6.037389 |
| **10** | 9 | 194.0 | 5.945449 |
| **11** | 1 | 174.0 | 5.332516 |

In [62]:

#Reasons for absence in March month

mar\_absent = df[df['Month\_of\_absence']==3].groupby('Reason\_for\_absence', as\_index=False)['Absenteeism\_time\_in\_hours'].sum()

mar\_absent = mar\_absent[mar\_absent['Absenteeism\_time\_in\_hours']>0]

mar\_absent.sort\_values('Absenteeism\_time\_in\_hours', ascending=False).reset\_index(drop=True)

Out[62]:

|  | **Reason\_for\_absence** | **Absenteeism\_time\_in\_hours** |
| --- | --- | --- |
| **0** | 13 | 62.0 |
| **1** | 22 | 56.0 |
| **2** | 19 | 51.0 |
| **3** | 28 | 43.0 |
| **4** | 27 | 41.0 |
| **5** | 11 | 40.0 |
| **6** | 14 | 33.0 |
| **7** | 23 | 28.0 |
| **8** | 26 | 24.0 |
| **9** | 1 | 16.0 |
| **10** | 25 | 13.0 |
| **11** | 18 | 8.0 |
| **12** | 6 | 8.0 |
| **13** | 21 | 8.0 |
| **14** | 8 | 8.0 |
| **15** | 7 | 8.0 |
| **16** | 17 | 8.0 |
| **17** | 10 | 4.0 |
| **18** | 12 | 3.0 |

The top four reasons for absence are:

* Diseases of the musculoskeletal system and connective tissue
* Patient follow-up
* Injury, poisoning and certain other consequences of external causes
* Dental consultation

In [63]:

#Reasons for absence in July month

july\_absent = df[df['Month\_of\_absence']==7].groupby('Reason\_for\_absence', as\_index=False)['Absenteeism\_time\_in\_hours'].sum()

july\_absent = july\_absent[july\_absent['Absenteeism\_time\_in\_hours']>0]

july\_absent.sort\_values('Absenteeism\_time\_in\_hours', ascending=False).reset\_index(drop=True)

Out[63]:

|  | **Reason\_for\_absence** | **Absenteeism\_time\_in\_hours** |
| --- | --- | --- |
| **0** | 22 | 72.0 |
| **1** | 26 | 56.0 |
| **2** | 1 | 40.0 |
| **3** | 19 | 37.0 |
| **4** | 23 | 24.0 |
| **5** | 11 | 21.0 |
| **6** | 28 | 18.0 |
| **7** | 25 | 17.0 |
| **8** | 13 | 16.0 |
| **9** | 6 | 16.0 |
| **10** | 14 | 12.0 |
| **11** | 7 | 12.0 |
| **12** | 5 | 8.0 |
| **13** | 18 | 8.0 |
| **14** | 21 | 8.0 |
| **15** | 15 | 8.0 |
| **16** | 9 | 6.0 |

The top four reasons for absence are:

* Patient follow-up
* Unjustified absence
* Certain infectious and parasitic diseases
* Injury, poisoning and certain other consequences of external causes

Unjustified absence might be due to personal reason(vacation with family) as it is holiday season.

###### Seasons Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Seasons-Vs-Absenteeism-time-in-hours)

In [64]:

season = df.groupby('Seasons')[['Absenteeism\_time\_in\_hours']].sum()

ax = season.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(season.values):

ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

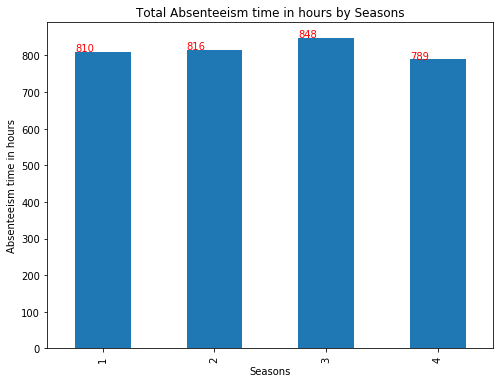
plt.xlabel('Seasons')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by Seasons')

Out[64]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by Seasons')

****

Winter season has highest absenteeism followed by Summer. It is also to be noted that March and July month has highest absenteeism.

###### Day of the week Vs Absenteeism time in hours[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Day-of-the-week-Vs-Absenteeism-time-in-hours)

In [65]:

day\_absent = df.groupby('Day\_of\_the\_week')[['Absenteeism\_time\_in\_hours']].sum()

ax = day\_absent.plot(kind='bar', figsize=(8,6), legend=False)

for i, v in enumerate(day\_absent.values):

ax.text(i-.25, v + 1, str(np.int(np.round(v))), color='red')

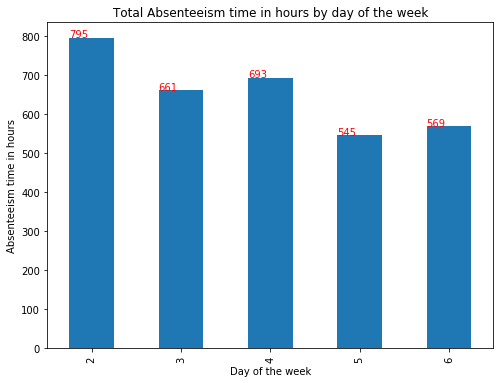
plt.xlabel('Day of the week')

plt.ylabel('Absenteeism time in hours')

plt.title('Total Absenteeism time in hours by day of the week')

Out[65]:

Text(0.5, 1.0, 'Total Absenteeism time in hours by day of the week')

****

Highest percentage of absenteeism is recorded on start of the week, Mondays followed by Tuesdays which can be due to the fact that most people travel over weekend and they tend to extend their holiday plan till monday/tuesday.

