A Project Report On

JOB SCAM ALERT

Mini project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

(2021-2025)

BY

J. KRISHNA CHAITANYA	**21241A12F6**
C. ABHINAV REDDY	**21241A12D9**
VINOD PAWAR	**21241A12G4**

Under the Esteemed Guidance

of

Dr. Y J Nagendra Kumar

Head of the Department

DEPARTMENT OF INFORMATION TECHNOLOGY

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

HYDERABAD

2023-24

CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled **"JOB SCAM ALERT"** done by **J. KRISHNA CHAITANYA (21241A12F6), C. ABHINAV REDDY (21241A12D9), VINOD PAWAR (21241A12G4)** of **B.Tech** in the Department of Information Technology, **Gokaraju Rangaraju Institute of Engineering and Technology** during the period 2021-2025 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

Dr. Y J Nagendra Kumar
Head of the Department
(Internal Guide)

(Project External)

ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our Internal guide**,** **Dr. Y J Nagendra Kumar, Head of the Department,IT**, **GRIET**. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. Y J Nagendra Kumar,** HOD IT, our Project Coordinators **Mrs.A. Pavithra** for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Name: J. Krishna Chaitanya
Email: chaitanyajuluri11@gmail.com
Contact No: 8520046269

Name: C. Abhinav Reddy
Email: chilkuriabhinavr@gmail.com
Contact No: 8639927360

Name: Vinod Pawar
Email: vinodpawar4069@gmail.com
Contact: 9121814860

DECLARATION

This is to certify that the mini-project entitled "**JOB SCAM ALERT"** is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

J. KRISHNA CHAITANYA 21241A12F6

C. ABHINAV REDDY 21241A12D9

VINOD PAWR 21241A12G4

TABLE OF CONTENTS

11 LIST OF DIAGRAMS

ABSTRACT

In response to the rise in online job scams, we developed Job Scam Detection, an innovative solution utilizing advanced machine learning techniques. Our software leverages Random Forest and Support Vector Machine (SVM) models to identify fraudulent job postings. This technology enables job seekers to safely navigate the digital job market, protecting them from scams and safeguarding their personal information. By improving the integrity of online job listings, Job Scam Detection fosters a more secure and trustworthy environment for both job seekers and employers. Our solution not only helps individuals avoid financial scams, such as fraudulent application fees and false promises of employment, but also empowers them to make informed decisions when applying for jobs online. The model has been saved and deployed using Streamlit, allowing users to easily check whether a job posting is fake or real. We are committed to continuously refining and optimizing our software to enhance its effectiveness, aiming to revolutionize how job seekers engage with the digital job market and providing them with confidence and peace of mind in their job search. Through ongoing research and development, we strive to stay ahead of emerging threats and ensure a seamless and secure experience for all stakeholders involved.

Keywords: Job scams, Machine learning, Random Forest, Support Vector Machine (SVM), Fraud detection, Streamlit deployment

Domain: Machine Learning

1. INTRODUCTION

1.1 Introduction to Project

The incidence of employment scams has been steadily increasing, with CNBC reporting that the number of such scams doubled from 2017 to 2018. The current economic climate, exacerbated by the coronavirus pandemic, has led to widespread job losses and high unemployment rates, creating fertile ground for scammers. These scammers prey on desperate job seekers by luring them with enticing but fraudulent job offers, aiming to extract sensitive personal information such as addresses, bank account details, and social security numbers, or to solicit money under the guise of application fees or investments. As a university student, I have personally received numerous scam emails offering lucrative job opportunities that turned out to be fraudulent. Addressing this issue is crucial, and advanced Machine Learning techniques, combined with Natural Language Processing (NLP), offer a promising solution.

To tackle this problem, we developed Job Scam Detection, a sophisticated tool employing advanced machine learning algorithms. Our project utilizes data from Kaggle, which includes a mix of genuine and fake job postings. Despite the small proportion of fraudulent postings, their identification is essential to protect job seekers. Our solution employs a variety of machine learning models, focusing on Random Forest classifiers and Support Vector Machines (SVM). The Random Forest model achieved an impressive accuracy of 97.22%, while the SVM model attained a 95% accuracy. The system is designed to save the trained models and deploy them using Streamlit, enabling users to easily determine whether a job posting is real or fake.

