

EMPLOYEE BURNOUT PREDICTION

Submitted in partial fulfillment of the requirements for the award
of

Bachelor of Computer Science

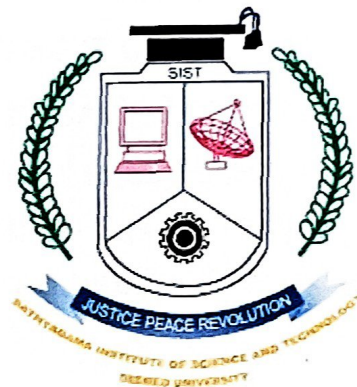
By

Name: HEMA K

(REGISTER NO 40290036)

Name: DHARSHINI S

(REGISTER NO 40290023)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

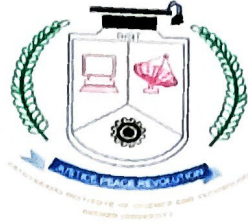
SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600199

MAY 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

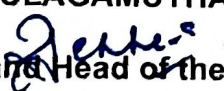
DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **HEMA K (40290036)** and **DHARSHINI S (40290023)** who carried out the project entitled "EMPLOYEE BURNOUT PREDICTION" under my supervision from January 2023 to May 2023.

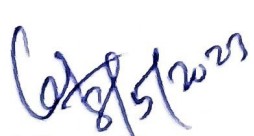

INTERNAL GUIDE

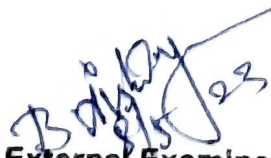
Dr. V.ULAGAMUTHALVI M.E., Ph.D.


Dean and Head of the Department

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva Voce examination held on 08-05-2023


Internal Examiner

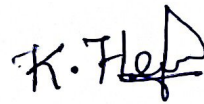

External Examiner

DECLARATION

I HEMA K (40290036) and DHARSHINI S (40290023) hereby declare that the Project Report entitled “EMPLOYEE BURNOUT PREDICTION” done by me under the guidance of Dr.V.ULAGAMUTHALVI M.E.,Ph.D. at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Computer Science.

DATE: 08-05-2023

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. REKHA CHAKRAVARTHI M.E., Ph.D.** Dean of Arts and Science 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.V.ULAGAMUTHALVI M.E., Ph.D.** for her 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 Computer Science who were helpful in many ways for the completion of the project.

ABSTRACT

Data Analysis is the process of bringing order and structure to collected data. It turns data into information teams can use. Analysis is done using systematic methods to look for trends, groupings, or other relationships between different types of data. Data visualization is the process of putting data into a chart, graph, or other visual format that helps inform analysis and interpretation. Data visuals present the analyzed data in ways that are accessible to and engage different stakeholders.

So this visualization can be done with any dataset(.csv files).We can select on any topic we wish like I mentioned above. Here I had chosen a dataset which contains data on Employee Burnout.

This dataset contains records of 2200 employees working in IT field. The dataset contains information like, Employee ID, Date of Joining, Gender, Company Type, WFH Setup Available, Designation, Resource Allocation, Mental Fatigue Score and Burn Rate.

TABLE OF CONTENTS

Chapter No.	Title.	Page No.
	ABSTRACT	V
	LIST OF FIGURES	VII
1.	INTRODUCTION	
	1.1 INTRODUCTION TO THE DATA ANALYTICS AND VISUALIZATION	10
	1.2 NEED FOR THE DATA ANALYTICS AND VISUALIZATION	11
	1.3 INTRODUCTION TO PYTHON FOR DATA ANALYTICS	12
2.	AIM AND SCOPE OF THE EXPERIMENT	
	2.1 AIM	
	2.2 SCOPE OF THE EXPERIMENT	14
	2.3 WHAT IS EMPLOYEE BURNOUT	
3.	LITERATURE SURVEY	
	3.1 LITERATURE REVIEW	16
4.	EXPERIMENTS OR REQUIREMENTS AND METHODS USED	
	4.1 SYSTEM SPECIFICATIONS	18
	4.2 PROJECT DESCRIPTION	18
	4.3 LIBRARY USED	18
	4.4 IMPORTING THE DATA SET FROM .CSV FILE	20

4.5 DATA PRE-PROCESSING	21
5. RESULTS, DISCUSSION AND PERFORMANCE ANALYSIS	
5.1 RESULT	25
5.2 PERFORMANCE ANALYSIS	25
5.3 SPLITTING DATA FOR TRAINING AND TESTING	25
5.4 LINEAR REGRESSION	26
5.5 FITTING THE ALGORITHM AND MAKING PREDICTIONS	27
5.6 CHECKING FOR ERRORS	28
5.7 CALCULATING THE COEFFICIENT OF DETERMINATION (R^2 SCORE)	28
5.8 DATA VISUALIZATION USING MATPLOTLIB AND SEABORN	29
6. SUMMARY AND DISCUSSIONS	32
APPENDIX	
A. SOURCE CODE	33
B. SCREEN SHOTS	35

LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE NO.
4.1	Code for importing libraries	20
4.2	Code for importing Dataset	21
4.3	Describing data.head()	22
4.4	Example code for the prep-work of data set with null values	23
4.5	Example code for the prep-work of data set with null values	23
4.6	code for splitting dataframe into two parts based on the column values	24
5.1	code for splitting the dataset into training and testing.	26
5.2	code to demonstrate fitting the algorithm to the model.	27
5.3	Resulting array of predictions made	28
5.4	Calculating the Mean Squared Error	28
5.5	Calculating the R ² Score	28
5.6	Plot of Burn Rate	29
5.7	Scatterplot of Mental fatigue Score and Burn Rate	30
5.8	Scatter Plot of Resource Allocation, Burn Rate and Gender	30

5.9	Scatterplot of Mental fatigue Score, Burn Rate and WFH Setup Available	31
6.1	Importing libraries and dataset	35
6.2	Example code for prep-work and filling up null values	35
6.3	Result for the prep-work of dataset with null values	36
6.4	Splitting the dataset into two subset	36
6.5	code to demonstrate fitting the algorithm to the model	38
6.6	Plot of Burn Rate	38
6.7	Scatterplot of Mental fatigue Score, Burn Rate and WFH Setup Available	39
6.8	Scatter Plot of Resource Allocation, Burn Rate and Gender	39
6.9	Scatterplot of Mental fatigue Score and Burn Rate	40

CHAPTER 1

INTRODUCTION

1.1 Introduction to the DATA ANALYTICS AND VISUALIZATION

Data Analysis is the process of bringing order and structure to collected data. It turns data into information teams can use. Analysis is done using systematic methods to look for trends, groupings, or other relationships between different types of data.

