A

Project Report

on

PREDECTING WORKFORCE PRODUCTIVITY IN THE GARMENT INDUSTRY: A DATA DRIVEN APPROACH

STAGE-2

Submitted for partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

AMIREDDY MADHUVIKA	**20K81A1205**
RAMPALLY NAVANEETH	**20K81A1246**
PANTULA SANATHAN	**20K81A1242**
PULI ANISH KUMAR	**20K81A1244**

Under the Guidance of

Mr V. CHANDRA PRAKASH

ASSISTANT PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous
Affiliated to JNTUH, Approved by AICTE
Accredited by NBA & NAAC A+, ISO 9001-2008 Certified
Dhulapally, Secunderabad-500 100
www.smec.ac.in

MARCH – 2024

St. MARTIN'S ENGINEERING COLLEGE
UGC Autonomous
NBA & NAAC A+ Accredited
Dhulapally, Secunderabad - 500 100
www.smec.ac.in

CERTIFICATE

This is to certify that the project entitled **"Predicting Workforce Productivity in The Garment Industry: A Data Driven Approach"** is being submitted by Madhuvika Prasad(20K81A1205), R.Navaneeth (20K81A1246), P. Sanathan(20K81A1242), P. Anish Kumar(20K81A1244) in fulfilment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** is recorded of bonafide work carried out by them. The result embodied in this report have been verified and found satisfactory.

Signature of Guide	Signature of HOD
Mr. V. CHANDRAPRAKASH	**Dr. V K SENTHIL RAGAVAN**
Assistant Professor	Professor and Head of Department
Department of Information Technology	Department of Information Technology

Internal Examiner **External Examiner**

Place:

Date:

St. MARTIN'S ENGINEERING COLLEGE
UGC Autonomous
NBA & NAAC A+ Accredited
Dhulapally, Secunderabad - 500 100
www.smec.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

DECLARATION

We, the students of '**Bachelor of Technology in Department of Information Technology**', session: 2020 - 2024**, St. Martin's Engineering College, Dhulapally, Kompally, Secunderabad,** hereby declare that the work presented in this Project Work entitled "**PREDICTING WORKFORCE PRODUCTIVITY IN THE GARMENT INDUSTRY: A DATA DRIVEN APPROACH"** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. This result embodied in this project report has not been submitted in any university for award of any degree.

A. MADHUVIKA	**20K81A1205**
R. NAVANEETH	**20K81A1246**
P. SANATHAN	**20K81A1242**
P. ANISH KUMAR	**20K81A1244**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowded our efforts with success.

First and foremost, we would like to express our deep sense of gratitude and indebtedness to our College Management for their kind support and permission to use the facilities available in the Institute.

We especially would like to express our deep sense of gratitude and indebtedness to **Dr. P. SANTOSH KUMAR PATRA**, Group Director, St. Martin's Engineering College, Dhulapally, for permitting us to undertake this project.

We wish to record our profound gratitude to **Dr. M. SREENIVAS RAO**, Principal, St. Martin's Engineering College, for his motivation and encouragement.

We are also thankful to **Dr. V K SENTHIL RAGAVAN**, Head of the Department, Information Technology, St. Martin's Engineering College, Dhulapally, Secunderabad. for his support and guidance throughout our project as well as Project Coordinator **Mr. A. BHASHA**, Assistant Professor, Information Technology department for his valuable support.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Mr. V. CHANDRAPRAKASH**, Assistant Professor, Information Technology, St. Martins Engineering College, Dhulapally, for her support and guidance throughout our project.

Finally, we express thanks to all those who have helped us successfully completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement. We express thanks to all those who have helped us in successfully completing the project.

A. MADHUVIKA	20K81A1205
R. NAVANEETH	20K81A1246
P. SANATHAN	20K81A1242
P.ANISH KUMAR	20K81A1244

ABSTRACT

The garment industry plays a significant role in the global economy, providing employment to millions of workers. However, optimizing workforce productivity in this sector is a complex task due to various factors such as skill levels, working conditions, and production processes. Historically, productivity improvements relied on traditional methods like time and motion studies. Traditional methods for improving workforce productivity in the garment industry often relied on time and motion studies, where experts observed and analyzed manual tasks. While effective to some extent, these methods may not capture the full complexity of factors affecting productivity. The primary challenge is to develop a predictive model that can accurately forecast workforce productivity in the garment industry. This involves collecting and analyzing various data points related to worker performance, process efficiency, and environmental conditions to identify patterns and factors influencing productivity. Therefore, the need of Efficient utilization of the workforce is crucial for competitiveness and sustainability in the garment industry. Predicting workforce productivity can help managers make informed decisions about resource allocation, training, and process optimization. A data-driven approach using advanced analytics and machine learning can provide more accurate predictions compared to traditional methods. This project, aims to enhance productivity optimization in the garment industry by leveraging advanced data analytics and machine learning techniques. By collecting and analyzing comprehensive datasets, this research endeavors to develop a system capable of autonomously and accurately predicting workforce productivity. Advanced algorithms can identify intricate relationships between various factors and productivity outcomes, providing valuable insights for process improvement and resource allocation. This data-driven approach holds great promise for revolutionizing how garment industry managers optimize workforce productivity, ultimately leading to increased efficiency and competitiveness in the global market.

LIST OF FIGURES

LIST OF ACRONYMS AND DEFINATIONS

S. No.	Acronym	Definition
01	ML	Machine Learning
02	CNN	Convolutional Neural Networks
03	DL	Deep Learning
04	UAV	Unmanned Aerial Networks
05	GAN	Generative Adversarial Networks
06	UML	Unified Modelling Language
07	DFD	Data Flow Diagram

CONTENTS

CHAPTER 1
INTRODUCTION

1.1History

The garment industry, a pivotal player in the global economy, has long been a major source of employment for millions of individuals. However, the challenge of optimizing workforce productivity within this sector has proven to be intricate, given the myriad of factors influencing efficiency, including skill levels, working conditions, and production processes. Traditional methods, such as time and motion studies, have historically been relied upon to enhance productivity. These approaches involved experts conducting observations and analyses of manual tasks, yielding some effectiveness. Yet, they often fell short of capturing the entire spectrum of complexities inherent in factors impacting productivity.

The pressing need, therefore, is to develop a predictive model capable of accurately forecasting workforce productivity in the garment industry. This necessitates the collection and analysis of diverse data points related to worker performance, process efficiency, and environmental conditions. The overarching objective is to discern patterns and identify the multifaceted factors influencing productivity. In the fiercely competitive garment industry, the efficient utilization of the workforce is imperative for both competitiveness and sustainability.

The prediction of workforce productivity holds considerable significance, offering managers valuable insights to make informed decisions regarding resource allocation, training initiatives, and process optimization. To surpass the limitations of traditional methods, a forward-looking strategy involves the adoption of advanced data analytics and machine learning techniques. This project endeavors to elevate productivity optimization within the garment industry by leveraging these cutting-edge methodologies.

By meticulously collecting and analyzing comprehensive datasets, the research aims to construct a system capable of autonomously and accurately predicting workforce productivity. Advanced algorithms, intrinsic to machine learning, can unravel intricate relationships between various factors and productivity outcomes. This, in turn, provides a wealth of insights for refining processes and judiciously allocating resources. The implementation of this data-driven approach holds immense promise, potentially revolutionizing how managers in the garment industry optimize workforce productivity. The ultimate goal is to foster increased efficiency and competitiveness in the dynamic global market landscape.

1.2Problem Statement

In the global economic landscape, the garment industry stands as a pivotal contributor, offering employment to millions of individuals. However, optimizing workforce productivity within this sector presents a multifaceted challenge, characterized by factors such as varying skill levels, diverse working conditions, and intricate production processes. Historically, efforts to improve productivity in the garment industry predominantly relied on traditional methodologies like time and motion studies. While these approaches were somewhat effective, they often fell short of comprehensively capturing the intricate array of factors influencing productivity.

The paramount challenge at hand is the development of a predictive model capable of accurately forecasting workforce productivity in the garment industry. This endeavor entails the meticulous collection and analysis of diverse data points pertaining to worker performance, process efficiency, and environmental conditions. The aim is to discern patterns and underlying factors that significantly impact productivity. Recognizing the pivotal role of efficient workforce utilization in ensuring competitiveness and sustainability in the garment industry, the imperative to predict workforce productivity becomes apparent. Such predictions can empower managers with insights for judicious decision-making regarding resource allocation, training initiatives, and process optimization.

To transcend the limitations of traditional methodologies, this project advocates for a data-driven approach, leveraging advanced analytics and machine learning techniques. By systematically collecting and scrutinizing comprehensive datasets, the research aspires to construct a system capable of autonomously and precisely predicting workforce productivity. Advanced algorithms, intrinsic to this data-driven methodology, hold the potential to unveil intricate relationships between diverse factors and productivity outcomes. The insights derived from this approach promise to be invaluable for refining processes and strategically allocating resources, thereby enhancing overall efficiency.

The significance of this data-driven approach lies in its potential to revolutionize how managers in the garment industry optimize workforce productivity. Through the application of advanced analytics and machine learning, this project seeks to usher in a new era of heightened efficiency, bolstering competitiveness on the global stage. In essence, the research endeavors to offer a transformative solution that goes beyond conventional practices, ultimately fostering increased efficiency and competitiveness within the dynamic landscape of the garment industry.

1.3Research Motivation

The impetus behind delving into the realm of predicting workforce productivity in the garment industry stems from the sector's pivotal role in the global economy, serving as a substantial source of employment for millions. Despite its economic significance, the inherent complexity of optimizing workforce productivity within this domain cannot be understated. Factors ranging from diverse skill levels and working conditions to intricate production processes contribute to the intricacies of this challenge.

Historically, attempts to enhance productivity in the garment industry have leaned heavily on conventional methods, such as time and motion studies. These traditional approaches, involving expert observation and analysis of manual tasks, have yielded results to a certain extent. However, they often fall short of encapsulating the entirety of factors that exert influence on productivity within this multifaceted industry.

The crux of the matter lies in the imperative to develop a predictive model capable of accurately forecasting workforce productivity in the garment industry. This undertaking necessitates the meticulous collection and analysis of a myriad of data points related to worker performance, process efficiency, and environmental conditions. The objective is to unveil patterns and discern the intricate interplay of factors that impact productivity comprehensively.

Efficient utilization of the workforce emerges as a linchpin for ensuring competitiveness and sustainability in the garment industry. Recognizing this, the anticipation of workforce productivity becomes a strategic lever for managers, empowering them to make well-informed decisions regarding resource allocation, training initiatives, and process optimization.

In light of the limitations associated with traditional methods, a paradigm shift towards a data-driven approach becomes imperative. Advanced analytics and machine learning, constituting the backbone of this approach, offer the promise of delivering more accurate predictions compared to their conventional counterparts. This project seeks to propel productivity optimization in the garment industry by harnessing the power of advanced data analytics and machine learning techniques.

The research endeavors to construct a system capable of autonomously and accurately predicting workforce productivity through the collection and analysis of comprehensive datasets. The deployment of advanced algorithms is poised to unravel intricate relationships between various factors and productivity outcomes, thereby furnishing invaluable insights for process improvement and resource allocation.

1.4 Applications

In the intricate landscape of the garment industry, where millions find employment and contribute significantly to the global economy, the optimization of workforce productivity stands as a formidable challenge. The multifaceted nature of this task is underscored by diverse factors, including varying skill levels, working conditions, and the intricacies of production processes.

Traditionally, improvements in productivity within the garment industry have been anchored in time-tested methods such as time and motion studies. While effective to a certain extent, these conventional approaches, relying on expert observation and analysis of manual tasks, may fall short of capturing the nuanced complexity inherent in the myriad factors influencing productivity

The central challenge lies in developing a forward-looking predictive model adept at accurately forecasting workforce productivity within the garment industry. This undertaking involves the systematic collection and analysis of an array of data points pertaining to worker performance, process efficiency, and prevailing environmental conditions. The goal is to discern patterns and unravel the intricate interplay of factors that exert influence on productivity.

Recognizing that efficient workforce utilization is a linchpin for maintaining competitiveness and sustainability in the garment industry, the need to predict workforce productivity becomes paramount. Anticipating and managing productivity can empower managers to make informed decisions regarding resource allocation, training initiatives, and process optimization.

In response to the limitations of traditional methods, a progressive shift towards a data-driven approach becomes imperative. Leveraging advanced analytics and machine learning, this project aspires to provide more accurate predictions compared to conventional methods. By meticulously collecting and analyzing comprehensive datasets, the research endeavors to construct a system capable of autonomously and accurately predicting workforce productivity.

The deployment of advanced algorithms in this data-driven approach is poised to unveil intricate relationships between various factors and productivity outcomes. This, in turn, promises to furnish valuable insights for refining processes and optimizing resource allocation within the garment industry.