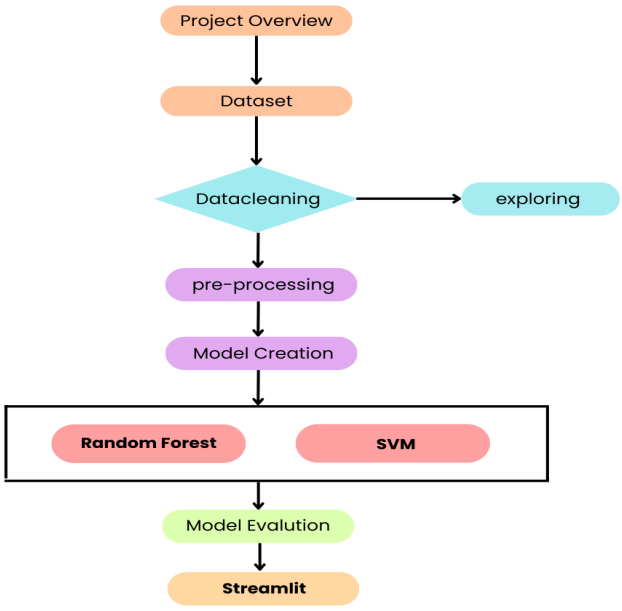
The project is structured into five stages: defining the problem, collecting data, cleaning and preprocessing data, modeling, and evaluation. The primary objective is to develop a classifier that can accurately differentiate between real and fake job postings. This classifier integrates both numeric and textual features to assess job postings comprehensively. Through this approach, Job Scam Detection aims to provide a reliable tool for job seekers, helping them navigate the job market safely and make informed decisions.

1.2 Motivation

The rise in employment scams has paralleled the increase in global economic uncertainty, particularly exacerbated by events like the coronavirus pandemic. With job losses mounting and unemployment rates soaring, individuals are increasingly vulnerable to fraudulent job offers that promise lucrative opportunities but ultimately aim to extract personal information or financial investments. As a university student, I have personally encountered deceptive job postings that initially appeared legitimate but later turned out to be scams. Addressing this issue requires innovative approaches, leveraging advanced Machine Learning techniques such as Natural Language Processing (NLP) to distinguish between authentic and fraudulent job postings. This project's motivation stems from the need to protect job seekers from falling victim to such scams by providing a reliable, technology-driven solution that ensures a safer job search experience.

1.3 METHODOLOGY

This section outlines the workflow of the proposed methodology for predicting fake job postings using machine learning techniques and natural language processing (NLP). The workflow involves several stages: problem definition, data collection, data cleaning and preprocessing, modeling, and evaluation. Figure 1 illustrates the workflow of the proposed model.

