An

Industry Oriented Mini Project Report On

GUARDING AGAINST JOB SCAMS: HARNESSING DECISION TREE ANALYSIS

Submitted in partial fulfillment of the requirements for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

T. PRANITHA	**217Z1A05H4**
S. SAISRIDHARNAIDU	**217Z1A05G6**
T. MADHAVASWAMY	**217Z1A05H6**

Under the Guidance of

MR.G. SRAVAN KUMAR

Assistant Professor

SCHOOL OF ENGINEERING

Department of Computer Science and Engineering

NALLA NARASIMHA REDDY

EDUCATION SOCIETY'S GROUP OF INSTITUTIONS

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi, Chowdariguda (V) Korremula

'x' Roads, Via Narapally, Ghatkesar (Mandal) Medchal (Dist), Telangana-500088

2024-2025

Nalla Narasimha Reddy Group of Institutions NNRG Integrated Campus

NALLA NARASIMHA REDDY

Education Society's Group of Institutions - Integrated Campus

(UGC AUTONOMOUS INSTITUTION)

UGC AUTONOMOUS A+ NAAC Accredited by NBA NATIONAL BOARD of ACCREDITATION CSE ECE

EAMCET/ECET/ICET/PGECET Code NNRG

SCHOOL OF ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report titled **"GUARDING AGAINST JOB SCAMS: HARNESSING DECISION TREE ANALYSIS"** is being submitted by **T. Pranitha(217Z1A05H4), S.SaiSridharNaidu(217Z1A05G6), T. Madhava Swamy(217Z1A05H6)** in Partial fulfilment for the award of **Bachelor of technology in Computer Science & Engineering** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide	**Head of the Department**
(Mr. G. Sravan Kumar)	(Dr. K. Rameshwaraiah)

Submitted for Viva Voce Examination held on…………………………….

External Examiner

DECLARATION

We T. Pranitha ,S. Sai Sridhar Naidu ,T. Madhava Swamy the students of **Bachelor of Technology in Computer Science and Engineering, Nalla Narasimha Reddy Education Society's Group Of Institutions**, Hyderabad, Telangana, hereby declare that the work presented in this project work entitled **GUARDING AGAINST JOB SCAMS: HARNESSING DECISION TREE ANALYSIS** 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. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

T.PRANITHA 217Z1A05H4 S.SAI SRIDHAR NAIDU 217Z1A05G6 T. MADHAVA SWAMY 217Z1A05H6

Date:

Signature:

ACKNOWLEDGEMENT

We express our sincere gratitude to our guide **Mr. G. Sravan Kumar**, Assistant Professor, in Computer Science and Engineering Department, NNRESGI, who motivated throughout the period of the project and also for his valuable and intellectual suggestions apart from his adequate guidance, constant encouragement right throughout our work.

We wish to record our deep sense of gratitude to our Project In-charge **Mrs. Ch Ramya**, Assistant Professor, in Computer Science and Engineering Department, NNRESGI, for giving her insight, advice provided during the review sessions and providing constant monitoring from time to time in completing our project and for giving us this opportunity to present the project work.

We profoundly express our sincere thanks to **Dr. K. Rameshwaraiah**, Professor & Head, Department of Computer Science and Engineering, NNRESGI, for his cooperation and encouragement in completing the project successfully.

We wish to express our sincere thanks to **Dr. G. Janardhana Raju**, Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

We wish to express our sincere thanks to **Dr. C. V. Krishna Reddy**, Director NNRESGI for providing the facilities for completion of the project.

Finally, we would like to thank Project Co-Ordinator, Project Review Committee (PRC) members, all the faculty members and supporting staff, Department of Computer Science and Engineering, NNRESGI for extending their help in all circumstances.

By:

T. PRANITHA 217Z1A05H4 S.SAI SRIDHAR NAIDU 217Z1A05G6 T. MADHAVA SWAMY 217Z1A05H6

ABSTRACT

The rise of online job portals has made job hunting easier but has also led to an increase in fake job postings. These scams can trick job seekers into losing money or sharing personal. This study aims to create a system that can predict and identify fake job postings using data mining techniques. We analyzed a large number of job advertisements using various data mining methods to find patterns that indicate whether a job posting is real or fake. We used text mining to extract important features from the job descriptions, like common words, sentence structures, and writing styles. Natural language processing (NLP) helped us understand the context and meaning behind the text. We then applied machine learning algorithms, such as decision trees, naïve bayes, and neural networks, to classify the job postings as real or fake based on these features. We found that specific words, unusual job titles, and how contact information is presented are key indicators of fake job postings. Our study highlights the effectiveness of data mining in protecting job seekers from online scams. The model we developed can be used by job portals to automatically detect and flag suspicious job postings, making job hunting safer for everyone. Future work will focus on using more data, improving the way we extract features, and testing advanced methods to make our predictions even more accurate. This research helps in the fight against online fraud in the job market, providing a practical solution to protect job seekers from scams.

Keywords : Natural Language Processing(NLP), Decision tree, Fraud detection, Job Portals, Text Mining.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 MOTIVATION

This project on predicting fake job posts is crucial because it addresses a significant issue in the job market: fraudulent job listings that deceive and exploit job seekers. By developing a model to detect these fake postings, you're not only enhancing the safety and trustworthiness of job boards but also protecting individuals from potential scams and financial loss. This work has the potential to create a more secure and reliable job search experience, making a tangible, positive impact on both job seekers and employers.

1.2 PROBLEM STATEMENT

The project's problem statement revolves around the pervasive issue of fake job postings on online platforms, posing significant challenges for both job seekers and organizations. The primary challenge lies in distinguishing genuine job opportunities from deceptive postings, which can lead to financial losses, identity theft, and wasted time for job seekers. The project aims to address this problem by developing and evaluating data mining techniques capable of accurately identifying fake job posts based on textual content, job descriptions, company details, and other relevant information. By assessing and comparing the effectiveness of various data mining techniques such as decision trees, support vector machines, and neural networks, the project seeks to improve prediction accuracy and contribute to the advancement of solutions in fake job post detection. Ultimately, the goal is to mitigate the negative impact on job seekers, enhance trust between organizations and job seekers, and provide actionable insights.

1.3 PURPOSE

The purpose of conducting a comparative study on fake job post prediction using different data mining techniques is to address the pressing issue of fraudulent job postings that adversely affect job seekers and organizations. By evaluating and comparing various data mining methods such as decision trees, support vector machines, and neural networks, the project aims to determine which approach is most effective in accurately identifying fake job posts. This comparative analysis will not only shed light on the performance metrics of each technique, including accuracy, precision, and recall, but also provide insights into their strengths and limitations. The ultimate goal is to contribute actionable recommendations for improving job post screening processes, thereby safeguarding job seekers from falling victim to deceptive practices. Additionally, the project seeks to advance academic knowledge in the field of fake job post detection.

1.4 SCOPE

This project aims to use the EMSCAD dataset to develop and compare decision trees and naïve bayes, for detecting fake job posts. It involves data preprocessing, model training, and performance evaluation. The goal is to identify the most effective technique for improving job post screening and protecting job seekers.

1.5 PROJECT OBJECTIVE

The objectives of this project are to develop and compare various machine learning models, including decision trees and naïve bayes, for detecting fake job posts using the EMSCAD dataset. The project will assess each model's performance in terms of accuracy and reliability. Ultimately, it aims to provide recommendations to improve job post screening processes and better safeguard job seekers from fraudulent.

2. LITERATURE SURVEY

2.1 INTRODUCTION

The project aims to conduct a comparative study on the prediction of fake job posts using various data mining techniques. The primary objective is to assess the effectiveness of different methods in identifying deceptive job postings, which can have detrimental effects on job seekers and organizations. The study will begin with a comprehensive literature review to understand the current state of research in fake job post detection and data mining approaches. Identified gaps in the literature will guide the research objectives, which include evaluating the performance of selected data mining techniques such as decision trees, support vector machines, and neural networks. The methodology involves acquiring a suitable dataset containing job postings, performing data preprocessing steps like cleaning and feature extraction, and dividing the data into training, validation, and testing sets.

The project will feature several key components focused on conducting a comparative study on fake job post prediction using different data mining techniques. Firstly, a robust dataset comprising a diverse range of genuine and fake job postings will be meticulously selected to ensure representativeness and relevance. Subsequently, thorough data preprocessing steps, including cleaning, normalization, and feature extraction using text mining techniques, will be applied to prepare the dataset for analysis. Feature engineering will be a crucial aspect, where advanced techniques will be explored to enhance the predictive capabilities of the data mining models. This may involve creating new features based on keyword frequency, sentiment analysis of job descriptions, or semantic similarity measures. Model selection will be another significant feature, with various data mining techniques such as decision trees (e.g., Random Forest), support vector machines (SVM), k-nearest neighbors (KNN), and neural networks (e.g., deep learning models like LSTM or CNN) being evaluated and compared. Each model's complexity, interpretability, and performance in predicting fake job posts will be considered during selection.

The system analysis for "A Comparative Study on Fake Job Post Prediction Using Different Data Mining Techniques" involves a comprehensive assessment of the project's requirements, functionalities, and components. The initial phase focuses on gathering requirements, understanding stakeholder needs, and defining project objectives such as acquiring a suitable dataset, selecting data mining techniques, and establishing evaluation metrics. The system's functionalities encompass several critical tasks, including data preprocessing (e.g., cleaning, normalization, feature extraction), model selection and training (e.g., decision trees, SVM, neural networks),hyperparameter tuning, performance evaluation using metrics like accuracy and recall ,visualization of results, and generation of a detailed project report summarizing findings and recommendations. Key components of the system include a Data Collection and Preprocessing Component responsible for acquiring and cleaning the dataset, a Model Selection and Training Component for selecting and training data mining models.