The transformative potential of this data-driven paradigm is substantial. If successful, the project holds the promise of revolutionizing how managers in the garment industry approach and enhance workforce productivity. Through increased efficiency and competitiveness on a global scale, the ripple effects of this endeavor may contribute to reshaping the operational landscape of the garment industry.

CHAPTER 2
LITERATURE SURVEY

To solve the common problem in garment industries, which is actual productivity prediction of the employees [1] proposed a DNN or Deep neural network model. The experiment resulted in a promising prediction showing a minimal Mean Absolute Error of 0.086 which is less than baseline performance error of 0.15. The MSE performance here is about 0.018 combining the Training and Validation performance. The data preprocessed and prepared for the experiment was collected from a renowned company that manufactures garments in Bangladesh. The model used is a deep learning model with capabilities like optimization and tuning of hyperparameters through manual search then practical judgements. Even though the learning procedure is consistent, the model proposed cannot reduce the error rate any further while thorough the machine learning model that we proposed, we can lower the error rate further to 0.13. In the recent world, data mining and machine learning is in a great demand for measuring some important aspects. Lots of people working in the garment industry are female and [2] is an approach to analyze the working performances of women based on their activities done previously, making a use of the machine learning algorithms. The algorithms used are Decision Tree Classifier, Logistic regression, Random Forest Classifier and Stochastic Gradient Descent. The best result found after such experiments are from the logistic regression model which is 69%. The dataset used in this procedure was very little in amount, which is 512 and that too was collected from some directly asked questions to some specified people. The classification procedure of models might have given a huge accuracy in this case, on the basis of the dataset that was experimented on, as well as the model used in classification but the approach, we talked about in our paper is way better. We are not proposing some classification model but a regression model experimenting on a huge data set for more perfection in result and receive more accuracy. For solving the productivity prediction problem of the garment employees, the data mining technique can be explored and used. The state-of-the-art data mining can be applied in industrial data analysis, meaningful insight revelation and predicting productivity performance of the garment employees according to [3]. This technique applies 8 data mining procedures containing 6 evaluation metrics. The datasets used are also of two kinds namely, 3CD or 3-class dataset and 2CD or 2-class dataset. The techniques were all individual and were compared strictly to figure out the best one of them. The class imbalance problem between two type datasets was solved by an oversampling technique called SMOTE on the training data.

This oversampling technique was more effective on the 2CD classification. Highest accuracy without oversampling procedure was 83.89% while oversampling produced 86.39%. The AUC score turned out to be 0.90 in case of the approach. On the other hand, the machine learning approach we proposed goes on to be a regression model with the same dataset and performs better compared to this approach in order to bring out more accuracy by reducing the error rate.This section discusses the main studies which focused on the usage of Machine Learning (ML) algorithms and ensemble learning algorithms in various sectors prediction issues. Ensemble learning algorithms such as decision tree, adaBoost, Naïve Bayes, Random Forest and SVM were applied by study

Bhatia, Arora, and Tomar 2016 [4] for presence of diabetic retinopathy, the results proved that the model could help in detecting symptoms earlier. Outperformed results were found in a study conducted.

Kruppa et al. 2013 [5] for credit risk prediction using framework of machine learning algorithms such as random forests (RF), k-nearest neighbors (KNN) and bagged knearest neighbors (BKN). Furthermore, a study by Balla, Rahayu, and Purnama 2021 [6] proved a promising result in predicting employee's productivity which is one of the most substantial factors in any organization. The study applied three classification algorithms namely, Neural Network (NN), Random Forest (RF) and Regressi Linier (RL). Random forest showed minimal values of correlation coefficient, MAE, and RMSE, which reflect that RF is very appropriate in predicting employee's productivity. Decision tree classification algorithms utilized by Attygalle and Abhayawardana 2021 [7] for investigating and visualizing employee productivity and any other social phenomenon with evidence. Moreover, decision tree methods and data mining tools employed by Ďurica, Frnda, and Svabova 2019 [8] to build a model for predicting financial difficulties of polish companies. The results presented prediction power around 98% and more.

In addition, Mahoto et al. 2021 [9] had used three machine learning algorithms (Multiclass Random Forest, Multiclass Logistic Regression, Multiclass one-vs-all) in order to help business workers to set product pricing and discounts depending on customer behavior, the model showed outstanding results in product price prediction. On the other hand, prediction model has been built by study Sorostinean, Gellert, and Pirvu 2021 [10] using decision tree methods and data mining tools for investigating the effect of decision tree methods and ensemble learning for improving performance prediction in assembly assistance system.

The results demonstrated that the gradient boosted decision trees was the best through all the decision treebased methods. Some studies evaluated worker ‘s performance of textile company by using ML and ensemble learning algorithm, such as study asSaad. [11] which applied different Machine learning algorithms including, decision tree and bagging algorithm to achieve the highest accuracy. The CHAID model produced high-level specificity and sensitivity. Four different ML algorithms including, support vector machine, optimized support vector machine (using genetic algorithm), random forest, XGBoost and Deep Learning were used.

El Hassani, El Mazgualdi, and Masrour [12] for predicting the overall equipment effectiveness (OEE) which is a performance measurement of manufacturing industry. Deep learning and random forest with cross validation manifest the best results for predicting OEE. Additionally, an approach built in study De Lucia,

Pazienza, and Bartlett [13] of ML and logistic regression used for financial performance prediction by focusing on predicting the accuracy of main financial indicators such as Return of Equity (ROE) and Return of Assets (ROA). The ML algorithms were performed perfectly for predicting ROE and ROA. All studies and research work mentioned above focused on combining two or more classifiers and how this integration of different techniques and algorithms can help in prediction. This research focuses on combining classification algorithms with bagging and Adaboost. In addition, the iterations from 1 to 100 are recorded to study how these combinations influence the accuracy, RMSE, and MAE values of predicting employees’ productivity. Detailed comparisons between our study and the studies mentioned.

CHAPTER 3
EXISTING SYSTEM

3.1 Overview

The project involves the development of predictive models, namely an Artificial Neural Network (ANN) and a Random Forest Regressor, to predict the actual productivity in a manufacturing setting based on various input features. The following paragraphs detail the different steps involved in this project.

- Dataset Upload: The project begins with the upload of the dataset from the "garments_worker_productivity.csv" file. The dataset is then loaded into a Pandas DataFrame for further analysis and modeling. Initial exploration through the display of the first few rows provides a glimpse into the structure of the data.
- Preprocessing: A preliminary step in data analysis involves identifying and handling missing values. The project checks for null values and subsequently drops the 'wip' column from the dataset. Descriptive statistics and information about the dataset, including its dimensions and data types, are presented for a comprehensive overview.
- Target Variable Declaration: The target variable 'actual_productivity' is identified, which the models aim to predict. The project further examines the dataset by visualizing pair plots and correlation heatmaps to discern potential relationships among variables.
- Label Encoding and Feature Engineering: Categorical columns such as 'quarter,' 'department,' and 'day' are label-encoded to facilitate their inclusion in the models. Additionally, the 'date' column is converted to a datetime object, and relevant components like year, month, and day are extracted and converted to integers. The original 'date' column is dropped from the dataset.
- Data Splitting: The dataset is split into features (X) and the target variable (y). Standard scaling is applied to the feature set. The data is then divided into training and testing sets using the train_test_split function from scikit-learn.

- Artificial Neural Network (ANN) Modeling: A sequential model with three layers (64 neurons, 32 neurons, and 1 neuron) is implemented using Keras. The model is compiled with mean squared error loss and the Adam optimizer. It is trained on the training set for 200 epochs with a batch size of 16. The model's performance is evaluated using the mean squared error and R-squared (R2) score on the test set.
- Random Forest Regressor Modeling: A Random Forest Regressor model is employed with 10 estimators. The model is trained on the training set, and predictions are made on the test set. The R-squared (R2) score, mean squared error, and mean absolute error are used to evaluate the performance of the Random Forest model.
- Performance Comparison: The project concludes with a comparison of the performance metrics, including R-squared (R2) score, mean squared error (MSE), and mean absolute error (MAE), between the ANN and Random Forest models. Visualizations, such as scatter plots, are generated to showcase the model predictions against actual values.

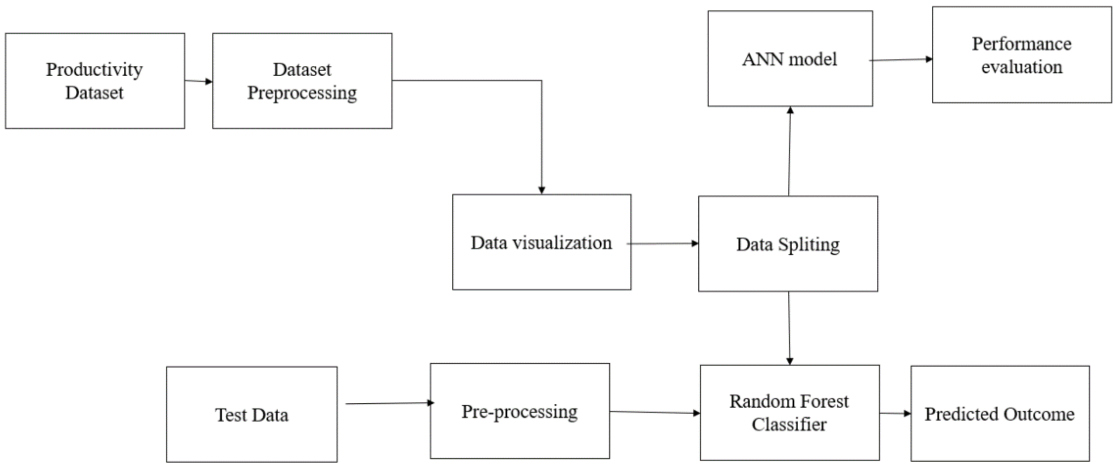


Figure 3.1: Architectural block diagram of existing system.

3.2 Data Preprocessing in Machine Learning

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

Why do we need Data Pre-processing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set

Handling Missing Values: Begin by identifying and addressing any missing values in the dataset. This might involve imputing missing values using mean or median for numerical features and mode for categorical features. Alternatively, you may choose to remove instances with missing values, depending on the impact on the dataset's integrity.

Removing Irrelevant Features: Analyze the dataset to identify features that do not contribute significantly to the prediction of defects. Removing these irrelevant features not only simplifies the model but also enhances its efficiency and reduces the risk of overfitting.

Normalization of Numerical Values: Normalize numerical features to ensure uniformity in their scales. This step is crucial, especially when using algorithms sensitive to the magnitude of values. Common normalization techniques include Min-Max scaling or Z-score normalization.

Encoding Categorical Variables: Ensemble learning algorithms typically require numerical input. Therefore, categorical variables need to be encoded. This can be achieved through techniques such as one-hot encoding, where categorical variables are converted into binary vectors, making them compatible with the algorithms.

Dealing with Imbalanced Data (if applicable): In software defect prediction, datasets may sometimes be imbalanced, with a disproportionate number of non-defective instances compared to defective ones. Addressing this imbalance can involve techniques like oversampling the minority class or undersampling the majority class to ensure the model is not biased towards predicting the majority class.

Feature Scaling (if applicable): Some ensemble learning algorithms, such as those based on distance metrics, benefit from feature scaling. Ensure that features are appropriately scaled to prevent certain features from dominating the learning process.

3.3 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting

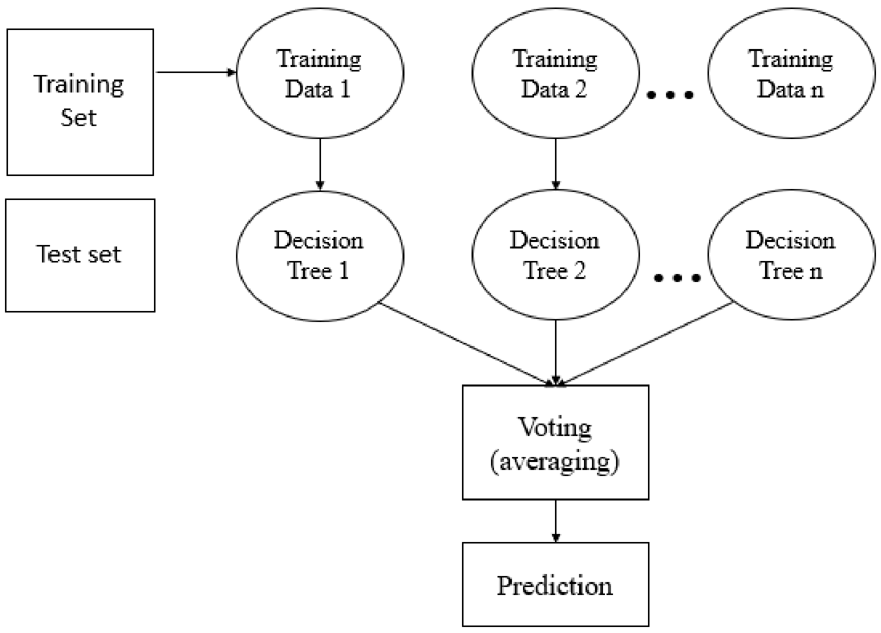


Fig. 3.2: Random Forest algorithm.