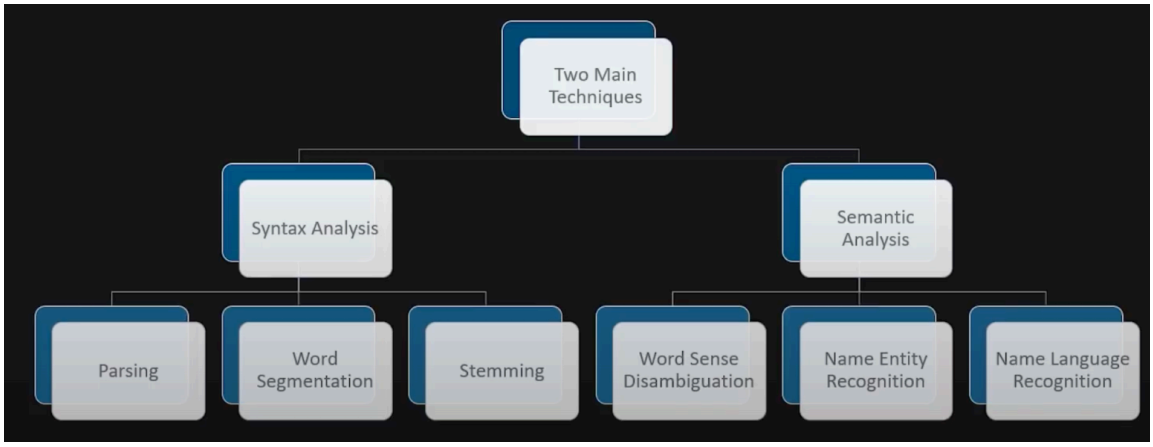


By following this methodology, the project aims to develop a robust classifier that effectively identifies fraudulent job postings, leveraging the strengths of both Random Forest and SVM classifiers. The project employs a comprehensive approach to process the dataset from Kaggle, which includes genuine and fake job postings. The classifier integrates both numeric and textual features to provide a holistic analysis of job postings. The final model is then deployed using Streamlit, enabling real-time predictions of job posting authenticity. This structured approach ensures that the developed model is both accurate and practical for real-world applications.

Natural Language Processing (NLP)

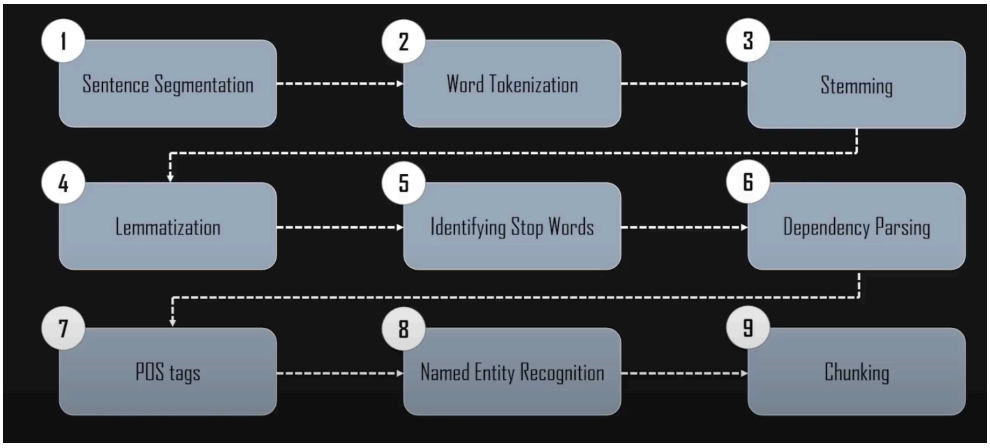
In this project, we utilized Natural Language Processing (NLP) to analyze and extract meaningful information from the text within job postings. NLP is a crucial component of artificial intelligence that enables computer programs to understand and interpret human language as it is spoken or written. By transforming raw text data into a structured format, NLP allows machine learning models to effectively classify job postings as either real or fake.

NLP employs various computational techniques to process and analyze large amounts of natural language data. It combines the principles of computational linguistics, which is the study of how language works, with advanced statistical models, machine learning, and deep learning. These technologies enable computers to analyze text data, understand context, and grasp the full meaning of the text, including the intentions and emotions of the writer.



In this project, we built an NLP pipeline that includes several key steps:

1. **Sentence Segmentation**: Dividing the text into individual sentences to simplify further analysis.
2. **Word Tokenization**: Breaking down sentences into individual words or tokens.
3. **Stemming and Lemmatization**: Reducing words to their base or root forms to ensure consistency in text analysis.
4. **Identifying Stop Words**: Removing common words that do not contribute significant meaning to the analysis, such as "and," "the," and "is."
5. **Dependency Parsing**: Analyzing the grammatical structure of sentences to understand the relationships between words.
6. **Part-of-Speech (POS) Tagging**: Assigning parts of speech to each word, such as nouns, verbs, adjectives, etc.
7. **Named Entity Recognition (NER)**: Identifying and classifying proper nouns within the text, such as names of people, organizations, and locations.
8. **Chunking**: Grouping words into meaningful chunks, such as noun phrases or verb phrases.

