

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Technology Degree in Information
Technology

By
CHERUKU NITHESH REDDY
(Reg.No:40120021)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **CHERUKU NITHESH REDDY (REG.NO:40120021)** who carried out the project entitled "**PREDICTING WORKING PATTERN OF EMPLOYEE**" under my supervision from January 2023 to April 2023

Internal Guide:

Dr.R Jeberson Retna Raj.M.Tech., Ph.D

Head of the Department:

Dr. R.Subhashini,M.E., Ph.D

Submitted for Viva-voce Examination held on 28/04/23

Internal Examiner

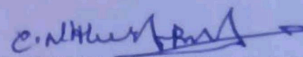
External Examiner

DECLARATION

I, CHERUKU NITHESH REDDY hereby declare that the Project Report entitled "**PREDICTING WORKING PATTERN OF EMPLOYEE**" Done by me under the guidance of **Dr.r Jeberson Retna Raj, M.Tech., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 28/04/23

PLACE: Chennai


SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. R. Subhashini, M.E., Ph.D., Heads of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Jeberson Retna Raj, M.Tech., Ph.D.**, his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



Elite NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to
CHERUKU NITHESH REDDY
for successfully completing the course



Foundation of Cloud IoT Edge ML

with a consolidated score of **87** %

Online Assignments	23.33/25	Proctored Exam	63.33/75
--------------------	----------	----------------	----------

Total number of candidates certified in this course: **1649**

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Feb-Apr 2023

(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur



Indian Institute of Technology Kanpur



Roll No: NPTEL23CS65533364924

To validate the certificate



No. of credits recommended: 2 or 3

ABSTRACT

The Main Objective of this project is to predict the working pattern of employee time in hours. Absenteeism is a serious workplace problem and an expensive occurrence for both employers and employees seemingly unpredictable in nature. A satisfactory level of attendance by employees at work is necessary to allow the achievement of objectives and targets by a department. Absenteeism results in financial losses, because of the resultant reduction in productivity and the cost of sick leave benefits or others are paid as wages for no work. Absenteeism reduces the satisfaction level of the employee and makes him unsecured about his job in the organization. Most researches had concluded that absence is a complex variable and that it is influenced by multiple causes, both personal and organizational. Based on that absence of the employees from work leads to deterioration and reduce production in the institutions which leads to heavy losses.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	iv
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
	1.1 PROBLEM STATEMENT	1
	1.2 DATA	1
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	5
	2.1 AIM	5
	2.2 Scope	5

3.	EXPRIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	6
	3.1 IMPORTED LIBRARIES	7
	3.2 PRE-PROCESSING	9
	3.2.1 MISSING VALUE ANALYSIS	11
	3.2.2 FEATURE SELECTION	13
	3.2.3 FEATURE SCALING	15
	3.3 MACHINE LEARNING TECHNIQUES USED	16
	3.3.1 DECISION TREE	17
	3.3.2 LINEAR REGRESSION	
4	RESULTS, AND DISCUSSION, PERFORMANCE ANALYSIS	18
	4.1 RESULTS AND PERFORMANCE ANALYSIS	18
5	SUMMARY AND CONCLUSIONS	19
	5.1 SUMMARY	19
	5.2 CONCLUSION	19
	REFERENCES	20
	SCREEN SHOTS AND SOURCE CODE	21-50

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME	PAGE NUMBER
FIGURE : 1.2	DATA SET	4
FIGURE: - 3	IDEATION MAP	6
FIGURE: - 3.1.1	PYTHON LIBRARIES	7
FIGURE: -3.2.1	Missing Value Table	10
FIGURE: - 3.2.2	CORELATION HEAT MAP	12
FIGURE: - 3.2.3	FEATURE SCALING GRAPHS	14
FIGURE: - 3.3	TECHNIQUES OF MACHINE LEARNING	15
FIGURE: 3.3.1	DECISION TREE	16
FIGURE: - 3.3.2	LINEAR REGRESSION	17

LIST OF ABBREVTIONS

ABBREVIATION	EXPANSION
np	Numpy
pd	pandas
sns	seaborn
sklearn	scikit learn
rmse	root mean square error

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Predict the working pattern of employee is a major problem faced by every employer which eventually lead to the backlogs, piling of the work, delay in deploying the project and can have a major effect on company finances.

1.2 DATA

The task is to build a regression model which will predict the number of absent hours in relation to the employee depend upon the various factor. Given below is the sample of the data set which is used to predict the number of absent hours related to the employee. There are 21 variables in our data in which 20 are independent variables and 1 (Absenteeism time in hours) is dependent variable. Since our target variable is continuous in nature.

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:

Certain infectious and parasitic diseases

- Neoplasms
- Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism.
- Endocrine, nutritional and metabolic diseases

- Mental and behavioural disorders
- Diseases of the nervous system
- Diseases of the eye and adnexa
- Diseases of the ear and mastoid process
- Diseases of the circulatory system
- Diseases of the respiratory system
- Diseases of the digestive system
- Diseases of the skin and subcutaneous tissue
- Diseases of the musculoskeletal system and connective Tissue.
- Diseases of the genitourinary system
- Pregnancy, childbirth and the puerperium
- Certain conditions originating in the perinatal period
- Congenital malformations, deformations and chromosomal Abnormalities.
- Symptoms, signs and abnormal clinical and laboratory.
- findings, not elsewhere classified.
- Injury, poisoning and certain other consequences of
- findings, not elsewhere classified.
- External causes of morbidity and mortality
- 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

6. Transportation expense

7.. Distance from Residence to Work (kilometers)

8. Service time

9. Age

10. Work load Average/day

11. Hit target

12. Disciplinary failure (yes=1; no=0)

13. Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))

14. Son (number of children)

15. Social drinker (yes=1; no=0)

16. Social smoker (yes=1; no=0)

17. Pet (number of pet)

18. Weight

19. Height

20. Body mass index

21. Absenteeism time in hours (target)

DATA SET INFORMATION

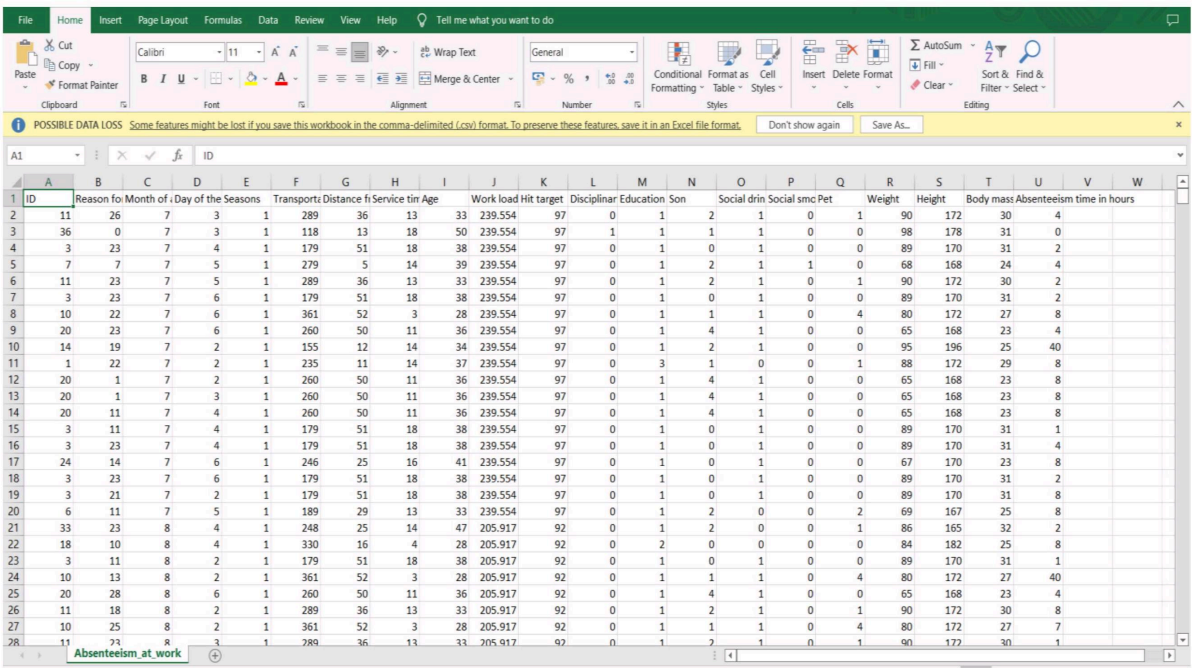


FIGURE : 1.2 DATA SET

The given data set contains 740 rows and 21 columns.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM:

The aim of this project is to find predict the working pattern of employee time in hours with given dataset to rectify the issue which eventually leads toward the absence of an employee and provide a proper solution to reduce the absenteeism in the company.

2.2 SCOPE:

Predict the working pattern of employee is not a new problem in the Indian organizations. It is a worldwide fact. Absenteeism is one of the common factors, which affects both the productivity of the employees, and also ruins the image of the Government among the public. So controlling and preventing absenteeism become the aim of the whole nation. The focus of the study is on employee's absenteeism at public organizations.

CHAPTER 3

EXPRIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED

Before feeding the data to the model we need to clean the data and convert it to a proper format. It is the most crucial part of data science project we spend almost 80% of time in it. For implementation of this project, we're following the traditional CRISP-DM Process and has been divided into three major sections. The first section involves the data preparations. In the second section some exploratory data analysis is conducted and finally third section provides solution to a predictive model, and fits the model.