3.3.1 Random Forest Algorithm

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively

3.3.2 Important Features of Random Forest

- **Diversity-** Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality-** Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization-**Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

- **Train-Test split**- In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability**- Stability arises because the result is based on majority voting/ averaging.

3.3.3 Assumptions For Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

3.3.4 Types of Ensembles

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

3.4 Advantages

The presented Tkinter-based surface identification project utilizing Decision Tree and Random Forest classifiers offers several advantages:

User-Friendly Interface: The graphical user interface (GUI) created with Tkinter enhances user interaction by providing buttons for various functionalities. This makes the application accessible and easy to use for individuals without programming expertise.

Dynamic Dataset Upload: The ability to upload datasets through the "Upload Dataset" button allows users to work with diverse datasets effortlessly. This dynamic approach supports the application's adaptability to different use cases and datasets.

Comprehensive Preprocessing: The "Preprocess Dataset" button automates preprocessing steps, such as handling missing values and label encoding. The generated count plot aids in visualizing the distribution of classes, offering insights into the dataset's characteristics.

Transparent Train-Test Splitting: The application transparently communicates the process of splitting the dataset into training and testing sets. Information about the total records and the sizes of the training and testing sets is provided, enhancing transparency in the data preparation phase.

Multiple Classifier Options: The inclusion of both Decision Tree and Random Forest classifiers offers flexibility to users. They can choose between different algorithms based on the nature of their data and the problem at hand, allowing for experimentation and model comparison.

Performance Metrics and Visualization: The application computes and displays essential performance metrics, including accuracy, confusion matrix, and classification report. The incorporation of ROC curves visually represents the models' performance, aiding users in assessing the classifiers' ability to discriminate between classes.

Prediction on Test Data: The "Prediction" button allows users to make predictions on new test data using the trained Decision Tree classifier. This functionality is valuable for real-world applications where the model is deployed on unseen data.

Comparison Graph: The "Comparison Graph" button generates a bar graph comparing performance metrics between the Decision Tree and Random Forest classifiers. This visual representation facilitates a quick and clear understanding of how different algorithms perform on the given dataset.

Scalability and Adaptability: The modular structure of the application makes it scalable and adaptable. Users can extend the functionality by adding more classifiers or incorporating additional preprocessing steps to suit specific project requirements.

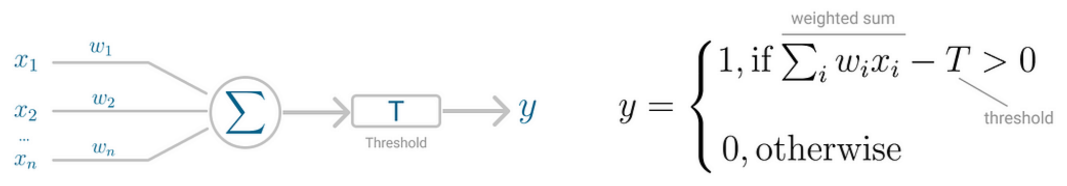
Educational Value: The project serves as an educational tool for individuals learning about machine learning and classification problems. The GUI-based approach and step-by-step functionalities make it suitable for educational purposes and practical experimentation.

CHAPTER 04

PROPOSED SYSTEM

4.1 ANN Classifier

Although today the Perceptron is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of perception, seeing, and recognizing images. Interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound, temperature, etc. — the "phenomenal world" with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information. Rosenblatt's perceptron machine relied on a basic unit of computation, the neuron. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights. The major difference in Rosenblatt's model is that inputs are combined in a weighted sum and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.



x_1
w_1
x_2
w_2
x_n
w_n
$\sum$
T
Threshold
y
weighted sum
$y = \begin{cases} 1, \text{if } \sum_i w_i x_i - T > 0 \\ 0, \text{otherwise} \end{cases}$
threshold

Fig. 4.1: Perceptron neuron model (left) and threshold logic (right).

Threshold *T* represents the activation function. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

Perceptron for Binary Classification

With this discrete output, controlled by the activation function, the perceptron can be used as a binary classification model, defining a linear decision boundary.

It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary. The perceptron loss function is defined as below:

$$\underbrace{D(w,c)}_{\text{distance}} = -\sum_{\underset{\text{misclassified observations}}{i\in \mathrm{M}}} \overset{\text{output}}{y_i}\,(x_i w_i + c)$$

To minimize this distance, perceptron uses stochastic gradient descent (SGD) as the optimization function. If the data is linearly separable, it is guaranteed that SGD will converge in a finite number of steps. The last piece that Perceptron needs is the activation function, the function that determines if the neuron will fire or not. Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense! The sigmoid function maps any real input to a value that is either 0 or 1 and encodes a non-linear function. The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.

But, if you look at Deep Learning papers and algorithms from the last decade, you'll see the most of them use the Rectified Linear Unit (ReLU) as the neuron's activation function. The reason why ReLU became more adopted is that it allows better optimization using SGD, more efficient computation and is scale-invariant, meaning, its characteristics are not affected by the scale of the input. The neuron receives inputs and picks an initial set of weights random. These are combined in weighted sum and then ReLU, the activation function, determines the value of the output.

.

EDA

The following EDA performed in this work:

- **Histogram Plots**: Histograms are generated for all numeric features in the dataset to visualize the distribution of each variable. This code snippet creates a figure with a size of 15x15, specifies the axis for plotting, and then generates histograms for each numeric feature in the DataFrame 'df.' These histograms help you understand the distribution of values within each feature.

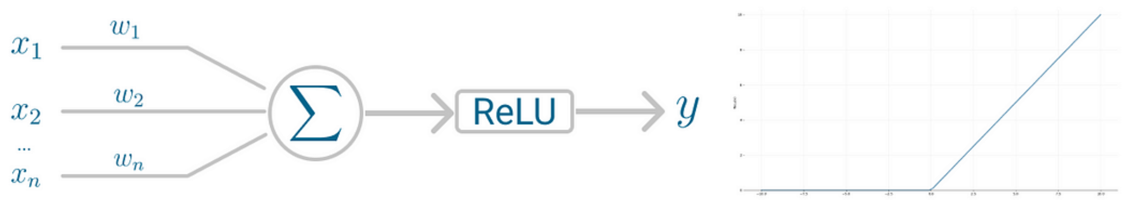


Fig. 4.2: Perceptron neuron model (left) and activation function (right).

Perceptron uses SGD to find, or you might say learn, the set of weight that minimizes the distance between the misclassified points and the decision boundary. Once SGD converges, the dataset is separated into two regions by a linear hyperplane. Although it was said the Perceptron could represent any circuit and logic, the biggest criticism was that it couldn't represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different. This was proved almost a decade later and highlights the fact that Perceptron, with only one neuron, can't be applied to non-linear data.

4.1.1 ANN

The ANN was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear. A ANN has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a ANN can use any arbitrary activation function. ANN falls under the category of feedforward algorithms because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer. If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning. This is where Backpropagation comes into play.

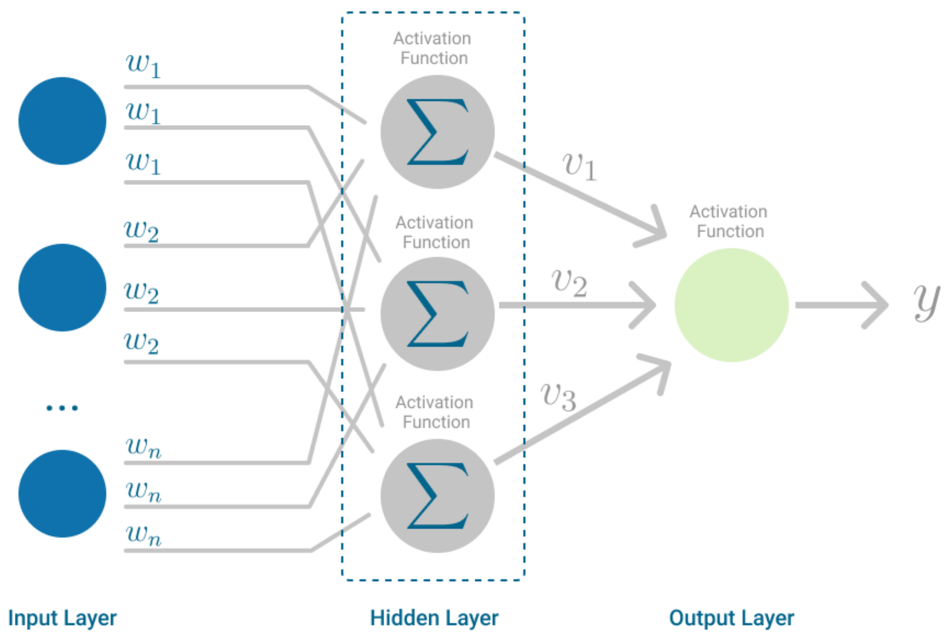


Fig. 4.3: Architecture of ANN.

Backpropagation: Backpropagation is the learning mechanism that allows the ANN to iteratively adjust the weights in the network, with the goal of minimizing the cost function. There is one hard requirement for backpropagation to work properly. The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a bounded derivative because Gradient Descent is typically the optimization function used in ANN. In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That's how the weights are propagated back to the starting point of the neural network. One iteration of Gradient Descent is defined as follows:

$$\underbrace{\Delta_w(t)}_{\text{Gradient Current Iteration}} = -\overset{\text{Bias}}{\varepsilon}\underbrace{\frac{\overset{\text{Error}}{dE}}{dw_{(t)}}}_{\text{Weight vector}} + \overset{\text{Learning Rate}}{\alpha}\underbrace{\Delta_{w(t-1)}}_{\text{Gradient Previous Iteration}}$$

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn't changed more than a specified convergence threshold, compared to the previous iteration.

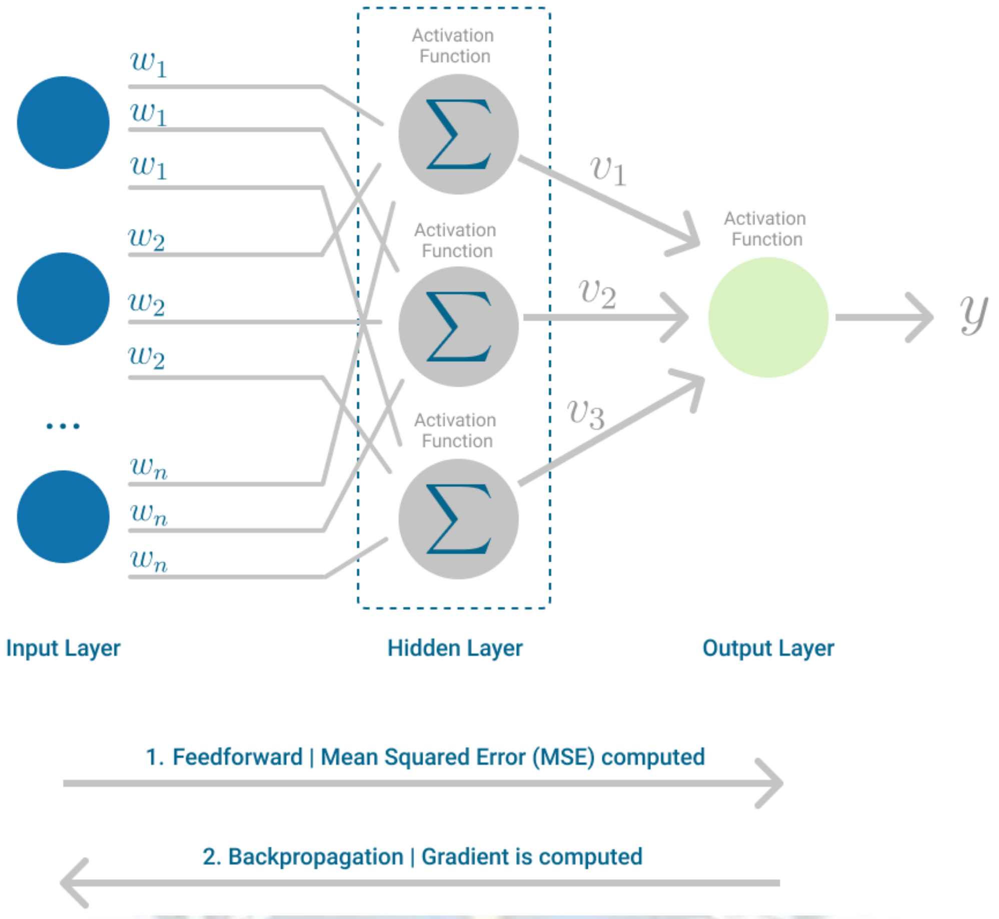


Fig. 4.4: ANN, highlighting the Feedforward and Backpropagation steps.