###### Effect on Absenteeism by Social drinking/smoking in employees[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Effect-on-Absenteeism-by-Social-drinking/smoking-in-employees)

In [66]:

drink\_only = df[(df['Social\_drinker'] == 1) & (df['Social\_smoker']==0)]['Absenteeism\_time\_in\_hours'].sum()

smoke\_only = df[(df['Social\_drinker'] == 0) & (df['Social\_smoker']==1)]['Absenteeism\_time\_in\_hours'].sum()

drink\_smoke\_both = df[(df['Social\_drinker'] == 1) & (df['Social\_smoker']==1)]['Absenteeism\_time\_in\_hours'].sum()

no\_drink\_smoke = df[(df['Social\_drinker'] == 0) & (df['Social\_smoker']==0)]['Absenteeism\_time\_in\_hours'].sum()

absent\_drink\_smoke = [drink\_only, smoke\_only, drink\_smoke\_both, no\_drink\_smoke]

pie\_labels = ['Social drinker', 'Social smoker', 'Social drinker & smoker', 'Non drinker/smoker']

fig, ax = plt.subplots(figsize=(8, 6))

def func(x, allvals):

absolute = int(x/100.\*np.sum(allvals))

return "{:.1f}%".format(x)

wedges, texts, autotexts = ax.pie(absent\_drink\_smoke, autopct=lambda x: func(x, absent\_drink\_smoke), textprops=dict(color='w'))

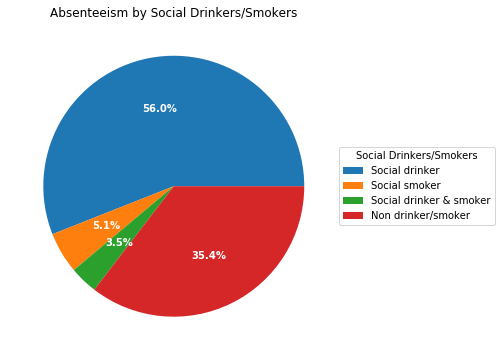
ax.legend(wedges, pie\_labels, title='Social Drinkers/Smokers', loc='right', bbox\_to\_anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=10, weight="bold")

ax.set\_title('Absenteeism by Social Drinkers/Smokers')

Out[66]:

Text(0.5, 1.0, 'Absenteeism by Social Drinkers/Smokers')

****

Looks like 57% of Social drinkers and interestingly 35% of Non drinkers/smokers are regular absentees.

#### Feature Engineering[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Feature-Engineering)

##### Feature Selection[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Feature-Selection)

Correlation Analysis:

The relationship between numerical variables in the dataset can be found using correlation matrix.

Hypothesis of Correlation Analysis:

1. There should be low or no correlation between independent numerical variables.
2. There should be high correlation between target and independent numerical variables.

In [67]:

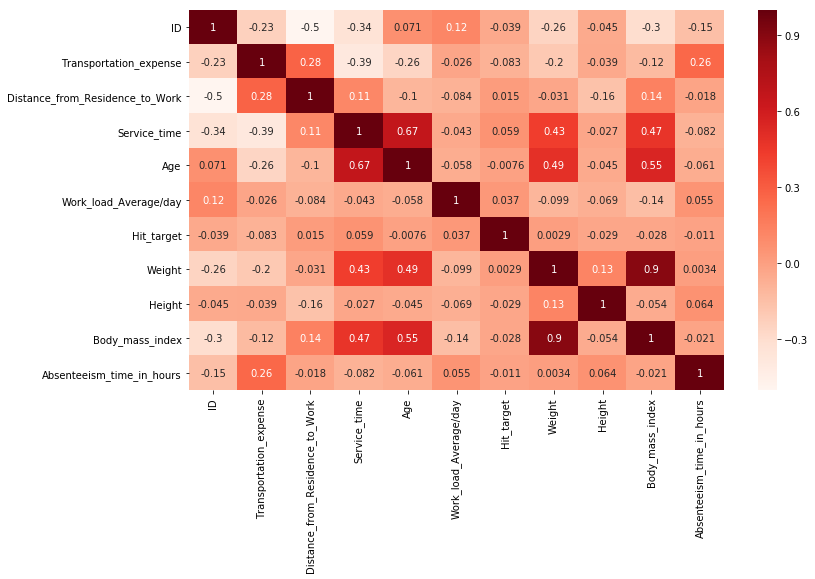
#Plot correlation matrix to find correlation between numerical variables

plt.figure(figsize=(12,7))

cor = df[num\_cols].corr()

sbn.heatmap(cor, annot=True, cmap=plt.cm.Reds)

plt.show()

****

Here, none of the independent attributes are highly correlated with the target and with other each other.

Chi-Square Test:

This test is used to derive the statistical significance of relationship between the categorical variables in the dataset. It returns probability for the computed chi-square distribution with the degree of freedom.

Hypothesis of Chi-Square test:

1. Null Hypothesis: The null hypothesis of the Chi-Square test is that no relationship exists on the categorical variables in the population; they are independent.
2. Alternate Hypothesis: The alternate hypothesis of the Chi-Square test is that there exists relationship between the categorical variables in the population; they are not independent.

If p-value is less than 0.05 then we reject the null hypothesis. And if p-value is greater than 0.05 then we accept the null hypothesis.

In [68]:

# Chi-Square test to find relationship among categorical attributes

factors\_paired = [(i,j) for i in cat\_cols for j in cat\_cols]

chi2, p\_val =[], []

for f in factors\_paired:

if f[0] != f[1]:

ch12, p, dof, ex = chi2\_contingency(pd.crosstab(df[f[0]], df[f[1]]))

chi2.append(ch12)

p\_val.append(p)

if p <0.05:

print(f[0]+":"+f[1])

else:

chi2.append(0)

p\_val.append(0)

p1 = np.array(p\_val).reshape((10,10))

df1 = pd.DataFrame(p1, index= cat\_cols, columns = cat\_cols)

print("Probability distribution is:")

df1

Month\_of\_absence:Reason\_for\_absence

Month\_of\_absence:Social\_drinker

Month\_of\_absence:Social\_smoker

Month\_of\_absence:Seasons

Month\_of\_absence:Pet

Month\_of\_absence:Education

Month\_of\_absence:Son

Disciplinary\_failure:Reason\_for\_absence

Disciplinary\_failure:Social\_smoker

Disciplinary\_failure:Seasons

Reason\_for\_absence:Month\_of\_absence

Reason\_for\_absence:Disciplinary\_failure

Reason\_for\_absence:Social\_drinker

Reason\_for\_absence:Social\_smoker

Reason\_for\_absence:Seasons

Reason\_for\_absence:Pet

Reason\_for\_absence:Education

Reason\_for\_absence:Son

Social\_drinker:Month\_of\_absence

Social\_drinker:Reason\_for\_absence

Social\_drinker:Social\_smoker

Social\_drinker:Pet

Social\_drinker:Education

Social\_drinker:Son

Social\_smoker:Month\_of\_absence

Social\_smoker:Disciplinary\_failure

Social\_smoker:Reason\_for\_absence

Social\_smoker:Social\_drinker

Social\_smoker:Pet

Social\_smoker:Education

Social\_smoker:Son

Seasons:Month\_of\_absence

Seasons:Disciplinary\_failure

Seasons:Reason\_for\_absence

Seasons:Pet

Seasons:Son

Pet:Month\_of\_absence

Pet:Reason\_for\_absence

Pet:Social\_drinker

Pet:Social\_smoker

Pet:Seasons

Pet:Education

Pet:Son

Day\_of\_the\_week:Son

Education:Month\_of\_absence

Education:Reason\_for\_absence

Education:Social\_drinker

Education:Social\_smoker

Education:Pet

Education:Son

Son:Month\_of\_absence

Son:Reason\_for\_absence

Son:Social\_drinker

Son:Social\_smoker

Son:Seasons

Son:Pet

Son:Day\_of\_the\_week

Son:Education

Probability distribution is:

Out[68]:

|  | **Month\_of\_absence** | **Disciplinary\_failure** | **Reason\_for\_absence** | **Social\_drinker** | **Social\_smoker** | **Seasons** | **Pet** | **Day\_of\_the\_week** | **Education** | **Son** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Month\_of\_absence** | 0.000000e+00 | 4.180979e-01 | 1.712661e-14 | 2.202100e-02 | 3.194987e-02 | 0.000000e+00 | 5.667973e-05 | 6.087582e-01 | 1.168501e-02 | 3.306114e-06 |
| **Disciplinary\_failure** | 4.180979e-01 | 0.000000e+00 | 1.020492e-12 | 6.666733e-01 | 2.660185e-03 | 2.425953e-02 | 7.200002e-01 | 2.460672e-01 | 9.669382e-01 | 4.570634e-01 |
| **Reason\_for\_absence** | 1.712661e-14 | 1.020492e-12 | 0.000000e+00 | 1.910487e-08 | 3.583594e-08 | 3.260335e-20 | 8.138594e-19 | 6.080070e-02 | 1.560248e-16 | 1.417274e-17 |
| **Social\_drinker** | 2.202100e-02 | 6.666733e-01 | 1.910487e-08 | 0.000000e+00 | 1.608149e-02 | 1.843045e-01 | 1.740071e-26 | 3.657264e-01 | 3.872295e-33 | 3.414471e-09 |
| **Social\_smoker** | 3.194987e-02 | 2.660185e-03 | 3.583594e-08 | 1.608149e-02 | 0.000000e+00 | 1.569992e-01 | 5.706486e-14 | 8.227146e-01 | 1.285777e-23 | 2.506498e-21 |
| **Seasons** | 0.000000e+00 | 2.425953e-02 | 3.260335e-20 | 1.843045e-01 | 1.569992e-01 | 0.000000e+00 | 1.772464e-04 | 3.948001e-01 | 1.107407e-01 | 2.965377e-06 |
| **Pet** | 5.667973e-05 | 7.200002e-01 | 8.138594e-19 | 1.740071e-26 | 5.706486e-14 | 1.772464e-04 | 0.000000e+00 | 4.046197e-01 | 2.446169e-27 | 3.179308e-88 |
| **Day\_of\_the\_week** | 6.087582e-01 | 2.460672e-01 | 6.080070e-02 | 3.657264e-01 | 8.227146e-01 | 3.948001e-01 | 4.046197e-01 | 0.000000e+00 | 6.167258e-01 | 1.241238e-08 |
| **Education** | 1.168501e-02 | 9.669382e-01 | 1.560248e-16 | 3.872295e-33 | 1.285777e-23 | 1.107407e-01 | 2.446169e-27 | 6.167258e-01 | 0.000000e+00 | 8.133766e-11 |
| **Son** | 3.306114e-06 | 4.570634e-01 | 1.417274e-17 | 3.414471e-09 | 2.506498e-21 | 2.965377e-06 | 3.179308e-88 | 1.241238e-08 | 8.133766e-11 | 0.000000e+00 |

We can observe that p value of Seasons vs Month\_of\_absence is 0.000000e+00 which means that it is highly dependent on Month of absence.

In [69]:

#Drop seasons as it is dependent on Month of absence

df = df.drop(['Seasons'],axis=1)

ANOVA:

Analysis Of Variance or ANOVA is used to find the relationship between independent categorical variables and numerical variable. This can be dependent or independent.

Hypothesis of ANOVA testing:

-> Null Hypothesis: Mean of all categories in a variable are same and numerical variable doesn't depend on it.

-> Alternate Hypothesis: Mean of at least one category in a variable is different and numerical variable depends on it.

If p-value is less than 0.05 then we reject the null hypothesis. And if p-value is greater than 0.05 then we accept the null hypothesis.

In [70]:

#ANOVA test

#model = ols('Absenteeism\_time\_in\_hours ~ C(Day\_of\_the\_week\_2)+C(Day\_of\_the\_week\_3)+C(Day\_of\_the\_week\_4)+C(Day\_of\_the\_week\_5)+C(Day\_of\_the\_week\_6)+C(Education\_0)+C(Education\_1)+C(Education\_2)+C(Education\_3)+C(Education\_4)+C(Social\_drinker\_0)+C(Social\_drinker\_1)+C(Social\_smoker\_0)+C(Social\_smoker\_1)+C(Month\_of\_absence\_1)+C(Month\_of\_absence\_2)+C(Month\_of\_absence\_3)+C(Month\_of\_absence\_4)+C(Month\_of\_absence\_5)+C(Month\_of\_absence\_6)+C(Month\_of\_absence\_7)+C(Month\_of\_absence\_8)+C(Month\_of\_absence\_9)+C(Month\_of\_absence\_10)+C(Month\_of\_absence\_11)+C(Month\_of\_absence\_12)+C(Son\_0)+C(Son\_1)+C(Son\_2)+C(Son\_3)+C(Son\_4)+C(Pet\_0)+C(Pet\_1)+C(Pet\_2)+C(Pet\_4)+C(Pet\_5)+C(Pet\_8)+C(Reason\_for\_absence\_1)+C(Reason\_for\_absence\_2)+C(Reason\_for\_absence\_3)+C(Reason\_for\_absence\_4)+C(Reason\_for\_absence\_5)+C(Reason\_for\_absence\_6)+C(Reason\_for\_absence\_7)+C(Reason\_for\_absence\_8)+C(Reason\_for\_absence\_9)+C(Reason\_for\_absence\_10)+C(Reason\_for\_absence\_11)+C(Reason\_for\_absence\_12)+C(Reason\_for\_absence\_13)+C(Reason\_for\_absence\_14)+C(Reason\_for\_absence\_15)+C(Reason\_for\_absence\_16)+C(Reason\_for\_absence\_17)+C(Reason\_for\_absence\_18)+C(Reason\_for\_absence\_19)+C(Reason\_for\_absence\_20)+C(Reason\_for\_absence\_21)+C(Reason\_for\_absence\_22)+C(Reason\_for\_absence\_23)+C(Reason\_for\_absence\_24)+C(Reason\_for\_absence\_25)+C(Reason\_for\_absence\_26)+C(Reason\_for\_absence\_27)+C(Reason\_for\_absence\_28)+C(Disciplinary\_failure\_0)+C(Disciplinary\_failure\_1)', data = df\_new).fit()