2.2 EXISTING SYSTEM

The current system for predicting fake job posts using data mining techniques relies heavily on manual screening processes and simplistic keyword-based filters. Job platforms and organizations typically employ rule-based systems that flag job postings containing specific keywords associated with fraudulent activities or scams. However, this approach has several drawbacks. Firstly, keyword-based filters may overlook subtle variations or new tactics used by scammers, leading to both false positives and false negatives. As the volume of job postings continues to grow, scalability becomes a significant challenge for manual review processes, resulting in delays in identifying and removing fake job posts and leaving job seekers vulnerable to scams. Furthermore, the existing system lacks proactive fraud prevention measures and comprehensive analysis capabilities such as predictive modeling and machine learning algorithms, which could provide deeper insights into the characteristics and patterns of fake job postings. Furthermore, the existing system lacks proactive fraud prevention measures and comprehensive analysis capabilities such as predictive modeling and machine learning algorithms, Therefore, there is a clear need for a more sophisticated and automated approach utilizing advanced data mining techniques to enhance fake job post prediction and prevention in a more efficient and effective manner.

LIMITATIONS

- The system is implemented by Conventional Machine Learning.
- The system doesn’t implement for analyzing large data set. ○ Scalability issues.
- Limited Detection.
- Lack of advanced techniques.

2.3 PROPOSED SYSTEM

The proposed system for "A Comparative Study on Fake Job Post Prediction Using Different Data Mining Techniques" encompasses several key enhancements to address the limitations of the existing approach. Firstly, the system will prioritize acquiring a comprehensive and diverse dataset with a balanced representation of genuine and fake job postings across various industries and regions, ensuring data quality and relevance. Advanced data preprocessing techniques, including natural language processing (NLP) methods, will be employed to clean, normalize, and extract features from textual job postings effectively. In terms of model selection, the proposed system will carefully choose a range of data mining techniques, such as decision trees, ensemble methods like Random Forest, naïve bayes including deep learning models, and anomaly detection algorithms.

ADVANTAGES

➢ The proposed has been implemented EMSCAD technique which is very accurate and fast.

➢ The system is very effective due to accurate detection of Fake job posts which creates inconsistency for the job seeker to find their preferable jobs causing a huge waste of their time.

3. SYSTEM ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

- **Data Collection and Preprocessing**: The system must be able to load, clean, and preprocess the EMSCAD dataset, including handling missing values and text vectorization.
- **Model Training**: The system should support training of various machine learning models, such as decision trees, support vector machines, and neural networks, using the pre processed data.
- **Model Evaluation**: The system must evaluate model performance using metrics like accuracy, precision, recall, and F1-score.
- **Comparison of Results**: The system should compare the performance of different models and provide a summary of which model performs best.
- **Recommendation Generation**: Based on model performance, the system must generate actionable recommendations for improving job post screening processes.
- **Reporting**: The system must generate a report summarizing the methodology, results, and recommendations.

3.2 NON-FUNCTIONAL REQUIREMENTS

- **Performance**: The system should process and analyse data efficiently, with acceptable training times for each model.
- **Scalability**: The system should be able to handle large datasets and adapt to future expansions or additional features.
- **Usability**: The system should be user-friendly, with a clear interface for inputting data, training models, and viewing results.
- **Accuracy**: The models should provide reliable predictions with high accuracy to effectively detect fake job posts.
- **Security**: The system should ensure that data is handled securely and protect sensitive information.
- **Maintainability**: The system should be designed for easy updates and maintenance, allowing for adjustments to models and processes as needed.

3.3 INTERFACE REQUIREMENTS User Requirements

- Data Input
- Model Selection
- Training and Testing
- Performance Evaluation
- Comparison and Visualization
- Recommendation Report
- User Interface
- Documentation and Support

System Requirements:

Hardware Requirements:

Processor	: Pentium –IV.
Hard Disk	: 20 GB.
Ram	: 4 GB.

Software Requirements:

Operating System	: Windows 7
Coding Language	: Python

Performance Requirements

The performance requirements for the project include efficient processing and analysis of the dataset, with model training and testing completing in a reasonable time. Models should achieve high accuracy in detecting fake job posts while handling increasing data volumes effectively. The system must optimize resource usage, ensure quick response times for user interactions, and support simultaneous processing of multiple tasks without performance issues.

4.SYSTEM DESIGN

4.1 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objectoriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts. Be independent of particular programming languages and development process. Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.

Data Flow Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

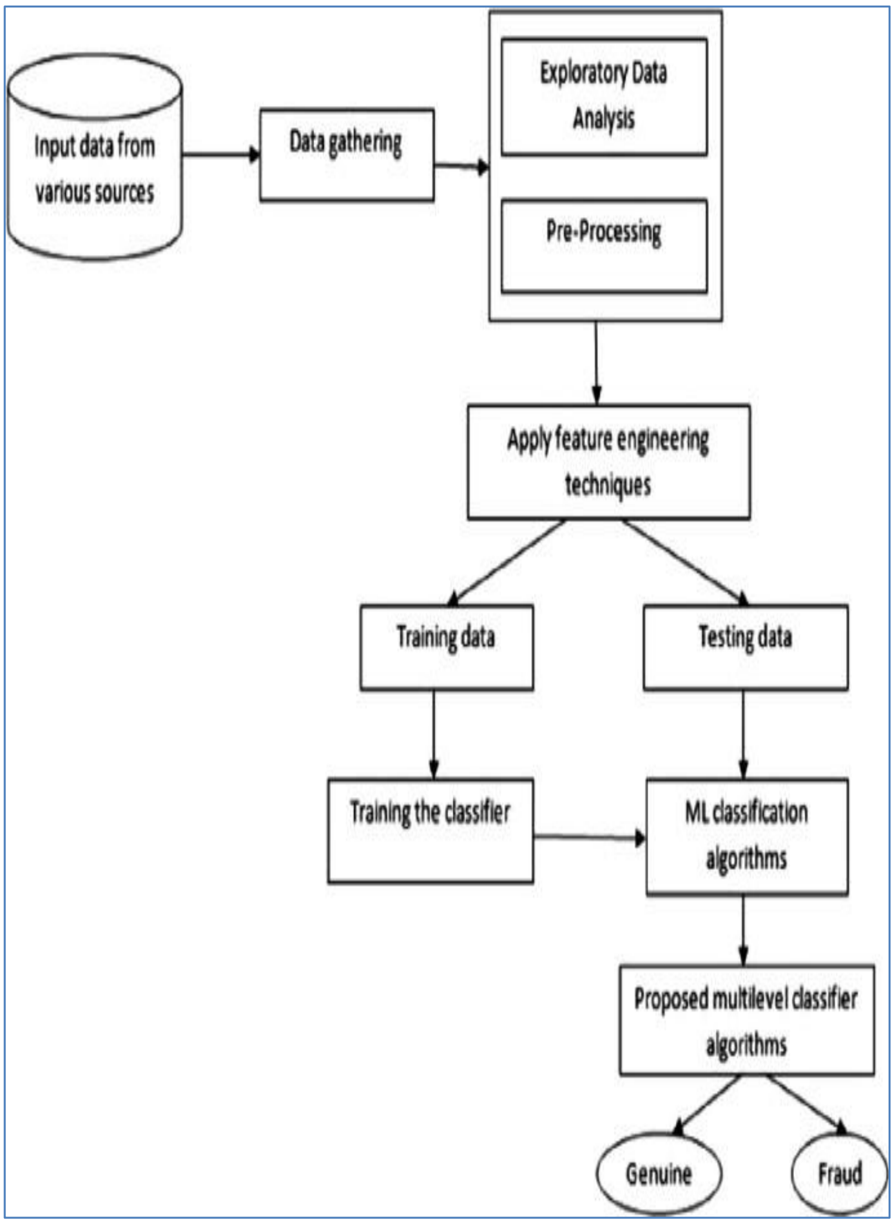

Input data from various sources
Data gathering
Exploratory Data Analysis
Pre-Processing
Apply feature engineering techniques
Training data
Testing data
Training the classifier
ML classification algorithms
Proposed multilevel classifier algorithms
Genuine
Fraud