4.2 Dis advantages of ANN model

Overfitting: Traditional ANNs are prone to overfitting, especially when dealing with small datasets or complex architectures. Overfitting occurs when the model learns the training data too well, including noise and outliers, leading to poor generalization on new data.

Vanishing Gradient Problem: In deep networks, the backpropagation algorithm can encounter difficulties propagating gradients backward through many layers. This can result in slow or stalled learning for early layers, known as the vanishing gradient problem.

Computational Intensity: Training large ANNs can be computationally intensive and time-consuming, especially when dealing with complex architectures and big datasets. This can limit their scalability and practicality in certain applications.

Need for Sufficient Data: ANNs, especially deep learning models, often require a large amount of labeled data for effective training. Insufficient data can lead to poor model performance and generalization.

Interpretability: ANNs, particularly deep neural networks, are often considered as "black-box" models, making it challenging to interpret and understand the decision-making process. Interpretability is crucial in applications where transparency and explainability are

CHAPTER 5

UML DAIGRAMS

5.1 Class Diagram

Class diagram is *a static diagram. It represents the static view of an application*.

Note:

A='No Fault'

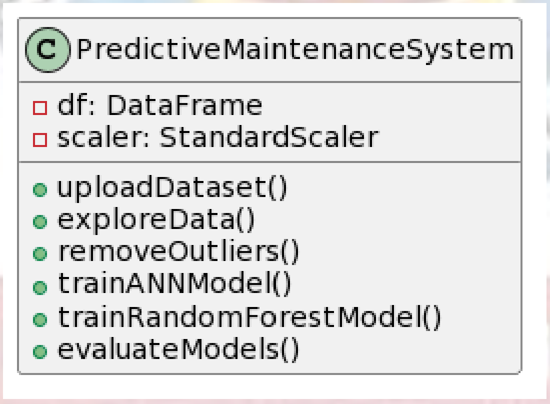
B='LLG Fault'

C='LLLG Fault'

D='LG Fault'

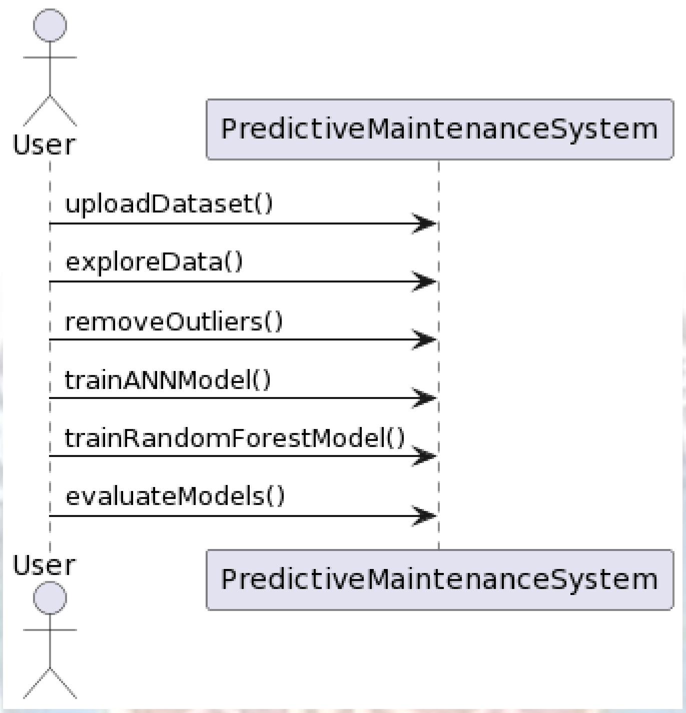
E='LLL Fault'

F='LL Fault'



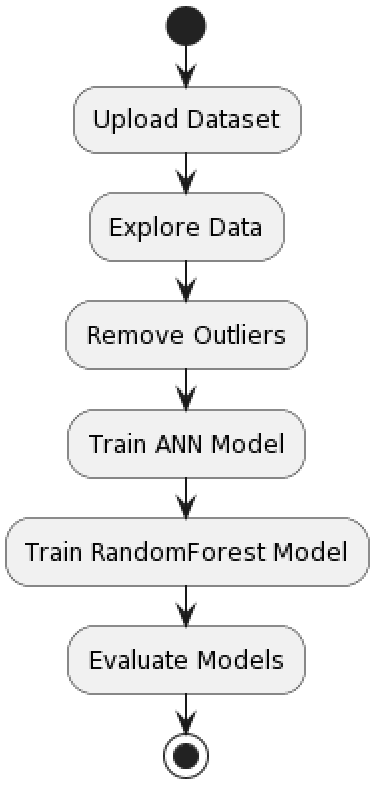
5.2 Sequence diagram

Sequence diagram is an interaction diagram that details how operations are carried out.

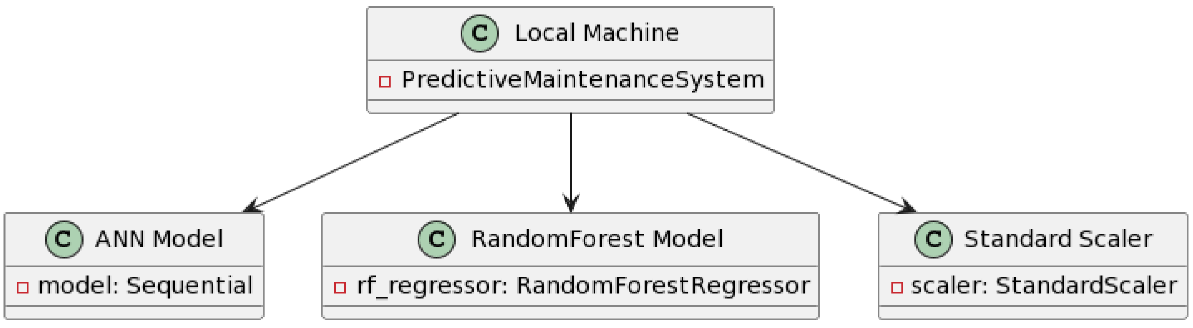


5.3 Activity Diagram

Activity diagram is another important diagram in UML to *describe the dynamic aspects of the system*

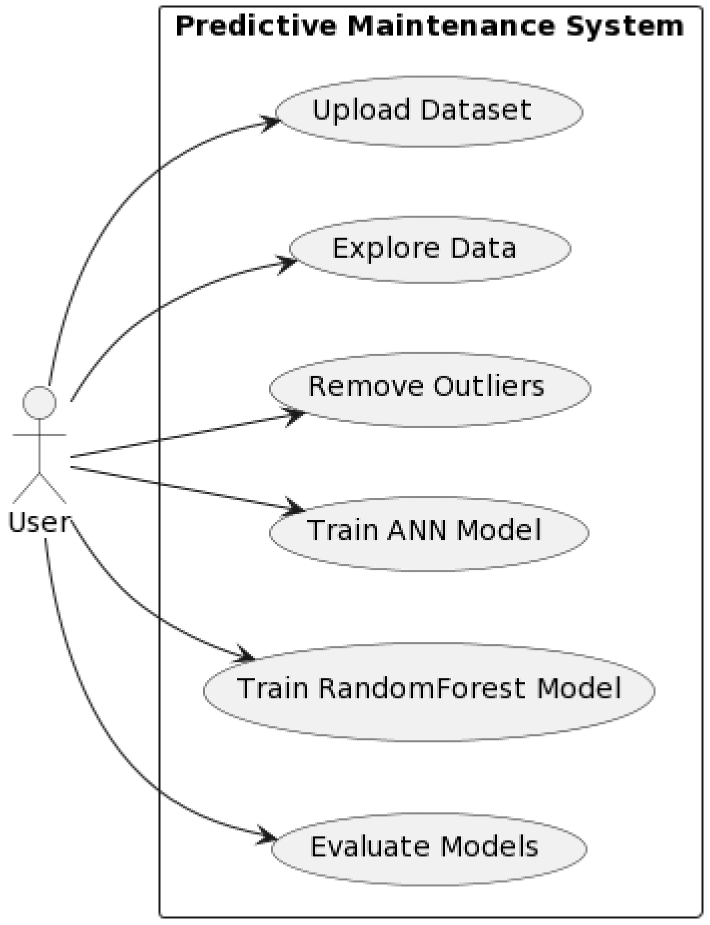


5.4 Deployement Diagram



5.4 Use Case Diagram

The purpose of use case diagram is *to capture the dynamic aspect of a system*.



CHAPTER 6
SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let’s see how Python dominates over other languages.

Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C.

Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

Improved Productivity

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

Simple and Easy

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.
Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.
The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands.

The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at Altisource’s in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as

"Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels.

Modules Used in Project

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little 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.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupiter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code.

Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.
The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

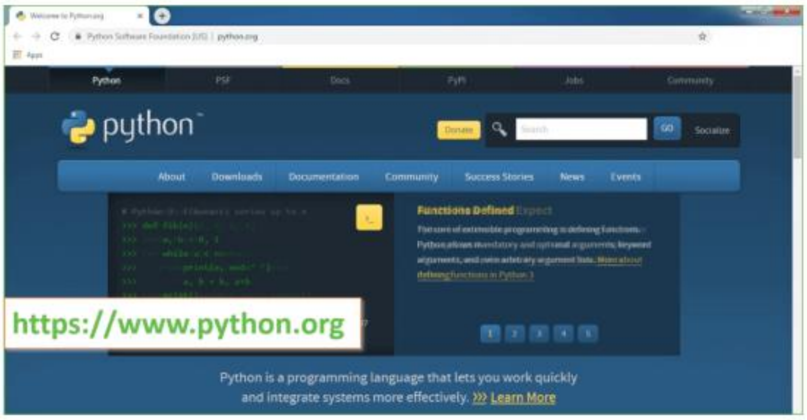
How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.
Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.
Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheat sheet here. The steps on

how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

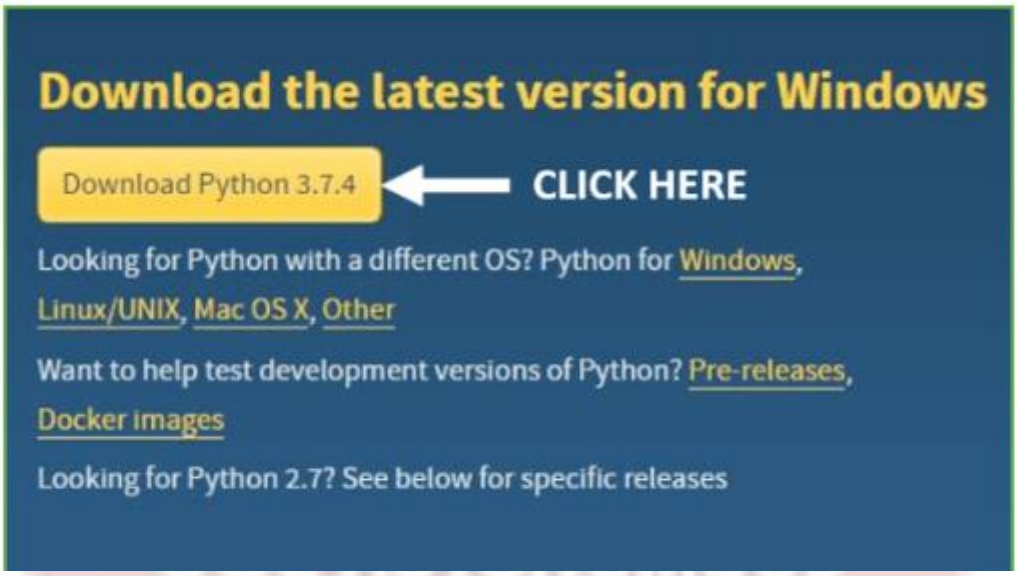
Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org



Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Download the latest version for Windows

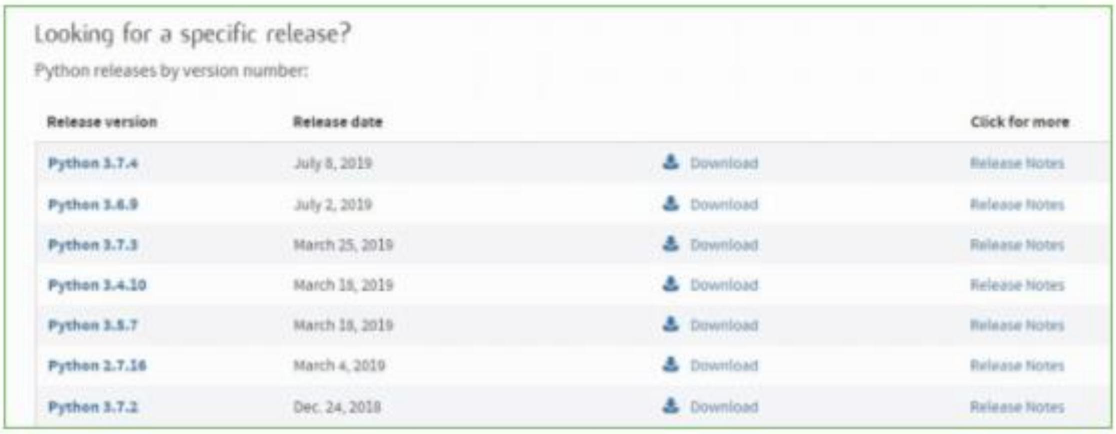
Download Python 3.7.4 ← CLICK HERE

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in yellow colour or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