#aov\_table = sm.stats.anova\_lm(model)

#aov\_table

In [71]:

#ANOVA test

model = ols('Absenteeism\_time\_in\_hours ~ C(Day\_of\_the\_week)+C(Social\_drinker)+C(Education)+C(Social\_smoker)+C(Month\_of\_absence)+C(Son)+C(Pet)+C(Reason\_for\_absence)+C(Disciplinary\_failure)', data = df).fit()

aov\_table = sm.stats.anova\_lm(model,typ=2)

aov\_table

Out[71]:

|  | **sum\_sq** | **df** | **F** | **PR(>F)** |
| --- | --- | --- | --- | --- |
| **C(Day\_of\_the\_week)** | 24.436598 | 4.0 | 0.886671 | 4.714536e-01 |
| **C(Social\_drinker)** | 61.387396 | 1.0 | 8.909660 | 2.944029e-03 |
| **C(Education)** | 76.733007 | 4.0 | 2.784224 | 2.590973e-02 |
| **C(Social\_smoker)** | 0.518578 | 1.0 | 0.075266 | 7.839068e-01 |
| **C(Month\_of\_absence)** | 74.198693 | 11.0 | 0.979006 | 4.640588e-01 |
| **C(Son)** | 114.587126 | 4.0 | 4.157744 | 2.476011e-03 |
| **C(Pet)** | 99.677447 | 5.0 | 2.893402 | 1.358614e-02 |
| **C(Reason\_for\_absence)** | 2231.300278 | 27.0 | 11.994335 | 2.172222e-41 |
| **C(Disciplinary\_failure)** | 16.543725 | 1.0 | 2.401128 | 1.217383e-01 |
| **Residual** | 4444.038154 | 645.0 | NaN | NaN |

In [72]:

#to get the attributes that are significant to the target

for i in range(0,len(aov\_table['PR(>F)'])):

if aov\_table['PR(>F)'][i] < 0.05:

print(aov\_table.index[i])

C(Social\_drinker)

C(Education)

C(Son)

C(Pet)

C(Reason\_for\_absence)

Thus, all the attributes except Social smoker are significant to the target. Hence, we can consider all of the attributes given above.

In [73]:

#Drop Social\_Smoker as it is not significant to the target

df = df.drop(['Social\_smoker'],axis=1)

In [74]:

df\_selected = df.copy()

Since our dataset consists of numerical and categorical features, we need to hot encode each category of categorical variables to perform data modelling.

In [75]:

cat\_cols = ['Son',

'Disciplinary\_failure',

'Social\_drinker',

'Pet',

'Reason\_for\_absence',

'Education',

'Day\_of\_the\_week',

'Month\_of\_absence']

In [76]:

#Hot encoding of categorical features

df\_sample = df[num\_cols]

df\_dummies = pd.get\_dummies(df[cat\_cols],drop\_first=True)

df = pd.concat([df\_sample, df\_dummies], axis=1)

df.reset\_index(drop=True)

Out[76]:

|  | **ID** | **Transportation\_expense** | **Distance\_from\_Residence\_to\_Work** | **Service\_time** | **Age** | **Work\_load\_Average/day** | **Hit\_target** | **Weight** | **Height** | **Body\_mass\_index** | **...** | **Month\_of\_absence\_3** | **Month\_of\_absence\_4** | **Month\_of\_absence\_5** | **Month\_of\_absence\_6** | **Month\_of\_absence\_7** | **Month\_of\_absence\_8** | **Month\_of\_absence\_9** | **Month\_of\_absence\_10** | **Month\_of\_absence\_11** | **Month\_of\_absence\_12** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 11 | 289.0 | 36.0 | 13.0 | 33.0 | 239554.0 | 97.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **1** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **2** | 7 | 279.0 | 5.0 | 14.0 | 39.0 | 239554.0 | 97.0 | 68.0 | 168.0 | 24.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **3** | 11 | 289.0 | 36.0 | 13.0 | 33.0 | 239554.0 | 97.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **4** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **5** | 10 | 354.0 | 52.0 | 3.0 | 28.0 | 239554.0 | 97.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **6** | 20 | 260.0 | 50.0 | 11.0 | 36.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **7** | 14 | 155.0 | 12.0 | 14.0 | 34.0 | 239554.0 | 97.0 | 95.0 | 169.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **8** | 1 | 235.0 | 11.0 | 14.0 | 37.0 | 239554.0 | 97.0 | 88.0 | 172.0 | 29.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **9** | 20 | 260.0 | 50.0 | 11.0 | 36.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **10** | 20 | 260.0 | 50.0 | 11.0 | 36.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **11** | 20 | 260.0 | 50.0 | 11.0 | 36.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **12** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **13** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **14** | 24 | 246.0 | 25.0 | 16.0 | 41.0 | 239554.0 | 97.0 | 67.0 | 170.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **15** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **16** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **17** | 6 | 189.0 | 29.0 | 13.0 | 33.0 | 239554.0 | 97.0 | 69.0 | 167.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **18** | 33 | 248.0 | 25.0 | 14.0 | 47.0 | 205917.0 | 92.0 | 86.0 | 165.0 | 32.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **19** | 18 | 330.0 | 16.0 | 4.0 | 28.0 | 205917.0 | 92.0 | 84.0 | 172.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **20** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 205917.0 | 92.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **21** | 10 | 361.0 | 52.0 | 3.0 | 28.0 | 205917.0 | 92.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **22** | 20 | 260.0 | 50.0 | 11.0 | 36.0 | 205917.0 | 92.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **23** | 11 | 289.0 | 36.0 | 13.0 | 33.0 | 205917.0 | 92.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **24** | 10 | 361.0 | 52.0 | 3.0 | 28.0 | 205917.0 | 92.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **25** | 11 | 289.0 | 36.0 | 13.0 | 33.0 | 205917.0 | 92.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **26** | 30 | 157.0 | 27.0 | 6.0 | 29.0 | 205917.0 | 92.0 | 75.0 | 169.0 | 26.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **27** | 11 | 289.0 | 36.0 | 13.0 | 33.0 | 205917.0 | 92.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **28** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 205917.0 | 92.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **29** | 3 | 179.0 | 51.0 | 18.0 | 38.0 | 205917.0 | 92.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **674** | 28 | 225.0 | 26.0 | 9.0 | 28.0 | 237656.0 | 99.0 | 69.0 | 169.0 | 24.0 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **675** | 16 | 118.0 | 15.0 | 24.0 | 46.0 | 275089.0 | 96.0 | 75.0 | 175.0 | 25.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **676** | 22 | 179.0 | 26.0 | 9.0 | 30.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **677** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **678** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **679** | 23 | 378.0 | 49.0 | 11.0 | 36.0 | 275089.0 | 96.0 | 65.0 | 174.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **680** | 36 | 118.0 | 13.0 | 18.0 | 50.0 | 275089.0 | 96.0 | 98.0 | 172.0 | 31.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **681** | 12 | 233.0 | 51.0 | 1.0 | 31.0 | 275089.0 | 96.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **682** | 22 | 179.0 | 26.0 | 9.0 | 30.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **683** | 36 | 118.0 | 13.0 | 18.0 | 50.0 | 275089.0 | 96.0 | 98.0 | 172.0 | 31.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **684** | 22 | 179.0 | 26.0 | 9.0 | 30.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **685** | 15 | 291.0 | 31.0 | 12.0 | 40.0 | 275089.0 | 96.0 | 73.0 | 171.0 | 25.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **686** | 22 | 179.0 | 26.0 | 9.0 | 30.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **687** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **688** | 12 | 233.0 | 51.0 | 1.0 | 31.0 | 275089.0 | 96.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **689** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **690** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **691** | 12 | 233.0 | 51.0 | 1.0 | 31.0 | 275089.0 | 96.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **692** | 5 | 235.0 | 20.0 | 13.0 | 43.0 | 264604.0 | 93.0 | 106.0 | 167.0 | 38.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **693** | 12 | 233.0 | 51.0 | 1.0 | 31.0 | 264604.0 | 93.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **694** | 9 | 228.0 | 14.0 | 16.0 | 40.0 | 264604.0 | 93.0 | 65.0 | 172.0 | 22.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **695** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 264604.0 | 93.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **696** | 9 | 228.0 | 14.0 | 16.0 | 40.0 | 264604.0 | 93.0 | 65.0 | 172.0 | 22.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **697** | 6 | 189.0 | 29.0 | 13.0 | 33.0 | 264604.0 | 93.0 | 69.0 | 167.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **698** | 34 | 118.0 | 10.0 | 10.0 | 37.0 | 264604.0 | 93.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **699** | 10 | 361.0 | 52.0 | 3.0 | 28.0 | 264604.0 | 93.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **700** | 28 | 225.0 | 26.0 | 9.0 | 28.0 | 264604.0 | 93.0 | 69.0 | 169.0 | 24.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **701** | 13 | 369.0 | 17.0 | 12.0 | 31.0 | 264604.0 | 93.0 | 70.0 | 169.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **702** | 11 | 289.0 | 36.0 | 13.0 | 33.0 | 264604.0 | 93.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **703** | 1 | 235.0 | 11.0 | 14.0 | 37.0 | 264604.0 | 93.0 | 88.0 | 172.0 | 29.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