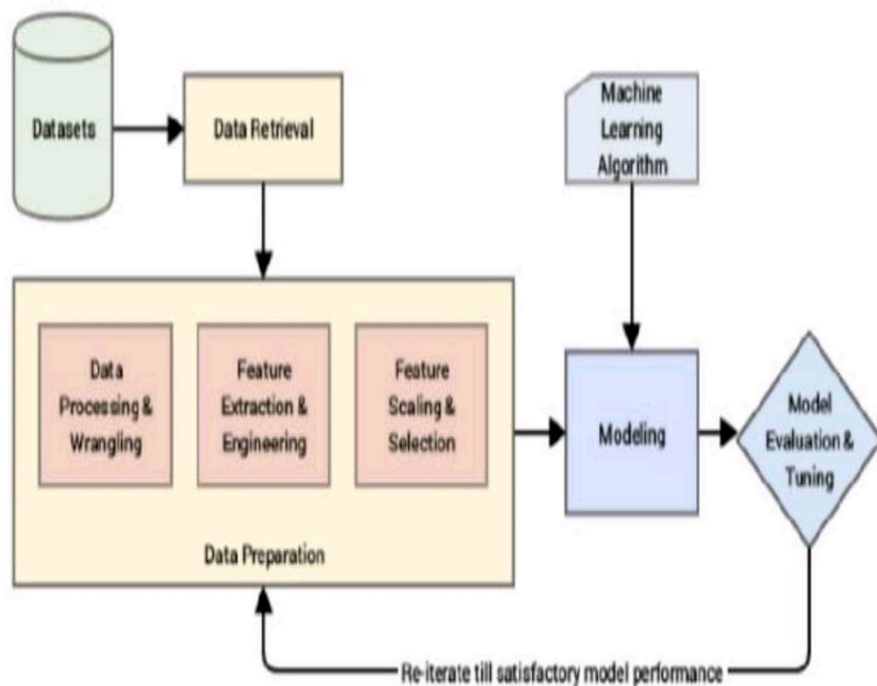


FIGURE: - 3 IDEATION MAP

3.1 IMPORTED LIBRARIES:

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let's look at the libraries that came in handy for mammographic dataset.

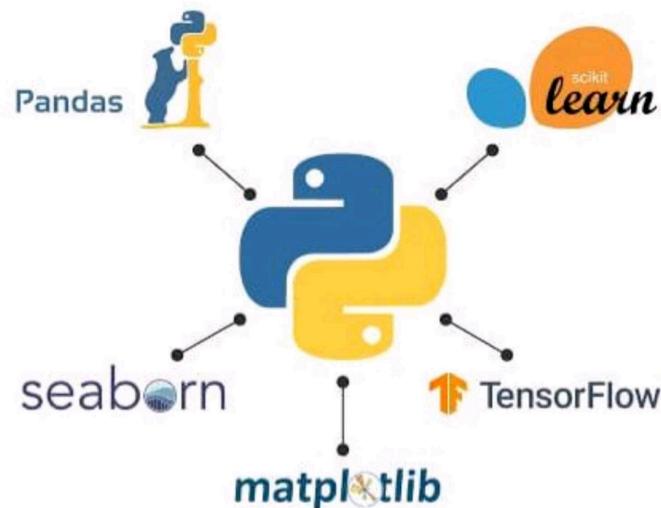


FIGURE: - 3.1.1 PYTHON LIBRARIES

LIBRARIES USED:

1.Numpy stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of countless of routines for processing those arrays. Using this mathematical and logical operations on arrays can be performed. The 21 differences in using Numpy from pandas is, it works on numerical data whereas pandas on tabular data

2.Pandas is a widely-used data analysis and manipulation library for python. It provides a lot of functions and methods that expedite the data analysis and preprocessing steps. IT also provides fast, flexible and expressive data structures working with relational or labelled or both easy and intuitive. Considered as

fundamental high-level building block in performing practical, real-world data analysis in python. Has powerful tools like Data Frame and Series for analysing

3. Matplotlib.pyplot is a state-based interface to matplotlib. It provides a MATLAB-like way of plotting. It makes changes to figures. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

4. sklearn stands for Scikit-learn, a machine learning library. It is imported for various classification, regression and clustering algorithms including k-means, random forest, support vector machines, gradient boosting and DBSCAN. It is designed using libraries Numpy and SciPy. From the sklearn library and from the tree inside the library Decision Tree Classifier. It is a class capable of performing multi-class classifier on a dataset. When compared with other classifiers, Decision Tree Classifier takes input as two arrays: an array X, a sparse or dense, of shape (n_samples, 22 n_features) holding training samples and an array Y of integer values, shape (n_samples), holding class labels for training sample. From sklearn another one called model_selection for training and testing the model imports train_test_split. It is a method setting a blueprint to analyze data and the using it to measure new data. Selecting a proper model allows to generate accurate results while making prediction. For proceeding, we need to train the model by using a specific dataset and test the model against another dataset. By default, sklearn train_test_split will make random partitions for two subsets. We can also specify a random state for the operation. First, we need to split the dataset and then allocate the size for train and test. For this mammographic dataset we need train size as 80% (0.80) and test size as 20% (0.20) with the random state of 100.

5. Seaborn is a library built on top of matplotlib. It used for data visualization and exploratory data analysis. They work easily with dataframes and panda's library. The graphs created can also be customized easily. It provides default styles and colour palettes to make statistical plots more attractive.

3.2 PRE - PROCESSING

The output of the Machine Learning data is fully depended upon the data feed in the algorithm, ML algorithm does not generate the proper outcome on raw data. So it's important to transform the data so that the model accuracy of the model can be increased. In other words, we must apply some cleaning and processing for a better outcome with respect to the Machine learning algorithm. Some Machine learning algorithm required well processed data like Decision Tree and random forest does not take null value. So, it is important to process the data before feeding it in the respective machine learning algorithm.

3.2.1 Missing Value Analysis

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data. There are several options to handle missing value, but it mainly depends upon the nature of the data set, which missing analytics produce the optimum solution. Missing of data can occur due to nonresponse occur for example privacy concern.

Moreover, if the value is missing completely at random, the data sample still represents the population but if the value is missing in some pattern, analysis produces bias output. Then missing values are filled with the mean of particular column of the dataset. Once we are good with the data with no null values and missing values and no string values then we can split the data into two parts. If a column has more than 30% of data as missing value either we ignore the entire column or we ignore those observations.

In the given data set the maximum percentage of missing value is 0.0%.


```
#Check if any missing values
df.isnull().sum()
```

```
: 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
```

FIGURE: -3.2.1 Missing Value Table

3.2.2 Feature Selection

Feature selection is another pre-processing technique which decreases the load over machine learning algorithm checking the correlation between other feature and check which feature is highly correlated to another feature. If two feature carries the same data, this will create a bias output. Feature selection is a process where you can select those features which contribute most to the prediction output. Moreover, having a relevant feature in the data set can decrease the accuracy of the model.

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

Figure represents the correlation heatmap with respect to the data. Darker color represents the positive correlation and lighter colour scheme represent the negative correlation with respect to other features. Body mass index is highly positively related towards weight feature in the dataset which is 0.9. Moreover, no other feature is correlated with respect to other feature. Moreover, body mass index is the ratio of weight with respect to the square of height, so by dropping the two features from the data set will reduce the time complexity with the same efficiency.

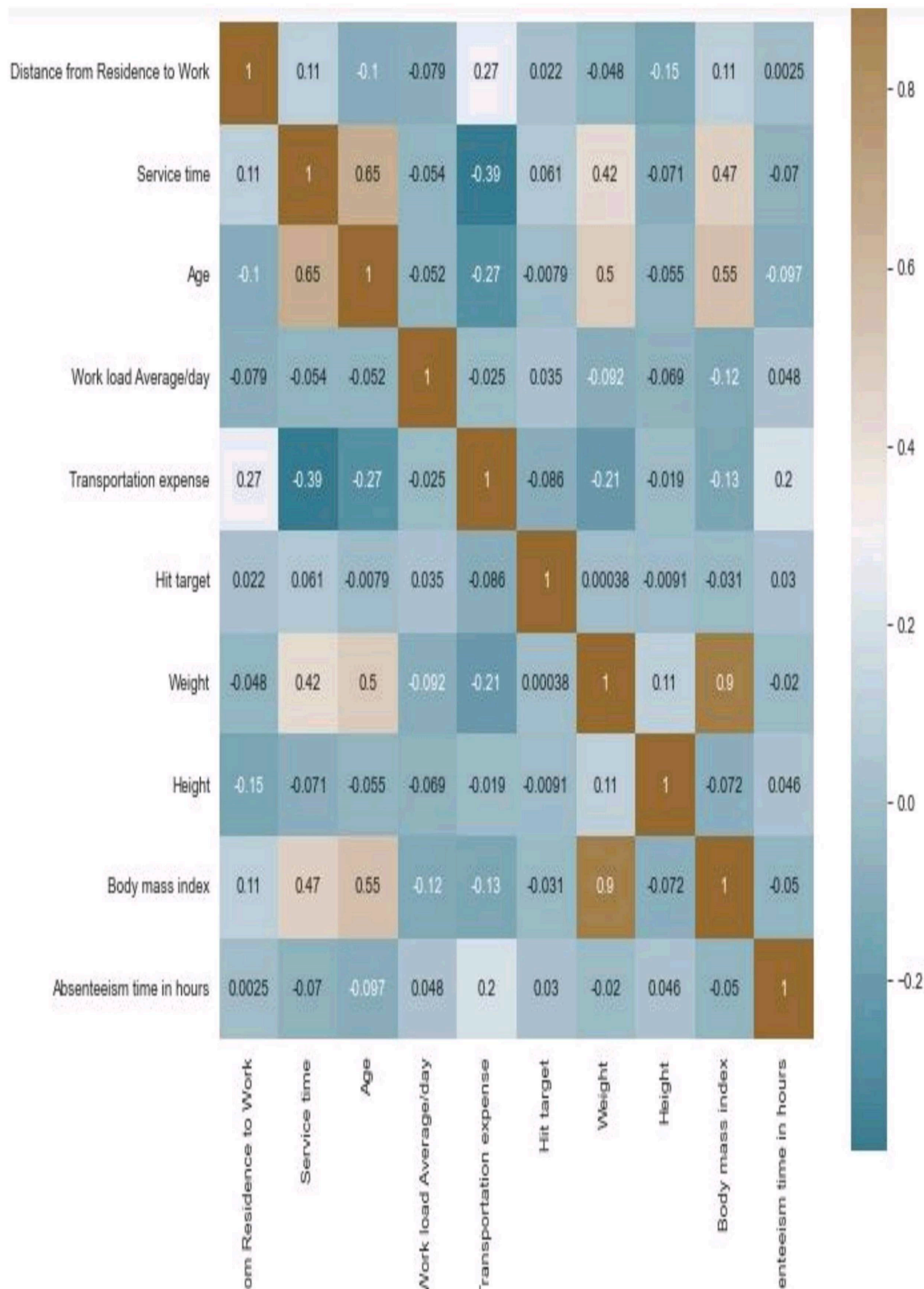


FIGURE:- 3.2.2 CO RELATION HEAT MAP

3.2.3 Feature Scaling

Feature scaling is a method to standardize the variable or the feature of the data set, it also knows as standardization of a data set. Some feature has magnitudes, unit or range and normalization should be performed when the scaling of the feature is irrelevant or have a high magnitude which will lead to the bases in the machine learning algorithm which will lean towards the higher magnitude dataset. Z-scoring/standardization is one technique in which the value of a feature is subtracted with the mean of the feature and their subtraction is divided by the standard deviation of the feature which will also set the feature in normalized bell shape form whereas another method involves the value of a feature is subtracted by the minimum value of the respective feature which is divided by the subtraction of the minimum value and maximum value of the respective feature. In our dataset, some feature has irrelevant scaling which will affect the machine learning algorithm. A feature like 'Transportation expense', 'Distance from Residence to Work', 'Service time', 'Workload Average/day ', 'Hit target', 'Height' and 'Weight' is scaled by applying the min-max scaling technique.

