A

MINI PROJECT REPORT

on

PREDICTING EMPLOYEES UNDER STRESS FOR PRE – EMPTIVE REMEDIATION USING MACHINE LEARNING ALGORITHM.

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

BATCH - 1

V. ABHINAV SAI :207Y1A1201

S. DIVYA :207Y1A1212

Under the guidance

of

Mr.CH.V.V.NARASIMHA RAJU

Assistant professor

DEPARTMENT OF INFORMATION TECHNOLOGY

MARRI LAXMAN REDDY

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(AUTONOMOUS)

(Affiliated to JNTUH, Approved by AICTE New Delhi and Accredited by NBA &NAAC With A grade)

(October 2023)

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

CERTIFICATE

This is to certify that the project titled **"Predicting Employees under stress for pre-emptive Remediation using Machine Learning"** is being submitted by **V. ABHINAV SAI (207Y1A1201)** in IV B. tech I semester in **INFORMATION TECHNOLOGY** is a record bonafide work carried out by him/her. The results embodied in this report have not been submitted to any other university for the award of any degree.

INTERNAL GUIDE **HOD.**

Principal. **External Examiner.**

DECLARATION

I here by declare that the Mini Project report entitled,**"Predicting Employees under stress for pre-emptive Remediation using Machine Learning"** submitted for the B.Tech degree is entirely my work and all ideas and references have been duly acknowledged.It does not contain any work for the award of any other degree.

Date:

V.ABHINAV SAI
(207Y1A1201)

ACKNOWLEDGEMENT

I am happy to express my deep sense of gratitude to the principal of the college **Dr.K.Venkateswara Reddy,** Professor, Department of Computer Science and Engineering, Marri Laxman Reddy Institute of Technology & Management, for having provided with adequate facilities to pursue my project.

I would like to thank **Dr.M.Nagalakshmi,** Assoc. Professor and Head, Department of Information technology, Marri Laxman Reddy Institute of Technology & Management, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

I am very grateful to my project guide **Mr.CH.V.V.Narasimha Raju,** Asst. Prof.,Department of Information Technology (IT), Marri Laxman Reddy Institute of Technology & Management, for his extensive patience and guidance throughout my project work.

I sincerely thank my seniors and all the teaching and non-teaching staff of the Department of Computer Science for their timely suggestions, healthy criticism and motivation during the course of this work.

I would also like to thank my classmates for always being there whenever I needed help or moral support. With great respect and obedience, I thank my parents and brother who were the backbone behind my deeds.

Finally, I express my immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to my need at the right time for the development and success of this work.

Abstract

With the ongoing COVID-19 pandemic, businesses and organizations have acclimated to unconventional and different working ways and patterns, like working from home, working with limited employees at office premises. With the new normal here to stay for the recent future, employees have also adapted to different working environments and customs, which has also resulted in psychological stress and lethargy for many, as they adapt to the new normal and adjust their personal and professional lives. In this work, data visualization techniques and machine learning algorithms have been used to predict employees stress levels. Based on data,we can develop a model that will assist to predict if an employee is likely to be under stress or not.

Many research has been done on this stress prediction. Most of the research papers use ML techniques for stress prediction. Many papers just present an idea of stress prediction, but no implementation is done. There are some search papers where implementation is done.These implementation papers use some ready tools such as WEKA tool, R Tool, Rapid Miner or Programming languages such as PYTHON or R Language. Using these ready tools and languages. It is easy to predict stress as they support ready libraries for stress prediction. ML techniques are efficient for processing training datasets and can predict human stress in less time with better results.

Here, the XGB classifier is used for the prediction process and the results are presented showing that the method facilitates getting a more reliable model performance. After performing interpretation utilizing XGB classifier it is determined that working hours, workload, age, and, role ambiguity have a significant and negative influenceon employee performance. The additional factors do not hold much significance when associated to the abovediscussed. Therefore, It is concluded that concluded that increasing working hours, role ambiguity, the workload would diminish employee representation in all perspectives.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

ABBREVATION	DESCRIPTION
KNN	K Nearest Neighbour
SVM	Support Vector Machine
API	Application Programming Interface
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
MLA	Multimodal Learning Analytics
UML	Unified Modelling Language

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION

On March 11, 2020 [3,] the WHO classified corona virus (COVID-19) a pandemic. It's now safe to assume that the virus has reached every corner of the globe and is generating widespread terror. The corona virus is responsible for the extremely contagious sickness known as COVID-19. The 'Corona virus' family is very diverse, including both cold and fatal viruses. According to WHO, 202 nations have confirmed cases of the virus as of March 31, 2020. The stock market and other sectors of economic development have slowed significantly as a result of this. Stress in the job may result from prolonged uncertainty and pressure. Many promising advances have been made in using machine learning and AI in commercial settings. The regular activities of employees are dissected and analysed.

Because of the long hours and heavy workload, they are unhappy with their current circumstances. The main focus of this research is to see how internal stress manifests in people's appearance on the job. Because of this, people become less healthy overall and less motivated to do their jobs. Thanks to COVID-19, however, mankind finds itself in a situation it has never been in before. The purpose of this paper is to investigate the toll that disasters like the current pandemic have on the workforce. Using machine learning techniques to determine whether a worker is anxious.

Leveraging advanced machine learning techniques, we aim to ascertain whether the signs of anxiety and stress can be detected in the behavior and appearance of employees on the job. In an era where technology offers promising avenues for understanding human experiences, we embark on a journey to uncover the hidden dimensions of workplace stress, offering insights that can inform interventions and support mechanisms for a healthier and more resilient workforce.

1.2 PROBLEM STATEMENT

The ongoing COVID-19 pandemic has ushered in a "new normal" in the way businesses operate, with many employees working from home or in limited-capacity office settings. This shift in working conditions and practices has introduced psychological stress and lethargy among employees as they navigate these changes in their personal and professional lives.

In response to this challenge, our project aims to utilize data visualization techniques and machine learning algorithms to predict employees' stress levels accurately. By analyzing various factors, we intend to develop a predictive model that can determine whether an employee is likely to be under significant stress. We have chosen the XGBoost (XGB) classifier for this predictive process, as it demonstrates superior model performance.
In summary, this project seeks to shed light on the impact of the COVID-19 pandemic on employee stress levels and performance by leveraging machine learning techniques. The insights gained will inform strategies for mitigating workplace stress and improving the overall quality of work life during these challenging times.

1.3 EXISTING SYSTEM

Workplace stress is a thriving concern for employees like human resource managers and so on. Although considerable scholarly and practical awareness has been dedicated to stress management over the years, the time has come for distinct perspectives and research. Extracting from the emerging field of organizational performance, this research proposes analysis conclusions including implications for combating occupational stress.

Specifically, data from a large sample of working employees beyond a variety of organizations and industries suggest that positive resources of efficacy, optimism, and resilience may be key to better understanding adaptation in perceived manifestations of stress. Numerous investigations and experimentation has been done inthe last few years, most of the studies have been conveyed in countries that endeavor to promote to enhance advanced economically and socially. stress has become one of the most widespread ‘occupational disorders ‘Aloft the past years, close to 3 billion employees‟ are experiencing stress at their workplace and it is influencing their overall job performances on regular basis.

1.4 PROPOSED SYSTEM

One of the best important steps while dealing with data is cleaning the data. Without doing that, if we go for model execution, we can not obtain better performance in model execution. Therefore, it ought to deal with nullvalues, zeros, NAN values.

Here, data has 3895 null values. The categorical null values are fixed by using mode, and numeric data is fixed by using mean, median, floor techniques. Here, altogether, it can drop

null values also, but by doing that, it maydissipate some data. So that is preferred for good model execution. This is the target variable count. By this, it shows that data is imbalanced and it can be balanced by using various techniques. It is also possible to implement resampling data, oversampling, under-sampling data, or either it can be done by using ‘smote’, hyperparameter techniques. It can also be overcome by using better algorithms that can give the most reliable performance model or can be done by using bagging and boosting techniques, algorithms like SVM Classifier, Logistic regression, right evaluation metrics, changing performances metrics, ROC curve.

ADVANTAGES

- The goal of the system is to test and train the large number of datasets with high accuracy.
- The proposed system developed a Machine Learning Algorithms to test and train the datasets.

CHAPTER 2

LITERATURE REVIEW

Many academics are intrigued by Predicting Employees under stress for pre-emptive Remediation using Machine Learning. Workplace stress is a thriving concern for employees like human resource managers and so on. Although considerable scholarly and practical awareness has been dedicated to stress management over the years, the time has come for distinct perspectives and research. Extracting from the emerging field of organizational performance, this research proposes analysis conclusions including implications for combating occupational stress. Specifically, data from a large sample of working employees beyond a variety of organizations and industries suggest that positive resources of efficacy, optimism, and resilience may be key to better understanding adaptation in perceived manifestations of stress.

The G. Azar et al. They have used various machine learning approaches and intelligent genetic algorithm to build a semi-automated system. They have compared the person's mental health with the DSMIV-TR. The further aim is to make the system fully automated. Through this experiment, they have proved that genetic algorithms can be applied for many real-time applications.

The goal of Fang Li was to forecast student stress. In light of this, the author integrated resources for mental health education into the cloud using data mining and a cloud platform, allowing them to share each other's high-quality resources. The author also covered three elements that affects the mental health of college students i.e., students, society and education. The author proposed several ways to support the management of college students' psychological health through the examination of the management system for the psychological health of students.

A.R. Subhani et al. used various machine learning frameworks to analyze and predict the levels of stress. The analysis of stress included the use of EEG signals. They proposed to implement Logistic Regression, Support Vector Machine, and Naive Bayes, as well as EEG feature extraction. The experiment's findings have provided the best accuracy for stress prediction. In the experiment, accuracy for level 2 stress was 94.6 percent, and accuracy for multi-level stress was 83.4 percent.

Aditya Vivek Thota et al. aimed at predicting stress of IT employees. Working professionals in the tech industry who participated in the OSMI mental health survey provided the data. The

best accuracy was achieved by boosting (75%), while the lowest accuracy was attained by bagging (69.43%). Among the other models Logistic Regression acquired 73%, KNN attained an accuracy of 73%, Decision Tree attained 70%, Random Forest attained 73%. The cross-validated AUC value for the random forest classifier was higher, indicating a more stable model.

Sandhya P et al. used various machine learning approaches to predict stress in IT Employees. The dataset was taken in the form of a questionnaire where employees were asked to fill in the details. The best accuracy was achieved by boosting (81.7%), while the lowest accuracy was attained by bagging (77.7%). Among the other models Logistic Regression acquired 79.9%, KNN attained an accuracy of 80.4%, Decision Tree attained 80.6%, and Random Forest attained 81.2%.

The goal of Monisha S. et al. is to forecast student stress. In order to assess reliable data and organize the components most likely to cause stress based on probabilistic characteristics, the authors employed the Naive Bayes technique. Authors have examined stress patterns using a variety of machine learning techniques. Vidit Laijawala et al. aim at mental health using data mining techniques. Authors have collected data from online available datasets. To forecast the mental health of individuals, various machine learning techniques are applied. In that, decision tree attained the highest accuracy of 82.2%, while random forests attained an accuracy of 79.3% and Naïve Bayes of 78.7%. The data has been analysed using WEKA tool.