Data Visualization is the discipline of trying to understand data by placing it in a visual Context so that pattern, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries:

MATPLOTLIB: It is a multi-platform data visualization library built on NumPy arrays, and designed to work with the broader SciPy stack. It was conceived by John Hunter in 2002, originally as a patch to I Python for enabling interactive MATLAB-style plotting via gnu plot from the I Python command line.

SEABORN: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

PLOTLY: Plotlib is the Python Library for interactive data visualizations. Plotlib allows you to plot superior interactive graphs than either Matplotlib or Seaborn.

1.2 Need for DATA ANALYTICS AND VISULIZATION

Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets.

Data visualization is one of the steps of the data science process, which states that after data has been collected, processed and modeled, it must be visualized for conclusions to be made. Data visualization is also an element of the broader data presentation architecture (DPA) discipline, which aims to identify, locate, manipulate, format and deliver data in the most efficient way possible.

1.3 INTRODUCTION TO PYTHON FOR DATA ANALYTICS

Python is a high level, dynamic programming language. Python3.4 version was used as it is a mature, versatile and robust programming language. It is an interpreted language which makes the testing and debugging extremely quickly as there is no compilation step. There are extensive open source libraries available for this version of python and a large community of users. Python is simple yet powerful, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data, i.e. spoken English using NLTK. Other high level programming languages such as R and Matplotlib were considered because they have many benefits such as ease of use but they do not offer the same flexibility and freedom that Python can deliver.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data. Prior to Pandas, Python was majorly used for data munging and preparation. It had very less contribution towards data analysis. Pandas solved this

problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data—load, prepare, manipulate, model and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient Data Frame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub-setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Matplotlib is the most popular python plotting library. It is a low-level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

Key features of Matplotlib:

- Semantic way to generate complex, subplot grids.
- Settings the aspect ratio of the axes box.
- Colored labels in legends.
- Ticks and labels.
- RcParams can be passed as Decorators.
- 3D plots now support minor ticks.

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive graphs.

Seaborn has a lot to offer. You can create graphs in one line that would take you multiple tens of lines in Matplotlib. Its standard designs are awesome and it also has a nice interface for working with pandas data frames.

Key features in Seaborn

- Built in themes for styling matplotlib graphics.
- Visualizing univariate and bivariate data.
- Fitting in and visualizing linear regression models.
- Plotting statistical time series data.
- Seaborn works well with NumPy and Pandas data structures.

Plotly allows users to import, copy and paste, or stream data to be analyzed and visualized. For analysis and styling graphs, Plotly offers a Python sandbox (NumPy supported), datagrid, and GUI. Python scripts can be saved, shared, and collaboratively edited in Plotly.

Key features of Plotly

- Charts
- Dashboards
- File Export
- App Manager
- Kubernetes Authentication
- Jobs Queue
- Snapshot Engine

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1. AIM:

To analyze and visualize the data using .csv or .tsv files and by using graphs to support our analysis from a data set.

2.2. SCOPE:

Our brain too interprets information fast and best when received through graphical processing. It makes complicated datasets easily understandable and comprehensible through the medium of graphical and visual communication. The strength and scope of data analytics and visualization is immense. To get full advantage of data analytics and visualization, first you should know about the various modern approaches on scope of data analytics visualization in the related fields.

In this project we are going to use a dataset named tain.csv file which contains the data of employees with details such as, Employee ID, Date of Joining, Gender, Company Type, WFH Setup Available, Designation, Resource Allocation, Mental Fatigue Score and Burn Rate.

2.3. WHAT IS EMPLOYEE BURNOUT

Burnout strikes employees when they have exhausted their physical or emotional strength. This usually occurs as a result of prolonged stress or frustration. Sometimes the cause is the work environment. Stressful jobs, lack of support and resources, and tight deadlines can all contribute to burnout. Other times, burnout has more to do with employees' expectations of themselves or their personal circumstances.

- Burnout can manifest itself in a variety of ways, including the following:
- Frustration or indifference toward work
- Persistent irritability
- Anger, sarcasm, or being argumentative

- Exhaustion
- Absenteeism

All of these things can drain an organization's morale—as well as its wallet. Burned-out employees can be costly in terms of productivity, and if burned-out employees quit, there are the costs of replacement searches and training.

While burnout can't always be prevented, it can be managed. In order to effectively manage employee burnout, people need to

- Understand the reasons for burnout
- Find mutually agreeable solutions
- Rekindle employee motivation

When employees experience burnout, they often feel exhausted, cynical, and detached from their work. They may also experience physical symptoms such as headaches, insomnia, and increased susceptibility to illness. Burnout can also lead to decreased job performance, reduced productivity, and increased absenteeism.

Burnout can occur in any industry or job, and it can affect employees at any level of an organization. It is important for employers to recognize the signs of burnout and take steps to prevent it, such as providing adequate resources, promoting work-life balance, and encouraging employees to take breaks and prioritize self-care.

CHAPTER 3

LITERATURE SURVEY

3.1. LITERATURE REVIEW:

[1] Title: Machine Learning Techniques for Stress Prediction in Working Employees.

Authors: U.Srinivasulu Reddy, Aditya Vivek Thota, A Dharun - 2020

Description:

Stress disorders are a common problem among today's IT professionals. Employees are more likely to be stressed as their lifestyles and work habits change. In this paper, we want to use machine learning techniques to analyze stress patterns in working adults and narrow down the factors that have a strong influence on stress levels.

Limitations:

- Less parameters used for stress prediction.
- Boosting algorithm is not suitable for real time application.
- Ready Tools used for stress prediction.

[2] Title: Classification Algorithms based Mental Health Prediction using Data Mining.

Authors: Vidit Laijawala, Aadesh Achaliya, Hardik Jatta, Vijaya Pinjarkar – 2020

Description:

Mental health reflects a person's emotional, mental, and social well-being. It affects the way people think, feel, and react in a particular situation. Depression, social anxiety, depression, compulsive mental disorders, substance abuse, work problems, and personality disorders are all factors that contribute to mental health problems leading to mental illness.

Limitations:

- Small datasets are used, resulting in less accurate results.
- Data mining techniques are used to predict mental health.
- Massive amounts of data are required.

[3] Title: Predictive Analysis of Student Stress Level Using Naive Bayesian Classification Algorithm .