Fig :4.1 Data Flow Diagram

Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

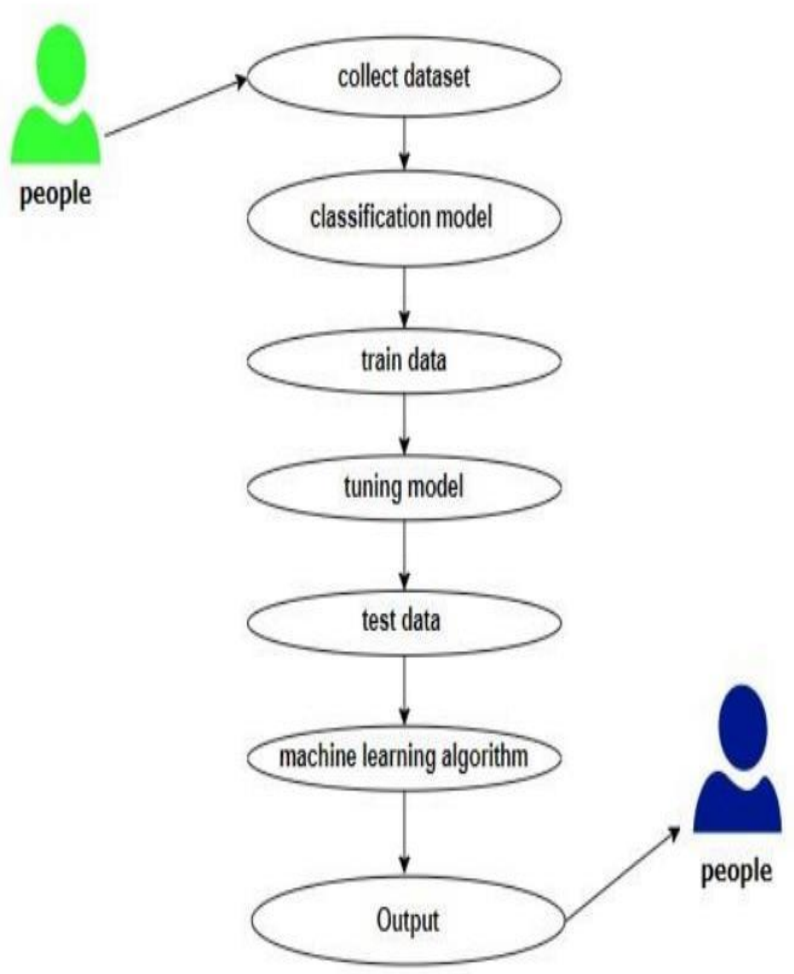


Fig :4.2 Use case Diagram

Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

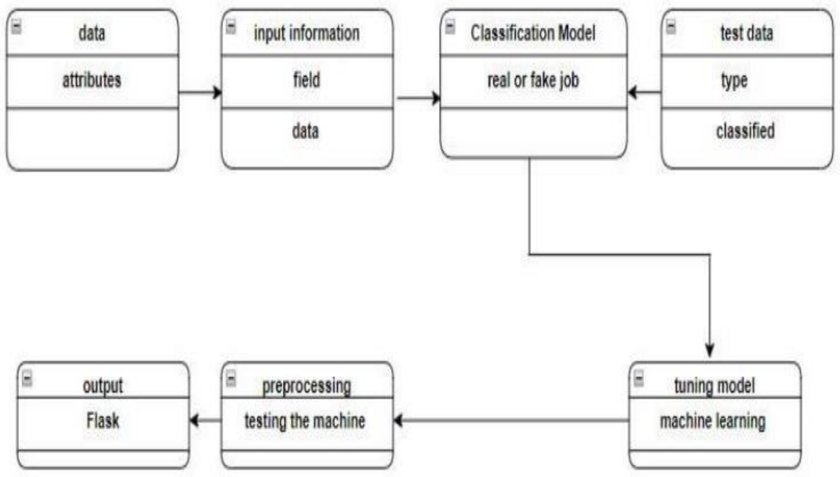


Fig :4.3 Class Diagram

Sequence Diagram:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

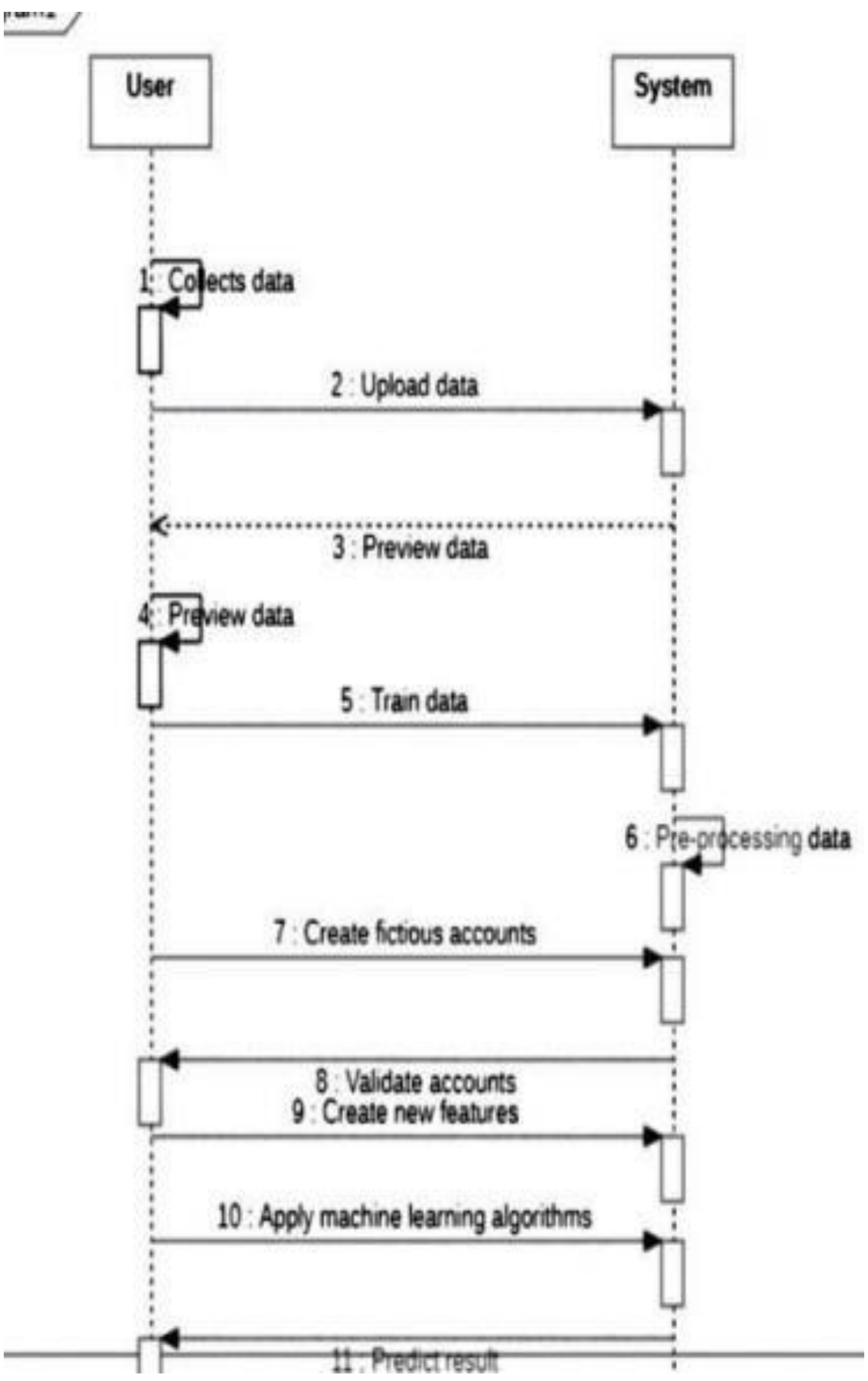


Fig :4.4 Sequence Diagram

Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

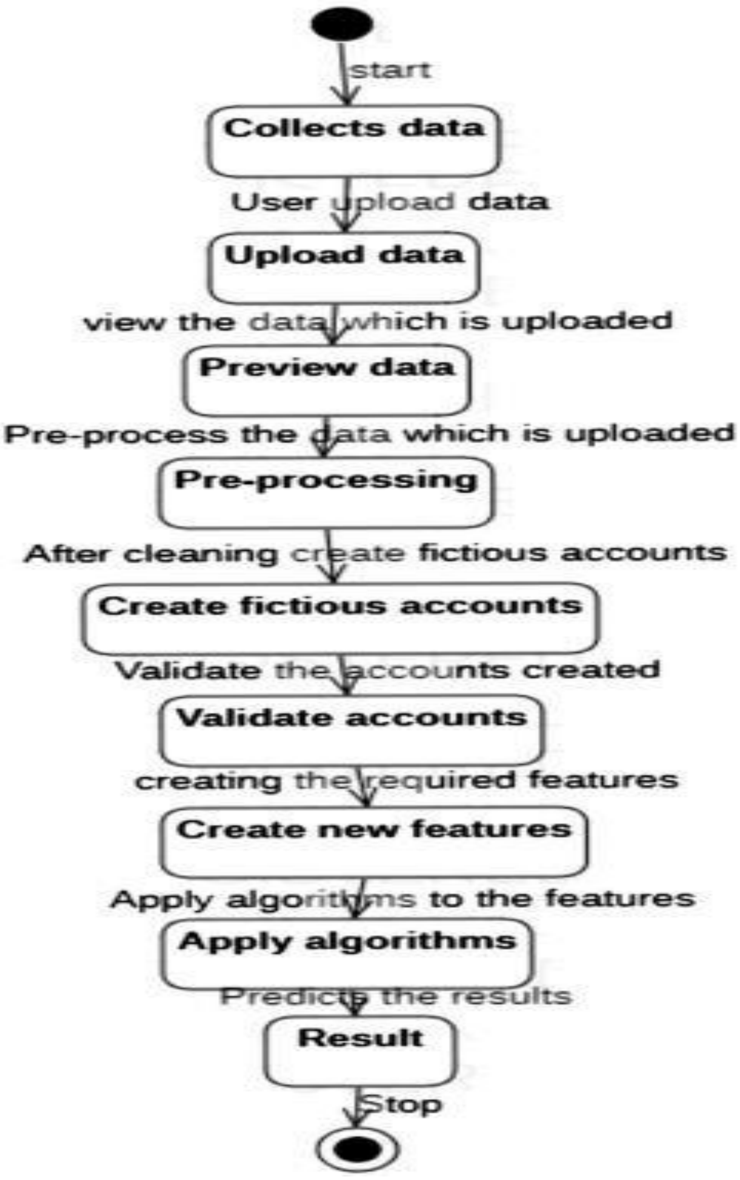

start
Collects data
User upload data
Upload data
view the data which is uploaded
Preview data
Pre-process the data which is uploaded
Pre-processing
After cleaning create fictious accounts
Create fictious accounts
Validate the accounts created
Validate accounts
creating the required features
Create new features
Apply algorithms to the features
Apply algorithms
Predicts the results
Result
Stop

Fig :4.5 Activity Diagram

4.2 MODULES:

- **Data Preprocessing Module**: Handles loading, cleaning, and preparing the EMSCAD dataset, including text vectorization and handling missing values.
- **Model Training Module**: Implements and trains various machine learning models such as decision trees, support vector machines, and neural networks on the pre processed data.

- **Model Evaluation Module**: Assesses the performance of each trained model using metrics like accuracy, precision, recall, and F1-score.
- **Comparison and Analysis Module**: Compares the performance of different models and provides visualizations and summaries of their effectiveness.
- **Recommendation Generation Module**: Generates actionable recommendations based on the analysis to improve job post screening processes.
- **User Interface Module**: Provides a user-friendly interface for interacting with the system, including data input, model selection, and result visualization.
- **Reporting Module**: Creates comprehensive reports summarizing the methodology, results, and recommendations from the analysis.

5. IMPLEMENTATION AND RESULTS

5.1 METHOD OF IMPLEMENTATION

What is 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.

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 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.

What is Machine Learning:

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Modules Used in Project:- 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 data-types 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.

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 Jupyter 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.

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**

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.