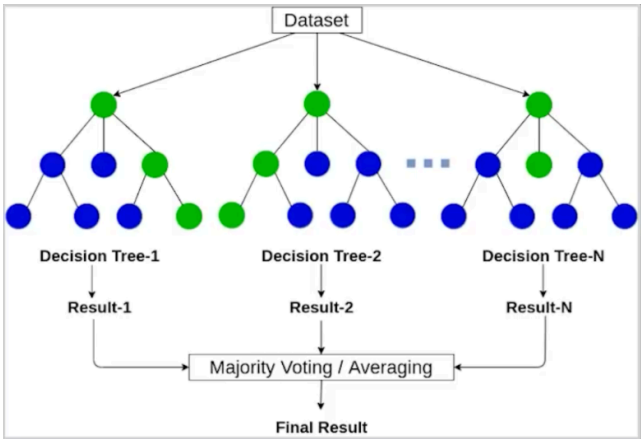


By applying these NLP techniques, we transformed unstructured textual data from job postings into structured numerical data that our machine learning models could use. This preprocessing step is essential for feature extraction and ensures that the text data is in a format suitable for machine learning algorithms.

We utilized Random Forest and Support Vector Machine (SVM) classifiers to build the predictive models. These models were trained on a dataset that included a mix of genuine and fake job postings. The use of NLP techniques allowed us to accurately identify and classify fake job postings, providing a significant improvement over manual and rule-based systems.

Random Forest

Random Forest is a powerful machine learning algorithm widely used for classification and regression tasks. It operates by constructing multiple decision trees during the training phase. Each tree in the forest is built using a random subset of the dataset and evaluates a random subset of features at each split. This variability among trees helps to reduce overfitting and improves the overall predictive performance of the model.



How Random Forest Works in Our Project

1. **Dataset Preparation**: The job postings dataset, comprising both genuine and fraudulent entries, is preprocessed and transformed using NLP techniques to create a structured numerical dataset.
2. **Decision Tree Construction**: During the training phase, the algorithm generates multiple decision trees. Each tree is trained on a different random subset of the dataset and examines different random subsets of features. This ensures that each tree provides a unique perspective on the data.

3. **Aggregation of Predictions**: When making predictions, each decision tree independently
4. classifies a job posting as either real or fake. The final classification is determined through majority voting among the trees. For instance, if most trees classify a job posting as fraudulent, the Random Forest algorithm will label it as fake.
5. **Reducing Overfitting**: The use of multiple trees and random subsets helps mitigate overfitting, a common problem in decision tree algorithms. By averaging the predictions of all trees, the Random Forest algorithm provides more stable and accurate results.
6. **Handling Complex Data**: The Random Forest algorithm is particularly effective at managing complex data structures, making it well-suited for our project. It can handle a large number of features derived from the text of job postings, ensuring reliable classifications.

Advantages of Random Forest in Our Project

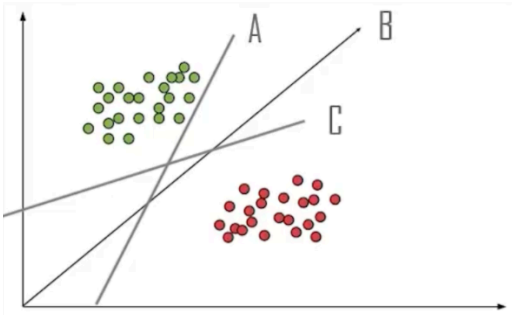
- **High Accuracy**: The ensemble nature of the algorithm, combining multiple decision trees, contributes to its high accuracy in classifying job postings.
- **Robustness**: The algorithm's ability to reduce overfitting ensures that the model generalizes well to new, unseen data.
- **Scalability**: Random Forest can handle large datasets and a high number of features, making it scalable and efficient for real-world applications.
- **Feature Importance**: The algorithm provides insights into feature importance, helping us understand which attributes of job postings are most indicative of fraud.

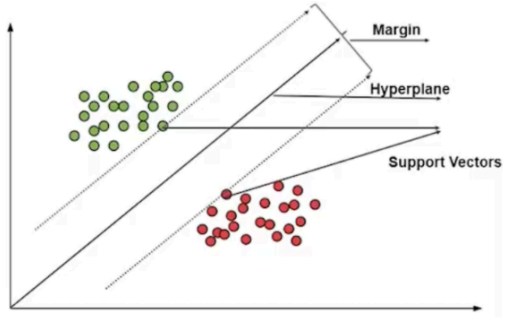
Support Vector Machine (SVM)

Support Vector Machine (SVM) is a versatile and robust machine learning algorithm that excels in both linear and nonlinear classification, regression, and outlier detection tasks. SVMs are renowned for their effectiveness across various domains, including text classification, image classification, spam detection, handwriting recognition, gene expression analysis, face detection, and anomaly detection. This adaptability stems from SVM's ability to handle high-dimensional data and intricate nonlinear relationships inherent in real-world datasets.