CHAPTER 3

REQUIREMENTS & DOMAIN INFORMATION

3.1 REQUIREMENT SPECIFICATION

3.1.1 SYSTEM REQUIREMENTS:

- Processor : Intel Core i3 and above
- RAM : 4 GB (min)
- Hard Disk : 20 GB
- Key Board : Standard Windows Keyboard
- Mouse : Two or Three Button Mouse

3.1.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows 7 Ultimate.
- Coding Language : Python.
- Front-End : Python.
- Back-End : Django-ORM
- Designing : Html, CSS, Java script.
- Data Base : MySQL (WAMP Server).

3.2 DOMAIN INFORMATION

3.2.1 PYTHON

Python's high level, interpreted, interactive, and object-oriented design make it a potent scripting language. Python was designed with legibility as a primary goal. Heavily reliant on English vocabulary, but without the punctuation or syntactical complexity of other languages. It is during runtime that Python code is processed by the interpreter, thus the word "interpreted." You don't need to build your code in advance; it will execute as expected. That looks a lot like PHP or PERL.

Python's interpreter allows you to type in commands and get immediate feedback, so you can write your own code right at the Python prompt. Python is compatible with the object-oriented

programming paradigm, which decouples data from code. Python is a great language for beginners because it can be used to create so many different kinds of software, including text editors, web browsers, and games.

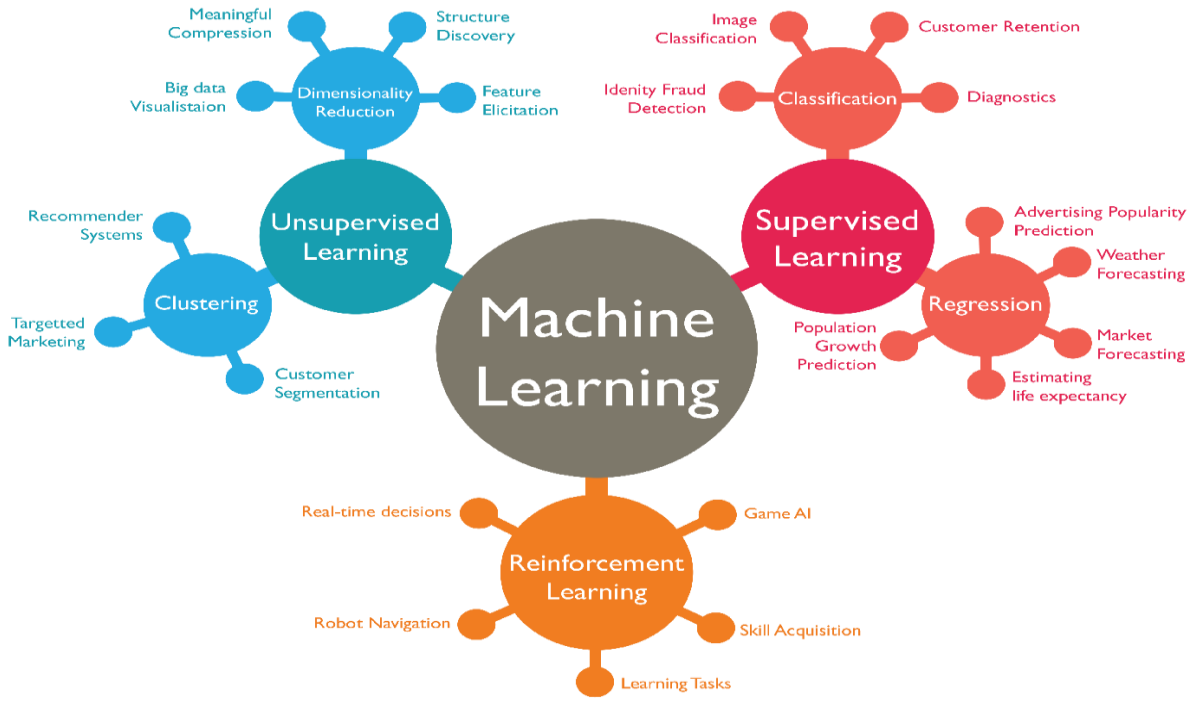
3.2.2 PYTHON DB-API

The Python DB-API is the industry standard for connecting Python programs to databases. This standard is followed by the majority of Python database interfaces. The appropriate database for your use case is available for selection. Database servers including Gadfly, small, I, PostgreSQL, Microsoft SQL Server 2000, Informix, Interpose, Oracle, and Sybase are all supported by the Python Database API. The DB API provides a bare minimum for interacting with databases via the use of Python structures and syntax. The following are all part of this API:

- Bringing in the API component.
- Establishing a link to the data source.
- Executing stored procedures and SQL statements.
- Connecting the dots

3.2.3 MACHINE LEARNING ALGORITHMS

The data, after being cleaned and normalized, is split into training and test data using a randomized 80-20 split. This is to ensure that the data used for testing does not contain any of the data used for training. Thus 20% of the data is reserved for testing purposes (see 4.4 Inference). The training dataset was used to train the four price prediction ML models chosen: Multiple Linear Regression, Lasso Regression, Ridge Regression, and Random Forest Regression. All machine learning algorithms used in this report were imported from the sklearn library. Some models were provided input parameters to implement. The motivations for the choice of input parameters are explained in this section for the models that require them.



This diagram represents Machine Learning Types and its Algorithms

Figure 3.2.3 Machine Learning

CLASSIFIER MODULES BASED ON DECISION TREES

A wide variety of applications have found success using decision tree classifiers. The capacity to extract descriptive decision-making information from the provided data is their primary strength. Training sets may be used to create a decision tree. The following is the technique for generating such a set given a set (S) of objects, each of which belongs to a class (C1, C2, and Ck): First, the objects in S must all be members of the same class, Cin, for the decision tree for S to have a leaf labelled with Cin. Step 2. If not, then T is a test with results O1, O2, on. The test divides the set S into the subsets S1, S2, Son, where each item in Si has the result Oil for T. For each possible result Oil, we construct a child decision tree by iteratively using the same technique recursively on the set Si, with T serving as the tree's root.

INCREASING THE GRADIENT

The machine learning method of gradient boosting may be used to a variety of problems, including classification and regression. It provides a prediction model in the form of a collection of low-powered prediction models, most often decision trees.[1][2] Gradient-boosted trees is the resultant technique where a decision tree is the weak learner; it often

performs better than random forest. Similar to previous boosting approaches, a gradient-boosted trees model is constructed in stages, but it generalizes the other methods by allowing optimization of any differentiable loss function. Classifies based on a similarity measure K-Nearest Neighbours (KNN) is a simple yet strong classification technique. Does not "learn" until the test example is provided; is non-parametric; and employs lazy learning. In order to categorize fresh data, we look for its K-nearest neighbours in the training data.

Example: Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset. Feature space means, space with categorization variables (non-metric variables).

CLASSIFICATION USING LOGISTIC REGRESSION

Using a collection of independent (explanatory) variables, logistic regression analyzes the correlation between a categorical dependent variable and those factors. When a dependent variable may only take on two values, like 0 and 1, or Yes and No, the statistical technique is known as logistic regression. When the dependent variable may take on three or more categories, such as marital status (Married, Single, Divorced, or widowed), multinomial logistic regression is often used. Although the dependent variable data is different from multiple regressions, the procedure's practical application is the same.

When it comes to evaluating categorical-response variables, logistic regression is in direct competition with discriminate analysis. Compared to discriminate analysis, logistic regression is seen by many statisticians as more flexible and suitable for modelling a wide range of circumstances. This is because, unlike discriminate analysis, logistic regression does not presume that the independent variables are regularly distributed.

Logistic regression, both binary and multinomial, may be calculated with this software using either numerical or categorical independent variables. Goodness of fit, odds ratios, confidence intervals, probability, and standard deviation are reported, as well as the regression equation. A full residual analysis is carried out, complete with diagnostic residual reports and graphs. It may conduct a search for the optimal regression model with the fewest possible independent variables. It helps find the optimal classification threshold by providing confidence intervals on predicted values and receiver operating characteristic (ROC) curves. By automatically categorizing rows that were not utilized in the study, it helps you verify your findings.

NAIVE BAYES

The naive bayes technique is a kind of supervised learning that makes the oversimplified assumption that the presence (or lack) of a given characteristic of a class has no bearing on the presence (or lack) of any other feature.

Nevertheless, its apparent strength and effectiveness belie this. When compared to other supervised learning methods, its performance is on par. There are a number of explanations put out in the published works. In this lesson, we focus on an explanation that makes use of the representation bias. Like linear discriminate analysis, logistic regression, and the support vector machine (SVM), the naive bays classifier is a kind of linear classifier. The key distinction is in how the classifier's parameters are estimated (the learning bias).

Naive Bays classifier is popular in academia, but not among practitioners who need actionable insights. The researchers discovered that, on the one hand, it is simple to code and implement, that its parameters are straightforward to estimate, that it can train quickly even with massive datasets, and that its accuracy is respectable when compared to other methods. But the end users don't get a simple model that's straightforward to comprehend and implement, so they don't see the point in using this method.

So, we provide the findings of the study in a different format. The classifier is less complicated to grasp, and its implementation is simplified. Part one of this course covers the naive bays classifier's theoretical foundations. We next apply the strategy to a dataset using Tanagra. Here, we look at how alternative linear methods, such logistic regression, linear discriminate analysis, and linear support vector machine, relate to our findings (the model's parameters). We find a significant degree of consistency in the findings. This is a major reason for the method's superior success.

DISTRIBUTED RANDOM FOREST

The ensemble learning technique known as random forests or random choice forests builds several decision trees during training and may be used for classification, regression, and other tasks. The output of a random forest is the class chosen by the majority of trees, which is useful for classification problems. The average or mean prediction of the individual trees is given for jobs requiring regression analysis. Decision trees tend to over fit their training data, however random decision forests eliminate this problem. While they are superior to choice trees, random forests are less precise than gradient enhanced trees. However, their efficiency might be

hampered by certain aspects of the data. Because of its ability to provide reliable predictions across a large variety of data types with little setup, random forests are often employed as "black box" models in enterprises.

SVM

To accurately predict labels for newly acquired instances, a discriminate machine learning approach in classification problems seeks to identify, on the basis of an independent and identically distributed (aid) training dataset, a discriminate function. A discriminate classification function takes a data point x and assigns it to one of the many classes that are part of the classification problem, as opposed to generative machine learning techniques that need calculations of conditional probability distributions. Discriminate techniques need less CPU resources and less training data than generative approaches, which are often employed when prediction requires outlier identification. This is particularly true for a multidimensional feature space and when just posterior probabilities are required. Learning a classifier, from a geometrical point of view, is the same as solving for the equation of a multidimensional surface that most effectively divides the feature space into distinct classes.

In contrast to popular classification methods like genetic algorithms (GAs) and perceptions, support vector machines (SVMs) always provide the same optimum hyper plane value since they solve the convex optimization issue analytically. The solutions for perceptions are quite sensitive to the parameters used to start and stop the process. Training provides uniquely specified SVM model parameters for a particular training set, but the perception and GA classifier models are distinct every time training is begun. This is because the SVM model relies on a kernel to translate the data from the input space to the feature space. Since the goal of GAs and perceptions is to achieve the lowest possible training error, there will be several hyper planes that satisfy this criterion.

SUPPLIER OF SERVICES

The Service Provider must provide a valid user name and password to access this section. Successful login grants access to features including searching, browsing, and training/testing data sets. Examine the Ratio of Predicted Employee Stress Types, View the Types of Employee Stress Used in Training, and Download the Predicted Data Sets. Get the Type Ratio Results of Employee Stress Predictions and a List of Remote Users.