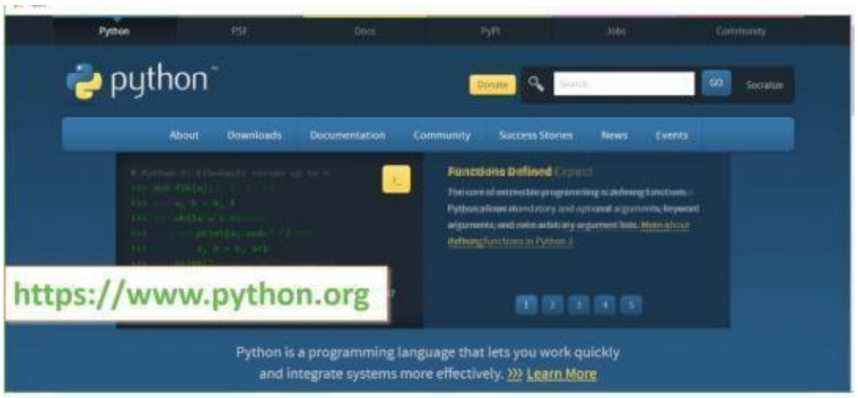
Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

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

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**

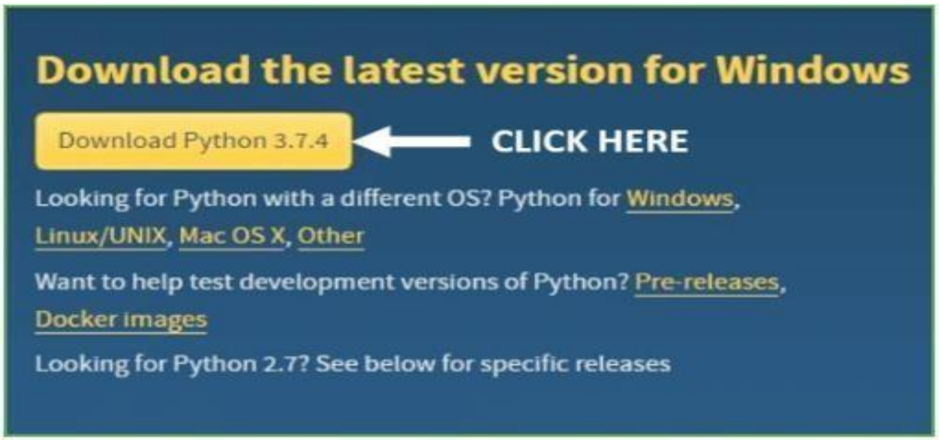


Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Fig 5.1: Download Python

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

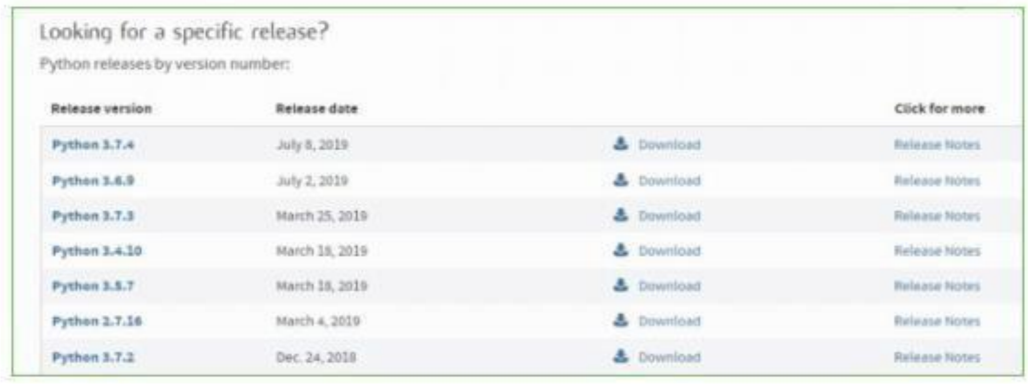
Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Fig 5.2: Python 3.7.4

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color 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?

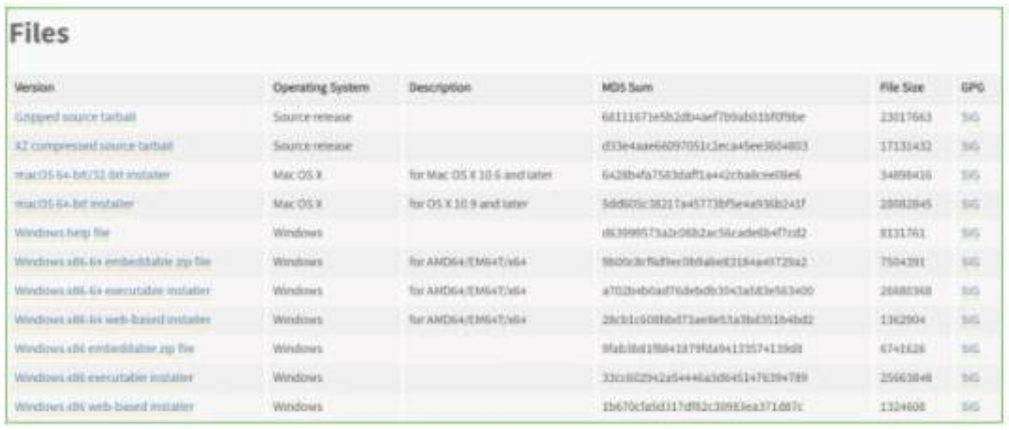
Python releases by version number:

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

Fig 5.3: Specific Release

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		d63999573a2c06b2ac56cade6b4f7cd2	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

Fig 5.4: Files

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.

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.

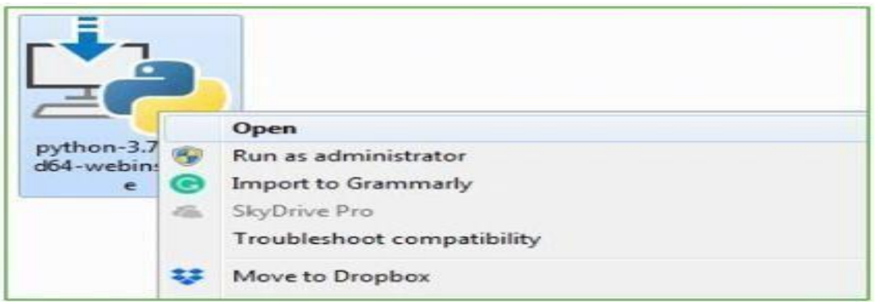
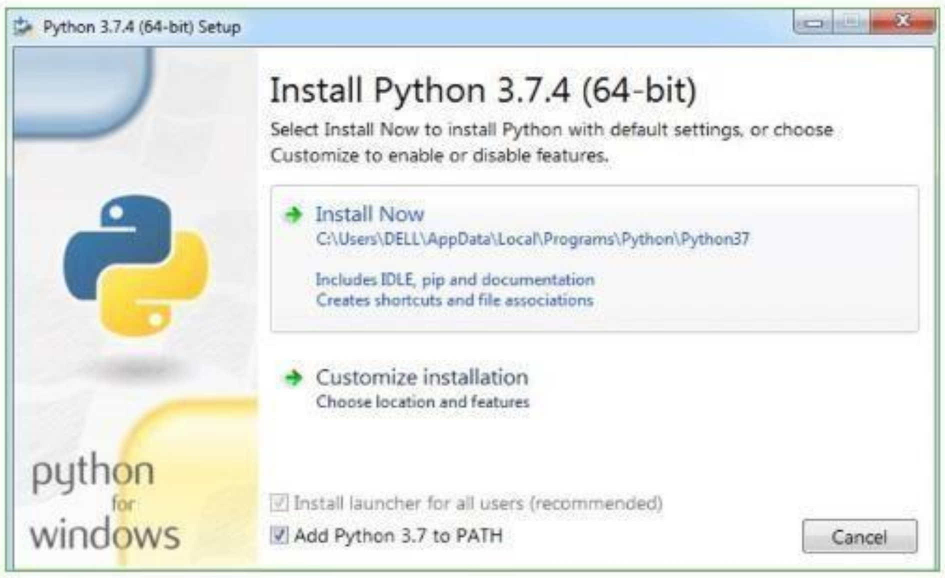


Fig 5.5: Open Python 3.7.4

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

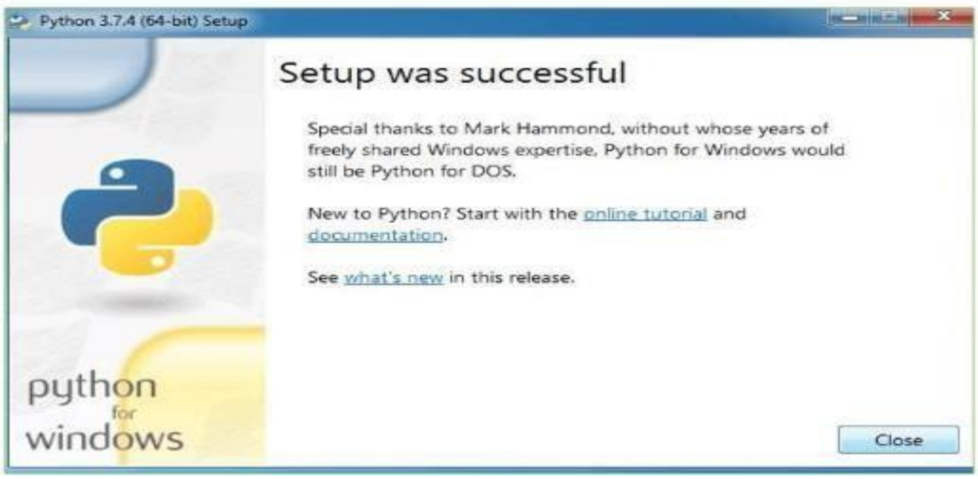
Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Cancel

Fig 5.6: Install Python 3.7.4

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.