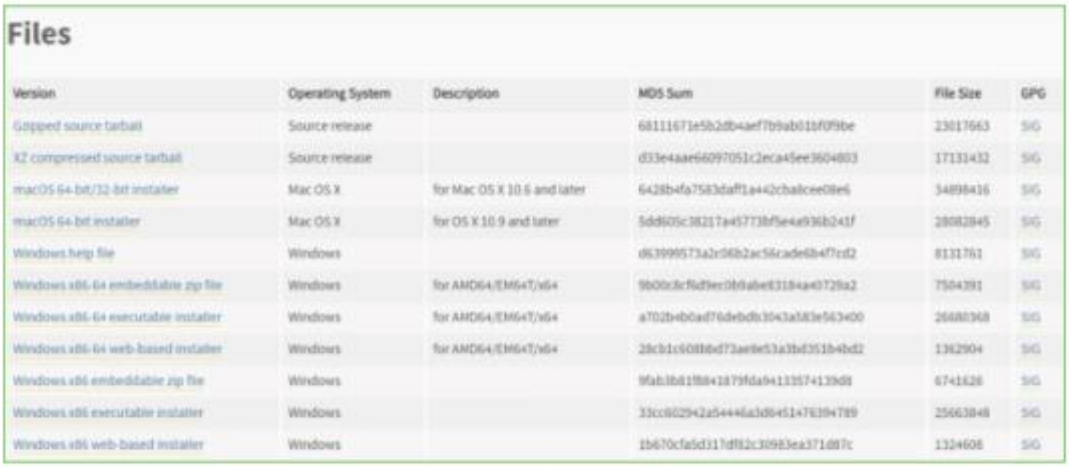
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c56b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

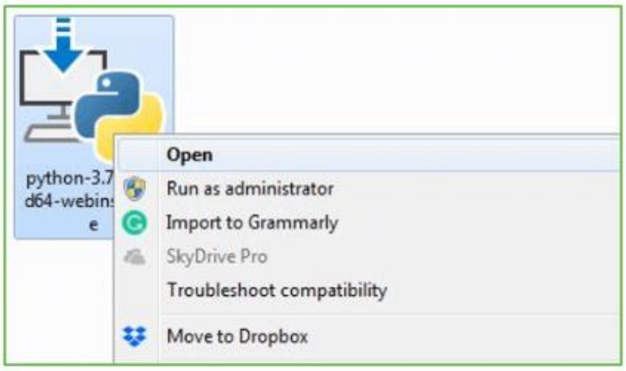
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

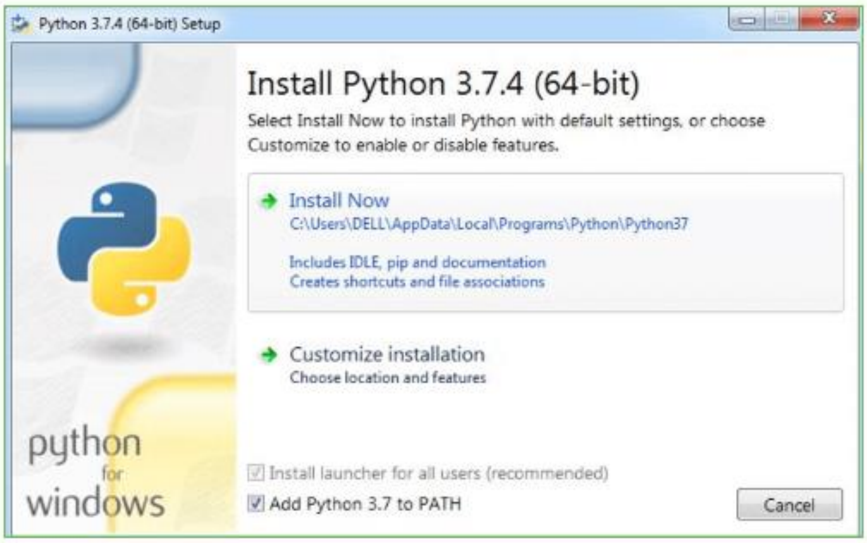
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

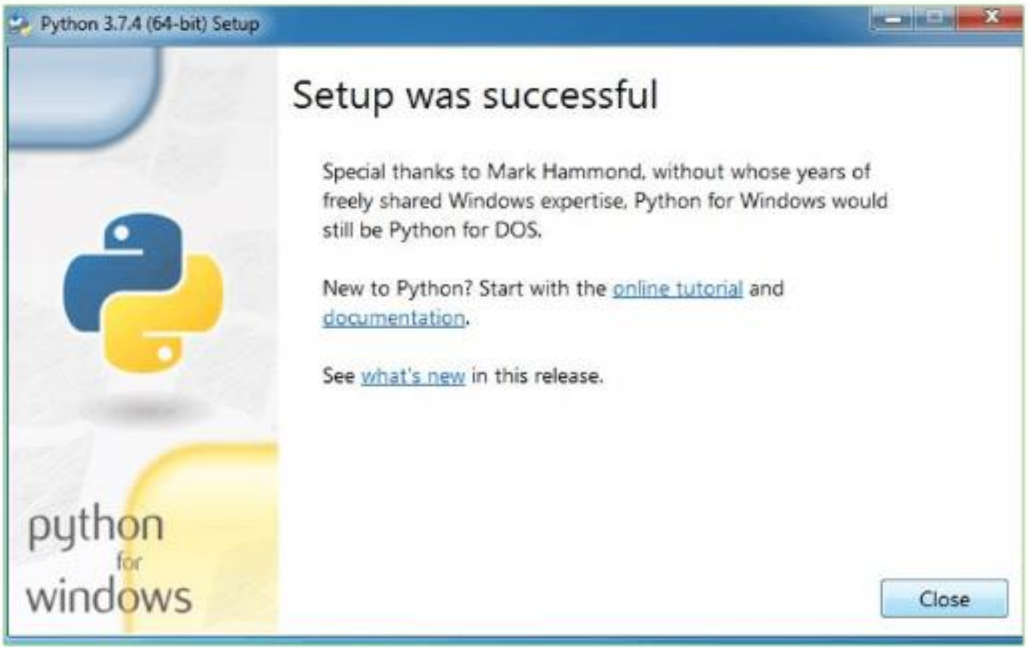
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[✓] Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

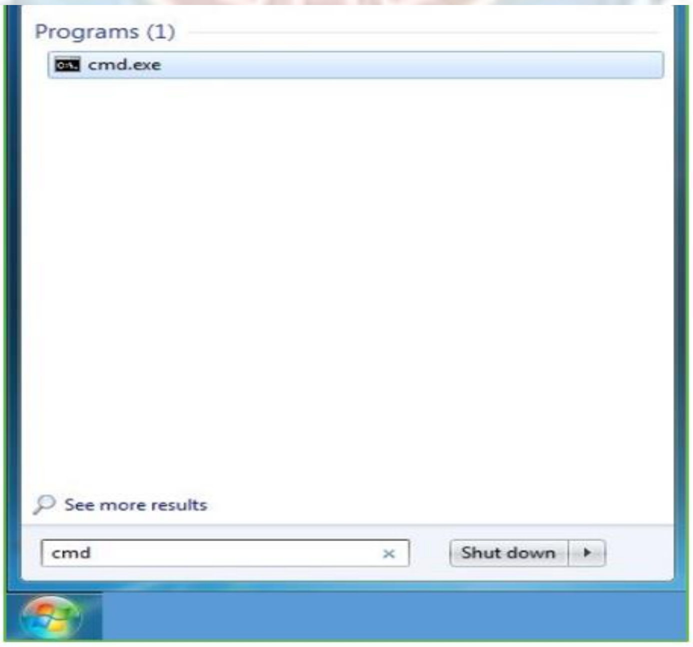
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

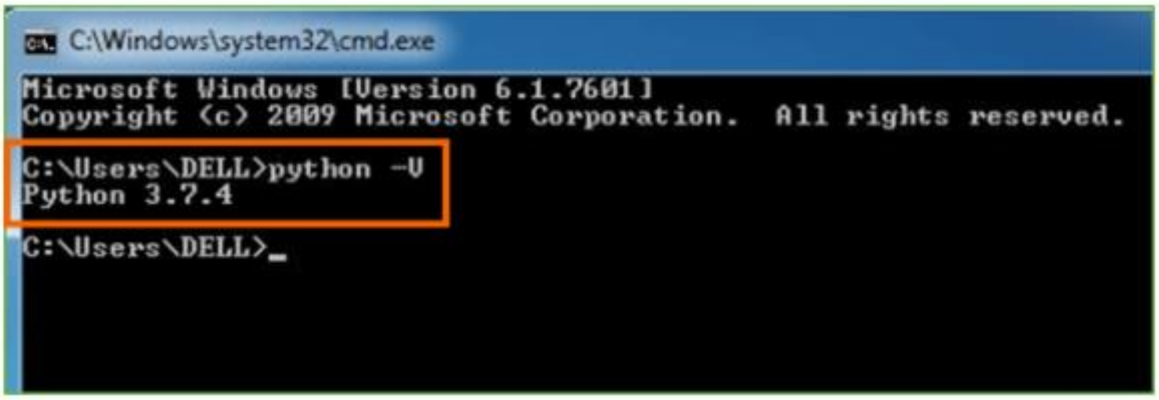
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

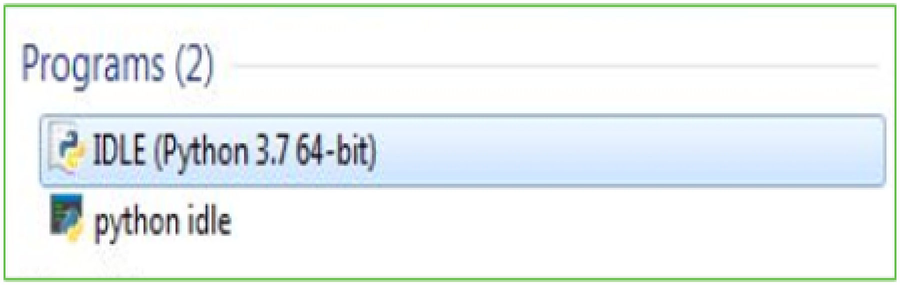
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

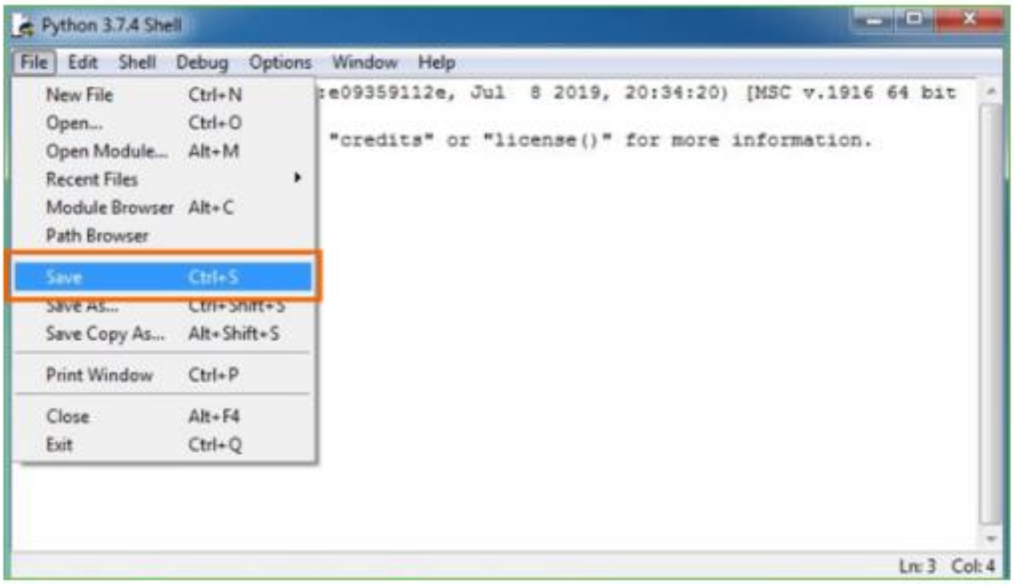
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

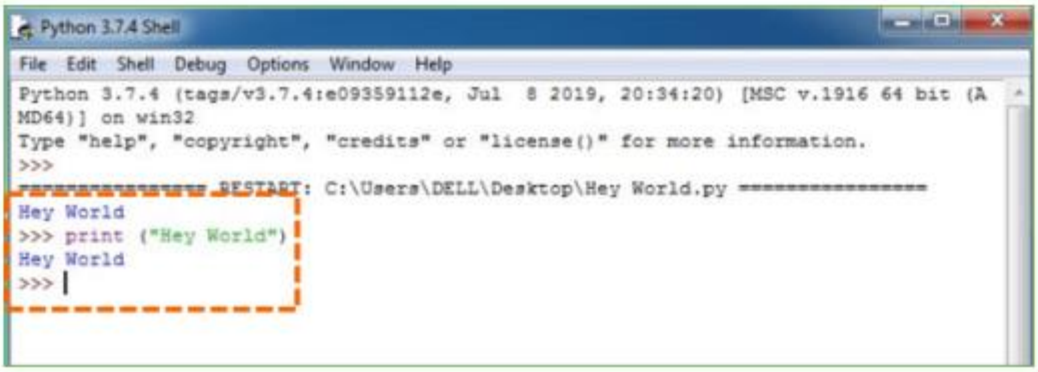


```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit

 "credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print (“Hey World”) and Press Enter.



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (A
MD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\DELL\Desktop\Hey World.py ================
Hey World
>>> print ("Hey World")
Hey World
>>>
```

CHAPTER 7
SYSTEM REQUIREMENTS SPECIFICATIONS

7.1 Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

7.2 Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system	:	Windows, Linux
Processor	:	minimum intel i3
Ram	:	minimum 4 GB
Hard disk	:	minimum 250GB

CHAPTER 8
FUNCTIONAL REQUIREMENTS

Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words, the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Classified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error, he/she has committed.