SVM in Job Scam Detection

1. **Optimal Hyperplane**: In our project, SVM seeks to find the hyperplane that maximizes the margin between the closest points of different classes. This margin is crucial as it determines the separation boundary between real and fraudulent job postings.
2. **Kernel Trick**: SVM utilizes kernel functions such as Polynomial, Gaussian Radial Basis Function (RBF), and others to handle nonlinear decision boundaries in the data. This flexibility allows SVM to capture intricate patterns and relationships within the textual features extracted from job postings.
3. **Classification Accuracy**: SVM aims to classify job postings accurately based on their textual content, leveraging NLP preprocessing techniques to convert unstructured text into meaningful numerical features that SVM can effectively process.
4. **Margin Maximization**: By maximizing the margin between classes, SVM reduces the risk of overfitting and improves generalization performance on unseen data, contributing to robust and reliable predictions in identifying fraudulent job postings.
5. **Real-world Applicability**: SVM's ability to handle high-dimensional data and nonlinear relationships makes it suitable for real-world applications where the distinction between legitimate and fraudulent job offers is critical for protecting job seekers from financial scams and deceptive practices.





1.4 Existing System

The existing system for detecting fake job postings primarily relies on manual review processes or basic rule-based filtering methods. These systems typically involve human reviewers who examine job postings for signs of fraud based on their experience and predefined rules. For instance, they might check for discrepancies in job details, unusual salary ranges, or suspicious email addresses. However, this approach has several limitations:

Scalability Issues: As the volume of job postings increases, manual review becomes impractical and inefficient.

Subjectivity and Human Error: The accuracy of detecting fake job postings depends heavily on the expertise and vigilance of the reviewers, making it prone to human error.

Slow Response Time: Manual review processes are time-consuming, resulting in delayed identification and removal of fake job postings.

Limited Adaptability: Rule-based systems are often static and cannot adapt quickly to new types of scams or evolving fraudulent techniques.

1.5 Proposed System

The proposed system leverages machine learning and natural language processing (NLP) techniques to automatically detect fake job postings. This system aims to address the limitations of the existing system by providing a scalable, accurate, and adaptive solution.

Key Components:

1. **Data Collection**: Gather job posting data from various sources.
2. **Data Preprocessing**: Clean and preprocess the data, including handling missing values and text preprocessing (tokenization, stopword removal, lemmatization).
3. **Feature Engineering**: Create new features from the existing data, such as combining text fields and generating character counts.

4. **Model Training**: Train machine learning models (Naive Bayes, SGD Classifier and LSTM) on the preprocessed data.
5. **Model Evaluation**: Evaluate the models using accuracy and F1-score metrics.
6. **Final Classification**: Combine the outputs of the models to determine the authenticity of job postings.

Flow Chart of the Proposed System:

1.5 Proposed System

The proposed system integrates advanced machine learning and natural language processing (NLP) techniques to automate the detection of fraudulent job postings, addressing inherent limitations in current methods. Key enhancements and components of the system include:

Data Collection: Job posting data is gathered from diverse sources, providing a comprehensive dataset that encompasses various attributes such as job title, location, and company details.

Data Preprocessing: The collected data undergoes rigorous preprocessing steps to enhance quality and usability. This includes handling missing values, text preprocessing (tokenization, stopword removal, lemmatization), and ensuring data integrity.

Feature Engineering: New features are engineered from the processed data to enrich the dataset. Techniques such as combining text fields, generating character counts, and extracting relevant keywords are employed to enhance model performance.

Model Selection and Training: The project focuses on utilizing Random Forest and Support Vector Machine (SVM) algorithms due to their demonstrated efficacy in classification tasks. These models are trained on the preprocessed dataset to learn patterns and distinguish between genuine and fake job postings.

Model Evaluation and Validation: The trained models are evaluated using appropriate metrics to assess their performance accurately. The evaluation criteria include accuracy, precision, recall, and F1-score, ensuring robustness and reliability in detecting fraudulent job postings.