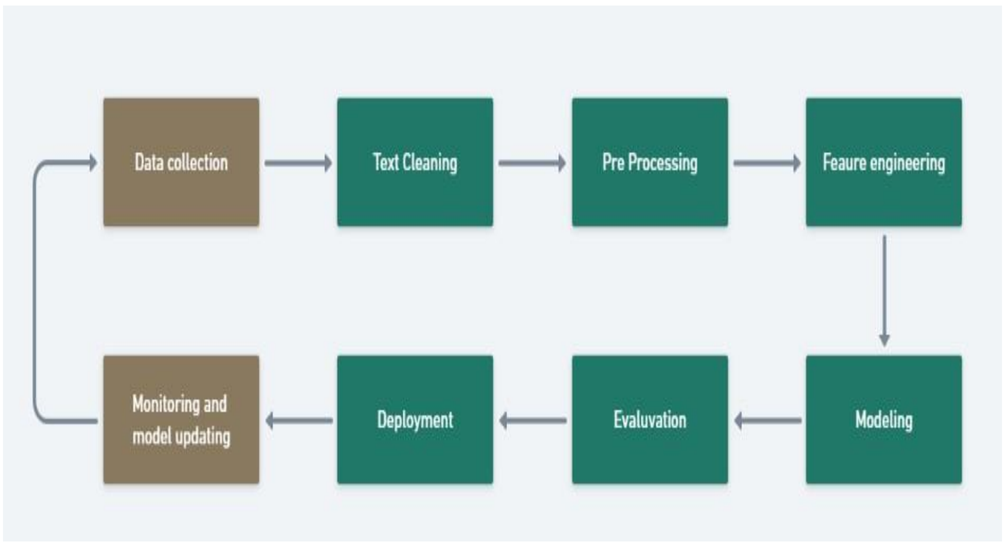
Fig 5.7: Installed Successfully

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.

5.2 EXPLAINATION OF KEY FUNCTION:

In Natural Language Processing (NLP), an NLP pipeline is a sequence of interconnected steps that systematically transform raw text data into a desired output suitable for further analysis or application.



Data Collection:

- **Job Post Features:** Typically includes job title, job description, company name, location, salary, and other relevant attributes.
- Labels: Binary classification indicating whether a job post is fake or genuine.
- **EMSCAD Dataset:** If you're using the EMSCAD dataset specifically, you should follow these steps:
- **Access:** Ensure you have the proper access to the EMSCAD dataset. This might involve requesting access from the dataset provider or downloading it from a data repository.
- **Format:** Verify the format of the dataset (CSV, JSON, Excel, etc.) and check the documentation to understand the structure and features.

Data Analysis

Handling Missing Values

- Identify any missing values in the dataset.
- Discuss strategies for handling them (e.g., imputation, removal).

Data Visualization

- **Distribution of Job Posts**: Plot the distribution of job types (fake vs. real).
- **Text Length Analysis**: Analyse and visualize the length of job descriptions for both classes.
- **Feature Correlations**: Use heatmaps to visualize correlations between numerical features.

Data Preprocessing:

- **Loading and Cleaning Data**: Read data, inspect, and clean it by removing missing values.
- **Categorical Encoding**: Convert categorical data into numerical format.
- **Text Processing**:

o Convert text to lowercase. o Remove noise like URLs, HTML
tags, and punctuation. o Remove stop words and perform stemming

- **Feature Extraction**: Convert text into numerical features using
- tokenization and Vectorization

Feature Engineering

- Discuss any new features created (e.g., text length, presence of specific keywords).
- Convert categorical variables into numerical format using encoding techniques (e.g., one-hot encoding).

Model Evaluation

- **Metrics**: Use accuracy, precision, recall, F1-score, and ROC-AUC to evaluate model performance.
- **Cross-Validation**: Implement k-fold cross-validation to ensure robustness.
- **Hyperparameter Tuning**: Use techniques like Grid Search or Random Search to optimize model parameters.

Naïve Bayes Algorithm:

The Naive Bayes algorithm is a simple yet powerful classification technique based on Bayes' theorem, often used for text classification tasks like spam detection and, in your case, fake job post prediction**.**

- Fast and efficient, especially with large datasets.
- Works well with high-dimensional data, typical in text classification.
- Simple to implement and interpret.

Decision Tree:

A decision tree is a flowchart-like structure where each internal node represents a feature (attribute), each branch represents a decision rule, and each leaf node represents an outcome (class label).

- Easy to interpret and visualize.
- Handles both numerical and categorical data well.
- Requires little data preprocessing (no need for scaling).

SOURCE CODE:

```
#Importing Libraries import
numpy as np import pandas
as pd import
matplotlib.pyplot as plt
import seaborn as sns import
string # Reading Dataset
# Dataset is from https://www.kaggle.com/amruthjithrajvr/recruitment-
scam data=pd.read_csv("./fake_job_postings.csv") # Reading top 5 rows
of our dataset data.head()
# To check the number of rows and
column data.shape data.columns
# let us check the missing values in our dataset
data.isnull().sum()
# Let us remove the columns which are not necessary
# axis =1 specifies that the values are column value and inplace=true to make these
changes permanent (ie. make these dropes of columns permanent in the data set)
# We have droped salary range because 70% approx null value
# also job_id and other irrelvent columns because they does not have any logical
meaning data.drop(['job_id', 'salary_range', 'telecommuting', 'has_company_logo',
'has_questions'],axis=1,inplace =
True) data.shape data.head()

# Fill NaN values with blank space
# inplace=true to make this change in the dataset permanent
data.fillna(' ', inplace=True)
#Create independent and Dependent Features
columns = data.columns.tolist()
# Filter the columns to remove data we do not want
columns = [c for c in columns if c not in ["fraudulent"]] #
Store the variable we are predicting  target = "fraudulent"
# Define a random state  state =
np.random.RandomState(42)
X = data[columns]
```

```
Y = data["fraudulent"]
X_outliers = state.uniform(low=0, high=1, size=(X.shape[0], X.shape[1]))
# Print the shapes of X & Y print(X.shape) print(Y.shape)
from imblearn.under_sampling import
RandomUnderSampler   under_sampler =
RandomUnderSampler() X_res, y_res =
under_sampler.fit_resample(X, Y) df1 =
pd.DataFrame(X_res) df3 = pd.DataFrame(y_res)
# the default behaviour is join='outer'
# inner join result = pd.concat([df1, df3],
axis=1, join='inner') display(result)
data=result; data.isnull().sum()

# data cleaning done
# Checking for distribution of class label(percentages belonging to real class and
percentages belonging to fraud class)
 # in the data 1 indicates fraud post
 # 0 indicating real post
 # Plotting pie chart for the data
 # function of Explode function: how the portion will appear (to understand change
explode=(0,0.5))          labels          =          'Fake',          'Real'
sizes=[data.fraudulent[data['fraudulent']==1].count(),
data.fraudulent[data['fraudulent']==
0].count()]
explode = (0, 0.1)
fig1, ax1 =
plt.subplots(figsiz
e=(8, 6)) #size of
the pie chart
ax1.pie(sizes,
explode=explode,
labels=labels,
autopct='%1.2f%
%',
shadow=True,
startangle=120)
#autopct
%1.2f%% for 2
digit precision
ax1.axis('equal')
plt.title("Proportio
n of Fraudulent",
size = 7)
plt.show()

# creating a dictionary(key-value pair) with top 10
country country = dict(data.country.value_counts()[:11])
del country[' '] #deleting country with space values
plt.figure(figsize=(10,6)) plt.title('Country-wise Job
Posting', size=20) plt.bar(country.keys(),
country.values()) #(xaxis,yaxis) plt.ylabel('No. of jobs',
size=10) plt.xlabel('Countries', size=10) # visualizing
jobs based on experience experience =
dict(data.required_experience.value_counts()) del
experience[' '] plt.figure(figsize=(10,6))
plt.bar(experience.keys(), experience.values())
plt.title('No. of Jobs with Experience')
plt.xlabel('Experience', size=10) plt.ylabel('No. of jobs',
size=10) plt.xticks(rotation=35) plt.show()
# NLTK :: Natural Language Toolkit import nltk
nltk.download("stopwords") from nltk.corpus
import stopwords #loading the stopwords
stop_words = set(stopwords.words("english"))
#converting all the text to lower case data['text']
= data['text'].apply(lambda x:x.lower())
#removing the stop words from the corpus
```

```
data['text'] = data['text'].apply(lambda x:' '.join([word for word in x.split() if word not
in
(stop_words)])) from sklearn.model_selection
import train_test_split
# Splitting dataset in train and test
X_train, X_test, y_train, y_test = train_test_split(data.text, data.fraudulent,
test_size=0.3)
# what does X-train and y_train
contain print(y_train) print(X_train)


# Naive Bayes Classifier
# we are using Multinomial Naive Bayes approach because the data here is not
symmetrical.
# generally if there are data in the form of this long text,it is advisable to
# %time will give the time taken by the system for execution
nb = MultinomialNB()
%time nb.fit(X_train_dtm, y_train) from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix import seaborn as sn import
matplotlib.pyplot as plt
# Assuming y_test and y_pred_nb are already defined
# Print accuracy score print("Classification Accuracy:",
accuracy_score(y_test, y_pred_nb))
# Print classification report
print("Classification Report\n")
print(classification_report(y_test,
y_pred_nb))
# Print confusion matrix print("Confusion Matrix\n") print(confusion_matrix(y_test,
y_pred_nb)) # Generate and visualize the confusion matrix cm =
confusion_matrix(y_test, y_pred_nb) # Plot the confusion matrix using seaborn
heatmap plt.figure(figsize=(10, 7)) sn.heatmap(cm, annot=True, fmt='d',
cmap='Blues')  # fmt='d' ensures integer formatting plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

```
#instantiate a Decision Tree Classifier dt = DecisionTreeClassifier()
# Model Accuracy print("Classification Accuracy:",
accuracy_score(y_test, y_pred_class)) print("Classification
Report\n") print(classification_report(y_test, y_pred_class))
print("Confusion Matrix\n") print(confusion_matrix(y_test,
y_pred_class))
# Confusion Matrix import seaborn as sn
cm =
confusion_matrix(y_test,y_pred_class)
plt.figure(figsize = (10,7)) sn.heatmap(cm,
annot=True, fmt='d') plt.xlabel('Predicted')
plt.ylabel('Truth')
#     convert     text     to     feature     vectors
input_data_features = vect.transform(input_text)