MIN MAX SCALING: -

$$\text{Min-Max scaling: } Y_i = [X_i - \min(X)] / [\max(X) - \min(X)]$$

Y_i is our new value

X_i is original cell value

$\min[X]$ is the minimum value of column

$\max[X]$ is the maximum value of column

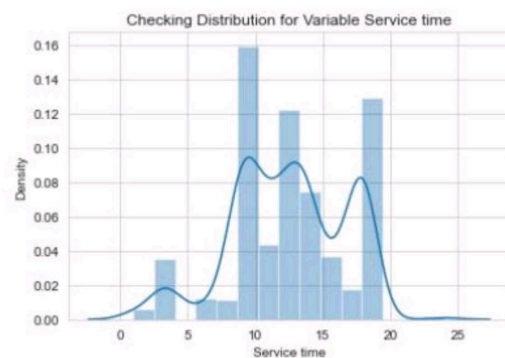
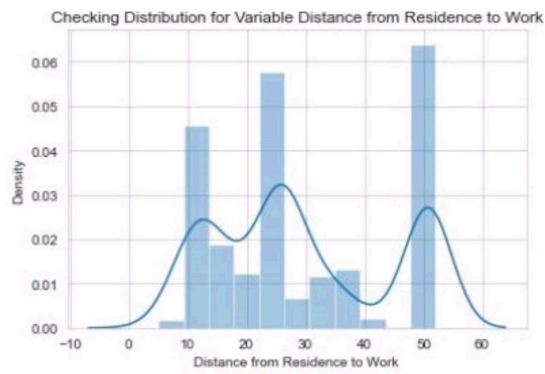
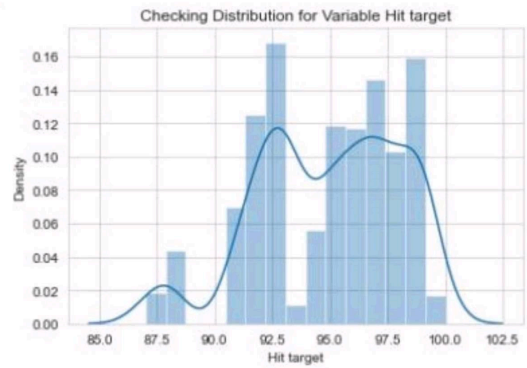
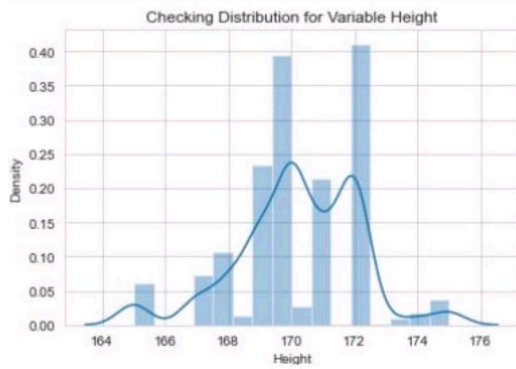
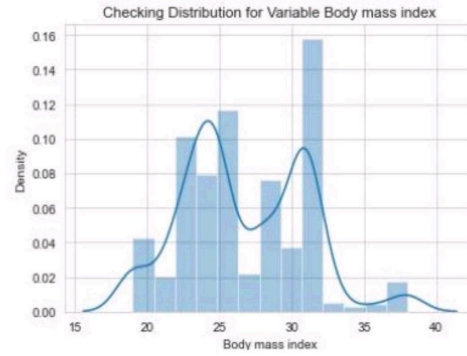
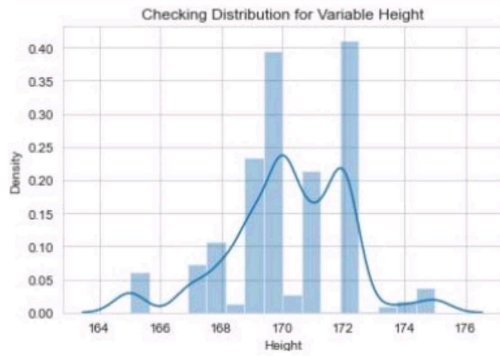
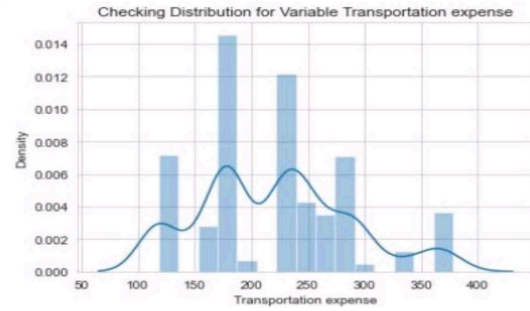
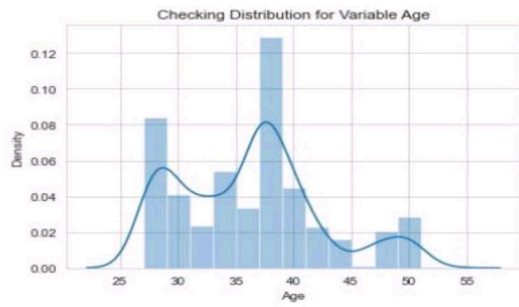


FIGURE: - 3.2.3 FEATURE SCALING GRAPH

3.3 MACHINE LEARNING TECHINQUES USED

In this work the following algorithms were used: Decision Tree, Linear Regression. The full dataset was divided into the training and test subsets. 80% of samples were used for training and 20% for testing. Division of all available samples into the training and test subsets is crucial for a correct methodology. The aim of all ML or DL methods is to achieve a "generalization" ability

Machine Learning Techniques (like Regression, Classification, Clustering, Anomaly detection, etc.) are used to build the training data or a mathematical model using certain algorithms based upon the computations statistic to make prediction without the need of programming, as these techniques are influential in making the system futuristic, models and promotes automation of things with reduced cost and manpower.

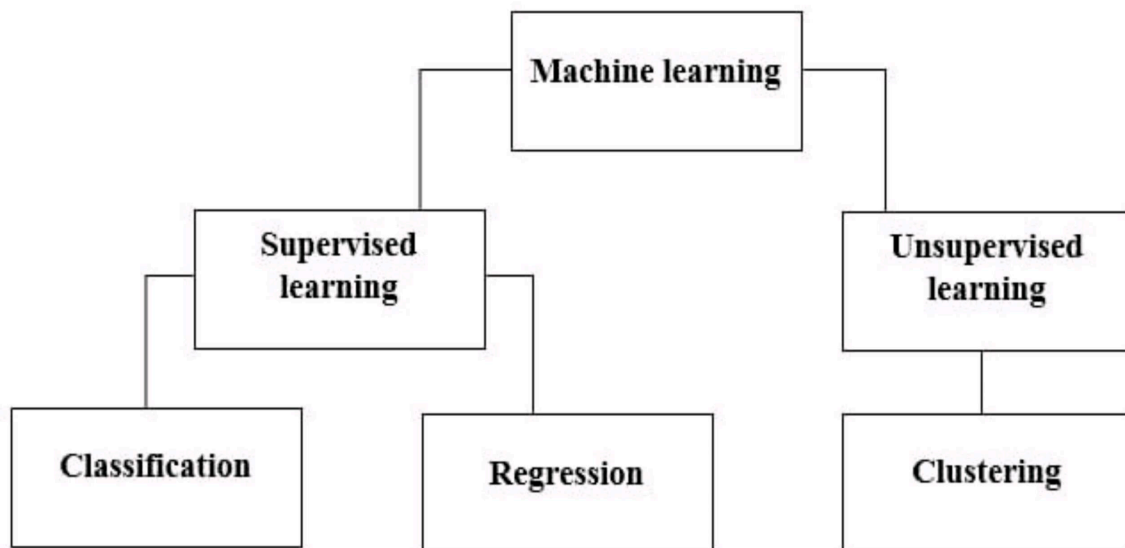


FIGURE :- 3.3 TECHNIQUES OF MACHINE LEARNING

ALGORITHM USED

3.3.1 DECISION TREE

Decision Tree is the most powerful and popular tool for classification and Fig 1.3.3- Decision Tree scatter plot. prediction A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers 17 or values. here it is important to continuous values are predicted with the help of a decision tree regression