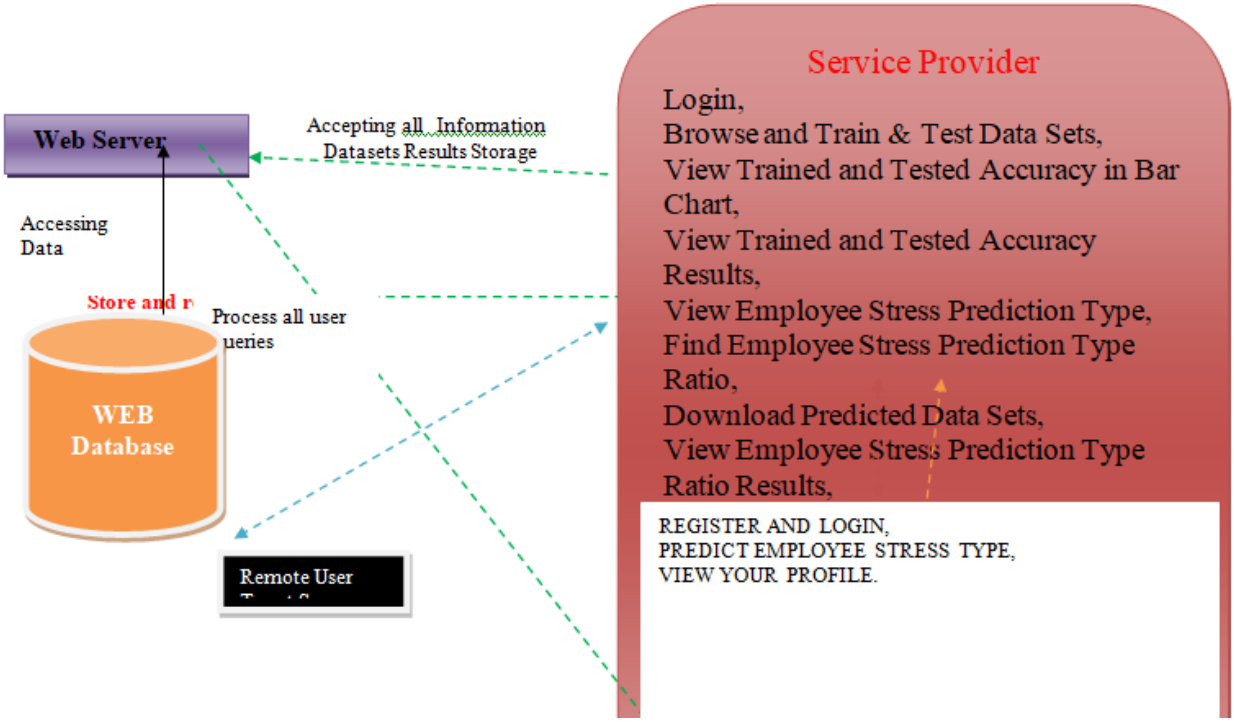
DISTANT USER

There are n people currently logged into this module. Users need to sign up first before they can do any actions. When a user signs up, their information is added to a database. He will be required to provide his valid user name and password when his registration has been approved. A user can perform things like PREDICT EMPLOYEE STRESS TYPE, VIEW YOUR PROFILE, and REGISTER AND LOGIN after they've successfully logged in.

CHAPTER 4

SYSTEM METHODOLOGY

4.1 ARCHITECTURE DIAGRAM



Architecture diagram is a visual representation that illustrates the structure, components, and relationships within a system, providing a high-level overview of its design and functionality.

Figure 4.1 Architecture Diagram

4.2 SYSTEM STUDY

4.2.1 FEASIBILITY STUDY

A business plan summarizing the project's broad strokes and some preliminary cost estimates is submitted and its feasibility is assessed at this stage. During system analysis, the feasibility of the proposed system will be examined. This is important to ensure the proposed solution won't break the bank for the company. Understanding the core requirements of the system is essential before beginning a feasibility analysis. When doing a feasibility study, it is important to keep in mind the following three things:

- Impossibility due to cost

- Probability in the field of technology
- Financial applicability

4.2.2 ECONOMIC PROSPECTS

The analysis's goal is to ascertain how much money will be spent on the system by the business. How much the company may spend on the system's creation is restricted. There has to be a thorough breakdown of all fees. Most of the used technologies are free and open source, thus the final product didn't break the bank. Only the customi
zed goods might be purchased.

4.2.3 TECHNOLOGY FEATURES

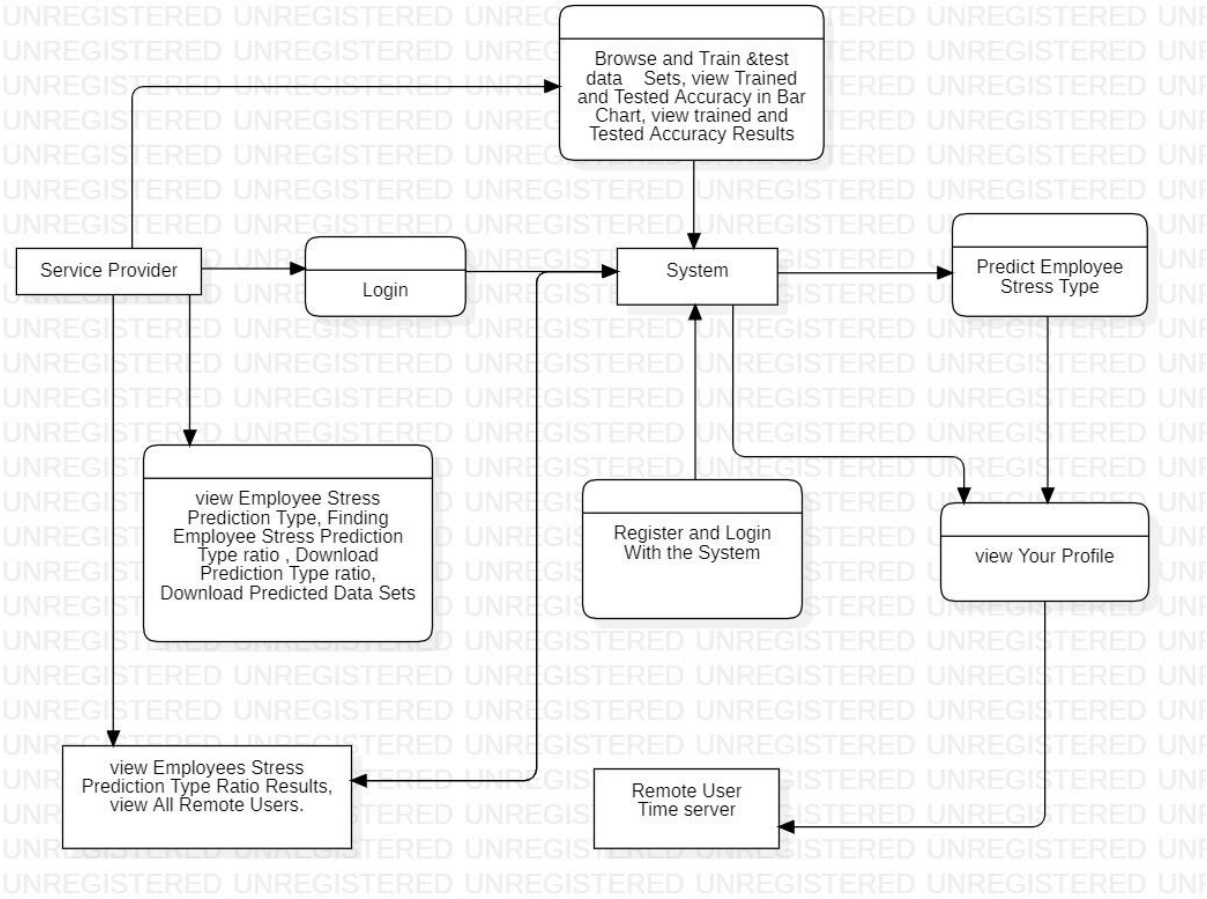
The goal of this study is to determine the system's technical feasibility, or whether or not it satisfies the system's technical standards. The existing system should not be over burdened by a whole new system. Because of this, our technical support systems will be taxed. Consequently, the client will be bombarded with unnecessary demands. The built system should have minimal needs as implementing it should take little tweaking.

4.2.4 SOCIAL POSSIBILITY

The study's objective is to learn how users of the system feel about it. The process includes instructing the user on how to get the most of the tool. The user should feel secure and not threatened when interacting with the system. The time and energy spent on boarding and educating new users is closely correlated with the system's success in gaining widespread acceptance. Before he can provide helpful criticism as the system's end user, he has to get more familiar with it.

4.3 System Design:

4.3.1 DATA FLOW DIAGRAM

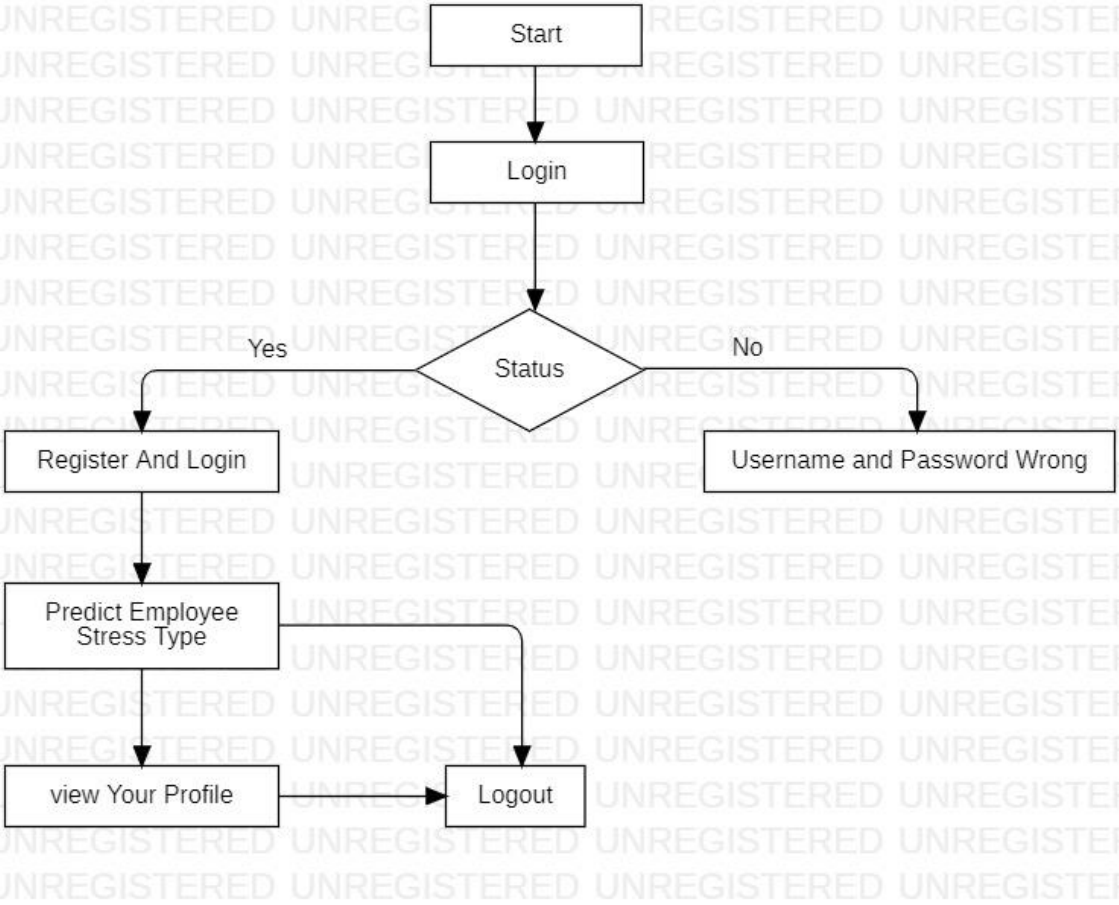


A Data Flow Diagram (DFD) is a visual representation of how data moves within a system, illustrating the processes, data sources, data destinations, and data transformations.