Authors: Monisha S, Meera R ,Vijay Swaminath R, Dr. Arun Raj L – 2020

Description:

Overall academic performance and social coercion have created a state of psychological stress for students. The stressors that are often talked about should be minimized to help students improve academically and socially, thus reducing health problems such as headaches, spectacular wear, and so on.

Limitations:

- This concept predicts stress of only college students, can't be applicable for working employees.
- Algorithms used here takes more time processing.
- Less efficient results.

[4] Title: Research on the College Students Psychological Health Management based on Data Mining and Cloud Platform.

Authors: Fang Li

Description:

College students' psychological problems have become increasingly visible in recent years, and it is difficult to meet their needs due to a lack of high-quality resources in mental health education. In this paper, author explores mental health management for college students using data mines and cloud platform .

Limitations:

- Applicable for education sector not for corporate world.
- This concept predicts stress of only college students, can't be applicable for working employees.
- Less parameter used in the project.

CHAPTER 4

EXPERIMENTAL OR REQUIREMENTS AND METHODS

4.1. SYSTEM SPECIFICATION

Hardware Requirements

1. Processor – Intel Core processors or any AMD chips.
2. RAM – 1 GB
3. Hard Disk – 40GB
4. Operating System – Windows 7 and above

Software requirements

1. Jupyter notebook (anaconda 3) or Google collab.
2. Python 3 or latest version.

4.2. PROJECT DESCRIPTION

We are going to analyze the data of Employee Burnout which contains information like Employee ID, Date of Joining, Gender, Company Type, WFH Setup Available, Designation, Resource Allocation, Mental Fatigue Score and Burn Rate. We have got this dataset from Kaggle. We are going to analyze the information using Linear Regression in this dataset and try to present in a visual manner using different libraries in python programming language like numpy, pandas, matplotlib and seaborn.

4.3. LIBRARIES USED

We will generally be working on this project using the standard libraries such as:

- Sklearn
- Pandas
- Matplotlib
- Seaborn

Sklearn is the most important library from which we are going to import and make use of the algorithms or the metrics or the train_test_split for making predictions using the model generated using it.

Pandas is the standard library one would be dealing with while doing a machine learning project. As I have mentioned above that I will be using the data set which is stored in an excel file, we can use pandas to read the excel file.

Similarly we will use Numpy to make use of the numpy arrays, while we are calculating the errors or handling the data of some kind.

Matplotlib is one of the major necessities as we are going to use it for creating plots and to analyze our predictions based on that.

Seaborn is also a library used for visualizing the data using the plots. It was my personal opinion to choose this library out of other visualizing modules and libraries present. Moving forward there may be some other libraries that are used offhand but all serve for the same use.

We will break down more about the libraries and their usage once we start using them. I will try to describe that particular usage and its overall description too once we get into the coding part.

As we have seen that the above are the necessary libraries which we will import as full but for some we will only import the necessary modules (just to have a quick knowledge of exactly what modules we will be using).

We will import a few specific functions and algorithms from modules of Sklearn library, they are namely:

- `train_test_split` from `model_selection` module of `sklearn`
- `from sklearn.linear_model import LinearRegression`
- `metrics` module of `sklearn`

`Model_selection` is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results

when making a prediction.

`train_test_split` is a function in sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

With this function you don't need to divide the dataset manually. By default, sklearn `train_test_split` will make random partitions for the two subsets. However, you can also specify a random state for the operation.

→ Linear Regression

Linear Regression is a linear model that assumes a linear relationship between input variables (independent variables 'x') and output variable (dependent variable-'y') such that 'y' can be calculated from a linear combination of input variables (x). For single input variable, method is referred to as Simple Linear Regression whereas for multiple input variables it is referred to as Multiple Linear Regression.

FIG 4.1 Describes the code to implement all the mentioned libraries.

```
In [1]:  import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

Fig 4.1: code for importing libraries

We will go deeper into the uses of all the libraries and modules mentioned above as we go through the coding phase.

4.4 Importing the data set from .csv file

As we discussed earlier we will import the data set which was saved in csv (.csv extension). We will use pandas to do it, I will use two different dataframe to import data set into. One for the dataset used for training, testing, model and predictions and the other for visualization. Fig 4.2 will show the code to import an excel file into a pandas dataframe.


```
In [2]: M df = pd.read_csv(r'C:\Users\Ishita\Documents\Project\train.csv')
df.head(5)

Out[2]:
```

	Employee ID	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
0	mfe32003000300033003200	30-09-00	Female	Service	No	2	3.0	3.8	0.15
1	mfe3700300033003500	20-11-08	Male	Service	Yes	1	2.0	5.0	0.36
2	mfe31003300320037003000	10-03-00	Female	Product	Yes	2	NaN	5.8	0.49
3	mfe32003400300032003000	03-11-00	Male	Service	Yes	1	1.0	2.6	0.20
4	mfe31003000340031003000	24-07-00	Female	Service	No	3	7.0	6.9	0.52

```
In [3]: M df.shape
Out[3]: (22750, 9)
```

Fig 4.2: code for importing data set

4.5 Data Pre-Processing

Data processing or data handling is one of the important steps needed to be taken before using the data from the given data set. Data processing is the task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative. Using Machine learning algorithms, mathematical modeling, and statistical knowledge, this entire process can be automated. The output of the process can be in any desired form like graphs, videos, charts, tables, images, and many more, depending on the task we are performing and the requirements of the machine.

The classic steps for data processing are as follows:

- Collection
- Preparation
- Input
- Processing
- Output
- Storage

Collection:

The most crucial step is to have the data of good quality and accuracy. The data we will be using is collected from UCL dataset for airfoil self-noise. Good data ensures that the results of the model are valid and can be trusted upon.

Preparation:

The collected data can be in a raw form which can't be directly fed to the machine. So, this is a process of collecting data sets from different sources, analyzing the data sets and then constructing a new data set for further processing and exploration. This preparation can be performed either manually or from the automatic approach. Data can also be prepared in numeric forms also which would fasten the model's learning.

For this preparation we have to first check for null values, for our self-noise data set we are lucky that we do not have any null values.

We can also see the data frame in a more detailed way in two different ways but the commands which is shown in fig 4.3 such as `data.head()` - which will just give us the generic top values of every column in the data set.