Integration with Streamlit: Post model training and validation, the best-performing model, particularly the Random Forest and SVM, is saved and integrated into a Streamlit-based application. This application provides a user-friendly interface where users can input job postings and receive real-time predictions on their authenticity.

System Objectives:

- **Identification of Genuine vs. Fake Job Postings:** The primary goal is to accurately classify job postings, thereby enabling job seekers to focus exclusively on legitimate opportunities.
- **Enhanced Data Handling:** Leveraging a Kaggle dataset ensures comprehensive coverage of job attributes, facilitating thorough analysis and detection.
- **Streamlined Data Processing:** Rigorous preprocessing ensures that the input data is cleaned and optimized for predictive modeling, enhancing the accuracy and reliability of the system.

By leveraging these advancements, the proposed system aims to revolutionize the detection of fake job postings, providing a scalable, accurate, and adaptive solution that addresses the evolving challenges in the digital job market.

2. REQUIREMENT ENGINEERING

2.1 Hardware Requirements

- Processor – i5 and above (64-bit OS).
- Memory – 4GB RAM (Higher specs are recommended for high performance)
- Input devices – Keyboard, Mouse

2.2 Software Requirements

- Anaconda Navigator/Jupiter Notebook
- Python
- Python Libraries
  1. pandas
  2. numpy
  3. matplotlib
  4. scikit-learn
  5. seaborn

3. LITERATURE SURVEY

Employment Scams Are on the Rise: This segment highlights the increasing prevalence of employment scams, exacerbated by economic uncertainties and events like the coronavirus pandemic. The study underscores the application of machine learning and natural language processing (NLP) to mitigate fraud in job postings, emphasizing the importance of protecting job seekers from deceptive schemes.[1]

Prediction of Fake Job Postings Using Machine Learning and Topic Modeling : This project focuses on predicting fake job postings using machine learning models deployed via platforms like Heroku. The study employs algorithms such as Naive Bayes and Decision Trees, integrating topic modeling techniques like Latent Dirichlet Allocation (LDA) to enhance classification accuracy. By analyzing textual features extracted from job descriptions, the research aims to provide real-time detection of fraudulent job listings and protect job seekers from financial scams.[2]

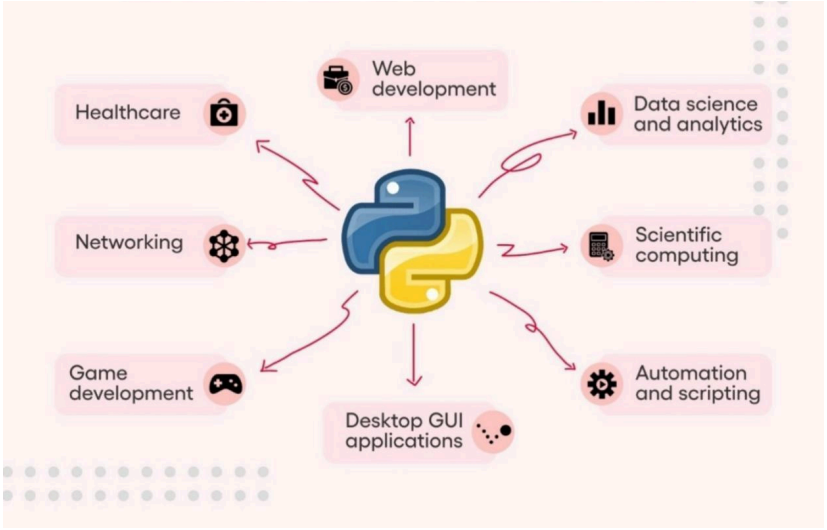
Job Posting Fraud Detection Using NLP and Machine Learning: This project addresses the detection of fake job postings by leveraging text analysis. With a highly imbalanced dataset (only 5% fraudulent postings), extensive preprocessing was necessary to handle missing values. Initially, a baseline RNN with Embedding and Bidirectional LSTM layers was deployed, achieving a validation AUC of 0.7974 and a test AUC of 0.7864. Comparison with a Small-BERT model significantly improved results, yielding a validation AUC of 0.9625 and a test AUC of 0.9637. This highlights the efficacy of advanced NLP models in distinguishing between real and fraudulent job postings [3].