704 rows × 68 columns

##### Feature Scaling[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Feature-Scaling)

In [77]:

#to get the distribution of numerical variables for normality check

for i in num\_cols:

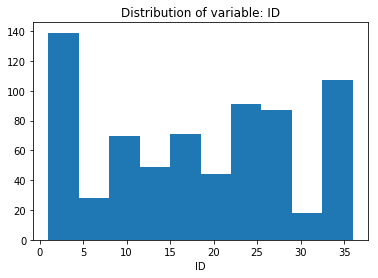
plt.figure()

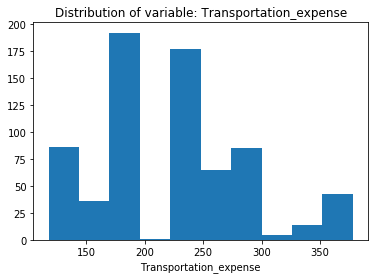
plt.clf()

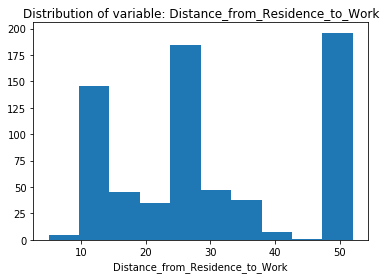
plt.hist(df[i])

plt.title("Distribution of variable: " + i)

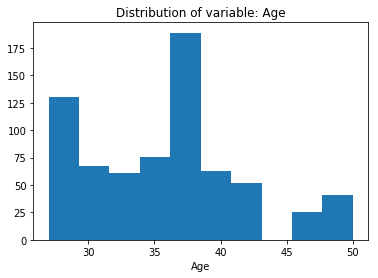
plt.xlabel(i)

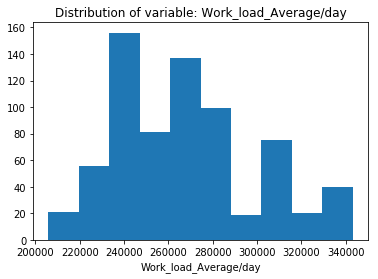
****

****

****

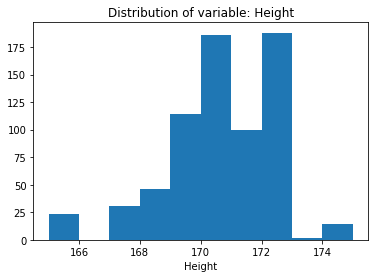
****

****

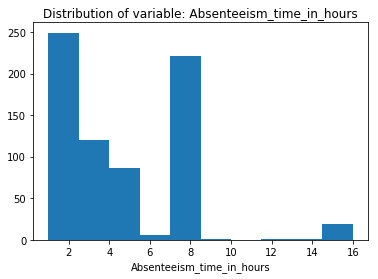
****

****

****

****

****

****

Thus, the distributions are not gaussian and hence we have to normalize the data to bring them in a common range.