This application must be able to produce output at different modules for different inputs.

Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements.

CHAPTER 9
SOURCE CODE

```
    import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from keras.models import Sequential
from keras.layers import Dense
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv(r"garments_worker_productivity.csv")
df.head()
sns.pairplot(df[['smv', 'wip', 'over_time', 'actual_productivity']])
plt.show()
#checking null values
df.isnull().sum()
df=df.drop('wip',axis=1)
# Dataset analysis
df.describe()
# dataset information
```

```
df.info()

#checking null values

df.isnull().sum()

#Labeling the text data columns

label_encoders = {}

categorical_columns = ['quarter', 'department', 'day']

for col in categorical_columns:

 le = LabelEncoder()

df[col] = le.fit_transform(df[col])

label_encoders[col] = le

df['date'] = pd.to_datetime(df['date'])

# Extract relevant components (e.g., year, month, day)

df['year'] = df['date'].dt.year

df['month'] = df['date'].dt.month

df['day'] = df['date'].dt.day

# Convert to integer

df['year'] = df['year'].astype(int)

df['month'] = df['month'].astype(int)

df['day'] = df['day'].astype(int)

# Drop the original date column if no longer needed

df = df.drop(columns=['date'])

df.head()

plt.figure(figsize=(10, 8))

correlation_matrix = df.corr()
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()

# Outliers visualization

plt.figure(figsize=(8, 6))

plt.boxplot(df['targeted_productivity'])

plt.title('Box Plot for Outlier Detection')

plt.show()

#Removing Outliers

feature_mean = df['targeted_productivity'].mean()

feature_std = df['targeted_productivity'].std()

threshold = 6

# Calculate Z-scores

z_scores = np.abs((df['targeted_productivity'] - feature_mean) / feature_std)

# Identify outliers

outliers = z_scores > threshold

# Remove outliers

df = df[~outliers]

df

outliers

plt.figure(figsize=(10, 6))

sns.boxplot(x='department', y='targeted_productivity', data=df)

plt.title('Box Plot of Targeted Productivity by Department')

plt.show()
```

```
#Selecting dependent and independent variable

X = df.drop(columns=['actual_productivity'])# Features

scaler = StandardScaler()

X= scaler.fit_transform(X)

X

y = df['actual_productivity']  # Target variable

X

# Splitting the dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#from sklearn.decomposition import PCA

#pca = PCA(n_components = 14)

#X_train = pca.fit_transform(X_trained)

X_train

# # ANN model

model = Sequential()

model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean_squared_error', optimizer='adam')

model.fit(X_train, y_train, epochs=200, batch_size=16, validation_split=0.1)

loss = model.evaluate(X_test, y_test)

predictions = model.predict(X_test)

# Evaluate the model

r2_score(y_test,predictions
mse = mean_squared_error(y_test, predictions)

print(f"Mean Squared Error: {mse}")

mae = mean_absolute_error(y_test,predictions)

mae

actual_values = y_test

predicted_values = predictions

# Generate a line for perfect predictions (y = x)

x = np.linspace(min(actual_values), max(actual_values), 100)

y = x

# Create the scatter plot

plt.figure(figsize=(8, 8))

plt.scatter(actual_values, predicted_values, c='blue', label='Predictions')

plt.plot(x, y, color='red', label='Perfect Predictions')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title('ANN Model Predictions')

plt.legend()

plt.show()

# ### RandomForest Regressor model

rf_regressor = RandomForestRegressor(n_estimators=10, random_state=42)

rf_regressor.fit(X_train, y_train)

y_pred_rf = rf_regressor.predict(X_test)

r2 = r2_score(y_test,y_pred_rf)

print(f"R-squared (R2): {r2}"
# Evaluate the model

mse = mean_squared_error(y_test, y_pred_rf)

print(f"Mean Squared Error: {mse}")

mae = mean_absolute_error(y_test,y_pred_rf)

mae

r2 = r2_score(y_test,y_pred_rf)

print(f"R-squared (R2): {r2}")

# In[77]:

actual_values = y_test

predicted_values = y_pred_rf

# Generate a line for perfect predictions (y = x)

x = np.linspace(min(actual_values), max(actual_values), 100)

y = x

# Create the scatter plot

plt.figure(figsize=(8, 8))

plt.scatter(actual_values, predicted_values, c='blue', label='Predictions')

plt.plot(x, y, color='red', label='Perfect Predictions')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title('RandomForest Regressor Model Predictions')

plt.legend()

plt.show()
```

RESULTS AND DISCUSSION

10.1 Implementation description

The project code performs a regression task using two different models: an Artificial Neural Network (ANN) and a RandomForest Regressor. Here's a breakdown of the code:

- Data Loading: Reads a dataset from a CSV file ("garments_worker_productivity.csv"). Displays the first few rows of the dataset.
- Data Visualization: Uses Seaborn to create a pairplot of selected columns: 'smv', 'wip', 'over_time', 'actual_productivity'.
- Data Preprocessing: Checks for null values in the dataset. Drops the 'wip' column from the dataset.
- Dataset Analysis: Displays descriptive statistics and information about the dataset.
- Label Encoding: Encodes categorical columns ('quarter', 'department', 'day') using LabelEncoder.
- Feature Engineering: Converts the 'date' column to datetime and extracts components like year, month, and day. Drops the original 'date' column.
- Correlation Heatmap: Displays a heatmap of the correlation matrix for numerical features.
- Outlier Detection: Uses a box plot to visualize the outliers in the 'targeted_productivity' column. Removes outliers based on a Z-score threshold.
- Box Plot Visualization: Displays a box plot of 'targeted_productivity' by department.
- Selecting Dependent and Independent Variables: Defines features (X) and target variable (y). Scales the features using StandardScaler.
- Data Splitting: Splits the dataset into training and testing sets.
- ANN Model: Defines and compiles a Sequential ANN model. Fits the model to the training data. Evaluates the model on the test data. Generates predictions

and calculates R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE). Plots the scatter plot of actual vs. predicted values.

- RandomForest Regressor Model: Defines and fits a RandomForest Regressor model. Generates predictions and calculates R-squared, MSE, and MAE. Plots the scatter plot of actual vs. predicted values.

10.2 Dataset Description

The dataset contains information of productivity and operational metrics in a manufacturing or production environment. Here's a brief description of each column:

- date: This column represents the date on which the measurements recorded.
- quarter: Indicates the quarter of the year to which the data corresponds. Quarters typically divide a year into four equal periods.
- department: Represents the department within the organization or factory where the observations were made.
- day: Indicates the day of the week for the recorded data.
- team: Represents the team or group within the department.
- targeted_productivity: This column contain the targeted or planned productivity levels for the specified department or team.
- smv: Stands for "Standard Minute Value" and represents the time required to complete a standard amount of work. It is often used as a measure of labor intensity.
- wip: Stands for "Work in Progress" and represents the number of items or tasks that are currently being worked on but are not yet completed.
- over_time: Indicates the amount of overtime worked by the team or department.
- incentive: Represents any incentives provided to the workers, potentially based on their productivity or performance.
- idle_time: Represents the amount of time during which no production or work is taking place.
- idle_men: Indicates the number of workers who are idle during the recorded period.
- no_of_style_change: Represents the number of times a change in product or production style occurred.

— no_of_workers: Indicates the total number of workers involved in the production process.

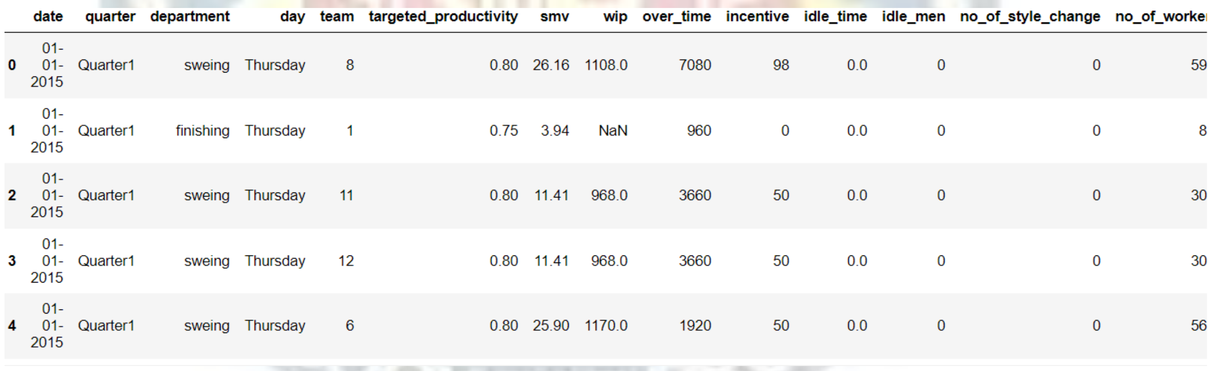
— actual_productivity: This column contains the actual productivity levels achieved during the recorded period. It is a key performance indicator (KPI) reflecting the efficiency of the production process.

10.3 Result Description

Figure 10.1: This figure presents a sample dataset, showing a snapshot or representation of the data that will be used for analysis or modeling.

Figure 10.2: A pairplot of each column in the dataset is displayed. A pairplot is a grid of scatterplots showing relationships between pairs of variables, providing a quick overview of the data's distribution and relationships.

Figure 10.3: This figure provides information about the dataset. It details such as the number of rows and columns, data types, and summary statistics for each column.



	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_worke
0	01-01-2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59
1	01-01-2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8
2	01-01-2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30
3	01-01-2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30
4	01-01-2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56

Figure 10.1: Displays the Sample dataset.