Figure 4.3.1 Data Flow Diagram

4.3.2 FLOW CHART DIAGRAM

REMOTE USER

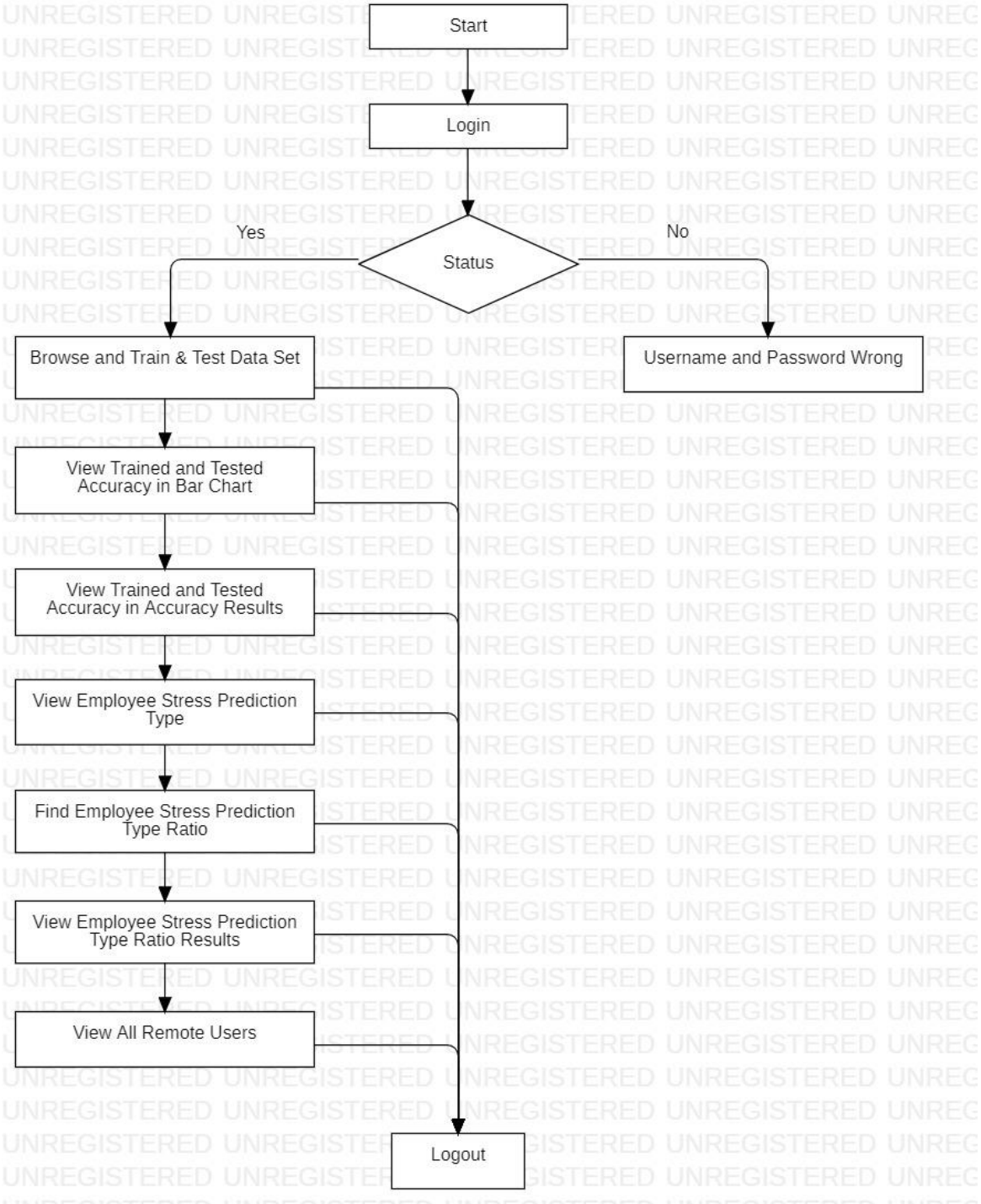


A flowchart diagram visually represents a process or algorithm using various shapes and arrows to illustrate the sequential steps and decision points.

Figure 4.3.2 Flow Chart Diagram: Remote user

4.3.3 FLOW CHART DIAGRAM

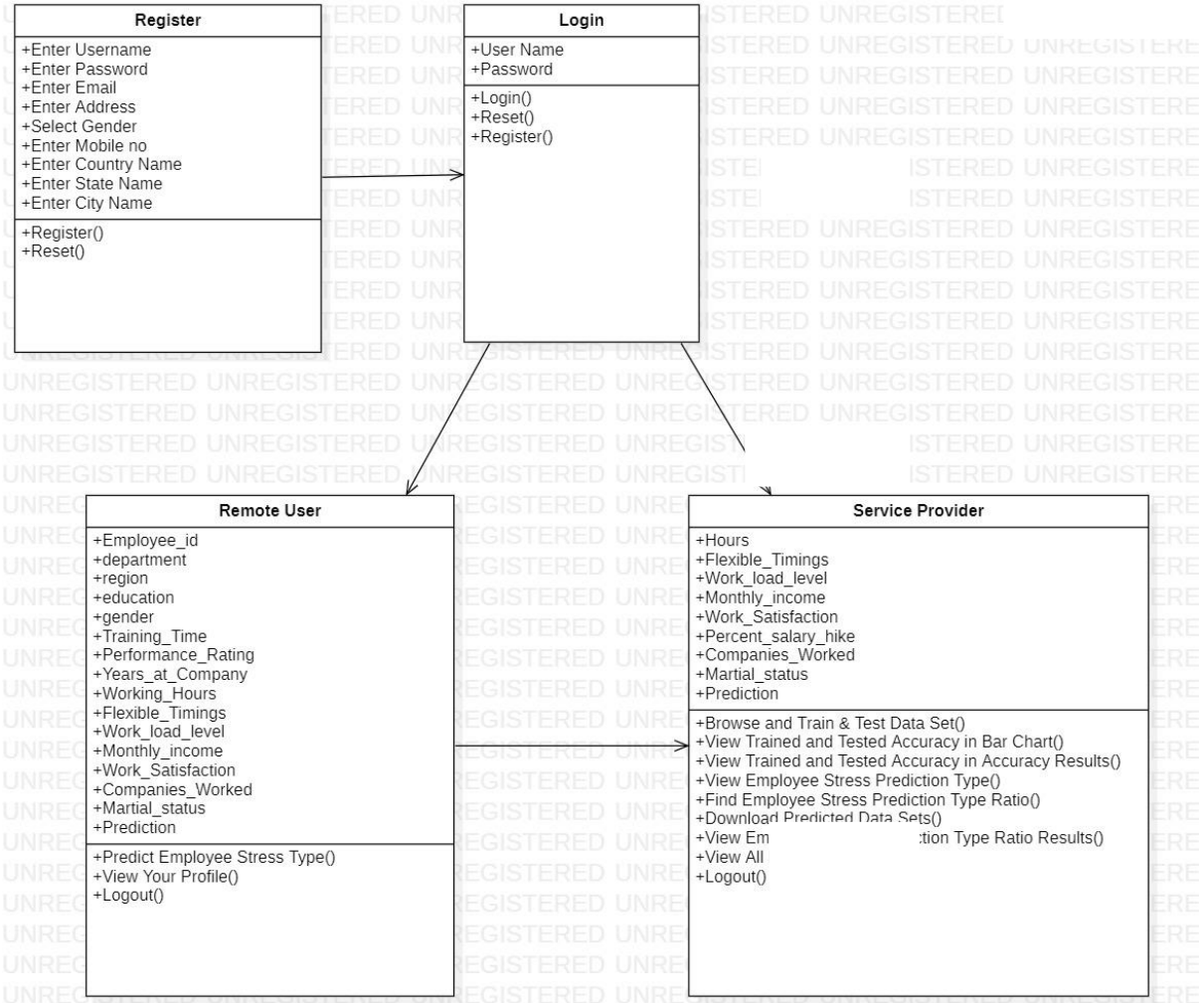
SERVICE PROVIDER



A flowchart diagram visually represents a process or algorithm using various shapes and arrows to illustrate the sequential steps and decision points.

Figure 4.3.3 Flow Chart Diagram: Service Provider Diagram

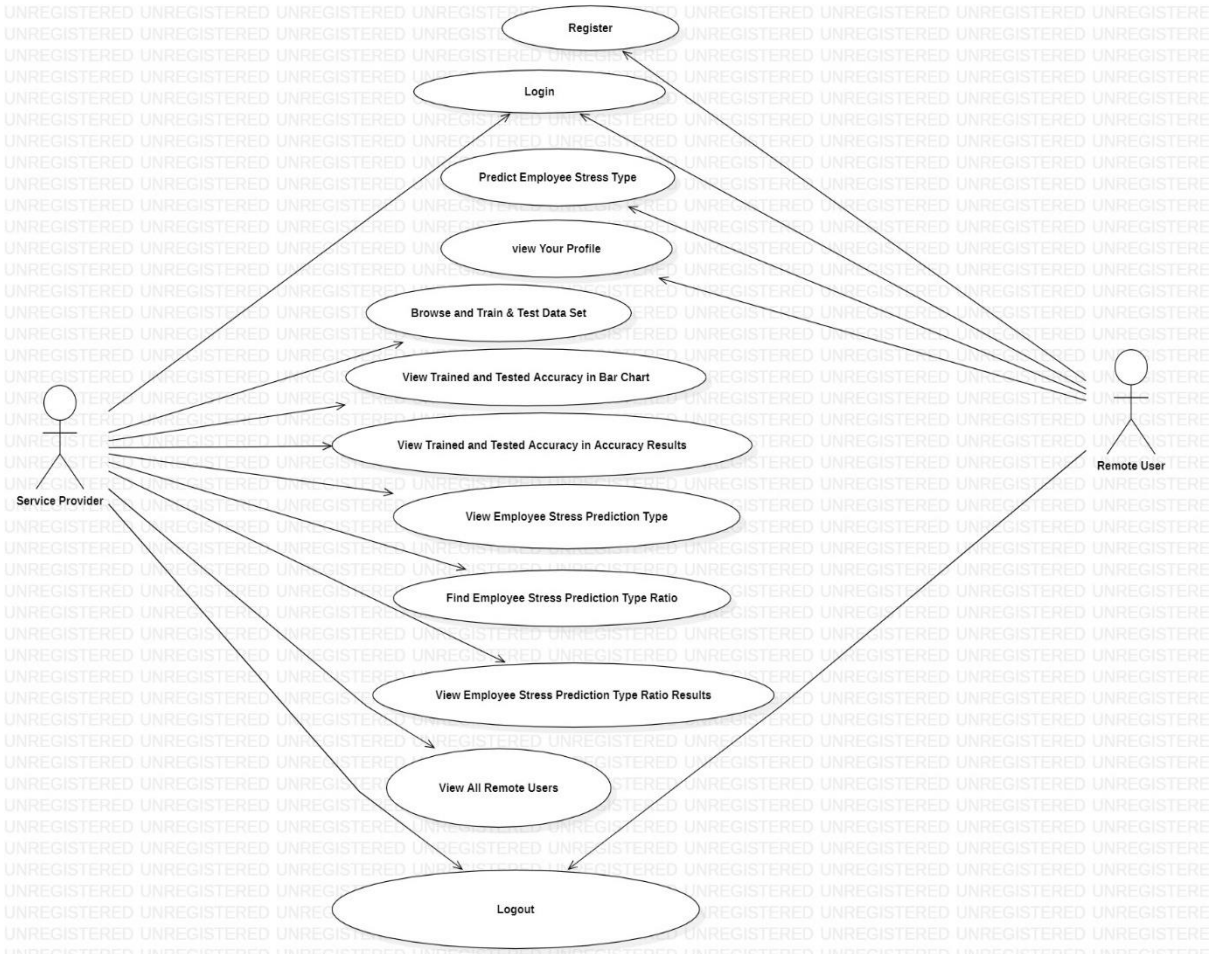
4.3.4 CLASS DIAGRAM



A class diagram is a visual representation of the structure and relationships between classes in an object-oriented system, depicting the attributes and methods each class possesses.