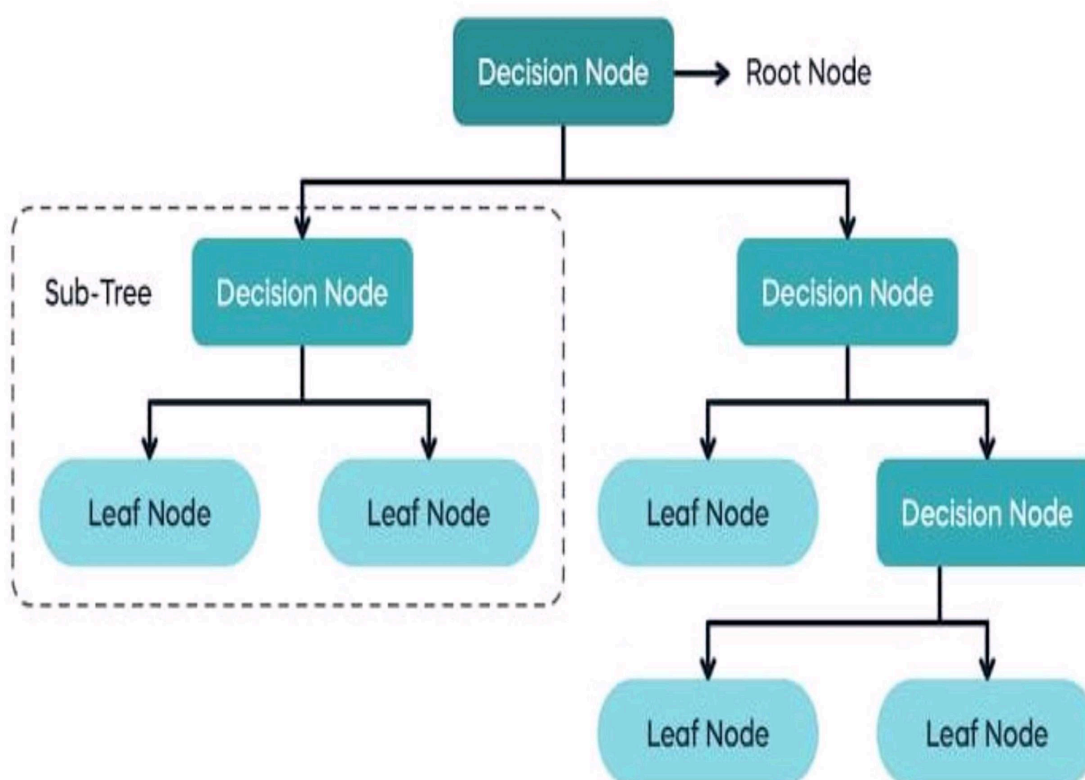


FIGURE: 3.3.1 DECISION TREE

3.3.2 LINEAR REGRESSION

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:

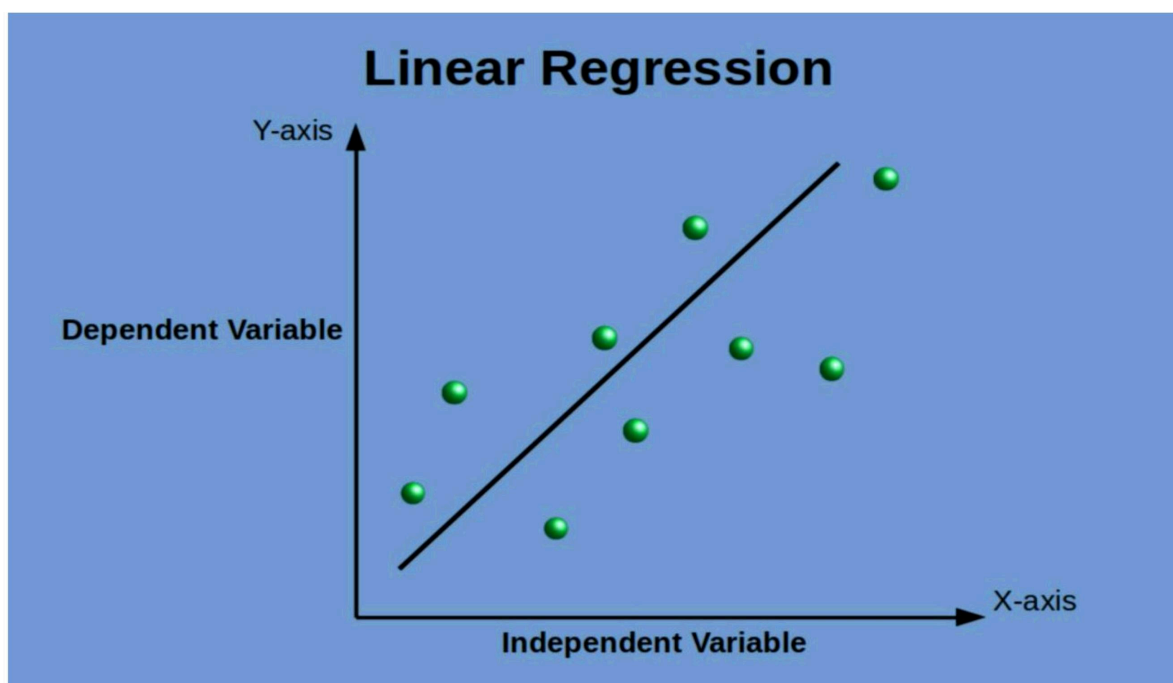


FIGURE : - 3.3.2 LINEAR REGRESSION

CHAPTER 4

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 RESULT AND PERFORMANCE ANALYSIS

	actual	pred
681	8.0	8.0
257	2.0	2.0
527	8.0	8.0
637	8.0	8.0
429	4.0	4.0

Root Mean Squared Error: 0.00014524951926243835
R^2 Score(coefficient of determination) = 0.9999999981868687

DECISION TREE

	actual	pred
681	8.0	8.0
257	2.0	2.0
527	8.0	8.0
637	8.0	8.0
429	4.0	4.0

Root Mean Squared Error: 0.07842449412709337
R^2 Score(coefficient of determination) = 0.9994714290308945

LINEAR REGRESSION

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 Summary:

This dataset consists of 740 observations and 21 columns. In the given data set, there are 21 variables and data types of all variables are either float 64 or into 64. From exploratory data analysis we have concluded that there are 10 continuous variables and 11 categorical variables in nature. Start looking for missing values and filling those with mean of particular column. Once we are good with the data with no null values and missing value and no string values then we can split the data into 2 parts. During applying different algorithms directly on the dataset, we need to train the data then we need to test the data. The purpose for this dataset is to be able to predict the absenteeism time in hours based upon the variables. The rmse and the r-squared value of the decision tree algorithm is RMSE: 0.0353 , R-squared: 0.9998. The rmse and the r-squared value of the Linear Regression

#RMSE: 0.0013 #R-squared: 0.9999

5.2 Conclusion

As manpower is the most important asset to an organization, absenteeism affects the same. The main reason for being default in Government organizations is low morale. Thus, government should to take proper measures to increase the morale and satisfaction of the employees through various training and development programs. It is also necessary to make the employees realize how important their role in the organization is. Being in public service, the employees should be aware of how their absence affects many people. Delayed promotion, fewer opportunities for promotion and advancement, low salary are some of the main reasons for absenteeism. Thus, this causes dissatisfaction among employees. Therefore, Government should take necessary steps to reduce absenteeism. To solve the problem of absenteeism of the employees at work in the companies, four models have been created in order to predict

the absenteeism. In this research four machine learning algorithms have been used that include; neural network algorithm, support vector machine algorithm, decision tree algorithm and logistic regression algorithm. The database used in this paper has 20 Attributes which created with records of absenteeism at work.

5.3 REFERENCES

- Bishop, *Pattern Recognition and Machine Learning*, 2006
- Ripley, *Pattern Recognition and Neural Networks*, 1996
- Duda, Hart, and Stork, *Pattern Classification*, 2nd Ed., 2002, Bertsekas and Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996
- For understanding sklearn and machine learning algorithms

https://www.tutorialspoint.com/scikit_learn/index.htm

SCREENSHOTS AND SOURCE CODE

```
#Import libraries
import os
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score
from fancyimpute import KNN
from sklearn.metrics import mean_squared_error

#import libraries for plots
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# Setting working directory
%os.chdir("C:/Users/Mayank/Desktop/Edwisor/Projects/Project 2")

# Loading data
emp_absent = pd.read_excel("Absenteeism_at_work.xlsx")
```

EXPLORATORY DATA ANALYSIS

```
emp_absent.shape
```

```
(740, 21)
```

```
# First 5 rows of data
emp_absent.head()
```

	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
0	11	26	7	3	1	289	36	13	33	239.554	...	0	1	2	1	0	1	9
1	36	0	7	3	1	118	13	18	50	239.554	...	1	1	1	1	0	0	9
2	3	23	7	4	1	179	51	18	38	239.554	...	0	1	0	1	0	0	8
3	7	7	7	5	1	279	5	14	39	239.554	...	0	1	2	1	1	0	6
4	11	23	7	5	1	289	36	13	33	239.554	...	0	1	2	1	0	1	9

5 rows × 21 columns

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
3	1	289	36	13	33	239.554	...	0	1	2	1	0	1	90	172	30	4
3	1	118	13	18	50	239.554	...	1	1	1	1	0	0	98	178	31	0
4	1	179	51	18	38	239.554	...	0	1	0	1	0	0	89	170	31	2
5	1	279	5	14	39	239.554	...	0	1	2	1	1	0	68	168	24	4
5	1	289	36	13	33	239.554	...	0	1	2	1	0	1	90	172	30	2

```
# Data Types of all the variables
emp_absent.dtypes
```

```

: ID int64
  Reason for absence int64
  Month of absence int64
  Day of the week int64
  Seasons int64
  Transportation expense int64
  Distance from Residence to Work int64
  Service time int64
  Age int64
  Work load Average/day float64
  Hit target int64
  Disciplinary failure int64
  Education int64
  Son int64
  Social drinker int64
  Social smoker int64
  Pet int64
  Weight int64
  Height int64
  Body mass index int64

```

```

Day of the week int64
Seasons int64
Transportation expense int64
Distance from Residence to Work int64
Service time int64
Age int64
Work load Average/day float64
Hit target int64
Disciplinary failure int64
Education int64
Son int64
Social drinker int64
Social smoker int64
Pet int64
Weight int64
Height int64
Body mass index int64
Absenteeism time in hours int64
dtype: object

```

```
# Number of Unique values present in each variable  
emp_absent.nunique()
```