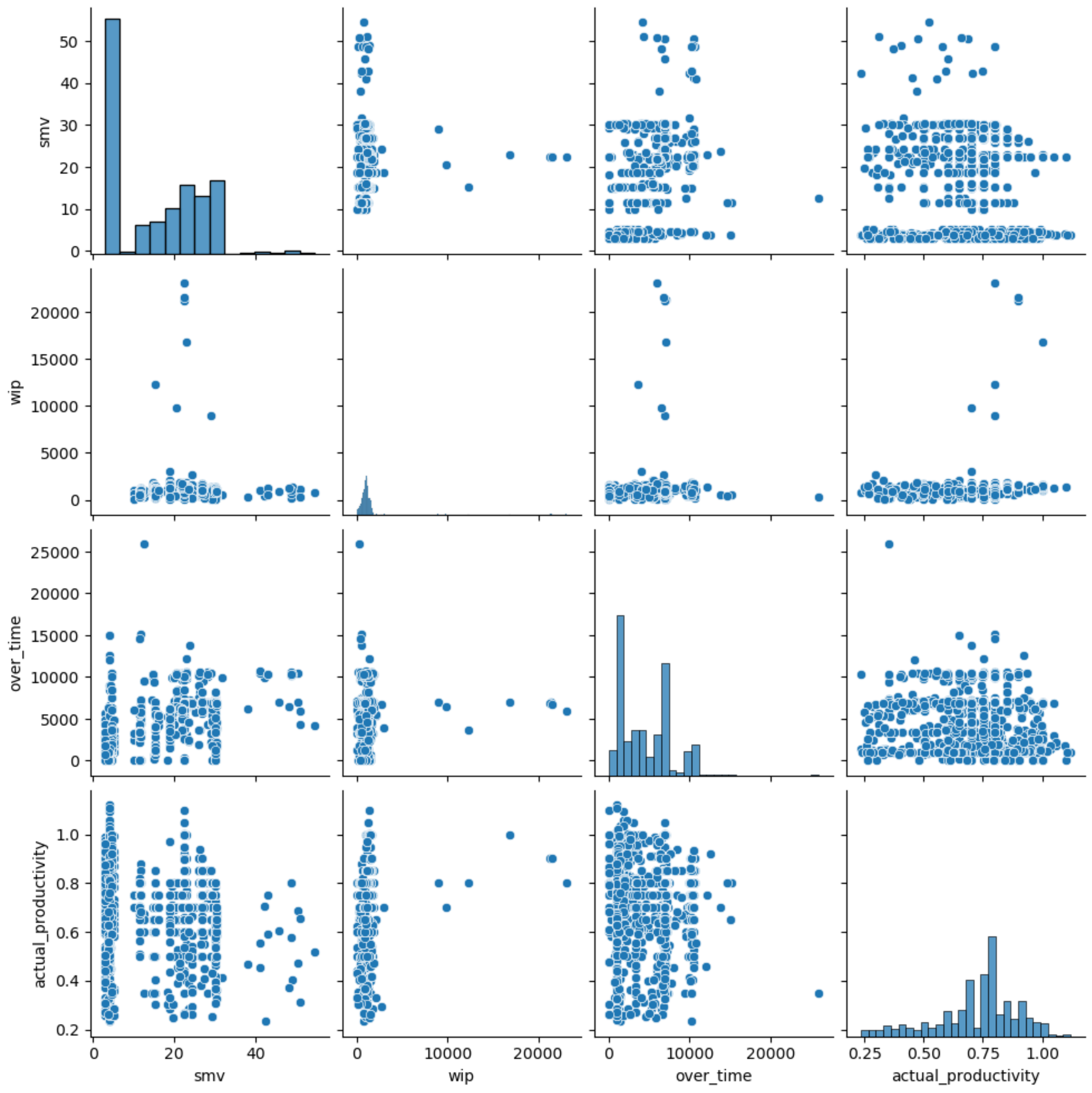
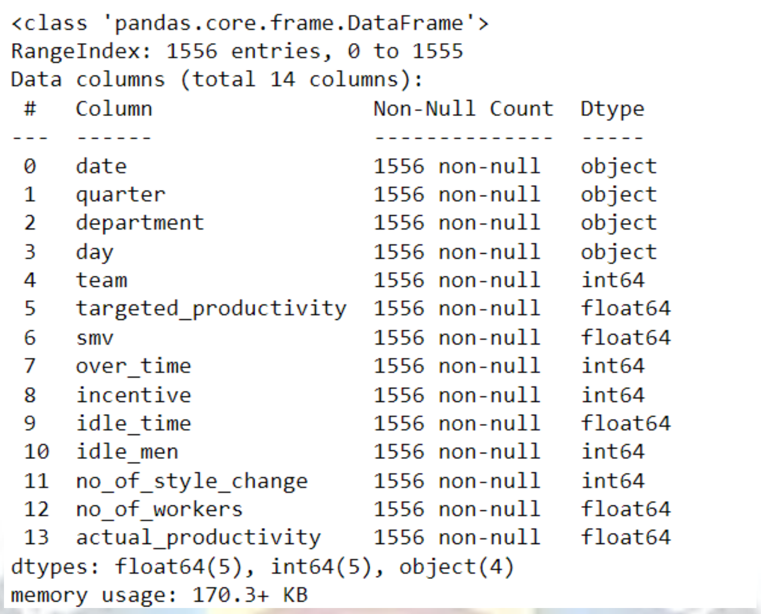


Figure 10.2: Pairplot of each columns in the dataset.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1556 entries, 0 to 1555
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   date                   1556 non-null   object
 1   quarter                1556 non-null   object
 2   department             1556 non-null   object
 3   day                    1556 non-null   object
 4   team                   1556 non-null   int64
 5   targeted_productivity  1556 non-null   float64
 6   smv                    1556 non-null   float64
 7   over_time              1556 non-null   int64
 8   incentive              1556 non-null   int64
 9   idle_time              1556 non-null   float64
 10  idle_men               1556 non-null   int64
 11  no_of_style_change     1556 non-null   int64
 12  no_of_workers          1556 non-null   float64
 13  actual_productivity    1556 non-null   float64
dtypes: float64(5), int64(5), object(4)
memory usage: 170.3+ KB
```

Figure 10.3: Presents the information about the dataset.

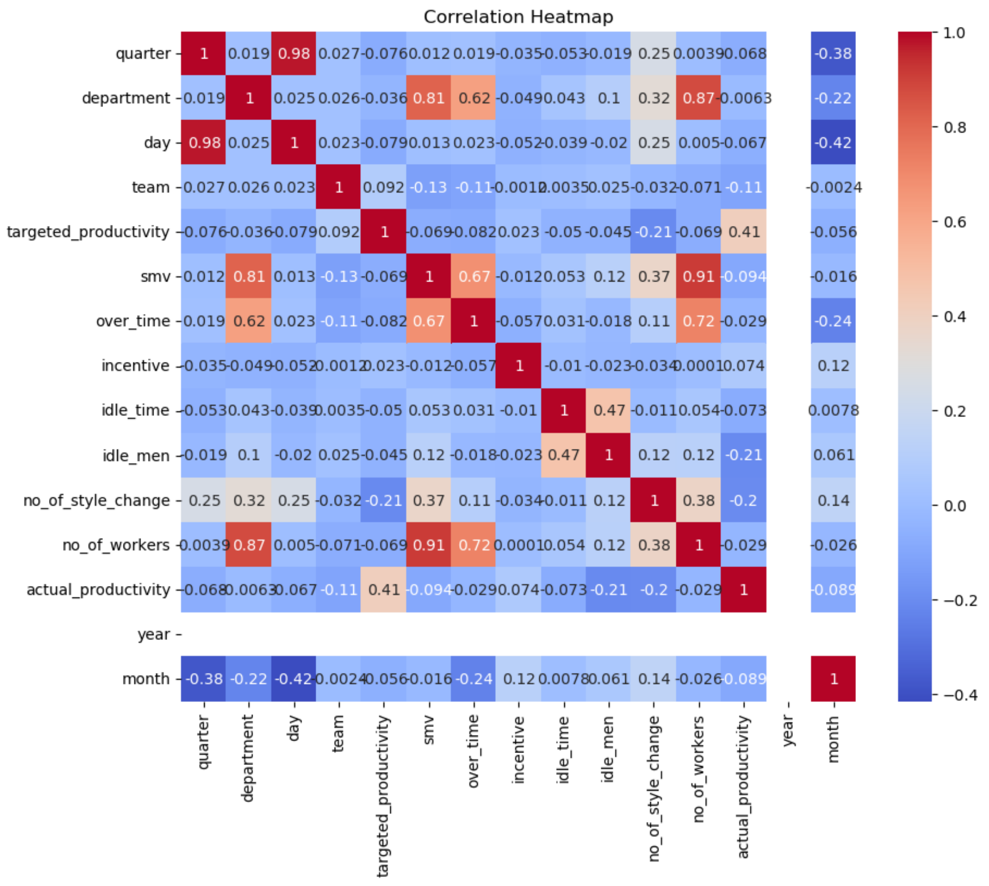


Figure 10.4: Displays the heatmap for dataset columns correletion.

Figure 10.4: A heatmap is shown, depicting the correlation between columns in the dataset. It helps to visualize the strength and direction of linear relationships between variables.

Figure 10.5: Two different aspects are mentioned for Figure 5. Firstly, it displays a boxplot designed to detect outliers in the dataset. Secondly, it shows a specific boxplot related to the "productivity" column, indicating the distribution and potential outliers in that particular feature.

Figure 10.6: This figure illustrates the performance metrics of an Artificial Neural Network (ANN) model. These metrics has accuracy, precision, recall, and F1 score, providing an evaluation of the model's effectiveness

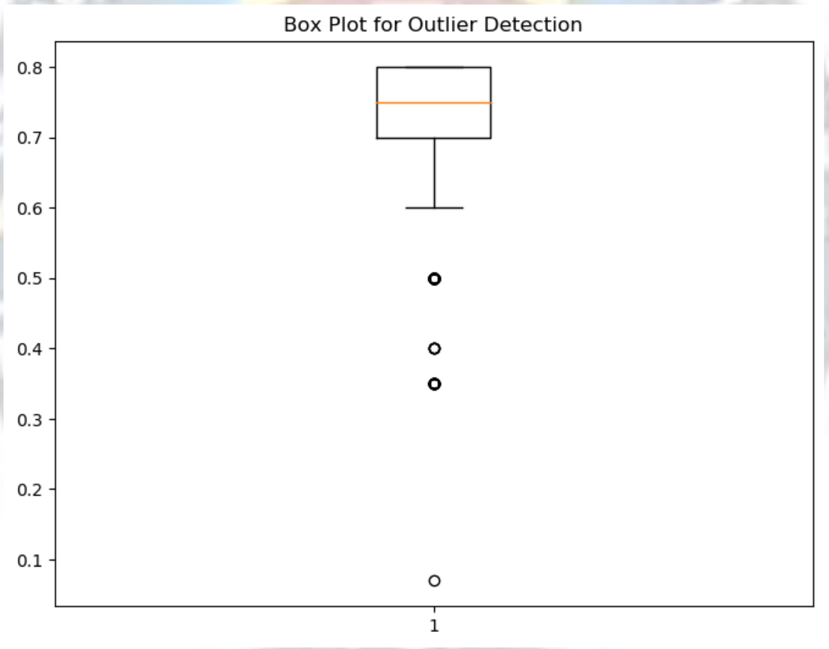


Figure 10.5: Displays the boxplot to detect outliers.

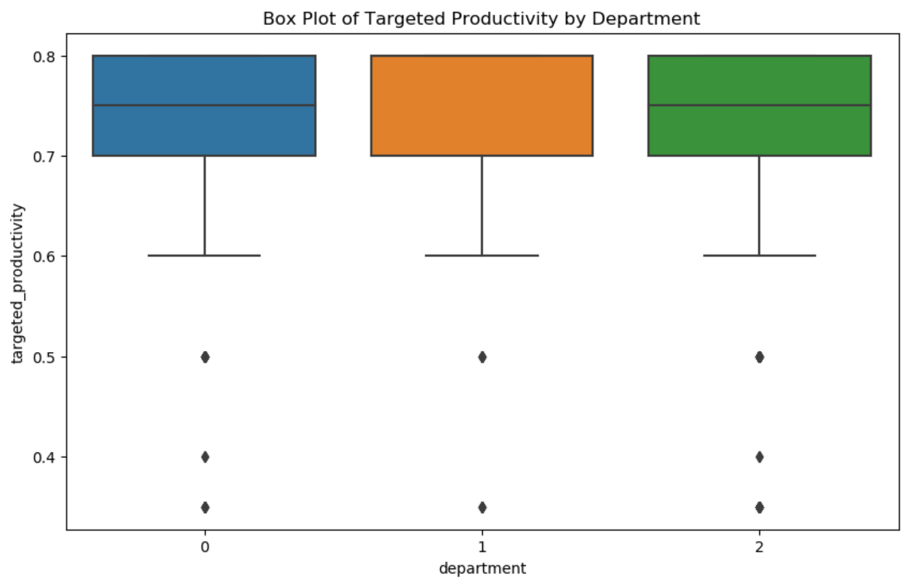
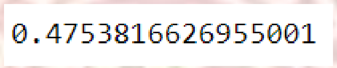
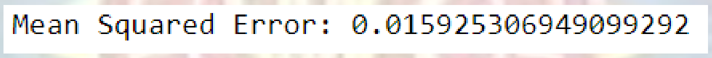


Figure 10.6: Displays the boxplot for productivity Figure 5



```
0.4753816626955001
```



```
Mean Squared Error: 0.015925306949099292
```

Figure 6: Performance metrices of ANN model.

Figure 10.7: The ANN model's predictions on actual data are displayed in this figure. It shows a comparison between the predicted values and the actual values in the dataset.

Figure 10.8: Similar to Figure 6, this figure presents the performance metrics, but for a Random Forest model. It allows for a comparison of the effectiveness of the ANN and Random Forest models.

Figure 10.9: The Random Forest model's predictions on actual data are displayed, showcasing the model's performance on real-world instances.

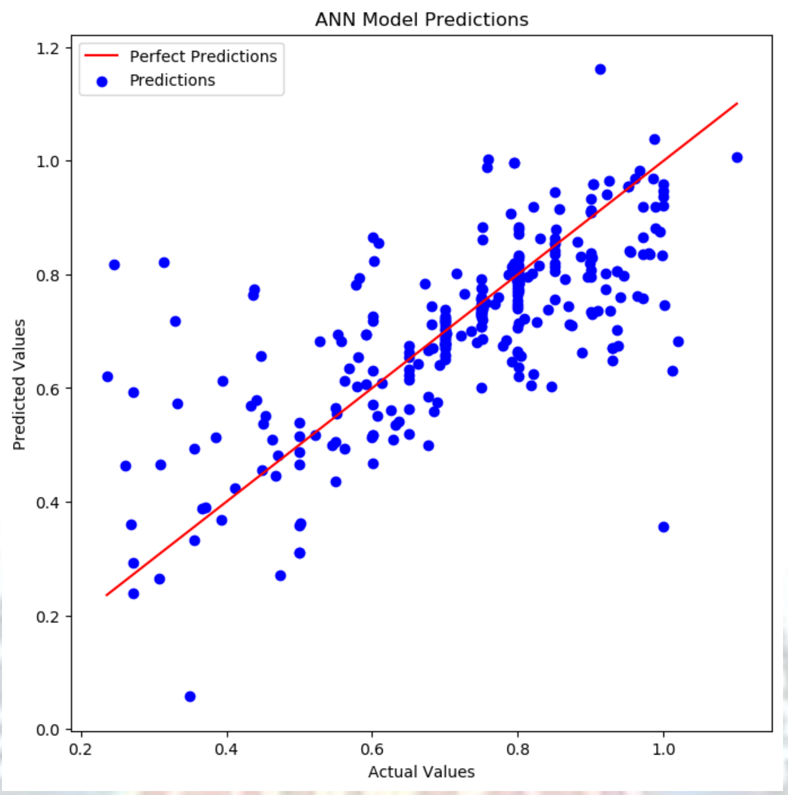
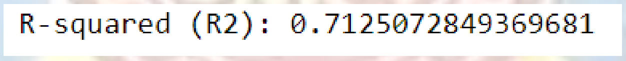
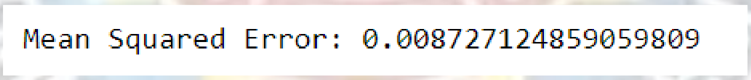


Figure 10.7: Displays the ANN model prediction on actual data.