Figure 4.3.4 Class Diagram

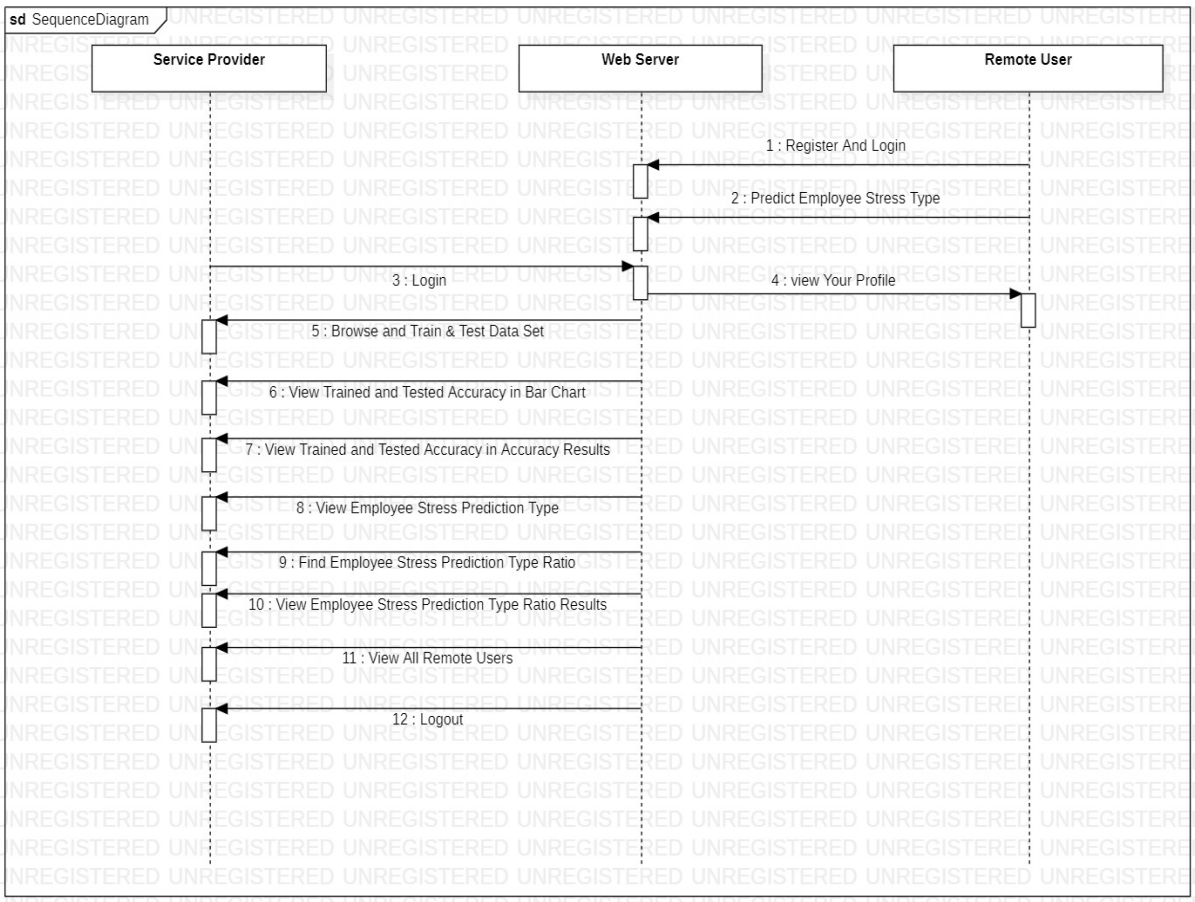
4.3.5 USE CASE DIAGRAM



A use case diagram is a visual representation that illustrates how different actors interact with a system to achieve specific goals or functionalities.

Figure 4.3.5 Use Case Diagram

4.3.6 SEQUENCE DIAGRAM



A sequence diagram visually represents the interactions and order of messages between objects or components in a system, and understanding the flow of a specific process within the system.

Figure 4.3.6 Sequence Diagram

CHAPTER - 5

EXPERIMENTATION AND ANALYSIS

5.1 EXPERIMENTATION

With the new normal here to stay for the recent future, employees have also adapted to different working environments and customs, which has also resulted in psychological stress and lethargy for many, as they adapt to the new normal and adjust their personal and professional lives. In this work, data visualization techniques and machine learning algorithms have been used to predict employees stress levels. Based on data, we can develop a model that will assist to predict if an employee is likely to be under stress or not.

5.2 SOURCE CODE

5.2.1 REMOTE USER:

```
from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import numpy as np # linear algebra
import pandas as pd
from sklearn.ensemble import VotingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
# Create your views here.
from Remote_User.models import
ClientRegister_Model,Predicting_Employee_Stress,detection_ratio,detection_accuracy
def login(request):
    if request.method == "POST" and 'submit1' in request.POST:
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            enter = ClientRegister_Model.objects.get(username=username,password=password)
            request.session["userid"] = enter.id
```

```
            return redirect('ViewYourProfile')
        except:
            pass
    return render(request,'RUser/login.html')
def Register1(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
        address = request.POST.get('address')
        gender = request.POST.get('gender')
        ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno,
                                            country=country, state=state, city=city, address=address,
gender=gender)
        obj = "Registered Successfully"
        return render(request, 'RUser/Register1.html', {'object': obj})
    else:
        return render(request,'RUser/Register1.html')
def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})
def Employee_Stress_Prediction_Type(request):
    if request.method == "POST":
        employee_id= request.POST.get('employee_id')
        department= request.POST.get('department')
        region= request.POST.get('region')
        education= request.POST.get('education')
```

```
gender= request.POST.get('gender')
recruitment_channel= request.POST.get('recruitment_channel')
Training_Time= request.POST.get('Training_Time')
age= request.POST.get('age')
Prformance_Rating= request.POST.get('Prformance_Rating')
Years_at_company= request.POST.get('Years_at_company')
Working_Hours= request.POST.get('Working_Hours')
Flexible_Timings= request.POST.get('Flexible_Timings')
Workload_level= request.POST.get('Workload_level')
Monthly_Income= request.POST.get('Monthly_Income')
Work_Satisfaction= request.POST.get('Work_Satisfaction')
Percent_salary_hike= request.POST.get('Percent_salary_hike')
companies_worked= request.POST.get('companies_worked')
Marital_Status= request.POST.get('Marital_Status')
df = pd.read_csv('Employee_Datasets.csv', encoding='latin-1')
df
df.columns
def apply_results(results):
    if (results == 'No'):
        return 0
    elif (results == 'Yes'):
        return 1
df['Results'] = df['Stress_status'].apply(apply_results)
X = df['employee_id']
y = df['Results']
print("RID")
print(X)
print("Results")
print(y)
cv = CountVectorizer(lowercase=False, strip_accents='unicode', ngram_range=(1, 1))
#X = cv.fit_transform(df['employee_id'].apply(lambda x: np.str_(X)))
X = cv.fit_transform(X)
models = []
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape
print("Naive Bayes")
from sklearn.naive_bayes import MultinomialNB
NB = MultinomialNB()
NB.fit(X_train, y_train)
predict_nb = NB.predict(X_test)
naivebayes = accuracy_score(y_test, predict_nb) * 100
print(naivebayes)
print(confusion_matrix(y_test, predict_nb))
print(classification_report(y_test, predict_nb))
models.append(('naive_bayes', NB))
# SVM Model
print("SVM")
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))
detection_accuracy.objects.create(names="SVM", ratio=svm_acc)
print("Logistic Regression")
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
y_pred = reg.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, y_pred) * 100)
```

```
        print("CLASSIFICATION REPORT")
        print(classification_report(y_test, y_pred))
        print("CONFUSION MATRIX")
        print(confusion_matrix(y_test, y_pred))
        models.append(('logistic', reg))
        print("KNeighborsClassifier")
        from sklearn.neighbors import KNeighborsClassifier
        kn = KNeighborsClassifier()
        kn.fit(X_train, y_train)
        knpredict = kn.predict(X_test)
        print("ACCURACY")
        print(accuracy_score(y_test, knpredict) * 100)
        print("CLASSIFICATION REPORT")
        print(classification_report(y_test, knpredict))
        print("CONFUSION MATRIX")
        print(confusion_matrix(y_test, knpredict))
        print("Decision Tree Classifier")
        dtc = DecisionTreeClassifier()
        dtc.fit(X_train, y_train)
        dtcpredict = dtc.predict(X_test)
        print("ACCURACY")
        print(accuracy_score(y_test, dtcpredict) * 100)
        print("CLASSIFICATION REPORT")
        print(classification_report(y_test, dtcpredict))
        print("CONFUSION MATRIX")
        print(confusion_matrix(y_test, dtcpredict))
        models.append(('DecisionTreeClassifier', dtc))
        detection_accuracy.objects.create(names="Decision Tree
Classifier",ratio=accuracy_score(y_test, dtcpredict) * 100)

        print("SGD Classifier")
        from sklearn.linear_model import SGDClassifier
        sgd_clf = SGDClassifier(loss='hinge', penalty='l2', random_state=0)
```

```
        sgd_clf.fit(X_train, y_train)
        sgdpredict = sgd_clf.predict(X_test)
        print("ACCURACY")
        print(accuracy_score(y_test, sgdpredict) * 100)
        print("CLASSIFICATION REPORT")
        print(classification_report(y_test, sgdpredict))
        print("CONFUSION MATRIX")
        print(confusion_matrix(y_test, sgdpredict))
        models.append(('SGDClassifier', sgd_clf))
        classifier = VotingClassifier(models)
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        employee_id = [employee_id]
        vector1 = cv.transform(employee_id).toarray()
        predict_text = classifier.predict(vector1)
        pred = str(predict_text).replace("[", "")
        pred1 = pred.replace("]", "")
        prediction = int(pred1)
        if prediction == 0:
            val = 'Low Stress'
        else:
            val = 'More Stress'
        print(val)
        print(pred1)
        Predicting_Employee_Stress.objects.create(employee_id=employee_id,
department=department,
region=region,
education=education,
gender=gender,
recruitment_channel=recruitment_channel,
Training_Time=Training_Time,
age=age,
Prformance_Rating=Prformance_Rating,
```

```
Years_at_company=Years_at_company,
Working_Hours=Working_Hours,
Flexible_Timings=Flexible_Timings,
Workload_level=Workload_level,
Monthly_Income=Monthly_Income,
Work_Satisfaction=Work_Satisfaction,
Percent_salary_hike=Percent_salary_hike,
companies_worked=companies_worked,
Marital_Status=Marital_Status,
        Prediction=val)
        return render(request, 'RUser/Employee_Stress_Prediction_Type.html',{'objs': val})
    return render(request, 'RUser/Employee_Stress_Prediction_Type.html')
```