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

```
#Transform data types
```

```
emp_absent['ID'] = emp_absent['ID'].astype('category')
```

```
emp_absent['Reason for absence'] = emp_absent['Reason for absence'].replace(0,20)
```

```
emp_absent['Reason for absence'] = emp_absent['Reason for absence'].astype('category')
```

```
emp_absent['Month of absence'] = emp_absent['Month of absence'].replace(0,np.nan)
```

```
emp_absent['Month of absence'] = emp_absent['Month of absence'].astype('category')
```

```
emp_absent['Day of the week'] = emp_absent['Day of the week'].astype('category')
```

```
emp_absent['Seasons'] = emp_absent['Seasons'].astype('category')
```

```
emp_absent['Disciplinary failure'] = emp_absent['Disciplinary failure'].astype('category')
```

```
emp_absent['Education'] = emp_absent['Education'].astype('category')
```

```
emp_absent['Son'] = emp_absent['Son'].astype('category')
```

```
emp_absent['Social drinker'] = emp_absent['Social drinker'].astype('category')
```

```
emp_absent['Social smoker'] = emp_absent['Social smoker'].astype('category')
```

```
emp_absent['Pet'] = emp_absent['Pet'].astype('category')
```

```
#Make a copy of dataframe
```

```
df = emp_absent.copy()
```

```
# From the EDA and problem statement file categorising the variables in two category "Continuous" and "Categorical"
continuous_vars = ['Distance from Residence to Work', 'Service time', 'Age', 'Work load Average/day', 'Transportation expense',
                  'Hit target', 'Weight', 'Height', 'Body mass index', 'Absenteeism time in hours']

categorical_vars = ['ID', 'Reason for absence', 'Month of absence', 'Day of the week',
                  'Seasons', 'Disciplinary failure', 'Education', 'Social drinker',
                  'Social smoker', 'Pet', 'Son']
```

MISSING VALUE ANALYSIS

```
#Creating dataframe with number of missing values
missing_val = pd.DataFrame(df.isnull().sum())

#Reset the index to get row names as columns
missing_val = missing_val.reset_index()

#Rename the columns
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_perc'})
missing_val

#Calculate percentage
missing_val['Missing_perc'] = (missing_val['Missing_perc']/len(df))*100

#Sort the rows according to decreasing missing percentage
missing_val = missing_val.sort_values('Missing_perc', ascending = False).reset_index(drop = True)

#Save output to csv file
missing_val.to_csv("Missing_perc.csv", index = False)

missing_val
```


	Variables	Missing_perc
0	Month of absence	0.405405
1	ID	0.000000
2	Disciplinary failure	0.000000
3	Body mass index	0.000000
4	Height	0.000000
5	Weight	0.000000
6	Pet	0.000000
7	Social smoker	0.000000
8	Social drinker	0.000000
9	Son	0.000000
10	Education	0.000000
11	Hit target	0.000000
12	Reason for absence	0.000000
13	Work load Average/day	0.000000
14	Age	0.000000
15	Service time	0.000000
16	Distance from Residence to Work	0.000000
17	Transportation expense	0.000000
18	Seasons	0.000000
19	Day of the week	0.000000
20	Absenteeism time in hours	0.000000

IMPUTE MISSING VALUE

```
#Actual value = 31
#Mean = 26.67
#Median = 25
#KNN = 30.80
print(df['Body mass index'].iloc[1])

#Set the value of first row in Body mass index as NAN
#create missing value
df['Body mass index'].iloc[1] = np.nan
```

31

```

#Impute with mean
df['Body mass index'] = df['Body mass index'].fillna(df['Body mass index'].mean())

#Impute with median
df['Body mass index'] = df['Body mass index'].fillna(df['Body mass index'].median())

#Apply KNN imputation algorithm
df = pd.DataFrame(KNN(k = 3).fit_transform(df), columns = df.columns)
df['Body mass index'].iloc[1]

```

```

Imputing row 1/740 with 0 missing, elapsed time: 0.100
Imputing row 101/740 with 0 missing, elapsed time: 0.100
Imputing row 201/740 with 0 missing, elapsed time: 0.100
Imputing row 301/740 with 0 missing, elapsed time: 0.100
Imputing row 401/740 with 0 missing, elapsed time: 0.100
Imputing row 501/740 with 0 missing, elapsed time: 0.100
Imputing row 601/740 with 0 missing, elapsed time: 0.100
Imputing row 701/740 with 0 missing, elapsed time: 0.100

31.00000068333565

```

```

#Round the values of categorical values
for i in categorical_vars:
    df.loc[:,i] = df.loc[:,i].round()
    df.loc[:,i] = df.loc[:,i].astype('category')

```

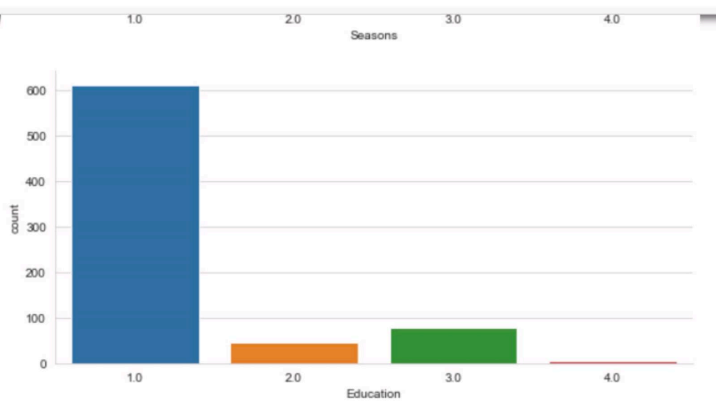
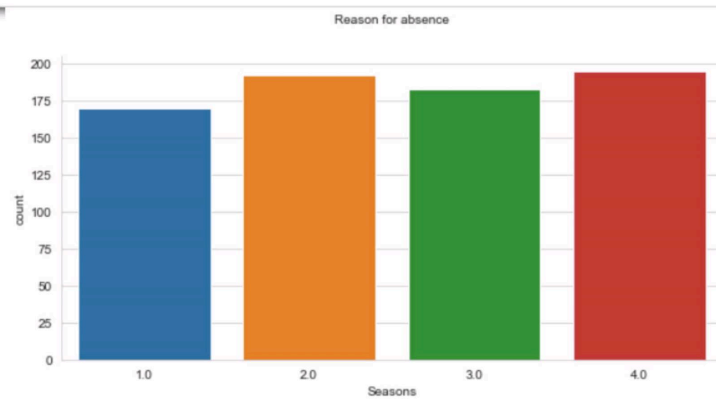
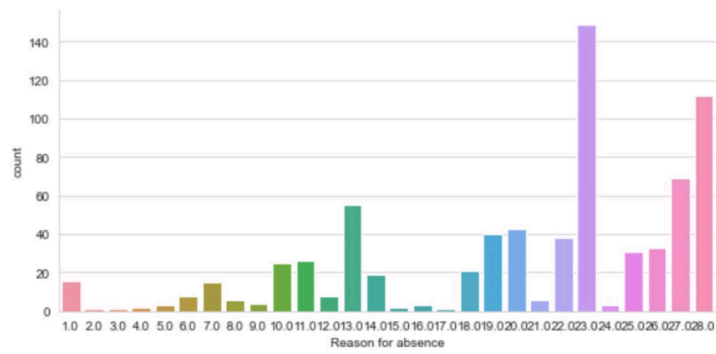
```
#Check if any missing values  
df.isnull().sum()
```

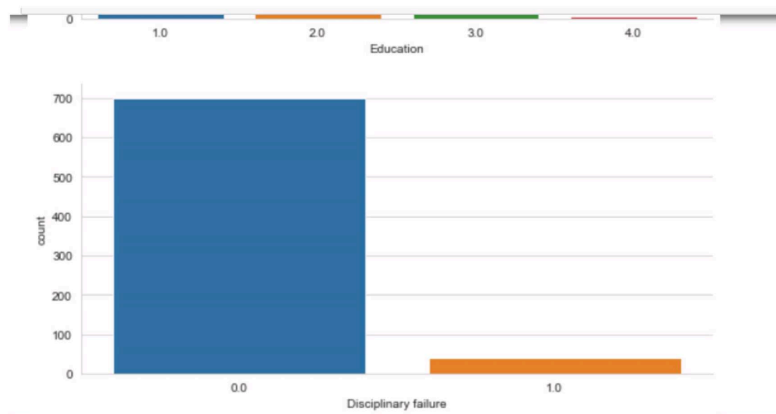
```
ID                                0  
Reason for absence                0  
Month of absence                  0  
Day of the week                   0  
Seasons                           0  
Transportation expense            0  
Distance from Residence to Work  0  
Service time                      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
```

DISTRIBUTION OF DATA USING GRAPHS

```
#Check the bar graph of categorical Data using factorplot  
sns.set_style("whitegrid")  
sns.factorplot(data=df, x='Reason for absence', kind= 'count',size=4,aspect=2)  
sns.factorplot(data=df, x='Seasons', kind= 'count',size=4,aspect=2)  
sns.factorplot(data=df, x='Education', kind= 'count',size=4,aspect=2)  
sns.factorplot(data=df, x='Disciplinary failure', kind= 'count',size=4,aspect=2)
```

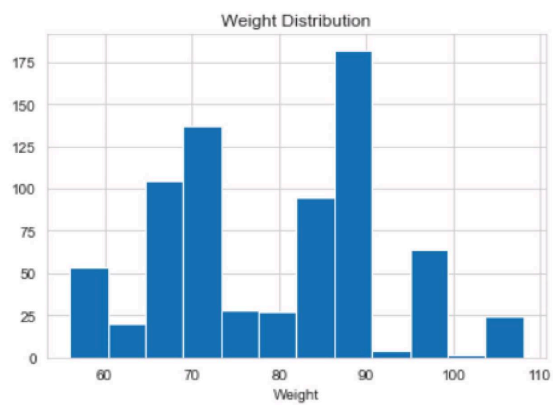
<seaborn.axisgrid.FacetGrid at 0x1ef0678b370>