```
R-squared (R2): 0.7125072849369681
```



```
Mean Squared Error: 0.008727124859059809
```

Figure 10.8: Performance metrices of Random Forest model.

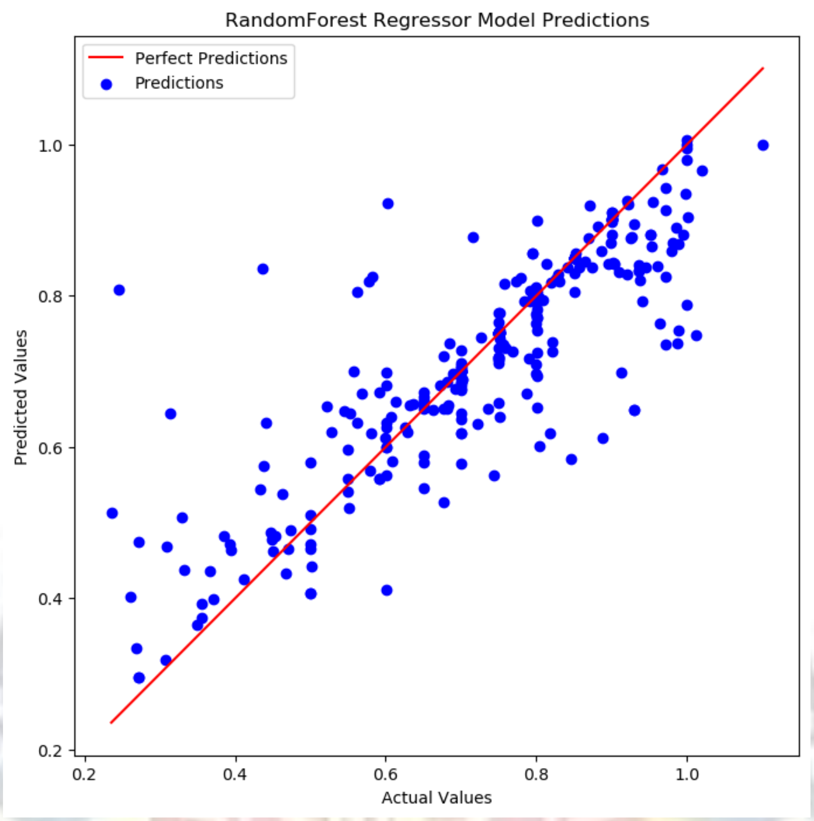


Figure 10.9: Displays the Random Forest model prediction on actual data.

CHAPTER 11
CONCLUSION AND FUTURE SCOPE

Conclusion:

In conclusion, the garment industry, a linchpin of the global economy, grapples with the intricate challenge of optimizing workforce productivity amidst a backdrop of diverse factors such as skill differentials, varied working conditions, and complex production processes. While historical strides have been made through traditional methods like time and motion studies, these approaches, albeit effective to a certain extent, may fall short of capturing the holistic complexity that influences productivity in this dynamic sector.

The crux of the matter lies in the imperative to develop a predictive model that transcends the limitations of traditional methodologies, accurately forecasting workforce productivity. This ambitious goal entails the systematic collection and analysis of a myriad of data points, encompassing worker performance, process efficiency, and environmental conditions. The essence of efficient workforce utilization becomes paramount, recognizing its critical role in fostering competitiveness and sustainability within the garment industry.

Predicting workforce productivity emerges as a strategic lever for managers, providing them with the insights needed to make informed decisions regarding resource allocation, training initiatives, and process optimization. The transformative potential lies in the adoption of a data-driven approach, utilizing advanced analytics and machine learning. This progressive project aspires to offer more precise predictions compared to traditional methods, thereby enhancing productivity optimization within the garment industry.

Through the meticulous collection and analysis of comprehensive datasets, this research endeavors to birth a system capable of autonomously and accurately predicting workforce productivity. The deployment of advanced algorithms holds the promise of unveiling intricate relationships between various factors and productivity outcomes, offering valuable insights for refining processes and optimizing resource allocation.

Future Scope

The future scope of predicting workforce productivity in the garment industry through a data-driven approach is expansive and holds the potential for transformative advancements. As the project aims to enhance productivity optimization using advanced analytics and machine learning techniques, several avenues for future exploration and refinement become evident.

- **Refinement of Predictive Models:**
  Continuous refinement of predictive models is essential to adapt to evolving industry dynamics. Incorporating more granular data points and refining algorithms can enhance the accuracy and reliability of productivity forecasts.
- **Integration of Real-Time Data:**
  Future iterations of the project could explore the integration of real-time data streams. This would allow for immediate adjustments and interventions based on current workforce conditions, further optimizing productivity on the fly.
- **Incorporation of Human-Centric Factors:**
  Delving deeper into the human-centric factors influencing productivity, such as employee motivation, satisfaction, and well-being, could provide a more comprehensive understanding. Integrating sentiment analysis or employee feedback into the predictive model could yield insights into the softer aspects of workforce dynamics.
- **Adaptation to Technological Advancements:**
  As technology continues to advance, exploring the integration of emerging technologies such as edge computing or the Internet of Things (IoT) can enhance data collection capabilities. This can lead to a more nuanced understanding of the production environment and workforce interactions.
- **Collaboration with Industry Stakeholders:**
  Collaborating with garment industry stakeholders, including manufacturers, workers, and technology providers, can contribute to a more holistic approach. Understanding industry-specific nuances and challenges can inform the refinement of the predictive model to align with practical needs.
- **Ethical Considerations and Bias Mitigation:**
  Future developments should emphasize ethical considerations in data usage, ensuring fairness and mitigating biases in predictive models. Implementing

transparent and accountable machine learning practices will be crucial for maintaining ethical standards.

- **Scaling Across Supply Chains:**

  Scaling the data-driven approach across entire supply chains within the garment industry can amplify its impact. Extending the system's capabilities to encompass various stages of production, from raw material sourcing to distribution, could optimize productivity on a broader scale.

- **Integration of Explainable AI:**

  To build trust and understanding, incorporating explainable AI techniques can provide insights into how the predictive model arrives at specific forecasts. This transparency is particularly important for industry stakeholders who may not have a background in machine learning.

- **Long-Term Sustainability Initiatives:**

  Consideration of long-term sustainability initiatives within the garment industry, such as environmentally friendly production practices and worker well-being, can be incorporated into the predictive model. This aligns with the broader global focus on sustainable and responsible business practices.

- **Global Collaboration and Benchmarking:**

  Encouraging global collaboration and benchmarking efforts can facilitate the sharing of best practices and standardized approaches to predicting workforce productivity. This collaborative approach can benefit the industry as a whole by fostering innovation and continuous improvement.

REFERENCES

[1] Al Imran, A., Amin, M. N., Rifat, M. R. I., Mehreen, S. (2019, April). Deep neural network approach for predicting the productivity of garment employees. In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 1402-1407). IEEE.

[2] Keya, M. S., Emon, M. U., Akter, H., Imran, M. A. M., Hassan, M. K., Mojumdar, M. U. (2021, January). Predicting performance analysis of garments women working status in Bangladesh using machine learning approaches. In 2021 6th International Conference on Inventive Computation Technologies (ICICT) (pp. 602-608). IEEE.

[3] Imran, A. A., Rahim, M. S., Ahmed, T. (2021). Mining the productivity data of the garment industry. International Journal of Business Intelligence and Data Mining, 19(3), 319-342.

[4] Alam, Mohammad, Rosima Alias, and Mohammad Azim. 2018. 'Social Compliance Factors (SCF) Affecting Employee Productivity (EP): An Empirical Study on RMG Industry in Bangladesh', 10: 87-96. https://www.researchgate.net/publication/326733299

[5] Bhatia, Karan, Shikhar Arora, and Ravi Tomar. 2016. "Diagnosis of diabetic retinopathy using machine learning classification algorithm." In 2016 2nd international conference on next generation computing technologies (NGCT), 347-51. IEEE DOI: 10.1109/NGCT.2016.7877439.

[6] Kruppa, Jochen, Alexandra Schwarz, Gerhard Arminger, and Andreas Ziegler. 2013. 'Consumer credit risk: Individual probability estimates using machine learning', Expert systems with applications, 40: 5125-31 DOI: https://doi.org/10.1016/j.eswa.2013.03.019. https://www.sciencedirect.com/science/article/pii/S0 957417413001693

[7] Balla, Imanuel, Sri Rahayu, and Jajang Jaya Purnama. 2021. 'GARMENT EMPLOYEE PRODUCTIVITY PREDICTION USING RANDOM FOREST', Jurnal Techno Nusa Mandiri, 18: 49-54 DOI: https://doi.org/10.33480/techno.v18i1.2210

[8] Attygalle, Dilhari, and Geethanadee Abhayawardana. 2021. 'Employee Productivity Modelling on a Work from Home Scenario During the Covid-19

Pandemic: A Case Study Using Classification Trees', Journal of Business and Management Sciences, 9: 92-100 DOI: 10.12691/jbms-9-3-1

[9] Ďurica, Marek, Jaroslav Frnda, and Lucia Svabova. 2019. 'Decision tree-based model of business failure prediction for Polish companies', Oeconomia Copernicana, 10: 453-69 DOI: 10.24136/oc.2019.022

[10] Mahoto, Naeem, Rabia Iftikhar, Asadullah Shaikh, Yousef Asiri, Abdullah Alghamdi, and Khairan Rajab. 2021. 'An Intelligent Business Model for Product Price Prediction Using Machine Learning Approach', 30: 147-59 DOI: 10.32604/iasc.2021.018944

[11] Sorostinean, Radu, Arpad Gellert, and BogdanConstantin Pirvu. 2021. 'Assembly Assistance System with Decision Trees and Ensemble Learning', Sensors, 21: 3580 DOI: https://doi.org/10.3390/s21113580

[12] Saad, Hamza. 2020. 'Use Bagging Algorithm to Improve Prediction Accuracy for Evaluation of Worker Performances at a Production Company', arXiv preprint arXiv:2011.12343 DOI: 10.4172/2169- 0316.1000257

[13] El Hassani, Ibtissam, Choumicha El Mazgualdi, and Tawfik Masrour. 2019. 'Artificial intelligence and machine learning to predict and improve efficiency in manufacturing industry', arXiv e-prints: arXiv: 1901.02256

[14] De Lucia, Caterina, Pasquale Pazienza, and Mark Bartlett. 2020. 'Does good ESG lead to better financial performances by firms? Machine learning and logistic regression models of public enterprises in Europe', Sustainability, 12: 5317 DOI: https://doi.org/10.3390/su12135317

[15] Ihya, Rachida, Abdelwahed Namir, Sanaa El Filali, Mohammed Ait Daoud, and Fatima Zahra Guerss. 2019. "J48 algorithms of machine learning for 58 Informatica 46 (2022) 49 –58 R. Obiedat et al. predicting user's the acceptance of an E -orientation systems." In Proceedings of the 4th International Conference on Smart City Applications, 1 -8. DOI: 10.1145/3368756.3368995.