5.2.2 SERVICE PROVIDER

```
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime
import xlwt
from django.http import HttpResponse
import numpy as np # linear algebra
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
# Create your views here.
from Remote_User.models import
ClientRegister_Model,Predicting_Employee_Stress,detection_ratio,detection_accuracy
def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
```

```
        if admin == "Admin" and password =="Admin":
            return redirect('View_Remote_Users')
    return render(request,'SProvider/serviceproviderlogin.html')
def View_Employee_Stress_Prediction_Type_Ratio(request):
    detection_ratio.objects.all().delete()
    ratio = ""
    kword = 'Low Stress'
    print(kword)
    obj = Predicting_Employee_Stress.objects.all().filter(Q(Prediction=kword))
    obj1 = Predicting_Employee_Stress.objects.all()
    count = obj.count();
    count1 = obj1.count();
    ratio = (count / count1) * 100
    if ratio != 0:
        detection_ratio.objects.create(names=kword, ratio=ratio)
    ratio1 = ""
    kword1 = 'More Stress'
    print(kword1)
    obj1 = Predicting_Employee_Stress.objects.all().filter(Q(Prediction=kword1))
    obj11 = Predicting_Employee_Stress.objects.all()
    count1 = obj1.count();
    count11 = obj11.count();
    ratio1 = (count1 / count11) * 100
    if ratio1 != 0:
        detection_ratio.objects.create(names=kword1, ratio=ratio1)
    obj = detection_ratio.objects.all()
    return render(request, 'SProvider/View_Employee_Stress_Prediction_Type_Ratio.html',
{'objs': obj})
def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})
def ViewTrendings(request):
```

```
    topic =
Predicting_Employee_Stress.objects.values('topics').annotate(dcount=Count('topics')).order_
by('-dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})
def charts(request,chart_type):
    chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})
def charts1(request,chart_type):
    chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})
def View_Employee_Stress_Prediction_Type(request):
    obj =Predicting_Employee_Stress.objects.all()
    return render(request, 'SProvider/View_Employee_Stress_Prediction_Type.html',
{'list_objects': obj})
def likeschart(request,like_chart):
    charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})
def Download_Trained_DataSets(request):
    response = HttpResponse(content_type='application/ms-excel')
    # decide file name
    response['Content-Disposition'] = 'attachment; filename="PredictedData.xls"'
    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')
    # adding sheet
    ws = wb.add_sheet("sheet1")
    # Sheet header, first row
    row_num = 0
    font_style = xlwt.XFStyle()
    # headers are bold
    font_style.font.bold = True
    # writer = csv.writer(response)
    obj = Predicting_Employee_Stress.objects.all()
    data = obj  # dummy method to fetch data.
```

```
    for my_row in data:
        row_num = row_num + 1
        ws.write(row_num, 0, my_row.employee_id, font_style)
        ws.write(row_num, 1, my_row.department, font_style)
        ws.write(row_num, 2, my_row.region, font_style)
        ws.write(row_num, 3, my_row.education, font_style)
        ws.write(row_num, 4, my_row.gender, font_style)
        ws.write(row_num, 5, my_row.recruitment_channel, font_style)
        ws.write(row_num, 6, my_row.Training_Time, font_style)
        ws.write(row_num, 7, my_row.age, font_style)
        ws.write(row_num, 8, my_row.Prformance_Rating, font_style)
        ws.write(row_num, 9, my_row.Years_at_company, font_style)
        ws.write(row_num, 10, my_row.Working_Hours, font_style)
        ws.write(row_num, 11, my_row.Flexible_Timings, font_style)
        ws.write(row_num, 12, my_row.Workload_level, font_style)
        ws.write(row_num, 13, my_row.Monthly_Income, font_style)
        ws.write(row_num, 14, my_row.Work_Satisfaction, font_style)
        ws.write(row_num, 15, my_row.Percent_salary_hike, font_style)
        ws.write(row_num, 16, my_row.companies_worked, font_style)
        ws.write(row_num, 17, my_row.Marital_Status, font_style)
        ws.write(row_num, 18, my_row.Prediction, font_style)
    wb.save(response)
    return response
def Train_Test_DataSets(request):
    detection_accuracy.objects.all().delete()
    df = pd.read_csv('Employee_Datasets.csv', encoding='latin-1')
    df
    df.columns
    def apply_results(results):
        if (results == 'No'):
            return 0
        elif (results == 'Yes'):
            return 1
```

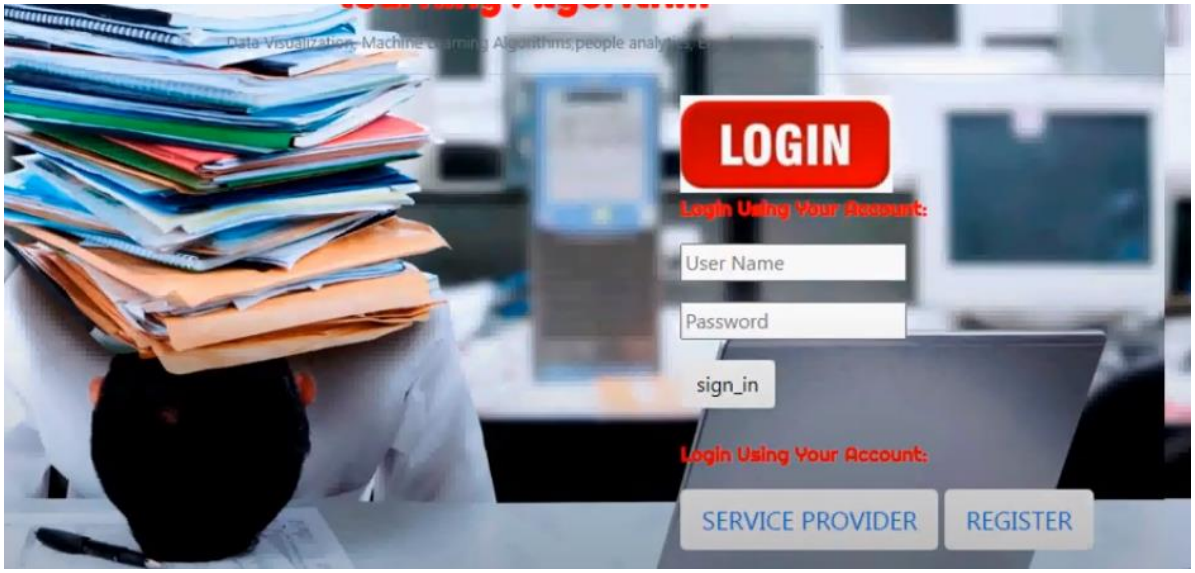
```
df['Results'] = df['Stress_status'].apply(apply_results)
X = df['employee_id']
y = df['Results']
print("RID")
print(X)
print("Results")
print(y)
cv = CountVectorizer(lowercase=False, strip_accents='unicode', ngram_range=(1, 1))
# X = cv.fit_transform(df['employee_id'].apply(lambda x: np.str_(X)))
X = cv.fit_transform(X)
models = []
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape
print("Naive Bayes")
from sklearn.naive_bayes import MultinomialNB
NB = MultinomialNB()
NB.fit(X_train, y_train)
predict_nb = NB.predict(X_test)
naivebayes = accuracy_score(y_test, predict_nb) * 100
print(naivebayes)
print(confusion_matrix(y_test, predict_nb))
print(classification_report(y_test, predict_nb))
models.append(('naive_bayes', NB))
detection_accuracy.objects.create(names="Naive Bayes", ratio=naivebayes)
# SVM Model
print("SVM")
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print(svm_acc)
```

```
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, predict_svm))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, predict_svm))
    models.append(('svm', lin_clf))
    detection_accuracy.objects.create(names="SVM", ratio=svm_acc)
    print("Logistic Regression")
    from sklearn.linear_model import LogisticRegression
    reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
    y_pred = reg.predict(X_test)
    print("ACCURACY")
    print(accuracy_score(y_test, y_pred) * 100)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, y_pred))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, y_pred))
    models.append(('logistic', reg))
    detection_accuracy.objects.create(names="Logistic Regression",
ratio=accuracy_score(y_test, y_pred) * 100)
    print("KNeighborsClassifier")
    from sklearn.neighbors import KNeighborsClassifier
    kn = KNeighborsClassifier()
    kn.fit(X_train, y_train)
    knpredict = kn.predict(X_test)
    print("ACCURACY")
    print(accuracy_score(y_test, knpredict) * 100)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, knpredict))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, knpredict))
    detection_accuracy.objects.create(names="KNeighborsClassifier",
ratio=accuracy_score(y_test, knpredict) * 100)
    print("Decision Tree Classifier")
```

```
    dtc = DecisionTreeClassifier()
    dtc.fit(X_train, y_train)
    dtcpredict = dtc.predict(X_test)
    print("ACCURACY")
    print(accuracy_score(y_test, dtcpredict) * 100)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, dtcpredict))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, dtcpredict))
    models.append(('DecisionTreeClassifier', dtc))
    detection_accuracy.objects.create(names="Decision Tree Classifier",
ratio=accuracy_score(y_test, dtcpredict) * 100)
    print("SGD Classifier")
    from sklearn.linear_model import SGDClassifier
    sgd_clf = SGDClassifier(loss='hinge', penalty='l2', random_state=0)
    sgd_clf.fit(X_train, y_train)
    sgdpredict = sgd_clf.predict(X_test)
    print("ACCURACY")
    print(accuracy_score(y_test, sgdpredict) * 100)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, sgdpredict))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, sgdpredict))
    models.append(('SGDClassifier', sgd_clf))
    detection_accuracy.objects.create(names="SGD Classifier", ratio=accuracy_score(y_test,
sgdpredict) * 100)
    predicts = 'Results.csv'
    df.to_csv(predicts, index=False)
    df.to_markdown
    obj = detection_accuracy.objects.all()
    return render(request,'SProvider/Train_Test_DataSets.html', {'objs': obj})
```

5.3 RESULT

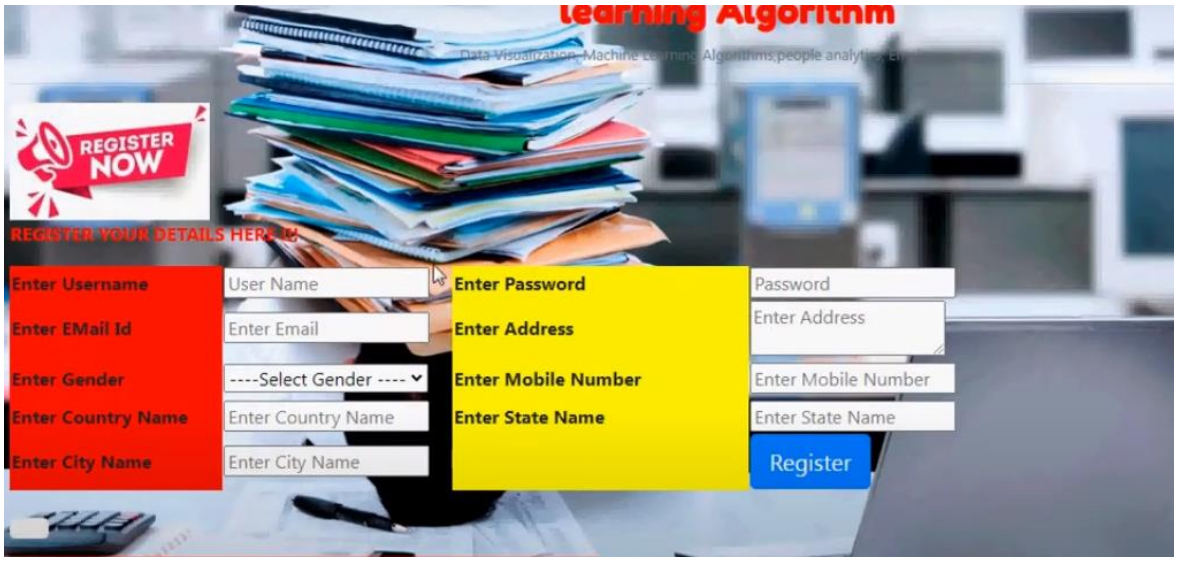
LOGIN PAGE



The Login page is the entry point for authorized users, where they provide their credentials (username and password) to access the system and its features securely.