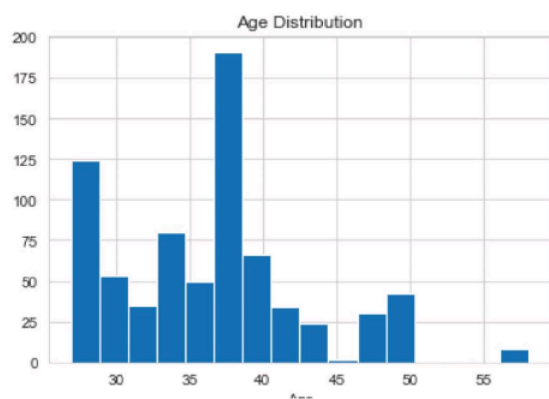
```
#Check the distribution of numerical data using histogram
plt.hist(data=df, x='Weight', bins='auto', label='Weight')
plt.xlabel('Weight')
plt.title("Weight Distribution")
```

Text(0.5, 1.0, 'Weight Distribution')



```
#Check the distribution of numerical data using histogram
plt.hist(data=df, x='Age', bins='auto', label='Age')
plt.xlabel('Age')
plt.title("Age Distribution")
```

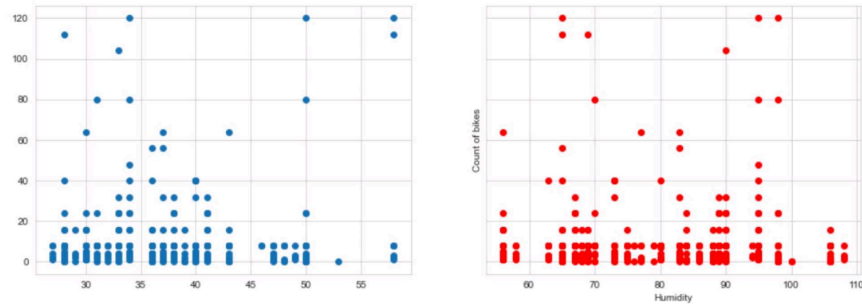
Text(0.5, 1.0, 'Age Distribution')



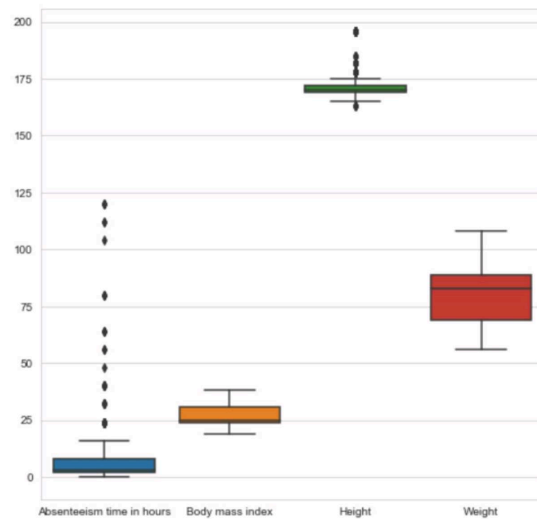
```
#Check the distribution of Temperature and Humidity against Bike rental count using scatter plot
fig, axs = plt.subplots(1,2, figsize=(15, 5), sharey=True)
axs[0].scatter(data=df, x='Age', y='Absenteeism time in hours')
axs[1].scatter(data=df, x='Weight', y='Absenteeism time in hours', color = 'red')
fig.suptitle('Scatter plot for Age and Weight')
plt.xlabel("Humidity")
plt.ylabel("Count of bikes")
```

Text(0, 0.5, 'Count of bikes')

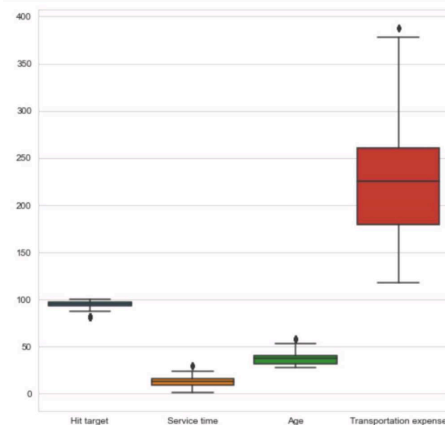
Scatter plot for Age and Weight



```
#Check for outliers in data using boxplot
sns.boxplot(data=df[['Absenteeism time in hours', 'Body mass index', 'Height', 'Weight']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```



```
#Check for outliers in data using boxplot
sns.boxplot(data=df[['Hit target', 'Service time', 'Age', 'Transportation expense']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```



OUTLINER ANALYSIS

```
#Check for outliers using boxplots
for i in continuous_vars:
    # Getting 75 and 25 percentile of variable "i"
    q75, q25 = np.percentile(df[i], [75,25])

    # Calculating Interquartile range
    iqr = q75 - q25

    # Calculating upper extream and Lower extream
    minimum = q25 - (iqr*1.5)
    maximum = q75 + (iqr*1.5)

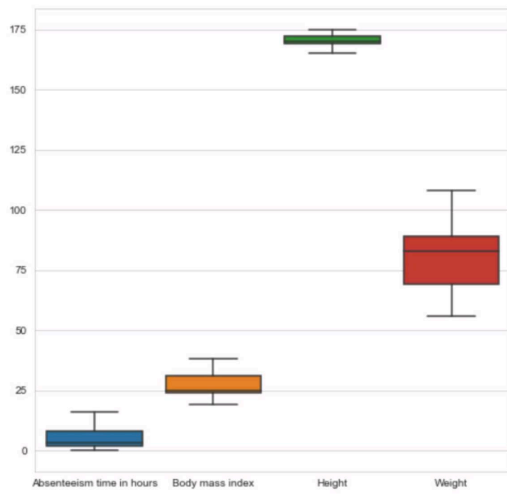
    # Replacing all the outliers value to NA
    df.loc[df[i]< minimum,i] = np.nan
    df.loc[df[i]> maximum,i] = np.nan

# Impute missing values with KNN
df = pd.DataFrame(KNN(k = 3).fit_transform(df), columns = df.columns)
# Checking if there is any missing value
df.isnull().sum()
```

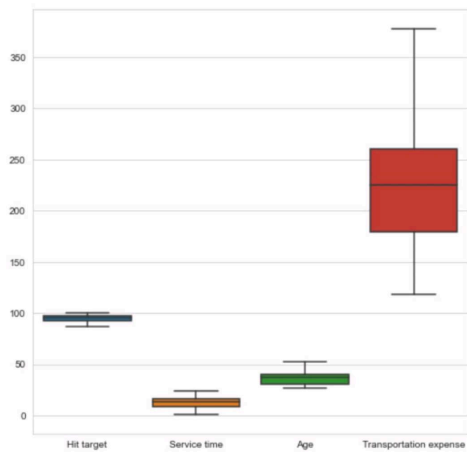
```
Imputing row 1/740 with 0 missing, elapsed time: 0.091
Imputing row 101/740 with 1 missing, elapsed time: 0.091
Imputing row 201/740 with 0 missing, elapsed time: 0.091
Imputing row 301/740 with 0 missing, elapsed time: 0.091
Imputing row 401/740 with 0 missing, elapsed time: 0.091
Imputing row 501/740 with 0 missing, elapsed time: 0.106
Imputing row 601/740 with 0 missing, elapsed time: 0.106
Imputing row 701/740 with 0 missing, elapsed time: 0.106
```

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

```
#Check for outliers in data using boxplot
sns.boxplot(data=df[['Absenteeism time in hours', 'Body mass index', 'Height', 'Weight']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```



```
#Check for outliers in data using boxplot
sns.boxplot(data=df[['Hit target', 'Service time', 'Age', 'Transportation expense']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```



FEATURE SELECTION

```
#Get dataframe with all continuous variables
df_corr = df.loc[:,continuous_vars]
```

```

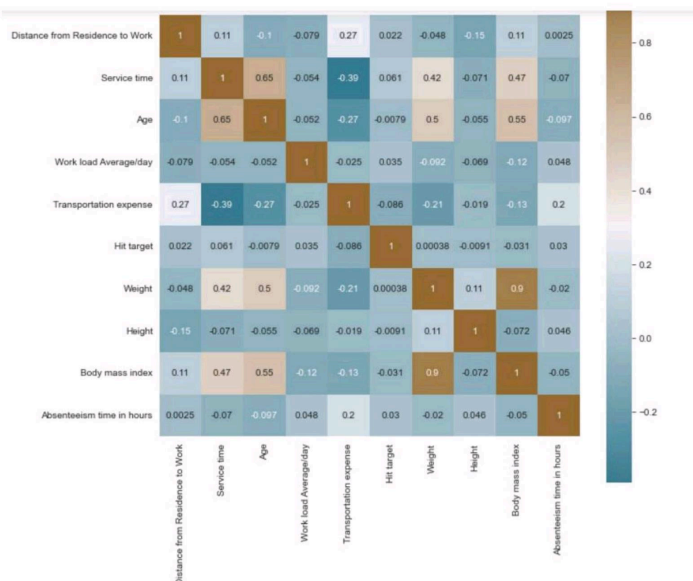
#Check for multicollinearity using correlation graph
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(10, 10))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
            cmap=sns.diverging_palette(220, 50, as_cmap=True),
            square=True, ax=ax, annot = True)

plt.plot()