```
df.head(5)

Out[2]:
```

	Employee ID	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
0	mfe32003000360033003200	30-09-08	Female	Service	No	2	3.0	3.8	0.18
1	mfe3700360033003500	30-11-08	Male	Service	Yes	1	2.0	5.0	0.36
2	mfe31003300320037003900	10-03-08	Female	Product	Yes	2	NaN	5.8	0.49
3	mfe32003400380032003900	03-11-08	Male	Service	Yes	1	1.0	2.6	0.20
4	mfe31003900340031003600	24-07-08	Female	Service	No	3	7.0	6.9	0.52

```
In [3]: df.shape
Out[3]: (22750, 9)
```

Fig 4.3 - Describing data.head()

Example for preparing the data if you had null values or other irrelevant data entries in the data set, fig 4.4 and 4.5 describes the code for the data prep work for null values:

I have encountered a scenario for another data set where they had to remove the null values, or there were some other values entered in the data instead of null values like "?" and we had to replace those with null values and finally remove those null values. Firstly I have dropped the missing values in the burn rate column and filled the missing values in resource allocation and mental fatigue score columns with its mean values. We can replace the continuous variables into some integer type if those variables are finite, like in the below example fig 4.6, for the gender and WFH available column I have done binary encoding by changing them into 0's and 1's.

```

In [6]: M #Checking the null items
df.isna().sum()

Out[6]: Date of Joining      0
Gender                      0
Company Type                0
WFH Setup Available         0
Designation                 0
Resource Allocation         1381
Mental Fatigue Score        2117
Burn Rate                   1124
dtype: int64

In [7]: M #Dropping rows with missing vales
df = df.dropna(subset=['Burn Rate'], axis=0).reset_index(drop=True)
df.shape

Out[7]: (21626, 8)

In [8]: M #Filling the remaining columns missing values with median
df = df
for col in ['Resource Allocation', 'Mental Fatigue Score']:
    df[col] = df[col].fillna(df[col].median())

In [9]: M df = df.drop('Date of Joining', axis=1)
df.head()

Out[9]:
   Gender  Company Type  WFH Setup Available  Designation  Resource Allocation  Mental Fatigue Score  Burn Rate
0  Female      Service                No          2             3.0              3.8         0.18
1   Male      Service                Yes          1             2.0              5.0         0.36
2  Female      Product                Yes          2             4.0              5.8         0.49

```

Fig 4.4: Example code for the prep-work of data set with null values

```

In [10]: M df.isna().sum()

Out[10]: Gender      0
Company Type      0
WFH Setup Available  0
Designation       0
Resource Allocation 0
Mental Fatigue Score 0
Burn Rate         0
dtype: int64

In [11]: M #Binary encoding
df['Gender'] = df['Gender'].replace({'Female' : 1, 'Male' : 0})
df['Company Type'] = df['Company Type'].replace({'Product' : 1, 'Service' : 0})
df['WFH Setup Available'] = df['WFH Setup Available'].replace({'Yes' : 1, 'No' : 0})

In [12]: M df.head()

Out[12]:
   Gender  Company Type  WFH Setup Available  Designation  Resource Allocation  Mental Fatigue Score  Burn Rate
0      1           0           0              2             3.0              3.8         0.18
1      0           0           1              1             2.0              5.0         0.36
2      1           1           1              2             4.0              5.8         0.49
3      0           0           1              1             1.0              2.6         0.20
4      1           0           0              3             7.0              6.9         0.52

```

Fig 4.5: Example code for the prep-work of data set with null values

Input:

Now the prepared data can be in the form that may not be machine-readable, so to convert this data to the readable form, some conversion algorithms are needed. For

this task to be executed, high computation and accuracy is needed.

For our self-noise data set we will now divide the dataframe into two parts, one with all the columns we know of the data to the other with the column of values for which we have to make the prediction. As stated in chapter 4.3 there are 7 columns in which the first six describe the means of the burn rate and for the last column we have to predict the values.

```
In [13]: # Splitting df into x and y
y = df['Burn Rate']
x = df.drop('Burn Rate', axis=1)

# train_test split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.2, shuffle=True, random_state=1)
x_train, x_test, y_train, y_test
```

```
Out[13]: (      Gender  company Type  WH Setup Available  Designation \
2426         0         0         0         1         3
16517        1         0         0         0         2
3341         1         0         0         0         3
270          1         0         0         0         4
15528         0         0         1         1         0
12216         1         0         1         1         2
14273         1         1         0         1         1
17363         1         1         1         1         2
6326         0         0         0         0         4
7518         0         0         1         1         1
20961         1         0         1         1         2
13487         0         0         0         0         2
11058         1         0         1         1         2
8107          0         0         0         0         1
1019          0         0         1         1         1
14261         1         0         1         1         5
14179         1         0         1         1         3
7098         1         0         0         0         3
10500         0         1         1         1         2
```

Fig 4.6: code for splitting dataframe into two parts based on the column values

Processing:

In this stage, results are procured by the machine in a meaningful manner which can be inferred easily by the user. Output can be in the form of reports, graphs, videos, etc. We will go through this phase in the next part.

CHAPTER 5

RESULT, PERFORMANCE ANALYSIS

5.1. RESULT:

We have successfully analyzed the data set and predicted the burn rate of the employees using Linear Regression. Then we successfully done Data Visualization using seaborn. We have plotted the graphs in different formats like scatter plots.

5.2. PERFORMANCE ANALYSIS:

Performance analysis is the technique of studying or comparing the performance of a specific situation in contrast to the aim and yet executed. In Human Resource, performance analysis can help to review an employee's contribution towards a project or assignment, which they allotted him or her.

We distinguish three basic steps in the performance analysis process: data collection, data transformation, and data visualization. Data collection is the process by which we get data about program performance from an executing program.

I had used countplot technique to count the values from data and plot the stripplot techniques to plot the graph in scatter graph.

Save fig is also a method to save the plotted graphs to our files.

Figsizes can be used to present the graph in a particular size we need.

5.3 *Splitting data for training and testing*

As already discussed in chapter 4.3, we will use `train_test_split` that is imported from the `model_selection` of `sklearn` library. We have already discussed about the `model_selection` module and its use, now we will understand about the `train_test_split` function:

`Sklearn train_test_split` has several parameters. A basic example of syntax will look like: `[ train_test_split(X, y, train_size = 0.8, test_size = 0.2, random_state = *) ]`.

- `X, y` - it is the dataset you are selecting to use.
- `train_size` - this parameter sets the size of the training data set. There are three options: `None` - which is default, `int` - which requires exact number of

samples and Float - which ranges from 0.1 to 1.0.

- `test_size` - this parameter specifies the size of the testing data set. The default state suits the training size. It will be set to 0.25 if the training size is set to default.
- `random_state` - the default mode performs a random split using `np.random`. Alternatively, you can add an integer using an exact number.

As shown in fig 5.1, I have preferred to keep the `train_size` to be default and the `test_size` to be 0.2, so that it will split the data set into that ratio so that I can get accurate predictions and also preferred to keep the `random_state` to 1. You can see the shapes of the train and test data set in the result of the fig 5.1.