Figure 5.3.1 Login Page

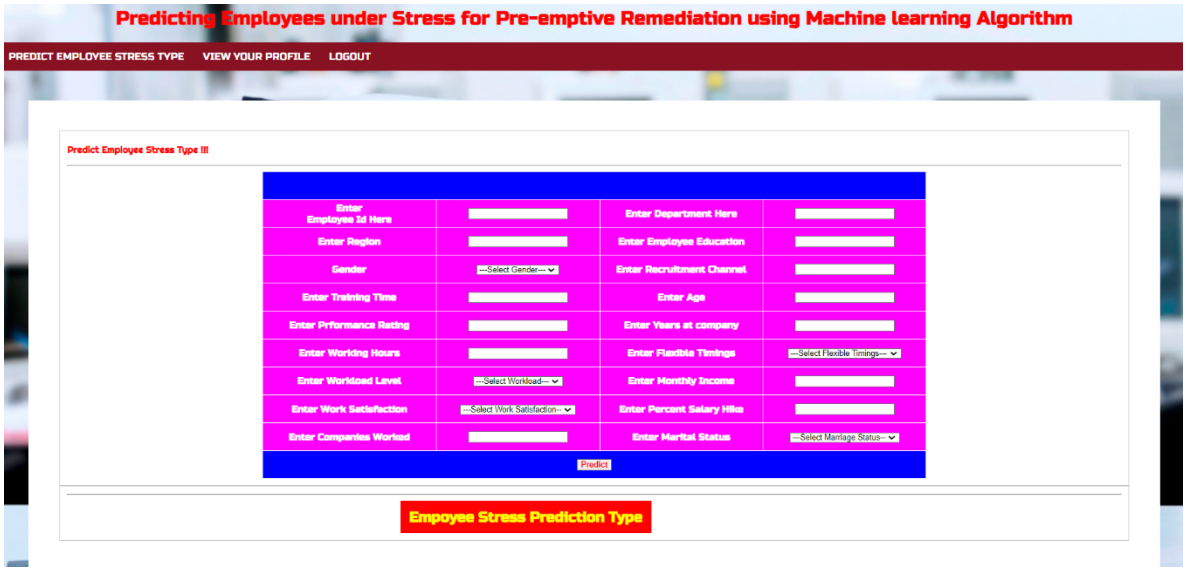
REGISTER PAGE



The registration page allows users to create new accounts by providing their personal information, enabling them to access the platform's features and services.

Figure 5.3.2 Register Page

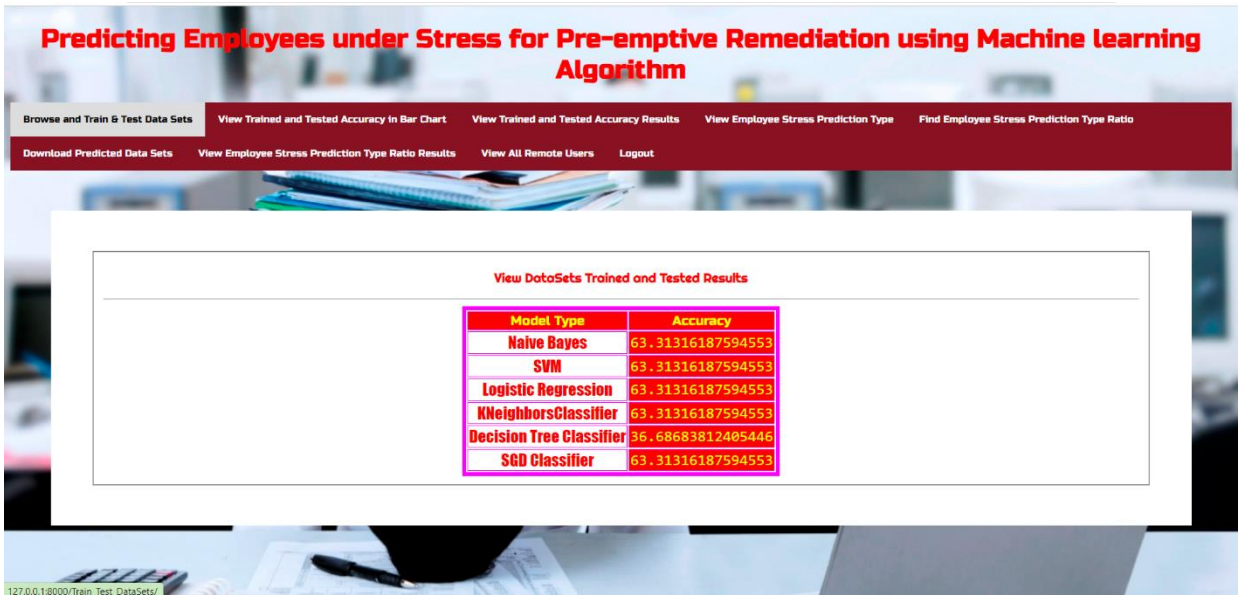
PREDICTING EMPLOYEE STRESS



We have to enter data to predict the employee stress in remote user module

Figure 5.3.3 Predicting employee Stress

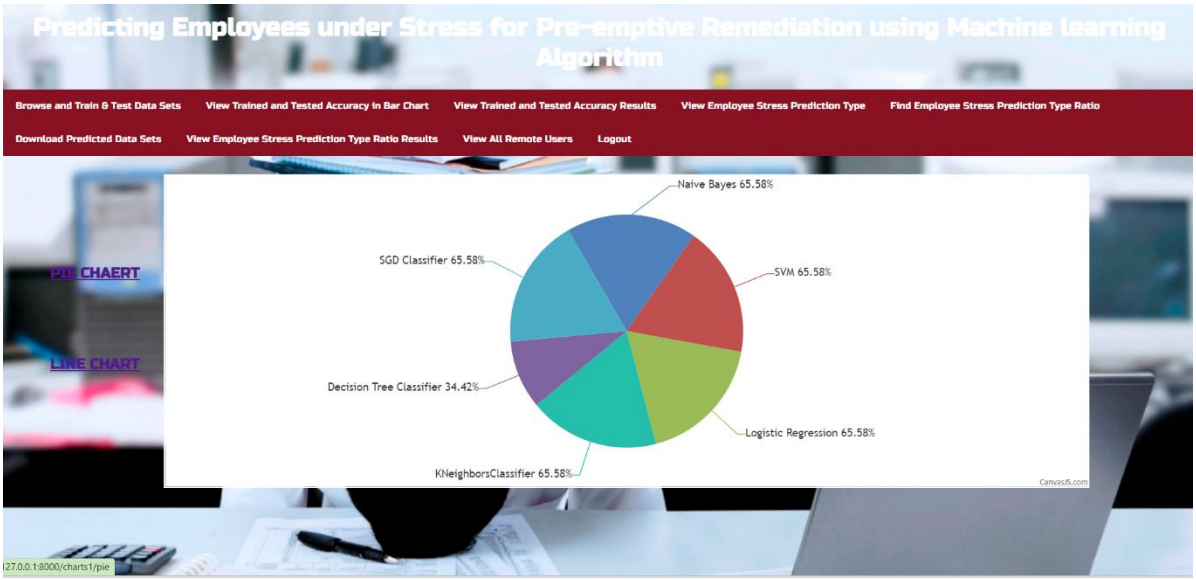
BROWSE AND TRAIN & TEST DATA SETS



Admin can see the accuracy of the algorithm which we have used in this project.

Figure 5.3.4 Browse and Train & Test Data Sets

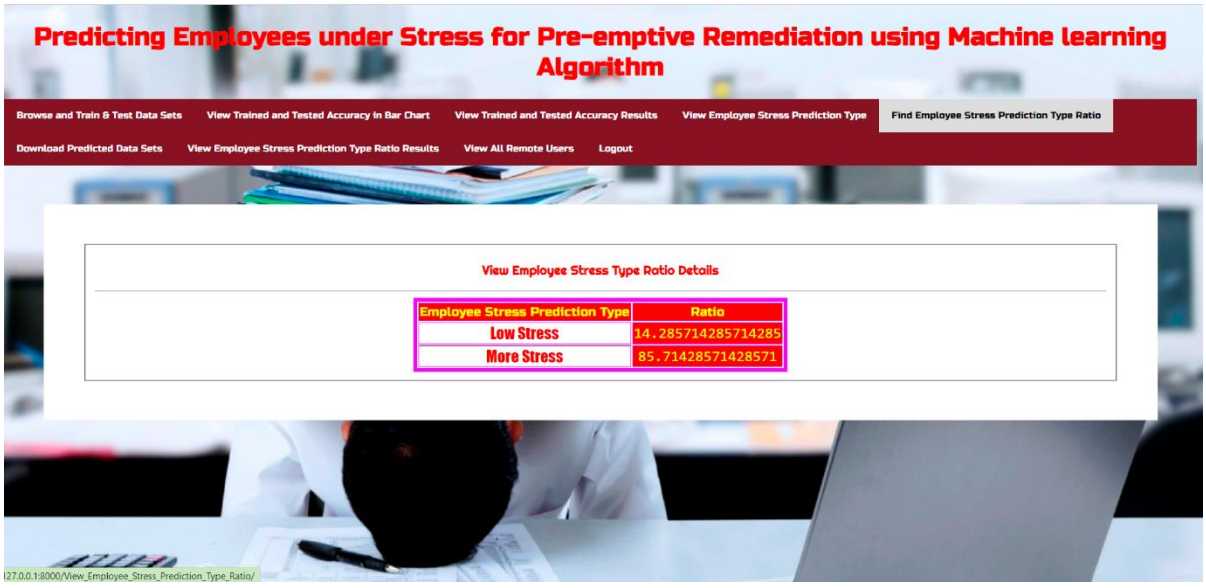
VIEW TRAINED AND TESTED ACCURACY IN PIE CHART



Admin can see the accuracy of the algorithm which we have used in this project in pie chart.

Figure 5.3.5 View Trained and Tested Accuracy in Pie Chart

FIND EMPLOYEE STRESS PREDICTION TYPE RATIO



Admin can Find Employee Stress Prediction Type Ratio in Service Provider

Figure 5.3.6 Find Employee Stress Prediction Type Ratio

VIEW EMPLOYEE STRESS PREDICTION TYPE RATIO PIE CHART

Admin can see the accuracy of the Employee Stress Prediction in pie chart.