# making prediction
prediction =
dt.predict(input_data_features)
print(prediction) if (prediction[0]==1):
print('Fraudulant Job') else:
  print('Real                              Job')
print(y_test.iloc[200])
```

5.3 OUTPUT SCREENS:

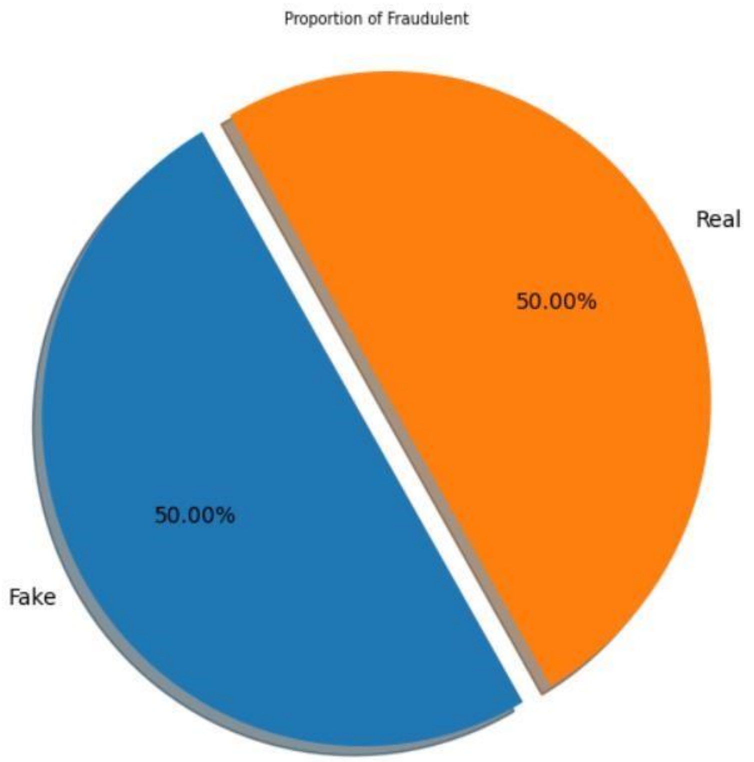


Fig 5.8 percentages belonging to real class and fraud class

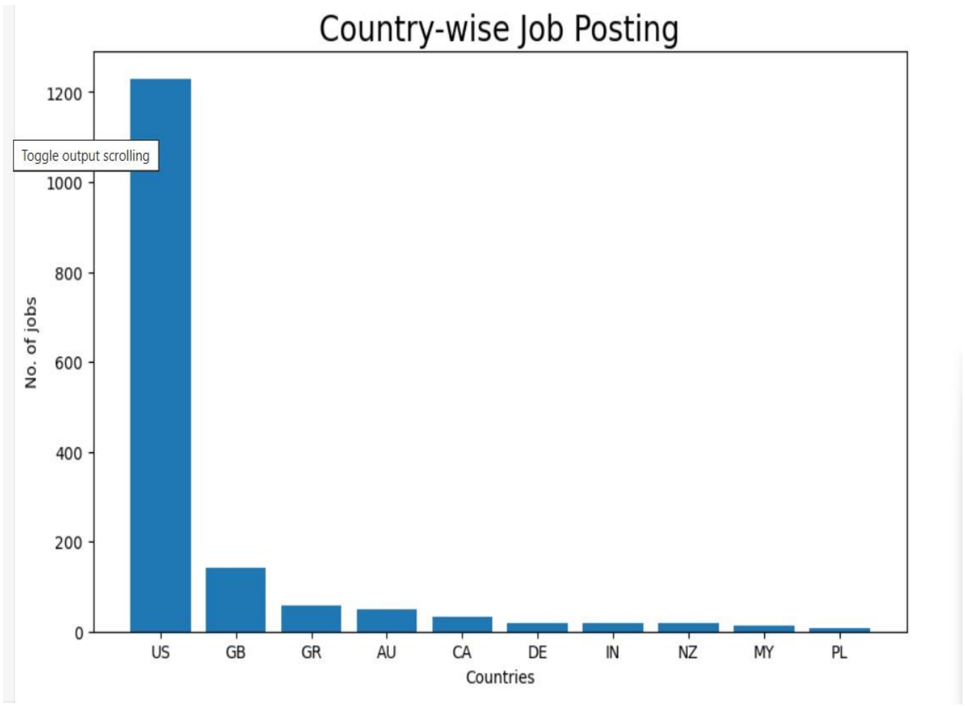


Fig 5.9 country-wise job postings

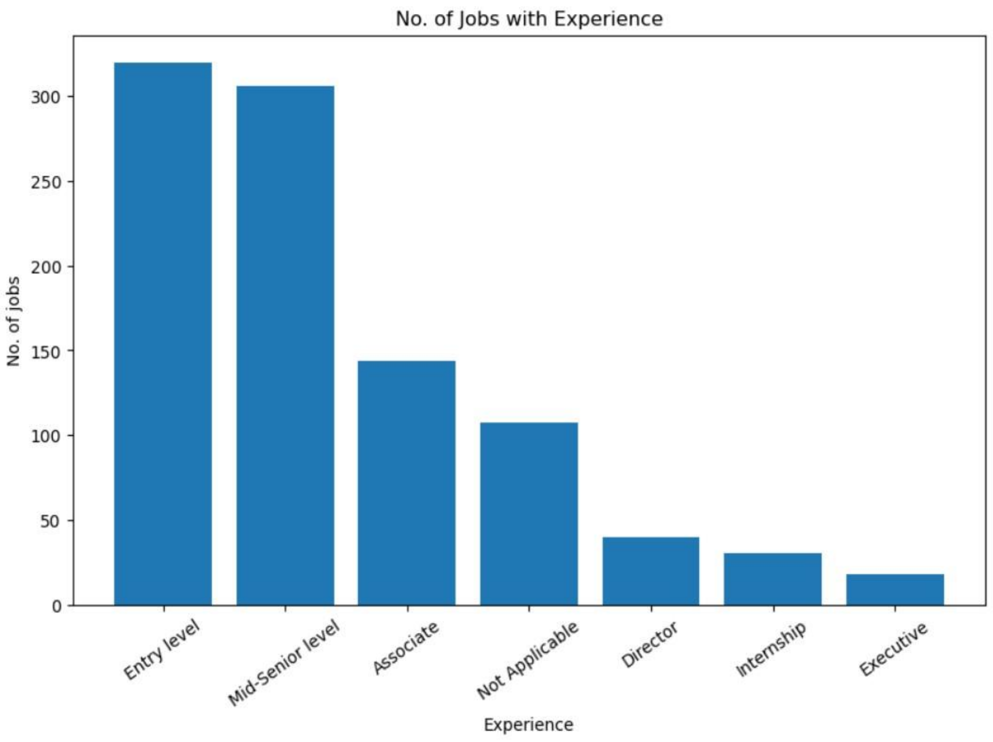


Fig 5.10 No. of jobs with Experience

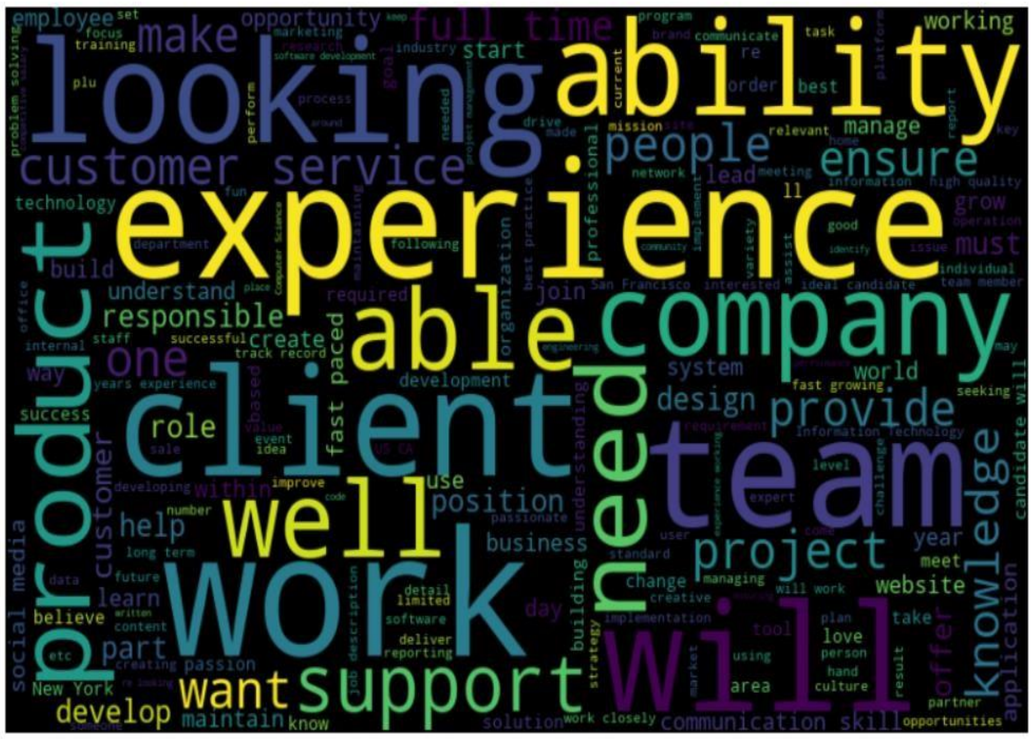


Fig 5.11 Common words in real job posting texts

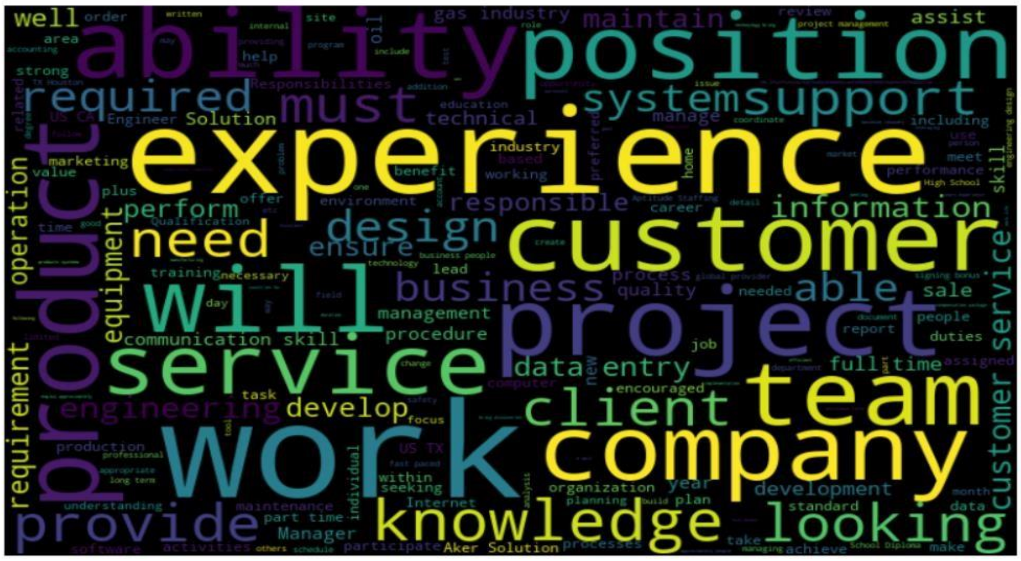
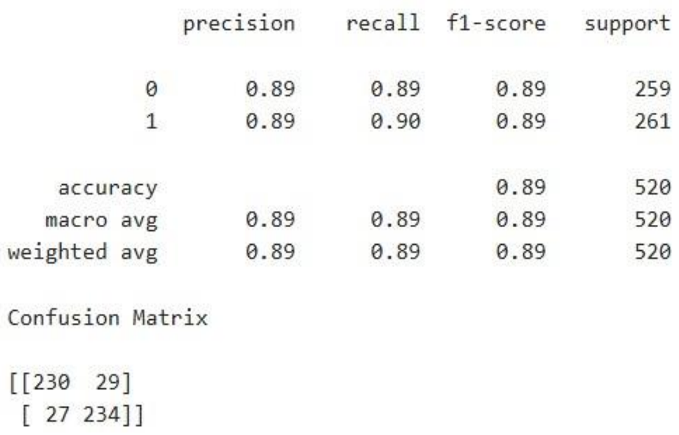


Fig 5.12 Common words in fraud job posting texts