```
In [13]: # Splitting df into X and Y
y = df['Burn Rate']
x = df.drop('Burn Rate', axis=1)

# Train test split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.2, shuffle=True, random_state=1)
x_train, x_test, y_train, y_test
```

	Gender	Company Type	WFH Setup Available	Designation \
2426	0	0	1	3
16517	1	0	0	2
3341	1	0	0	3
270	1	0	0	4
15528	0	0	1	0
12216	1	0	1	2
14273	1	1	0	1
17363	1	1	1	2
6326	0	0	0	4
7518	0	0	1	1
20961	1	0	1	2
13487	0	0	0	2
11058	1	0	1	2
8107	0	0	0	2
1019	0	0	1	1
14261	1	0	1	5
14179	1	0	1	1
7098	1	0	0	3
10508	0	1	1	2

Fig 5.1: code for splitting the dataset into training and testing.

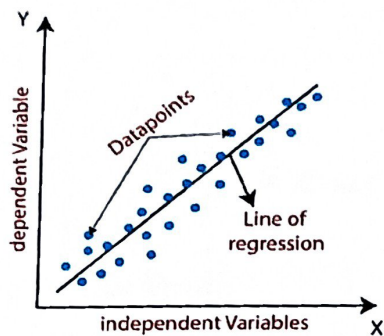
5.4 Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price, etc.**

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:



$$\rightarrow y = a_0 + a_1x + \epsilon$$

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a_0 = intercept of the line (Gives an additional degree of freedom)

a_1 = Linear regression coefficient (scale factor to each input value).

ϵ = random error

The values for x and y variables are training datasets for Linear Regression model representation.

5.5 Fitting the algorithm and making predictions

To fit the LinearRegression algorithm to the model we can simply do as shown in fig 5.2, and similarly to make the prediction using the model, do as shown in the same fig.

```
In [16]: from sklearn.linear_model import LinearRegression
emp = LinearRegression()
emp.fit(x_train, y_train)

Out[16]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [17]: y_pred = emp.predict(x_test)
y_pred

Out[17]: array([0.15613814, 0.52853132, 0.42575767, ..., 0.83384877, 0.14814639,
0.23077081])
```

Fig 5.2: code to demonstrate fitting the algorithm to the model.

As instructed above the resulting array of the code for values prediction is shown in fig 5.3

```
In [17]: M y_pred = emp.predict(x_test)
          y_pred
Out[17]: array([0.15613814, 0.52853132, 0.42575767, ..., 0.83384877, 0.14814639,
                0.23077081])
```

Fig 5.3: Resulting array of predictions made

5.6 Checking for Errors

Errors are one of the ways to get an idea of how we have performed till now when compared to the initial values.

As part of the project we were asked to find out Mean Squared Error. And these can be found easily using the metrics module which is imported from sklearn library.

Here as shown in fig 5.4, we can see from the documentation which function to be used to find Mean Absolute Error and Mean Squared Error respectively.

```
In [20]: M from sklearn.metrics import mean_squared_error
          mean_squared_error(y_test, y_pred)
Out[20]: 0.00508859669854793
In [21]: M print('Mean squared error: %.2f'
          % mean_squared_error(y_test, y_pred))
Mean squared error: 0.01
```

Fig 5.4: Calculating the Mean Squared Error

5.7 Calculating the Coefficient of Determination (R^2 Score)

```
In [18]: M from sklearn.metrics import r2_score
          r2_score(y_test, y_pred)
Out[18]: 0.8705651919089659
In [19]: M print('Coefficient of determination ( $R^2$  Score): %.2f'
          % r2_score(y_test, y_pred))
Coefficient of determination ( $R^2$  Score): 0.87
```

Fig 5.5: Calculating the R^2 Score

Here we have obtained **87% accuracy** which is considered as a very good precision value, also we have to state that data set and data splitting (test and train data) plays a huge role in making good predictions.

Why does a model improve with more data? The best way to answer this is to think in terms of how humans learn. We increase our knowledge of the world through experiences, and the more times we practice a skill, the better we get.

As we used a machine learning technique, a machine learning model also “learns from experience” in the sense that each time it looks at another training data point, it learns a little more about the relationships between the features and labels. Assuming that there are relationships in the data, giving the model more data will allow it to better understand how to map a set of features to a label.

For our case, as the model sees more observations of factors that cause burn-rate, it better understands how to take those observations and predict the employee burnout. Practice improves human abilities and machine learning model performance alike.

5.8 Data Visualization using Matplot and Seaborn.

```
In [23]: sns.distplot(df['Burn Rate'])  
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x20debb1b8d0>
```

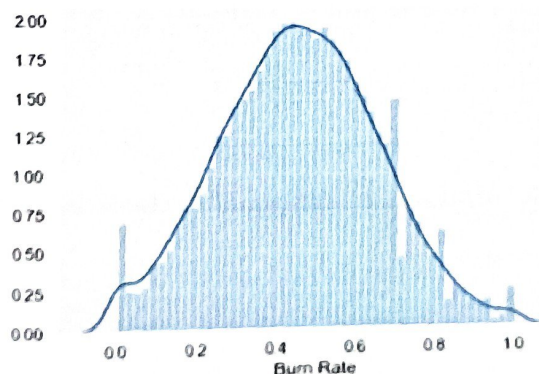


Fig 5.6: Plot of Burn Rate

```
In [24]: sns.scatterplot(x=df['Mental Fatigue Score'], y=df['Burn Rate'])
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x20debc74630>
```

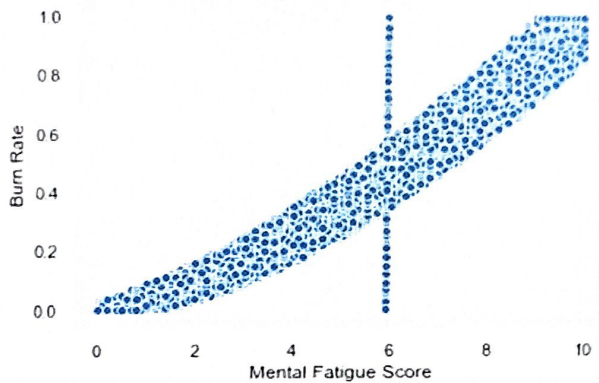


Fig 5.7: Scatterplot of Mental fatigue Score and Burn Rate