In [78]:

#Normalization on dataset

for i in df.columns:

df.loc[:,i] = (df.loc[:,i] - np.min(df.loc[:,i]))/(np.max(df.loc[:,i])-np.min(df.loc[:,i]))

In [79]:

df.head()

Out[79]:

|  | **ID** | **Transportation\_expense** | **Distance\_from\_Residence\_to\_Work** | **Service\_time** | **Age** | **Work\_load\_Average/day** | **Hit\_target** | **Weight** | **Height** | **Body\_mass\_index** | **...** | **Month\_of\_absence\_3** | **Month\_of\_absence\_4** | **Month\_of\_absence\_5** | **Month\_of\_absence\_6** | **Month\_of\_absence\_7** | **Month\_of\_absence\_8** | **Month\_of\_absence\_9** | **Month\_of\_absence\_10** | **Month\_of\_absence\_11** | **Month\_of\_absence\_12** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.285714 | 0.657692 | 0.659574 | 0.521739 | 0.260870 | 0.244925 | 0.769231 | 0.653846 | 0.7 | 0.578947 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.057143 | 0.234615 | 0.978723 | 0.739130 | 0.478261 | 0.244925 | 0.769231 | 0.634615 | 0.5 | 0.631579 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.171429 | 0.619231 | 0.000000 | 0.565217 | 0.521739 | 0.244925 | 0.769231 | 0.230769 | 0.3 | 0.263158 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 0.285714 | 0.657692 | 0.659574 | 0.521739 | 0.260870 | 0.244925 | 0.769231 | 0.653846 | 0.7 | 0.578947 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **5** | 0.057143 | 0.234615 | 0.978723 | 0.739130 | 0.478261 | 0.244925 | 0.769231 | 0.634615 | 0.5 | 0.631579 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 68 columns

In [80]:

df\_normalized = df.copy()

#### Applying Machine Learning algorithms[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Applying-Machine-Learning-algorithms)

We can split the data into train and test datasets. Training dataset is used for building training model and test dataset is for validating our model. This is done to understand the robustness, accuracy and performance of the model built.

In [81]:

df.iloc[:,10].head()

Out[81]:

0 0.200000

2 0.066667

3 0.200000

4 0.066667

5 0.066667

Name: Absenteeism\_time\_in\_hours, dtype: float64

In [82]:

#Obtain test and training dataset

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(df.iloc[:, df.columns != 'Absenteeism\_time\_in\_hours'],df.iloc[:,10],test\_size=0.2,random\_state = 42)

In [83]:

#Dimensions of training dataset

print(X\_train.shape)

print(Y\_train.shape)

(563, 67)

(563,)

In [84]:

#Dimensions of test dataset

print(X\_test.shape)

print(Y\_test.shape)

(141, 67)

(141,)

#### Linear Regression[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Linear-Regression)

In [85]:

#Building linear regression model on training dataset

LR\_model = LinearRegression().fit(X\_train, Y\_train)

In [86]:

#Predict target values on training and test dataset

Ypred\_train\_LR = LR\_model.predict(X\_train)

Ypred\_test\_LR = LR\_model.predict(X\_test)

In [87]:

#Calculating RMSE values for training and test dataset

RMSE\_train\_LR = np.sqrt(mean\_squared\_error(Y\_train, Ypred\_train\_LR))

RMSE\_test\_LR = np.sqrt(mean\_squared\_error(Y\_test, Ypred\_test\_LR))

In [88]:

#RMSE calculation for training data

print("RMSE value of training data is:")

print(RMSE\_train\_LR)

#RMSE calculation for test data

print("RMSE value of test data is:")

print(RMSE\_test\_LR)

#R^2 calculation for train data

print("R2 score of training data is:")

print(r2\_score(Y\_train, Ypred\_train\_LR))

#R^2 calculation for test data

print("R2 score of test data is:")

print(r2\_score(Y\_test, Ypred\_test\_LR))

RMSE value of training data is:

0.15426840125452113

RMSE value of test data is:

0.20847435587305038

R2 score of training data is:

0.4674263799330639

R2 score of test data is:

0.32782913521020485

#### Decision Tree model[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Decision-Tree-model)

In [89]:

#Building decision tree model on training dataset

DT\_model = DecisionTreeRegressor(max\_depth=2,random\_state = 42).fit(X\_train,Y\_train)

In [90]:

DT\_model

Out[90]:

DecisionTreeRegressor(criterion='mse', max\_depth=2, max\_features=None,

max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=42, splitter='best')

In [91]:

#Predict target values on training and test dataset

Ypred\_train\_DT = DT\_model.predict(X\_train)

Ypred\_test\_DT = DT\_model.predict(X\_test)

In [92]:

#Calculating RMSE values for training and test dataset

RMSE\_train\_DT = np.sqrt(mean\_squared\_error(Y\_train, Ypred\_train\_DT))

RMSE\_test\_DT = np.sqrt(mean\_squared\_error(Y\_test, Ypred\_test\_DT))

In [93]:

#RMSE calculation for training data

print("RMSE value of training data is:")

print(RMSE\_train\_DT)

#RMSE calculation for test data

print("RMSE value of test data is:")

print(RMSE\_test\_DT)

#R^2 calculation for train data

print("R2 score of training data is:")

print(r2\_score(Y\_train, Ypred\_train\_DT))

#R^2 calculation for test data

print("R2 score of test data is:")

print(r2\_score(Y\_test, Ypred\_test\_DT))

RMSE value of training data is:

0.1915355006658154

RMSE value of test data is:

0.23766373118278583

R2 score of training data is:

0.17903573087970592

R2 score of test data is:

0.12642494390809056

#### Random Forest model[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Random-Forest-model)

In [94]:

#Building random forest model on training dataset

RF\_model = RandomForestRegressor(n\_estimators = 200,random\_state=42).fit(X\_train,Y\_train)

In [95]:

#Predict target values on training and test dataset

Ypred\_train\_RF = RF\_model.predict(X\_train)

Ypred\_test\_RF = RF\_model.predict(X\_test)

In [96]:

#calculating RMSE for training and test dataset

RMSE\_train\_RF = np.sqrt(mean\_squared\_error(Y\_train, Ypred\_train\_RF))

RMSE\_test\_RF = np.sqrt(mean\_squared\_error(Y\_test, Ypred\_test\_RF))

In [97]:

#RMSE calculation for training data

print("RMSE value of training data is:")

print(RMSE\_train\_RF)

#RMSE calculation for test data

print("RMSE value of test data is:")

print(RMSE\_test\_RF)

#R^2 calculation for training data

print("R2 score of training data is:")

print(r2\_score(Y\_train, Ypred\_train\_RF))

#R^2 calculation for test data

print("R2 score of test data is:")

print(r2\_score(Y\_test, Ypred\_test\_RF))

RMSE value of training data is:

0.07031719189101368

RMSE value of test data is:

0.19711876793050032

R2 score of training data is:

0.8893506846553185

R2 score of test data is:

0.39906104877806736

#### Gradient boosting model[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Gradient-boosting-model)

In [98]:

#Build Gradient Boosting Model on trainig dataset

GBM\_model = GradientBoostingRegressor(random\_state=42).fit(X\_train,Y\_train)

In [99]:

#Predict target values on training and test dataset

Ypred\_train\_GBM = GBM\_model.predict(X\_train)

Ypred\_test\_GBM = GBM\_model.predict(X\_test)

In [100]:

#calculating RMSE for training and test dataset

RMSE\_train\_GBM = np.sqrt(mean\_squared\_error(Y\_train, Ypred\_train\_GBM))

RMSE\_test\_GBM = np.sqrt(mean\_squared\_error(Y\_test, Ypred\_test\_GBM))

In [101]:

#RMSE calculation for training data

print("RMSE value of training data is:")

print(RMSE\_train\_GBM)

#RMSE calculation for test data

print("RMSE value of test data is:")

print(RMSE\_test\_GBM)

#R^2 calculation for train data

print("R2 score of training data is:")

print(r2\_score(Y\_train, Ypred\_train\_GBM))

#R^2 calculation for test data

print("R2 score of test data is:")

print(r2\_score(Y\_test, Ypred\_test\_GBM))

RMSE value of training data is:

0.12735078584139936

RMSE value of test data is:

0.21084018397799467

R2 score of training data is:

0.6370649166304359

R2 score of test data is:

0.3124865864780011

We can conclude that although Random Forest Model yields maximum R^2 value, there is very much difference in the R2 values of training and test dataset, thus there is a chance that it might nt generalize well. Hence, we can predict test dataset using GBM Model.

In [102]:

#importance = RF\_model.feature\_importances\_

#importance = pd.DataFrame(importance, index=X\_train.columns, columns=["Importance"])

#importance = importance.sort\_values('Importance',ascending=False)

#importance

#### Prediction of absent hours for the year 2011:[¶](http://localhost:8888/nbconvert/html/Employee%20absenteeism.ipynb?download=false#Prediction-of-absent-hours-for-the-year-2011:)

Thus, from the data given it is clear that it is past data. Now, we are expected to make project losses every month in 2011 if same trend of absenteeism continues. Thus, assuming that the given data is from year 2010, we can frame the sample dataset for 2011 by increasing the age and service time of the employees and keeping the rest of the data constant assuming there will not be much change to it.

In [103]:

df\_data\_2011 = df\_selected.drop(['Absenteeism\_time\_in\_hours'],axis=1).copy()

In [104]:

#Increment Age and Service time in the dataset

for i in num\_cols:

if i == 'Age':

df\_data\_2011[i] = df\_data\_2011[i] + 1.0

elif i == 'Service\_time':

df\_data\_2011[i] = df\_data\_2011[i] + 1.0

Since the data is preprocessed in dataset for 2011, let's continue with the below steps:

In [105]:

num\_cols\_2011 = df\_data\_2011.\_get\_numeric\_data().columns

cat\_cols\_2011 = ['Son',

'Disciplinary\_failure',

'Social\_drinker',

'Pet',

'Reason\_for\_absence',

'Education',

'Day\_of\_the\_week',

'Month\_of\_absence']

In [106]:

#Hot encoding of categorical features

df\_sample\_2011 = df\_data\_2011[num\_cols\_2011]

df\_dummies\_2011 = pd.get\_dummies(df\_data\_2011[cat\_cols\_2011],drop\_first=True)

df\_data\_2011 = pd.concat([df\_sample\_2011, df\_dummies\_2011], axis=1)

df\_data\_2011.reset\_index(drop=True)

Out[106]:

|  | **ID** | **Transportation\_expense** | **Distance\_from\_Residence\_to\_Work** | **Service\_time** | **Age** | **Work\_load\_Average/day** | **Hit\_target** | **Weight** | **Height** | **Body\_mass\_index** | **...** | **Month\_of\_absence\_3** | **Month\_of\_absence\_4** | **Month\_of\_absence\_5** | **Month\_of\_absence\_6** | **Month\_of\_absence\_7** | **Month\_of\_absence\_8** | **Month\_of\_absence\_9** | **Month\_of\_absence\_10** | **Month\_of\_absence\_11** | **Month\_of\_absence\_12** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 11 | 289.0 | 36.0 | 14.0 | 34.0 | 239554.0 | 97.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **1** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **2** | 7 | 279.0 | 5.0 | 15.0 | 40.0 | 239554.0 | 97.0 | 68.0 | 168.0 | 24.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **3** | 11 | 289.0 | 36.0 | 14.0 | 34.0 | 239554.0 | 97.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **4** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **5** | 10 | 354.0 | 52.0 | 4.0 | 29.0 | 239554.0 | 97.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **6** | 20 | 260.0 | 50.0 | 12.0 | 37.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **7** | 14 | 155.0 | 12.0 | 15.0 | 35.0 | 239554.0 | 97.0 | 95.0 | 169.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **8** | 1 | 235.0 | 11.0 | 15.0 | 38.0 | 239554.0 | 97.0 | 88.0 | 172.0 | 29.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **9** | 20 | 260.0 | 50.0 | 12.0 | 37.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **10** | 20 | 260.0 | 50.0 | 12.0 | 37.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **11** | 20 | 260.0 | 50.0 | 12.0 | 37.0 | 239554.0 | 97.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **12** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **13** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **14** | 24 | 246.0 | 25.0 | 17.0 | 42.0 | 239554.0 | 97.0 | 67.0 | 170.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **15** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **16** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 239554.0 | 97.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **17** | 6 | 189.0 | 29.0 | 14.0 | 34.0 | 239554.0 | 97.0 | 69.0 | 167.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **18** | 33 | 248.0 | 25.0 | 15.0 | 48.0 | 205917.0 | 92.0 | 86.0 | 165.0 | 32.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **19** | 18 | 330.0 | 16.0 | 5.0 | 29.0 | 205917.0 | 92.0 | 84.0 | 172.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **20** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 205917.0 | 92.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **21** | 10 | 361.0 | 52.0 | 4.0 | 29.0 | 205917.0 | 92.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **22** | 20 | 260.0 | 50.0 | 12.0 | 37.0 | 205917.0 | 92.0 | 65.0 | 168.0 | 23.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **23** | 11 | 289.0 | 36.0 | 14.0 | 34.0 | 205917.0 | 92.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **24** | 10 | 361.0 | 52.0 | 4.0 | 29.0 | 205917.0 | 92.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **25** | 11 | 289.0 | 36.0 | 14.0 | 34.0 | 205917.0 | 92.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **26** | 30 | 157.0 | 27.0 | 7.0 | 30.0 | 205917.0 | 92.0 | 75.0 | 169.0 | 26.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **27** | 11 | 289.0 | 36.0 | 14.0 | 34.0 | 205917.0 | 92.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **28** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 205917.0 | 92.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **29** | 3 | 179.0 | 51.0 | 19.0 | 39.0 | 205917.0 | 92.0 | 89.0 | 170.0 | 31.0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **674** | 28 | 225.0 | 26.0 | 10.0 | 29.0 | 237656.0 | 99.0 | 69.0 | 169.0 | 24.0 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **675** | 16 | 118.0 | 15.0 | 25.0 | 47.0 | 275089.0 | 96.0 | 75.0 | 175.0 | 25.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **676** | 22 | 179.0 | 26.0 | 10.0 | 31.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **677** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **678** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **679** | 23 | 378.0 | 49.0 | 12.0 | 37.0 | 275089.0 | 96.0 | 65.0 | 174.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **680** | 36 | 118.0 | 13.0 | 19.0 | 51.0 | 275089.0 | 96.0 | 98.0 | 172.0 | 31.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **681** | 12 | 233.0 | 51.0 | 2.0 | 32.0 | 275089.0 | 96.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **682** | 22 | 179.0 | 26.0 | 10.0 | 31.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **683** | 36 | 118.0 | 13.0 | 19.0 | 51.0 | 275089.0 | 96.0 | 98.0 | 172.0 | 31.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **684** | 22 | 179.0 | 26.0 | 10.0 | 31.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **685** | 15 | 291.0 | 31.0 | 13.0 | 41.0 | 275089.0 | 96.0 | 73.0 | 171.0 | 25.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **686** | 22 | 179.0 | 26.0 | 10.0 | 31.0 | 275089.0 | 96.0 | 56.0 | 171.0 | 19.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **687** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **688** | 12 | 233.0 | 51.0 | 2.0 | 32.0 | 275089.0 | 96.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **689** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **690** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 275089.0 | 96.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **691** | 12 | 233.0 | 51.0 | 2.0 | 32.0 | 275089.0 | 96.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **692** | 5 | 235.0 | 20.0 | 14.0 | 44.0 | 264604.0 | 93.0 | 106.0 | 167.0 | 38.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **693** | 12 | 233.0 | 51.0 | 2.0 | 32.0 | 264604.0 | 93.0 | 68.0 | 169.0 | 21.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **694** | 9 | 228.0 | 14.0 | 17.0 | 41.0 | 264604.0 | 93.0 | 65.0 | 172.0 | 22.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **695** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 264604.0 | 93.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **696** | 9 | 228.0 | 14.0 | 17.0 | 41.0 | 264604.0 | 93.0 | 65.0 | 172.0 | 22.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **697** | 6 | 189.0 | 29.0 | 14.0 | 34.0 | 264604.0 | 93.0 | 69.0 | 167.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **698** | 34 | 118.0 | 10.0 | 11.0 | 38.0 | 264604.0 | 93.0 | 83.0 | 172.0 | 28.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **699** | 10 | 361.0 | 52.0 | 4.0 | 29.0 | 264604.0 | 93.0 | 80.0 | 172.0 | 27.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **700** | 28 | 225.0 | 26.0 | 10.0 | 29.0 | 264604.0 | 93.0 | 69.0 | 169.0 | 24.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **701** | 13 | 369.0 | 17.0 | 13.0 | 32.0 | 264604.0 | 93.0 | 70.0 | 169.0 | 25.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **702** | 11 | 289.0 | 36.0 | 14.0 | 34.0 | 264604.0 | 93.0 | 90.0 | 172.0 | 30.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **703** | 1 | 235.0 | 11.0 | 15.0 | 38.0 | 264604.0 | 93.0 | 88.0 | 172.0 | 29.0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

704 rows × 67 columns

In [107]:

#Normalization on dataset

for i in df\_data\_2011.columns:

df\_data\_2011.loc[:,i] = (df\_data\_2011.loc[:,i] - np.min(df\_data\_2011.loc[:,i]))/(np.max(df\_data\_2011.loc[:,i])-np.min(df\_data\_2011.loc[:,i]))

In [108]:

df\_data\_2011.head()

Out[108]:

|  | **ID** | **Transportation\_expense** | **Distance\_from\_Residence\_to\_Work** | **Service\_time** | **Age** | **Work\_load\_Average/day** | **Hit\_target** | **Weight** | **Height** | **Body\_mass\_index** | **...** | **Month\_of\_absence\_3** | **Month\_of\_absence\_4** | **Month\_of\_absence\_5** | **Month\_of\_absence\_6** | **Month\_of\_absence\_7** | **Month\_of\_absence\_8** | **Month\_of\_absence\_9** | **Month\_of\_absence\_10** | **Month\_of\_absence\_11** | **Month\_of\_absence\_12** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.285714 | 0.657692 | 0.659574 | 0.521739 | 0.260870 | 0.244925 | 0.769231 | 0.653846 | 0.7 | 0.578947 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.057143 | 0.234615 | 0.978723 | 0.739130 | 0.478261 | 0.244925 | 0.769231 | 0.634615 | 0.5 | 0.631579 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.171429 | 0.619231 | 0.000000 | 0.565217 | 0.521739 | 0.244925 | 0.769231 | 0.230769 | 0.3 | 0.263158 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 0.285714 | 0.657692 | 0.659574 | 0.521739 | 0.260870 | 0.244925 | 0.769231 | 0.653846 | 0.7 | 0.578947 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **5** | 0.057143 | 0.234615 | 0.978723 | 0.739130 | 0.478261 | 0.244925 | 0.769231 | 0.634615 | 0.5 | 0.631579 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 67 columns

In [109]:

#Predict using Random Forest model

Ypred\_2011= GBM\_model.predict(df\_data\_2011)

In [110]:

#Adding predicted absent hours to the test dataset

df\_data\_2011['Predicted\_Absent\_hours'] = Ypred\_2011

In [111]:

#Summary of the predicted hours of absent

df\_data\_2011['Predicted\_Absent\_hours'].describe()

Out[111]:

count 704.000000

mean 0.238070

std 0.135608

min 0.030604

25% 0.116654

50% 0.208680

75% 0.356300

max 0.897303

Name: Predicted\_Absent\_hours, dtype: float64

In [112]:

df\_normalized['Absenteeism\_time\_in\_hours'].describe()

Out[112]:

count 704.000000

mean 0.242330

std 0.221025

min 0.000000

25% 0.066667

50% 0.133333

75% 0.466667

max 1.000000

Name: Absenteeism\_time\_in\_hours, dtype: float64

In [113]:

df\_data\_2011['Predicted\_Absent\_hours'].head()

Out[113]:

0 0.368500

2 0.119028

3 0.374129

4 0.178333

5 0.112028

Name: Predicted\_Absent\_hours, dtype: float64