```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89       259
           1       0.89      0.90      0.89       261

    accuracy                           0.89       520
   macro avg       0.89      0.89      0.89       520
weighted avg       0.89      0.89      0.89       520

Confusion Matrix

[[230  29]
 [ 27 234]]
```

Fig 5.13 Classification Report for naïve bayes

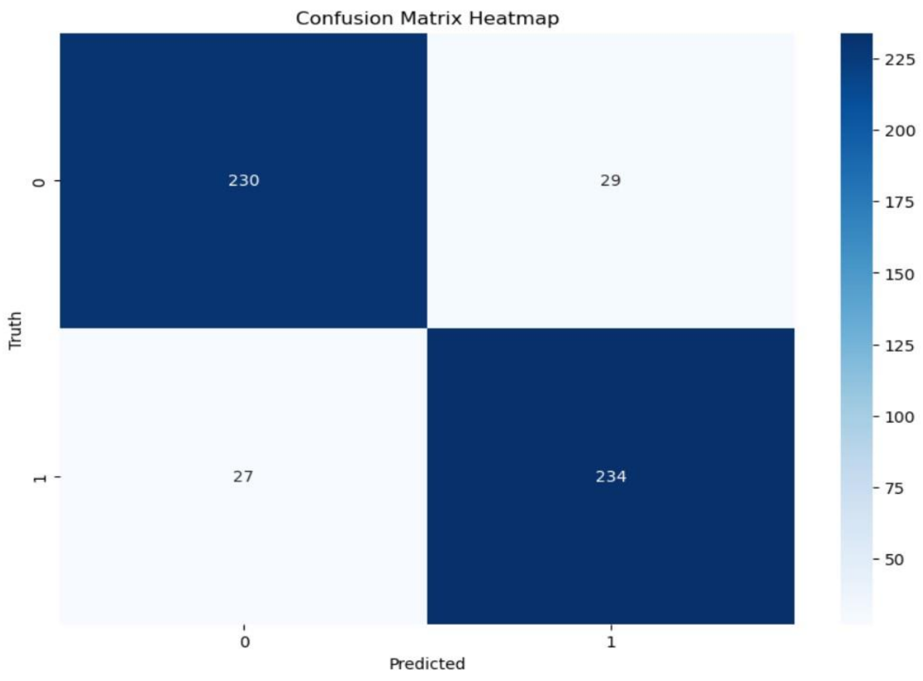
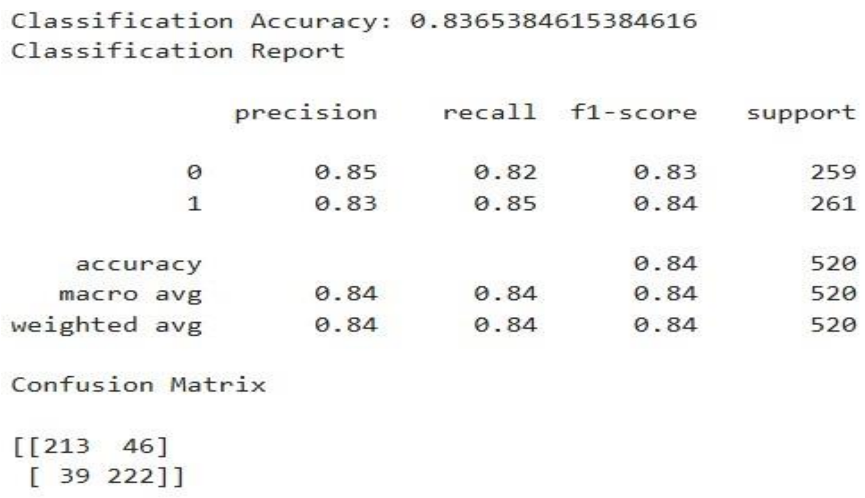


Fig 5.14 Naïve Bayes prediction



```
Classification Accuracy: 0.8365384615384616
Classification Report

              precision    recall  f1-score   support

           0       0.85      0.82      0.83       259
           1       0.83      0.85      0.84       261

    accuracy                           0.84       520
   macro avg       0.84      0.84      0.84       520
weighted avg       0.84      0.84      0.84       520

Confusion Matrix

[[213  46]
 [ 39 222]]
```

Fig 5.15 Classification Report of Decision tree

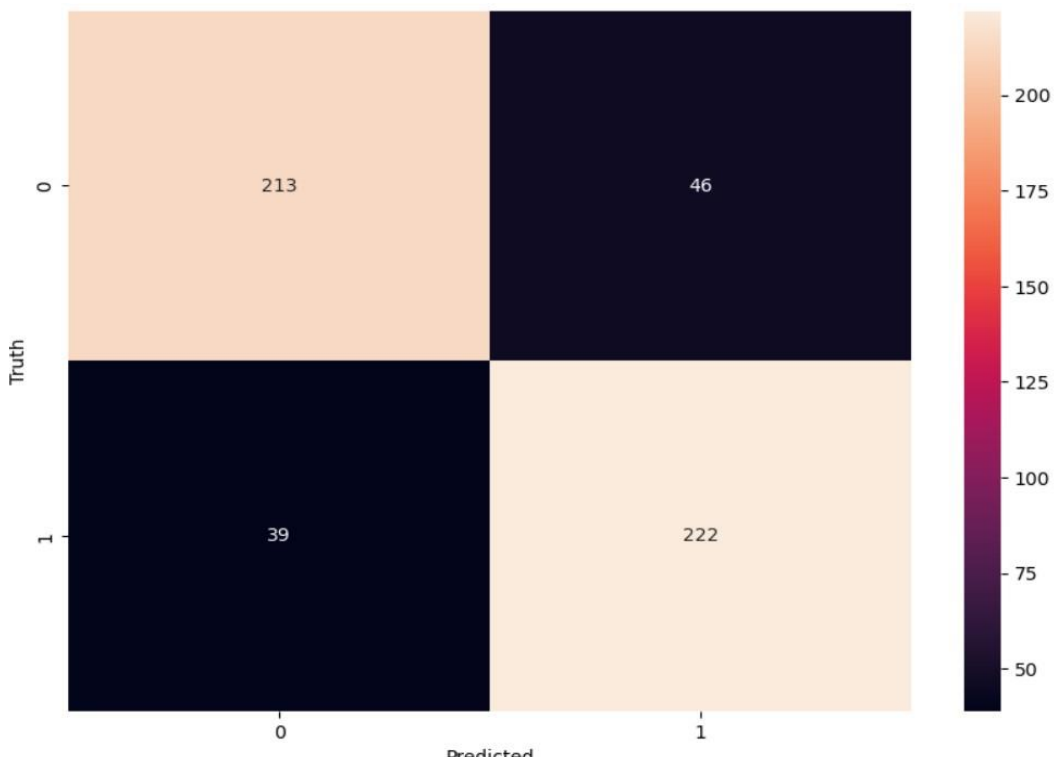
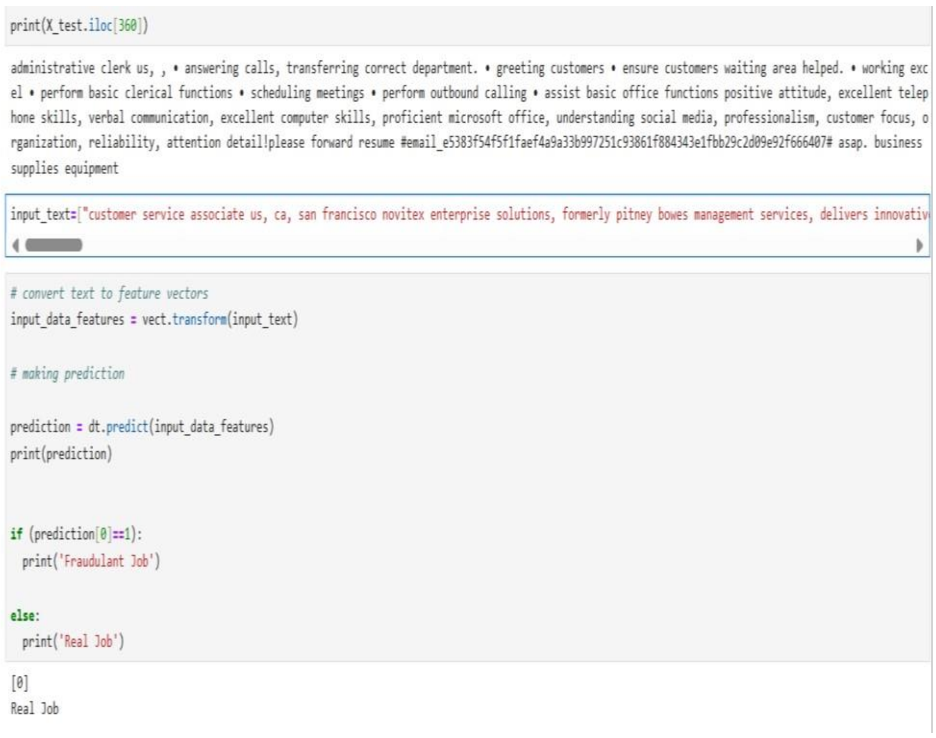


Fig 5.16 Decision Tress prediction