```
In [25]: sns.scatterplot(x="Resource Allocation", y="Burn Rate", hue="Gender", data=df, linewidth=0.0)
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x20debcfc5f8>
```

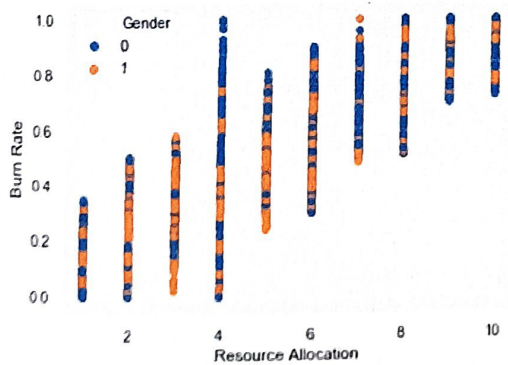


Fig 5.8: Scatter Plot of Resource Allocation, Burn Rate and Gender


```
In [26]: sns.relplot(x="Burn Rate", y="Mental Fatigue Score", hue="WFH Setup Available", data=df)
Out[26]: <seaborn.axisgrid.FacetGrid at 0x20deb9a0e10>
```

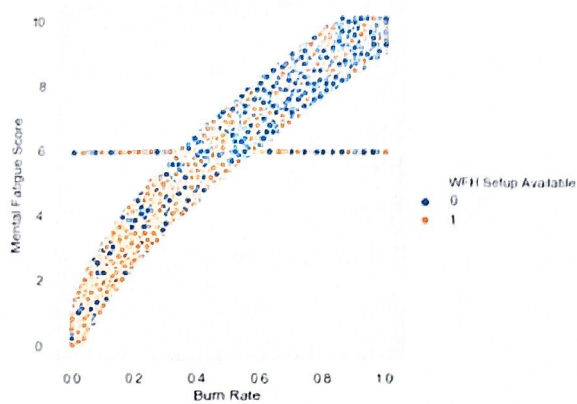


Fig 5.9: Scatterplot of Mental fatigue Score, Burn Rate and WFH Setup Available

CHAPTER 6

SUMMARY AND CONCLUSION

We have used all type of python techniques and imported different type of libraries from python and used the jupyter platform to implement the libraries and packages. I have selected the data set from kaggle which contains data related Employees details and their information. We had even visualized and compared the burn rate of the employees. These all analytics and visualizations are done using python and python imported packages We have used matplotlib, seaborn, plotly, numpy, pandas and many other types of visualization techniques using Python in Jupyter Notebook (Anaconda 3) Software.

We can conclude that all the different type of analytics and visualizations are shown through Python techniques. We have illustrated the .csv file which contains the data of Employees burn out. We have used different type of packages that are imported from Python and implemented them to visualize the data set. We can use any type of data set and even .tsv files to data visualize the information in the formats we like using Python.

APPENDIX

A. Source Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
df = pd.read_csv(r'C:\Users\lshita\Documents\Project\train.csv')
df.head(5)
df.shape
df.info()
#Dropping the employee ID Column
df = df.drop('Employee ID', axis=1)
#Checking the null items
df.isna().sum()
#Dropping rows with missing vales
df = df.dropna(subset=['Burn Rate'], axis=0).reset_index(drop=True)
df.shape
#Filling the remaining columns missing values with median
df = df
for col in ['Resource Allocation', 'Mental Fatigue Score']:
    df[col] = df[col].fillna(df[col].median())
df = df.drop('Date of Joining', axis=1)
df.head()
df.isna().sum()
#Binary encoding
df['Gender'] = df['Gender'].replace({'Female' : 1, 'Male' : 0})
df['Company Type'] = df['Company Type'].replace({'Product' : 1, 'Service' : 0})
df['WFH Setup Available'] = df['WFH Setup Available'].replace({'Yes' : 1, 'No' : 0})
df.head()
#Splitting df into X and Y
y = df['Burn Rate']
x = df.drop('Burn Rate', axis=1)

#Train_test split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.2,
```



```

shuffle=True, random_state=1)

x_train, x_test, y_train, y_test
x_train
y_train
from sklearn.linear_model import LinearRegression
emp = LinearRegression()
emp.fit(x_train, y_train)
y_pred = emp.predict(x_test)
y_pred
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
print('Coefficient of determination (R^2 Score): %.2f'
      % r2_score(y_test, y_pred))
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
print('Mean squared error: %.2f'
      % mean_squared_error(y_test, y_pred))
import seaborn as sns
%matplotlib inline
sns.set(color_codes=True)
sns.distplot(df['Burn Rate'])
sns.scatterplot(x=df['Mental Fatigue Score'], y=df['Burn Rate'])
sns.scatterplot(x="Resource Allocation", y="Burn Rate", hue="Gender",
data=df, linewidth=0.0)
sns.relplot(x="Burn Rate", y="Mental Fatigue Score", hue="WFH Setup
Available", data=df)
sns.scatterplot(x="Resource Allocation", y="Burn Rate", hue="Gender",
data=df, linewidth=0.0)