```



```

#Variable Reduction
to_drop = ['Weight']
df = df.drop(to_drop, axis = 1)

# Updating the Continuous and Categorical Variables
continuous_vars.remove('Weight')

#Make a copy of clean data and export it as excel file
clean_data = df.copy()

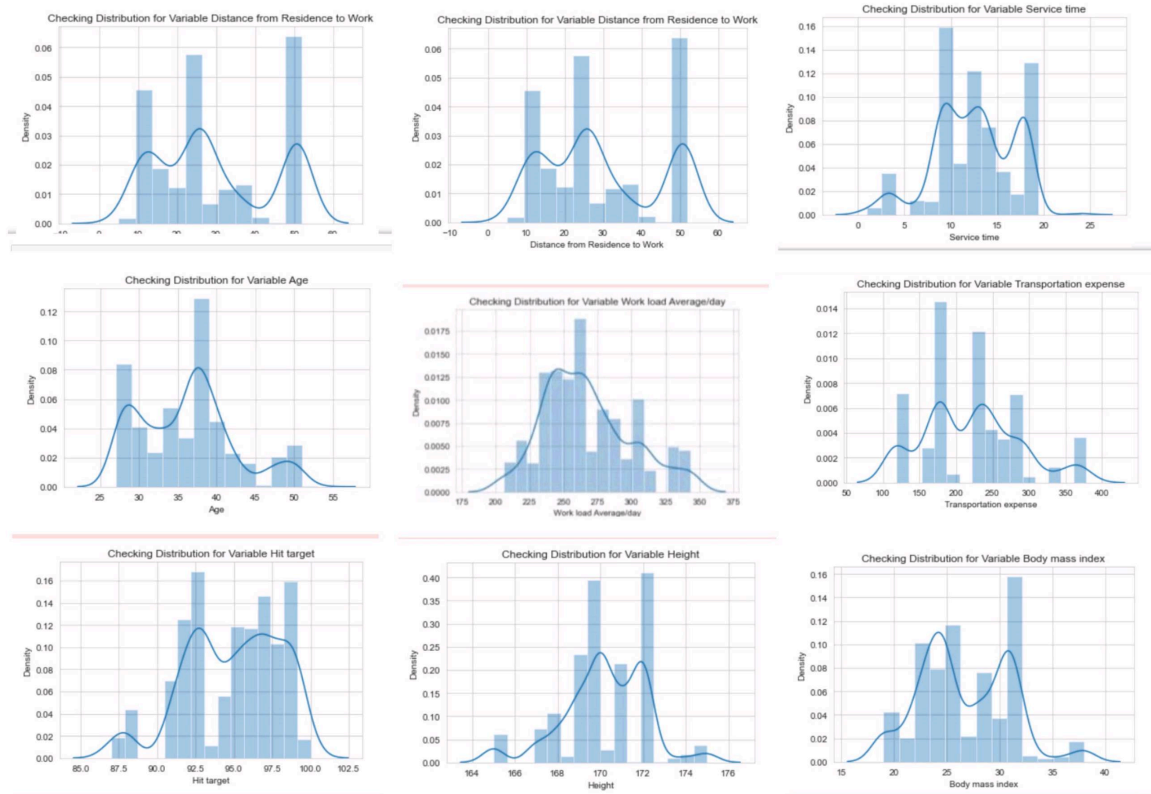
continuous_vars

['Distance from Residence to Work',
 'Service time',
 'Age',
 'Work load Average/day',
 'Transportation expense',
 'Hit target',
 'Height',
 'Body mass index',
 'Absenteeism time in hours']

```

FEATURE SCALING

```
#Normality check
for i in continuous_vars:
    if i == 'Absenteeism time in hours':
        continue
    sns.distplot(df[i],bins = 'auto')
    plt.title("Checking Distribution for Variable "+str(i))
    plt.ylabel("Density")
    plt.show()
```



```
#Normalization of continuous variables
for i in continuous_vars:
    if i == 'Absenteeism time in hours':
        continue
    df[i] = (df[i] - df[i].min())/(df[i].max()-df[i].min())
```

MACHINE LEARNING MODELS

```
#Create dummy variables of factor variables
df = pd.get_dummies(data = df, columns = categorical_vars)

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

```
#Get number of rows and columns
df.shape
```

```
(740, 115)
```

```
#Observe the first row
df.head(1)
```

	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target	Height	Body mass index	Absenteeism time in hours	ID_1.0 ...	Pet_1.0	Pet_2.0	Pet_4.0	Pet_5.0	Pet_8
0	0.657692	0.659574	0.521739	0.230769	0.244925	0.769231	0.699999	0.578947	4.0	0 ...	1	0	0	0	0

```
1 rows x 115 columns
```

```
#Observe the first row
df.head(1)
```

	Work load Average/day	Hit target	Height	Body mass index	Absenteeism time in hours	ID_1.0 ...	Pet_1.0	Pet_2.0	Pet_4.0	Pet_5.0	Pet_8.0	Son_0.0	Son_1.0	Son_2.0	Son_3.0	Son_4.0
0	0.244925	0.769231	0.699999	0.578947	4.0	0 ...	1	0	0	0	0	0	0	1	0	0

```
#Splitting data into train and test data
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:, df.columns != 'Absenteeism time in hours'], df.iloc[:, 8], test_size=0.2, random_state=1)
```

```
# test data
```

```
import train_test_split
test = train_test_split(df.iloc[:, df.columns != 'Absenteeism time in hours'], df.iloc[:, 8], test_size=0.2, random_state=1)
```

DECISION TREE

```
# Importing Libraries for Decision Tree
from sklearn.tree import DecisionTreeRegressor

#Build decision tree using DecisionTreeRegressor
dt_model = DecisionTreeRegressor(random_state = 1).fit(X_train,y_train)

#Perdict for test cases
dt_predictions = dt_model.predict(X_test)

#Create data frame for actual and predicted values
df_dt = pd.DataFrame({'actual': y_test, 'pred': dt_predictions})
print(df_dt.head())

#Define function to calculate RMSE
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, dt_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, dt_predictions)))
```

	actual	pred
681	8.0	8.0
257	2.0	1.0
527	8.0	8.0
637	8.0	2.0
429	4.0	1.0

Root Mean Squared Error: 3.9710640104494304
R^2 Score(coefficient of determination) = -0.35522928984886937

LINEAR REGRESSION

#RMSE: 40145e+8 #R-squared: -1.4181e+24

```
# Importing Libraries for Linear Regression
from sklearn.linear_model import LinearRegression

#Train the model
lr_model = LinearRegression().fit(X_train , y_train)

#Perdict for test cases
lr_predictions = lr_model.predict(X_test)

#Create data frame for actual and predicted values
df_lr = pd.DataFrame({'actual': y_test, 'pred': lr_predictions})
print(df_lr.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, lr_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, lr_predictions)))
```

	actual	pred
681	8.0	4.052734
257	2.0	4.863281
527	8.0	6.464844
637	8.0	5.417969
429	4.0	2.687500

Root Mean Squared Error: 1425414659635.2578
R^2 Score(coefficient of determination) = -1.7461495887001298e+23

DIMENSION REDUCTION USING PCA

```
#Get the target variable
target = df['Absenteeism time in hours']
```

```
#Get the number of rows and columns of data
df.shape
```

(740, 115)


```

#Import library for PCA
from sklearn.decomposition import PCA

#Converting data to numpy array
X = df.values

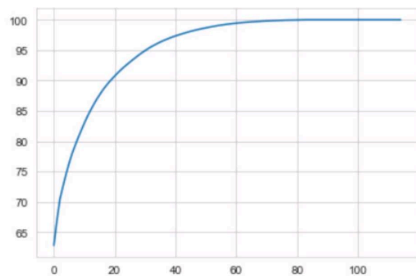
#Data has 116 variables so no of components of PCA = 115
pca = PCA(n_components=115)
pca.fit(X)

#Proportion of variance explained
var = pca.explained_variance_ratio_

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

#Draw the plot
plt.plot(var1)
plt.show()

```



```

#Selecting 45 components since it explains almost 95+ % data variance
pca = PCA(n_components=45)

#Fitting the selected components to the data
pca.fit(X)

#Splitting data into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, target, test_size=0.2, random_state = 1)

```

DECISION TREE

```

#Build decision tree using DecisionTreeRegressor
dt_model = DecisionTreeRegressor(random_state = 1).fit(X_train,y_train)

#Predict for test cases
dt_predictions = dt_model.predict(X_test)

#Create data frame for actual and predicted values
df_dt = pd.DataFrame({'actual': y_test, 'pred': dt_predictions})
print(df_dt.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, dt_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, dt_predictions)))

```

	actual	pred
681	8.0	8.0
257	2.0	2.0
527	8.0	8.0
637	8.0	8.0
429	4.0	4.0

Root Mean Squared Error: 0.07842390053401137
R^2 Score(coefficient of determination) = 0.9994714380674777

LINEAR REGRESSION

#RMSE: 0.0013 #R-squared: 0.9999

```
# Importing Libraries for Linear Regression
from sklearn.linear_model import LinearRegression

#Train the model
lr_model = LinearRegression().fit(X_train , y_train)

#Predict for test cases
lr_predictions = lr_model.predict(X_test)

#Create data frame for actual and predicted values
df_lr = pd.DataFrame({'actual': y_test, 'pred': lr_predictions})
print(df_lr.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, lr_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, lr_predictions)))
```

	actual	pred
681	8.0	8.0
257	2.0	2.0
527	8.0	8.0
637	8.0	8.0
429	4.0	4.0

Root Mean Squared Error: 0.008415393369067563
R^2 Score(coefficient of determination) = 0.999993913777196

SOURCE CODE

#Import libraries

import os

import pandas as pd

import numpy as np

from sklearn.metrics import r2_score

from fancyimpute import KNN

from sklearn.metrics import mean_squared_error

#import libraries for plots

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

Setting working directory

#os.chdir("C:/Users/Mayank/Desktop/Edwisor/Projects/Project 2")

Loading data

```

emp_absent = pd.read_excel("Absenteeism_at_work.xlsx")
emp_absent.shape

# First 5 rows of data
emp_absent.head()

# Data Types of all the variables
emp_absent.dtypes

# Number of Unique values present in each variable
emp_absent.nunique()

# Transform data types
emp_absent['ID'] = emp_absent['ID'].astype('category')

emp_absent['Reason for absence'] = emp_absent['Reason for absence'].replace(0,20)
emp_absent['Reason for absence'] = emp_absent['Reason for absence'].astype('category')

emp_absent['Month of absence'] = emp_absent['Month of absence'].replace(0,np.nan)
emp_absent['Month of absence'] = emp_absent['Month of absence'].astype('category')

emp_absent['Day of the week'] = emp_absent['Day of the week'].astype('category')
emp_absent['Seasons'] = emp_absent['Seasons'].astype('category')
emp_absent['Disciplinary failure'] = emp_absent['Disciplinary failure'].astype('category')
emp_absent['Education'] = emp_absent['Education'].astype('category')
emp_absent['Son'] = emp_absent['Son'].astype('category')
emp_absent['Social drinker'] = emp_absent['Social drinker'].astype('category')
emp_absent['Social smoker'] = emp_absent['Social smoker'].astype('category')
emp_absent['Pet'] = emp_absent['Pet'].astype('category')

```