Figure 5.3.7 View Employee Stress Prediction Type Ratio Pie Chart

VIEW EMPLOYEE STRESS PREDICTION TYPE RATIO LINE CHART

Admin can see the accuracy of the Employee Stress Prediction which we have used in this project in line chart.

Figure 5.3.8 View Employee Stress Prediction Type Ratio Line Chart

5.4 SYSTEM TESTING

Testing's goal is to unearth flaws. The goal of testing is to expose every possible flaw or vulnerability in a product. It is the act of exercising software with the goal of verifying that the programs expected behaviour occurs in the context of a given scenario. The software works as intended and does not experience any unexpected or unwelcome failures. Multiple testing formats exist. There are several kinds of tests, and each one caters to a different need.

DIFFERENT EXAMS

PARTS CHECKING

The goal of unit testing is to ensure that the underlying logic of a program is working as intended and that legitimate inputs will result in expected outputs. Validation of internal code flow and all decision branches is essential. It is the process of testing the application's software components separately once they have been developed. This is an invasive kind of testing that requires specific information about the structure being tested. Unit tests are simple tests that check one business process, application, or system configuration at a time. Each possible branch of a business process may be tested independently to verify it conforms to the standards and produces the intended outcomes.

EXAMINING THE WHOLE

The goal of an integration test is to verify that all of a program's components work together seamlessly. The focus of testing is on the simplest possible results of any given screen or field. Even if each part was satisfactory on its own, as shown by passing unit tests, integration tests indicate that the whole is proper and consistent. The goal of integration testing is to reveal issues that manifest as a result of putting different parts together.

PROVE THE FUNCTIONALITY

The purpose of functional testing is to systematically prove that the system works as intended by the business and technical requirements, the system documentation, and the user guides. Functional tests are organized and prepared with a particular emphasis on requirements, important functions, or unique test cases. Additionally, testing must consider comprehensive coverage of the identified business process flows, data fields, established procedures, and

subsequent activities. Additional tests are identified and the effective value of existing tests is assessed before functional testing is complete.

EXAMINATION OF THE SYSTEM

System testing verifies that the totality of an integrated software solution is up to snuff. It puts a set up to the test to see whether it will provide consistent and reliable outcomes. The system integration test based on configuration is an example of a system test. Process descriptions and flows provide the backbone of system testing, with an emphasis on the integration points and dependencies that are driven in advance.

WHITE BOX EVALUATIONS

White Box Testing is performed when the software tester is familiar with the code, the programming language, and the software's internals. That is the point. Areas that cannot be accessed from a "black box" level are tested using this method.

CONTRAST-BASED TESTING

During black box testing, testers pretend they know nothing about the code's internals, architecture, or language. Like other types of tests, black box tests need a definite source document like a specification or requirements document to guide their creation. This kind of testing assumes no knowledge of the internal workings of the program being tested. Without considering how the program really operates, the test just offers inputs and reacts to outputs.

5.4.1 UNIT TESTING

Though it is more conventional to do coding and unit testing as separate stages of the software development lifecycle, unit testing is often performed as part of a combined code and unit test phase.

METHODS AND APPROACHES TO TESTING

Functional tests will be prepared in great detail, and manual testing in the field will be undertaken. Goals for the Exam All input fields must be functional. To access a page, click on the corresponding link. Input screen, messages, and answers must be instantaneous. To be evaluated Features Make sure the entries are in the right format by checking them. In order to

avoid repeated submissions, if a user clicks on a link, they should be sent to the intended location.

5.4.2 FUNCTIONAL TESTING

To simulate failures brought on by interface faults, software integration testing progressively integrates two or more software components on a single platform. The goal of an integration test is to ensure that different parts of a software system or, more generally, different software applications within a firm, work together without causing any problems. All the test cases were successful, and the tests have been completed. There were no problems at all.

5.4.3 A TEST OF ACCEPTANCE

The end user's input is crucial throughout the User Acceptance Testing phase of any project. This process also verifies that the system is fully working. All the test cases were successful, and the tests have been completed. There were no problems at all.

TESTING THE SYSTEM

Methods of testing the following techniques are used for testing:

- Unit Tests.
- To Integrate Tests.
- UAT or User Acceptance Testing.
- Checking the Results.
- Tests for Validity.

PARTS CHECKING

Unit testing isolates and verifies individual modules inside a larger whole. The purpose of unit testing is to find and fix bugs by repeatedly exercising individual control flow pathways inside a module. Each component is tested separately to guarantee that the whole works as intended. Thus, the term "Unit Testing" was coined.

Individual modules are tested, and their interfaces are checked to make sure they conform to the design specifications. The predicted outcomes of each critical processing step have been tested. We also test every possible error-handling scenario.

5.4.4 VERIFYING VALIDITY

The following fields undergo validation tests.

SPACE FOR TEXT

Only as many characters as are fewer than or equal to the size of the text box may be entered. In certain tables, the text fields are organized numerically, whereas in others, they are organized alphabetically. When an incorrect input is made, an error notice will appear.

NUMBERED LIST

Only digits from 0 to 9 are allowed in the numerical field. An alert appears whenever a character is entered that is not allowed. The correctness and functionality of the different modules are verified. Each component is tested using real-world data. Each component is examined separately before being included into the whole. When a program is tested, it is run with its intended input data, and any bugs are identified based on the results.

A good test is one that reports errors for bad input and generates results that show where the system is broken.

TEST DATA PREPARATION

All the above is tested by using different types of test data. The system testing process relies heavily on the preparation of test data. The research system is put to the test when the test data has been prepared. Errors are found and fixed using the aforementioned testing procedures when testing the system with test data, and the fixes are documented for future reference.

USING ACTUAL TEST RESULTS

Real-world test data are ones that are culled from a real company's records. After a system has been largely built, programmers or analysts may typically ask for some data from regular users. The system administrator then uses this information to do limited testing. Sometimes, developers or analysts manually insert data that was extracted from live files.

CREATING FAKE TEST DATA

Since all possible permutations of format and value may be tested using artificial test data, it is built specifically for use in testing. In other words, all conceivable login and control pathways via the program may be tested with the use of fake data, which can be swiftly created by a data generating utility software in the information systems department.

EDUCATION OF THE USER

New systems always need user training to familiarize end users with the system's inner workings and ensure effective use by the target audience. This was done so that the potential users could see how the project normally operates. Users with a solid grasp of computing concepts should have no trouble getting up and running with this system.

MAINTENANCE

This encompasses a broad variety of tasks, including as fixing bugs in the code or the design. We have better specified the user's needs throughout the system development process to lessen the need for future maintenance. This system was designed to provide maximum utility in response to specified criteria. More features, tailored to future needs, may be conceivable as technology advances. Maintenance will be less of a hassle since the code and layout are intuitive and straightforward.

APPROACH TO TESTING

System testing strategies combine system test cases with design methodologies in a methodical progression toward a finished piece of software. A software testing strategy should allow for both low-level tests (used to ensure that a small chunk of source code has been correctly implemented) and high-level tests (used to validate major system functions against user requirements), and the two must work together seamlessly throughout the testing process.

The software testing process is the last check of the requirements, designs, and implementations that make up the program. An unusual aspect of software development is testing. Therefore, the proposed system undergoes a battery of tests prior to user acceptability testing.

PROTOCOL EVALUATION

After software has been tested and verified, it must be integrated with the rest of the system components. The purpose of system testing is to ensure that the system as a whole performs as expected. The system is also put through a series of checks for inconsistencies between its stated goal, existing specs, and official documentation.

TESTING ON UNITS

Unit testing involves verifying those individual modules function in accordance with the requirements established during the design phase. The aims of unit testing are to test the internal logic of the modules, making it vital for verifying the code created during the development process. The testing is done while still in the development phase. All modules were determined to function properly and provide the desired results throughout this round of testing. The most recent technical developments will be accounted for in due time. Many of the networking system's components will be designed as generic building blocks throughout development, making them available to other projects for usage or interaction.

CHAPTER - 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

We use XGB classifier to assess how well our model performs and make any necessary adjustments. Like a decision tree-based algorithm, this method uses the gradient boosting framework to conduct analysis, and the confusion matrix reveals the percentage of right predictions made by the model. XG Boost is around 10 times more robust than other gradient boosting methods and has remarkable predictive potential. Regularization is one of its many features, and it helps prevent over fitting and boost performance. As a result, it is also known as the "regularized boosting" method. The values might be either true (positive) or false (negative) or both (in all four directions). Measures how well a categorization model does its job.

6.2 FUTURE ENHANCEMENT

While this project has made significant strides in predicting employee stress levels and understanding the factors that influence workplace stress, there are several avenues for future research and development to build upon this work. In conclusion, the future scope of this project is broad and dynamic, offering opportunities to refine and expand the research to benefit employees and organizations in a rapidly evolving work landscape. By addressing these future directions, we can continue to advance the field of workplace stress management and create healthier, more productive work environments.

REFERENCES

[1] Shekhar Pandey, Supriya Muthuraman, Abhilash Shrivastava. The International Symposium on Intelligent Systems Technologies and Applications (2018)

[2] Ramachandran, R; Rajeev, D.C; Krishnan, S.G; Subathra.P. International Journal of Applied Engineering Research (2015),Research India Publications, Volume 10, Number 10

[3] Ramin Zibaseresht: How to Respond to the Ongoing Pandemic Outbreak of the Coronavirus Disease (COVID-19) (WHO- World Health Organization) (2020), ISSN 2349- 8870.

[4] Chen, Tianqi; Guestrin, Carlos; "XG Boost: A scalable Tree Boosting System". Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA (2016).ACM. pp. 785-794.

[5] Jolliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002,XXIX,487p.28 illus.ISBN 978-0-387-95442-4.

[6] Manuela Aparicio and Carlos J. Costa "Data Visualization".Communication Design Quarterly (2014). DOI: 10.1145/2721882.2721883.

[7] Ali.M., Alqahtani.A.,Jones.M.W., Xie.X ."Clustering andclassification for Time Series Data in Visual Analytics: A SurveyIEEE Access 7,8930535, pp. 181314-181338.

[8] Mouubayedd.A, Injadat.M.,Nassif, Lutfuyya, H. Shami, AE-learning: Challenges and Research Opportunities Using MachineLearning and Data Analytics (2018) IEEE Access 6,8417405. pp.39117-39138.

[9] Kim., Soyata, T., Behnagh, R.F. Towards Emotionally Aware AI Smart Classroom: Current Issues and Directions for Engineering and Education (2018) IEEE pp. 5308-5331.

[10] PriyonesiS.Madeh; EI-Diraby Tamer E. "Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems". Journal of Transprotation Engineering, Pavements (2020). DOI: 10.1061/JPEODX.0000175.

[11] Basu, S., and R. Bhattacharyya (2018). India Inc. is working to reduce the mounting stress among employees. drawn from "The Economic Times."

[12] Dataset from Kaggle's 2017 OSMI Mental Health in Tech Survey.

[13] J. Van den Broeck, S. A. Cunningham, R. Eeckels, & K. Herbst (2005). Data cleaning entails identifying, diagnosing, and correcting data irregularities. 2(10), e267 in PLoS medicine.

[14] Predictive analysis utilising categorization techniques in the healthcare industry. Journal of Computing and Linguistics International, ISSN: 2456-8848, Vol. I, Issue. I, June-2017.

[15] Tomar, D., and S. Agarwal (2013). a review of data mining techniques used in healthcare. 241-266 in International Journal of Bioscience and Biotechnology, volume 5(5).

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, & J. Vanderplus.