```

B. Screen Shots

Fig no.6.1: Importing libraries and dataset

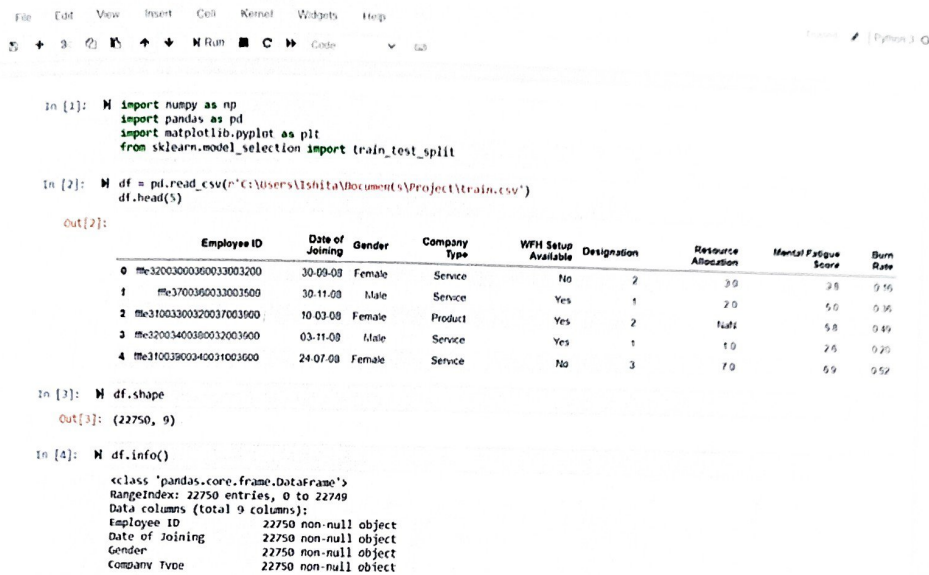


Fig.no.6.2: Example code for prep-work and filling up null values

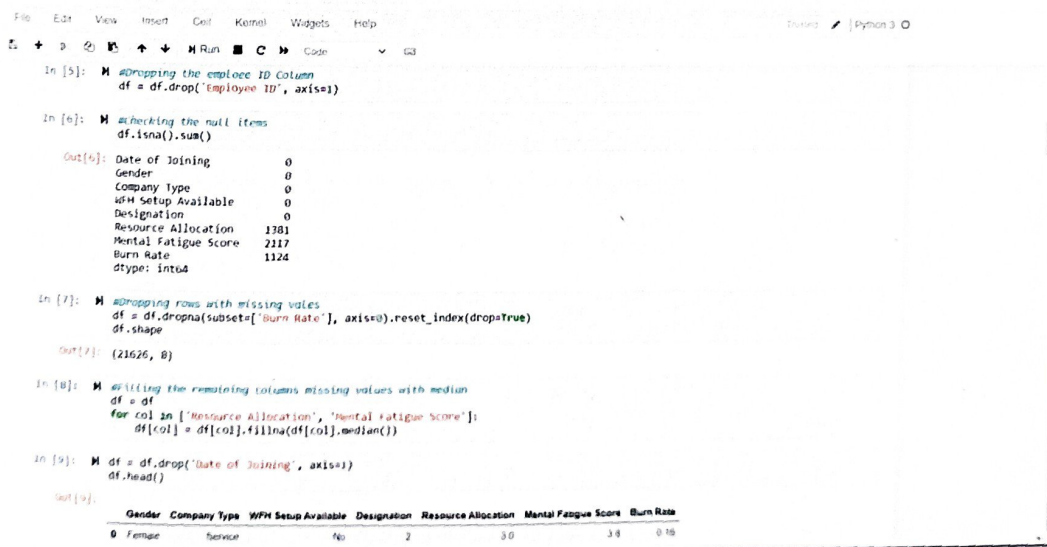


Fig no.6.3: Result for the prep-work of dataset with null values

```

File Edit View Insert Cell Kernel Widgets Help
df[col] = df[col].fillna(df[col].median())

In [9]: M df = df.drop('Date of Joining', axis=1)
df.head()

Out[9]:

```

	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
0	Female	Service	No	2	30	38	0.16
1	Male	Service	Yes	1	20	50	0.36
2	Female	Product	Yes	2	40	58	0.49
3	Male	Service	Yes	1	10	26	0.20
4	Female	Service	No	3	70	69	0.52

```

In [10]: M df.isna().sum()

Out[10]: Gender      0
Company Type      0
WFH Setup Available  0
Designation       0
Resource Allocation 0
Mental Fatigue Score 0
Burn Rate         0
dtype: int64

In [11]: M #Binary encoding
df['Gender'] = df['Gender'].replace({'Female': 1, 'Male': 0})
df['Company Type'] = df['Company Type'].replace({'Product': 1, 'Service': 0})
df['WFH Setup Available'] = df['WFH Setup Available'].replace({'Yes': 1, 'No': 0})

In [12]: M df.head()

Out[12]:

```

Fig no.6.4: Splitting the dataset into two subset

```
File Edit View Insert Cell Kernel Widgets Help
+ 3 2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
In [14]: M_x_train
Out[14]:
```

	Gender	Company Type	WPH Setup Available	Designation	Resource Allocation	Mental Fatigue Score
2428	0	0	1	3	40	59
18517	1	0	0	2	60	62
3341	1	0	0	3	60	69
270	1	0	0	4	70	72
16528	0	0	1	0	20	45
12216	1	0	1	2	40	59
14273	1	1	0	1	20	47
17363	1	1	1	2	50	63
6328	0	0	0	4	80	82
7618	0	0	1	1	20	36
20961	1	0	1	2	40	62
13487	0	0	0	2	40	67
11058	1	0	1	2	30	61
8107	0	0	0	2	50	59
1018	0	0	1	1	50	33
14261	1	0	1	5	80	90
14178	1	0	1	1	20	59
7088	1	0	0	3	60	58
19608	0	1	1	2	40	50
7630	1	0	1	1	10	41
6460	1	0	1	1	10	41


```
In [15]: y_train
Out[15]:
2426 0.50
16517 0.53
3341 0.59
270 0.69
15528 0.33
12216 0.50
14273 0.39
17363 0.41
6316 0.21
7518 0.21
20401 0.41
11047 0.67
11058 0.44
8307 0.68
1029 0.15
14261 0.75
14179 0.10
7098 0.42
19508 0.25
7630 0.28
9152 0.54
13330 0.21
7130 0.50
14392 0.42
10270 0.57
10334 0.33
5708 0.18
16950 0.69
13222 0.60
9543 0.70
...
19281 0.31
```

```
In [12]: df.head()
```

```
Out[12]:
  Gender  Company Type  WFH Setup Available  Designation  Resource Allocation  Mental Fatigue Score  Burn Rate
0      1            0              0           2             30              38           0.16
1      0            0              1           1             20              60           0.36
2      1            1              1           2             40              58           0.49
3      0            0              1           1             10              26           0.20
4      1            0              0           3             70              69           0.52
```

```
In [13]: #splitting df into x and y
```

```
y = df['Burn Rate']
x = df.drop('Burn Rate', axis=1)
```

```
#train test split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.2, shuffle=True, random_state=1)
```

```
x_train, x_test, y_train, y_test
```

```
Out[13]: (
  Gender  Company Type  WFH Setup Available  Designation \
2426      0            0              1           3
16517      1            0              0           2
3341       1            0              0           3
270        1            0              0           4
15528      0            0              1           0
14273      1            0              1           2
17363      1            1              0           1
6316       0            0              1           2
7518       0            0              0           4
20401      0            0              1           1
7098       1            0              1           2
...      ...      ...      ...      ...
19281      1            0              1           2
)
```