```
print(X_test.iloc[360])
```

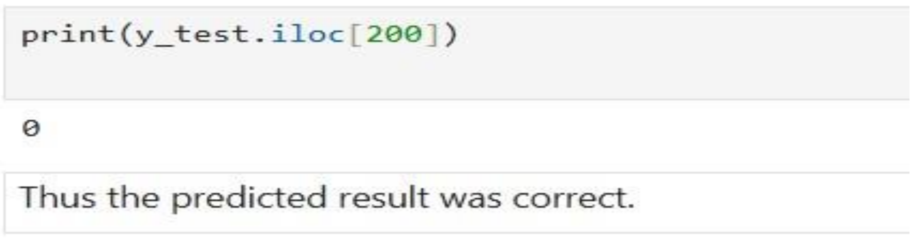
```
administrative clerk us, , • answering calls, transferring correct department. • greeting customers • ensure customers waiting area helped. • working exc
el • perform basic clerical functions • scheduling meetings • perform outbound calling • assist basic office functions positive attitude, excellent telep
hone skills, verbal communication, excellent computer skills, proficient microsoft office, understanding social media, professionalism, customer focus, o
rganization, reliability, attention detail!please forward resume #email_e5383f54f5f1faef4a9a33b997251c93861f884343e1fbb29c2d09e92f666407# asap. business
supplies equipment
```

```
input_text=["customer service associate us, ca, san francisco novitex enterprise solutions, formerly pitney bowes management services, delivers innovativ
```

```
# convert text to feature vectors
input_data_features = vect.transform(input_text)

# making prediction

prediction = dt.predict(input_data_features)
print(prediction)


if (prediction[0]==1):
  print('Fraudulant Job')

else:
  print('Real Job')
```

```
[0]
Real Job
```

Fig 5.17 Selecting the input text

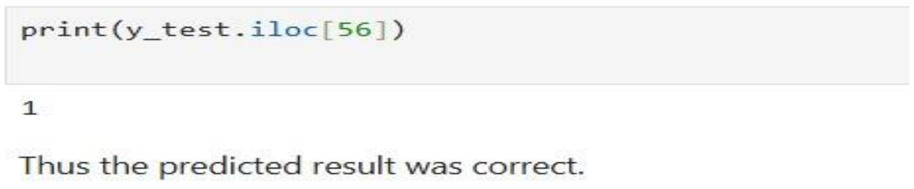


```
print(y_test.iloc[200])
```

```
0
```

Thus the predicted result was correct.

Fig 5.18 The given input text is real.



```
print(y_test.iloc[56])
```

```
1
```

Thus the predicted result was correct.

Fig 5.19 The given input text is fake

6. SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be Rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be

manuals.

Functional testing is centered on the following items:

Exercised.

Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Thoroughness: Enables comprehensive testing of all code paths, helping to identify hidden errors.

Early Detection of Bugs: Issues can be caught early in the development cycle, reducing costs and time to fix.
Optimization: Helps in identifying performance bottlenecks or inefficient code.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

User-Centric: Focuses on user experience and expected outcomes, aligning closely with user needs.
Unbiased Testing: Testers evaluate the software from an outsider's perspective, which can help identify issues overlooked by developers.
Early Detection of Issues: Can help catch bugs before the software is released.
Black box testing is essential for validating that software meets its design and functional requirements before it is released to users. It complements other testing methods, such as white box testing, which focuses on the internal workings of the software.

6.2 VARIOUS TESTCASE SCENARIOS

Table no:6.1 TestCases

Testcases ID	**Test case Description**	**Testcase steps**	**Test data**	**Expected result**	**Actual result**	**Status**

TC01	Verify dataset loading and preprocessing	1. Upload the EMSCAD dataset. 2. Perform preprocessing steps	EMSCAD dataset	Dataset is loaded and preprocessed without errors.	Successfully loaded.	Success

Testcases ID	Testcase Description	Testcase steps	Test data	Expected result	Actual result	Status
TC02	Check model training functionality	1. Select a model (e.g., decision tree). 2. Initiate training on the dataset.	EMSCAD dataset, decision tree model.	Model trains successfully and completes within a reasonable time.	Completes successfully.	Success
TC03	Validate model evaluation metrics.	1. Train a model. 2. Evaluate the model using accuracy, precision, recall, and F1score metrics.	Trained model, test data.	Metrics are calculated correctly and displayed.	It shows the metrics successfully.	Success

TC04	Ensure correct comparison of model performance	1. Train multiple models. 2. Compare their performance metrics.	Decision tree, naïve bayes, neural network models	Performanc e comparison is accurate and reflects model differences.	Invalid.	Fail
TC05	Confirm recommendation generation	1. Complete model evaluation. 2. Generate recommendation s based on model performance.	Evaluation results	Recommen dations are relevant and actionable.	Confirms.	Success
Testcas es ID	**Testcase Description**	**Testcase steps**	**Test data**	**Expected result**	**Actual result**	**Status**
TC06	Test user interface functionality	1. Access the system. 2. Upload data. 3. Select and train a model. 4. View results.	EMSCAD dataset, various models	Interface is intuitive and all functionaliti es work as expected.	functionalities work as expected.	Success
TC07	Check report generation accuracy	1. Complete model evaluation and comparison. 2. Generate a report summarizing findings.	Evaluation and comparison results	Report accurately summarizes methodolog y, results, and recommend ations.	Accuracy generated.	Success
TC08	Verify system performance under load	1. Simultaneously initiate multiple model trainings. 2. Monitor system response and resource usage.	Multiple datasets and models	System handles concurrent processes efficiently with acceptable response times.	Multiple datasets are not loaded.	fail

7.CONCLUSION AND FUTURE ENHANCEMENT

7.1PROJECT CONCLUSION:

Job scam detection remains a critical issue globally, and our study using the EMSCAD dataset has shed light on the effectiveness of different classification algorithms. We applied both Decision Tree and Naive Bayes algorithms to identify fake job posts. Our findings revealed that the Decision Tree algorithm provided a robust classification performance, offering clear interpretability of decision rules. The Naive Bayes algorithm, known for its simplicity and efficiency, also demonstrated strong performance, particularly in handling high-dimensional data. While Decision Trees excelled in providing detailed insights into feature importance, Naive Bayes achieved competitive accuracy with lower computational requirements. This comparative analysis underscores the strengths of both algorithms in job scam detection and highlights their potential for enhancing job market safety.

7.2 FUTURE ENHANCEMENT:

The future scope of a comparative study on fake job post prediction using different data mining techniques:

Advanced Machine Learning Algorithms: As machine learning algorithms evolve, newer and more sophisticated models may emerge that can better distinguish between genuine and fake job postings.

Natural Language Processing (NLP): With advancements in NLP, more accurate text analysis can be performed to identify suspicious patterns, misleading information, or inconsistencies in job postings.

Feature Engineering: Future studies can focus on identifying and extracting new features or variables from job postings that can provide better insights into their authenticity. This could include linguistic cues, job requirements, company details, and more.

Real-time Monitoring and Analysis: Developing real-time monitoring systems that can continuously scan and analyze job postings from various platforms can be an interesting area of research. This can help in early detection and prevention of fake job posting.

8.REFERENCES

8.1 PAPER REFERENCES :

[1] S. Vidros, C. Kolias , G. Kambourakis ,and L. Akoglu, “Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset”, Future Internet 2017, 9, 6; doi:10.3390/fi9010006.

[2] B. Alghamdi, F. Alhar by, “An Intelligent Model for Online Recruitment Fraud Detection”, Journal of Information Security, 2019, Vol 10, pp. 155176, https://doi.org/10.4236/iis.2019.103009.

[3] Tin Van Huynh1, Kiet Van Nguyen, Ngan Luu-Thuy Nguyen1, and AnhGia-Tuan Nguyen, “Job Prediction: From Deep Neural Network Models to Applications”, RIVF International Conference on Computing and Communication Technologies (RIVF), 2020.

[4] Jiawei Zhang, Bowen Dong, Philip S. Yu, “FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network”, IEEE 36thInternational Conference on Data Engineering (ICDE), 2020.

[5] Scanlon, J.R. and Gerber, M.S., “Automatic Detection of Cyber Recruitment by Violent Extremists”, Security Informatics, 3, 5, 2014, https://doi.org/10.1186/s13388-014-0005-5. [6] Y. Kim, “Convolutional neural networks for sentence classification,” arXivPrepr. arXiv1408.5882, 2014.

[7] Jindal, A., & Liu, B. (2009). "Review Spam Detection." Proceedings of the 2009 International Conference on Web Search and Data Mining.

[8] Agerri, R., & De Meo, P. (2017). "Text Mining for Job Offer Classification: A Case Study." *Journal of Computer and System Sciences*.

8.2 LINKS:

- Recruitment Scam (kaggle.com)
- Datasets - UCI Machine Learning Repository
- ResearchGate | Find and share research
- Machine Learning Online Courses | Coursera

8.3 TEXT BOOKS:

- "Pattern Recognition and Machine Learning" by Christopher M. Bishop
- "Machine Learning: A Probabilistic Perspective" by Kevin P. Murphy
- "Data Mining: Practical Machine Learning Tools and Techniques" by Ian H. Witten, Eibe Frank, and Mark A. Hall
- "Introduction to Machine Learning" by Ethem Alpaydin

- "Natural Language Processing with Python" by Steven Bird, Ewan Klein, and Edward Loper
- "Text Mining with R: A Tidy Approach" by Julia Silge and David Robinson
- "Machine Learning for Text" by Charu C. Aggarwal.
- Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques. Morgan Kaufmann.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.