```

#Make a copy of dataframe
df = emp_absent.copy()

# From the EDA and problem statement file categorising the variables in two category
" Continuos" and "Categorical"
continuous_vars = ['Distance from Residence to Work', 'Service time', 'Age', 'Work load
Average/day', 'Transportation expense',
    'Hit target', 'Weight', 'Height', 'Body mass index', 'Absenteeism time in hours']

categorical_vars = ['ID', 'Reason for absence', 'Month of absence', 'Day of the week',
    'Seasons', 'Disciplinary failure', 'Education', 'Social drinker',
    'Social smoker', 'Pet', 'Son']

#Creating dataframe with number of missing values
missing_val = pd.DataFrame(df.isnull().sum())

#Reset the index to get row names as columns
missing_val = missing_val.reset_index()

#Rename the columns
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_perc'})
missing_val

#Calculate percentage
missing_val['Missing_perc'] = (missing_val['Missing_perc']/len(df))*100

#Sort the rows according to decreasing missing percentage
missing_val = missing_val.sort_values('Missing_perc', ascending =
False).reset_index(drop = True)

#Save output to csv file
missing_val.to_csv("Missing_perc.csv", index = False)

```

```
missing_val
```

```
#Actual value = 31
```

```
#Mean = 26.67
```

```
#Median = 25
```

```
#KNN = 30.80
```

```
print(df['Body mass index'].iloc[1])
```

```
#Set the value of first row in Body mass index as NAN
```

```
#create missing value
```

```
df['Body mass index'].iloc[1] = np.nan
```

```
#Actual value = 31
```

```
#Mean = 26.67
```

```
#Median = 25
```

```
#KNN = 30.80
```

```
print(df['Body mass index'].iloc[1])
```

```
#Set the value of first row in Body mass index as NAN
```

```
#create missing value
```

```
df['Body mass index'].iloc[1] = np.nan
```

```
#Round the values of categorical values
```

```
for i in categorical_vars:
```

```
    df.loc[:,i] = df.loc[:,i].round()
```

```
    df.loc[:,i] = df.loc[:,i].astype('category')
```

```
#Check if any missing values
```

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

```
#Check the bar graph of categorical Data using factorplot
```

```

sns.set_style("whitegrid")
sns.factorplot(data=df, x='Reason for absence', kind= 'count',size=4,aspect=2)
sns.factorplot(data=df, x='Seasons', kind= 'count',size=4,aspect=2)
sns.factorplot(data=df, x='Education', kind= 'count',size=4,aspect=2)
sns.factorplot(data=df, x='Disciplinary failure', kind= 'count',size=4,aspect=2)

#Check the distribution of numerical data using histogram
plt.hist(data=df, x='Weight', bins='auto', label='Weight')
plt.xlabel('Weight')
plt.title("Weight Distribution")

#Check the distribution of numerical data using histogram
plt.hist(data=df, x='Age', bins='auto', label='Age')
plt.xlabel('Age')
plt.title("Age Distribution")

#Check the distribution of Temperature and Humidity against Bike rental count using
scatter plot
fig, axs = plt.subplots(1,2, figsize=(15, 5), sharey=True)
axs[0].scatter(data=df, x='Age', y='Absenteeism time in hours')
axs[1].scatter(data=df, x='Weight', y='Absenteeism time in hours', color = 'red')
fig.suptitle('Scatter plot for Age and Weight')
plt.xlabel("Humidity")
plt.ylabel("Count of bikes")

#Check for outliers in data using boxplot
sns.boxplot(data=df[['Absenteeism time in hours','Body mass index','Height','Weight']])
fig=plt.gcf()
fig.set_size_inches(8,8)

#Check for outliers in data using boxplot
sns.boxplot(data=df[['Hit target','Service time','Age','Transportation expense']])

```



```

fig=plt.gcf()
fig.set_size_inches(8,8)

#Check for outliers using boxplots
for i in continuous_vars:
    # Getting 75 and 25 percentile of variable "i"
    q75, q25 = np.percentile(df[i], [75,25])

    # Calculating Interquartile range
    iqr = q75 - q25

    # Calculating upper extream and lower extream
    minimum = q25 - (iqr*1.5)
    maximum = q75 + (iqr*1.5)

    # Replacing all the outliers value to NA
    df.loc[df[i]< minimum,i] = np.nan
    df.loc[df[i]> maximum,i] = np.nan

# Impute missing values with KNN
df = pd.DataFrame(KNN(k = 3).fit_transform(df), columns = df.columns)
# Checking if there is any missing value
df.isnull().sum()

#Check for outliers in data using boxplot
sns.boxplot(data=df[['Absenteeism time in hours','Body mass index','Height','Weight']])
fig=plt.gcf()
fig.set_size_inches(8,8)

#Check for outliers in data using boxplot
sns.boxplot(data=df[['Hit target','Service time','Age','Transportation expense']])

```

```

fig=plt.gcf()
fig.set_size_inches(8,8)

#Get dataframe with all continuous variables
df_corr = df.loc[:,continuous_vars]

#Check for multicollinearity using corelation graph
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(10, 10))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
            cmap=sns.diverging_palette(220, 50, as_cmap=True),
            square=True, ax=ax, annot = True)
plt.plot()

#Variable Reduction
to_drop = ['Weight']
df = df.drop(to_drop, axis = 1)

# Updating the Continuous and Categorical Variables
continuous_vars.remove('Weight')

#Make a copy of clean data and export it as excel file
clean_data = df.copy()

continuous_vars

#Normality check

```

```

for i in continuous_vars:
    if i == 'Absenteeism time in hours':
        continue
    sns.distplot(df[i],bins = 'auto')
    plt.title("Checking Distribution for Variable "+str(i))
    plt.ylabel("Density")
    plt.show()

#Normalization of continuous variables
for i in continuous_vars:
    if i == 'Absenteeism time in hours':
        continue
    df[i] = (df[i] - df[i].min())/(df[i].max()-df[i].min())

#Create dummy variables of factor variables
df = pd.get_dummies(data = df, columns = categorical_vars)

# Copying dataframe
df1 = df.copy()

#Get number of rows and columns
df.shape

#Observe the first row
df.head(1)

#Splitting data into train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( df.iloc[:, df.columns != 'Absenteeism
time in hours'], df.iloc[:, 8], test_size

# Importing libraries for Decision Tree

```

```

from sklearn.tree import DecisionTreeRegressor

#Build decision tree using DecisionTreeRegressor
dt_model = DecisionTreeRegressor(random_state = 1).fit(X_train,y_train)

#Predict for test cases
dt_predictions = dt_model.predict(X_test)

#Create data frame for actual and predicted values
df_dt = pd.DataFrame({'actual': y_test, 'pred': dt_predictions})
print(df_dt.head())

#Define function to calculate RMSE
def RMSE(y_actual,y_predicted):
    rmse = np.sqrt(mean_squared_error(y_actual,y_predicted))
    return rmse

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, dt_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, dt_predictions)))

# Importing libraries for Random Forest
from sklearn.ensemble import RandomForestRegressor

#Build random forest using RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators = 500, random_state =
1).fit(X_train,y_train)

#Predict for test cases
rf_predictions = rf_model.predict(X_test)

#Create data frame for actual and predicted values

```

```

df_rf = pd.DataFrame({'actual': y_test, 'pred': rf_predictions})
print(df_rf.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, rf_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, rf_predictions)))

# Importing libraries for Linear Regression
from sklearn.linear_model import Linear Regression

#Train the model
lr_model = Linear Regression().fit(X_train , y_train)

#Predict for test cases
lr_predictions = lr_model.predict(X_test)

#Create data frame for actual and predicted values
df_lr = pd.DataFrame({'actual': y_test, 'pred': lr_predictions})
print(df_lr.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, lr_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, lr_predictions)))

#Get the target variable
target = df['Absenteeism time in hours']

#Get the number of rows and columns of data
df.shape

#Import library for PCA
from sklearn.decomposition import PCA

```

```

#Converting data to numpy array
X = df.values

#Data has 116 variables so no of components of PCA = 115
pca = PCA(n_components=115)
pca.fit(X)

#Proportion of variance explained
var= pca.explained_variance_ratio_

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

#Draw the plot
plt.plot(var1)
plt.show()

#Selecting 45 components since it explains almost 95+ % data variance
pca = PCA(n_components=45)

#Fitting the selected components to the data
pca.fit(X)

#Splitting data into train and test data
X_train, X_test, y_train, y_test = train_test_split(X,target, test_size=0.2, random_state
= 1)

#Build decsion tree using DecisionTreeRegressor
dt_model = DecisionTreeRegressor(random_state = 1).fit(X_train,y_train)

#Perdict for test cases

```



```

dt_predictions = dt_model.predict(X_test)

#Create data frame for actual and predicted values
df_dt = pd.DataFrame({'actual': y_test, 'pred': dt_predictions})
print(df_dt.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, dt_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, dt_predictions)))

#Build random forest using RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators = 500, random_state =
1).fit(X_train,y_train)

#Perdict for test cases
rf_predictions = rf_model.predict(X_test)

#Create data frame for actual and predicted values
df_rf = pd.DataFrame({'actual': y_test, 'pred': rf_predictions})
print(df_rf.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, rf_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, rf_predictions)))

# Importing libraries for Linear Regression
from sklearn.linear_model import LinearRegression

#Train the model
lr_model = LinearRegression().fit(X_train , y_train)

#Perdict for test cases

```

```
lr_predictions = lr_model.predict(X_test)

#Create data frame for actual and predicted values
df_lr = pd.DataFrame({'actual': y_test, 'pred': lr_predictions})
print(df_lr.head())

#Calculate RMSE and R-squared value
print("Root Mean Squared Error: "+str(RMSE(y_test, lr_predictions)))
print("R^2 Score(coefficient of determination) = "+str(r2_score(y_test, lr_predictions)))
```