Fig.no.6.5: code to demonstrate fitting the algorithm to the model

```
In [16]: from sklearn.linear_model import LinearRegression
emp = LinearRegression()
emp.fit(x_train, y_train)

Out[16]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [17]: y_pred = emp.predict(x_test)
y_pred

Out[17]: array([0.15613814, 0.52853132, 0.42575767, ..., 0.83388877, 0.18816639,
0.23077081])

In [18]: from sklearn.metrics import r2_score
r2_score(y_test, y_pred)

Out[18]: 0.8705651948089659

In [19]: print('Coefficient of determination (R^2 Score): %.2f'
% r2_score(y_test, y_pred))

Coefficient of determination (R^2 Score): 0.87

In [20]: from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)

Out[20]: 0.00508859669854793

In [21]: print('Mean squared error: %.2f'
% mean_squared_error(y_test, y_pred))

Mean squared error: 0.01
```

Fig no.6.6: Plot of Burn Rate

```
In [23]: sns.distplot(df['Burn Rate'])

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x20debb1b8d0>
```

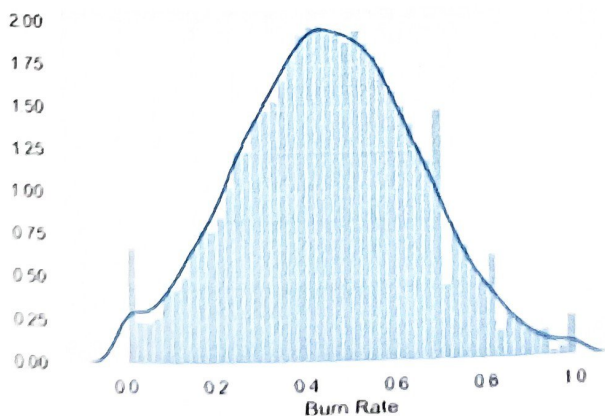


Fig no.6.7: Scatterplot of Mental fatigue Score, Burn Rate and WFH Setup Available

```
In [26]: M sns.relplot(x="Burn Rate", y="Mental Fatigue Score", hue="WFH Setup Available", data=df)
Out[26]: <seaborn.axisgrid.FacetGrid at 0x20deb9a0e10>
```

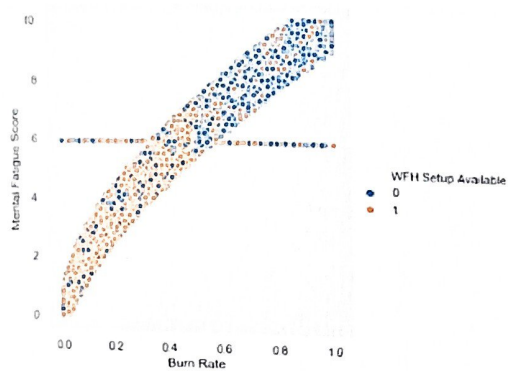


Fig no.6.8: Scatter Plot of Resource Allocation, Burn Rate and Gender

```
In [25]: M sns.scatterplot(x="Resource Allocation", y="Burn Rate", hue="Gender", data=df, linewidth=0.0)
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x20deb9c5f8>
```

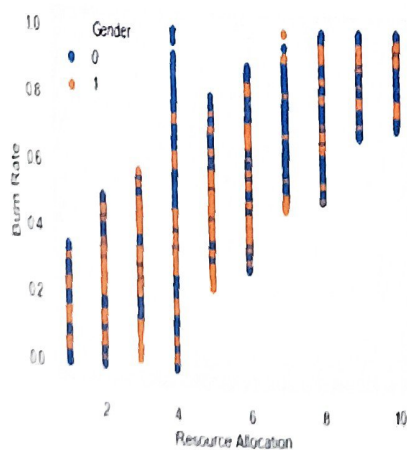
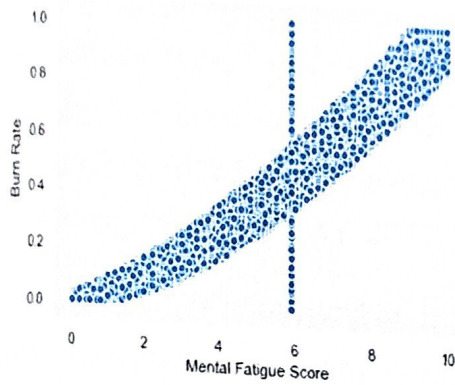


Fig no.6.9: Scatterplot of Mental fatigue Score and Burn Rate

```
In [24]: sns.scatterplot(x=df['Mental Fatigue Score'], y=df['Burn Rate'])
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x20debc74630>
```



REFERENCES

1. Arnold B. Bakker and Patricia L. Costa, "Chronic job burnout and daily functioning: A theoretical analysis", *Burnout Research*, vol. 1, no. 3, pp. 112-119, 2014.
2. Kirsi Ahola, Salla Toppinen-Tanner and Johanna Seppänen, "Interventions to alleviate burnout symptoms and to support return to work among employees with burnout: Systematic review and metaanalysis", *Burnout research*, vol. 4, pp. 1-11, 2017.
3. P. Zhernova, Y. Bodyanskiy, B. Yatsenko and I. Zavgorodnii, "Detection and prevention of professional burnout using machine learning methods", In *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics Telecommunications and Computer Engineering (TCSET)*, pp. 218-221, 2020, February.
4. Oussama Batata, Vincent Augusto and Xiaolan Xie, "Caregivers burnout prediction using supervised learning", In *2018 IEEE International Conference on Systems Man and Cybernetics (SMC)*, pp. 1749-1754, 2018.
5. U. S. Reddy, A. V. Thota and A. Dharun, "Machine learning techniques for stress prediction in working employees", In *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1-4, 2018, December
6. Wendy Sanchez, Alicia Martinez and Miguel Gonzalez, "Towards job stress recognition based on behavior and physiological features", *International conference on ubiquitous computing and ambient intelligence*, 2017.
7. Henry R. Young, David R. Glerum, Wei Wang and Dana L. Joseph, "Who are the most engaged at work? A meta-analysis of personality and

employee engagement", *Journal of Organizational Behavior*, vol. 39, no. 10, pp. 1330-1346, 2018.

8. Cho Yoon Jik and Hyun Jin Song, "Determinants of turnover intention of social workers: Effects of emotional labor and organizational trust", *Public Personnel Management*, vol. 46, no. 1, pp. 41-65, 2017.
9. Nina Geuens, Peter Van Bogaert and Erik Franck, "Vulnerability to burnout within the nursing workforce—The role of personality and interpersonal behaviour", *Journal of clinical nursing*, vol. 26, no. 23-24, pp. 4622-4633, 2017.
10. Anis Khedhaouria and Alain Cucchi, "Technostress creators personality traits and job burnout: A fuzzy-set configurational analysis", *Journal of Business Research*, vol. 101, pp. 349-361, 2019.