Impact of Employment Scams and Detection Methods: Employment scams, as described by the FBI, involve criminals deceiving victims under the guise of job opportunities to extract personal information or money. In 2020 alone, over 16,000 people reported being victims of such scams, resulting in losses exceeding $59 million. Major job platforms like Indeed and LinkedIn have implemented measures to combat fraudulent postings. This project develops a job posting classifier using textual data to identify fake job listings, aiming to safeguard job seekers from financial and identity-related risks associated with employment scams [4].

4. TECHNOLOGY

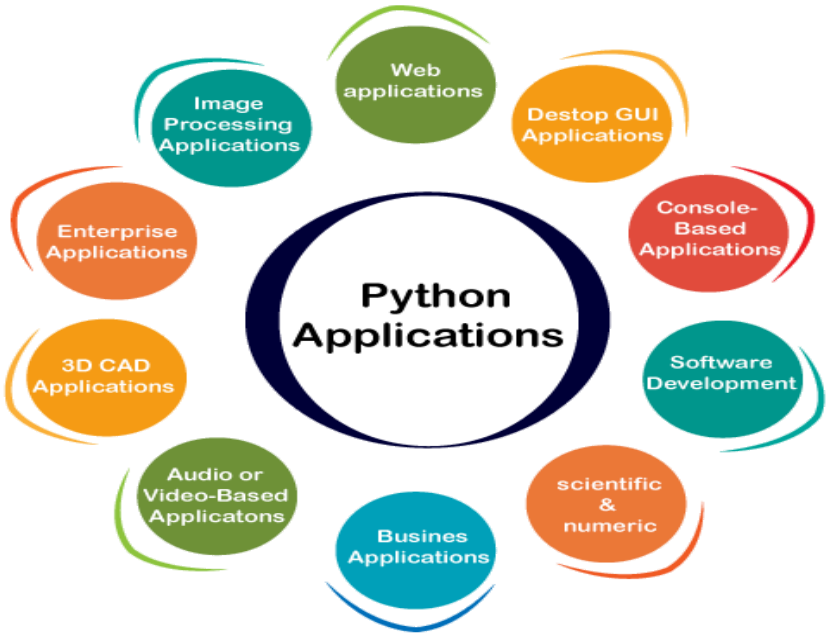
4.1 ABOUT PYTHON

Python's environment has evolved significantly, enhancing its capabilities for statistical analysis. It strikes a fine balance between scalability and elegance, placing a premium on efficiency and code readability. Python is renowned for its emphasis on program readability, featuring a straightforward syntax that is beginner-friendly and encourages concise code expression through indentation. Noteworthy aspects of this high-level language include dynamic system functions and automatic memory management.



4.2 APPLICATIONS OF PYTHON

Python is used in many application domains. It makes its presence in every emerging field. It is the fastest-growing programming language and may be used to create any type of application.



Applications of python

It is used in various fields:

- Web Applications. We can use Python to develop web applications. ...
- Desktop GUI Applications. ...
- Console-based Application. ...
- Software Development. ...
- Scientific and Numeric. ...
- Business Applications. ...
- Audio or Video-based Applications. ...
- 3D CAD Applications.

4.3 PYTHON IS WIDELY USED IN MACHINE LEARNING

Python is widely favored in machine learning for its flexibility and open-source nature. It provides extensive functionality for mathematical computations and scientific operations, making it indispensable in developing and deploying machine learning models. Python's simple syntax and vast libraries accelerate the development process, reducing coding time significantly.

This makes it a preferred choice for machine learning practitioners seeking efficiency and robustness in their projects.

The major Python libraries used in machine learning are as follows:

4.3.1 PANDAS

Pandas is a Python library used for statistical analysis, data cleaning, exploration, and manipulation. Typically, datasets contain both useful and extraneous information. Pandas helps to make this data more readable and relevant.

4.3.2 NUMPY

NumPy is a Python library utilized for numerical data reading, cleaning, exploration, and manipulation. It provides powerful data structures for efficient computation with large arrays and matrices, making the data more accessible and manageable.

4.3.3 MATPLOTLIB

Matplotlib is a Python library for plotting graphs. Built on NumPy arrays, it allows for the creation of a wide range of graph types, from basic plots to bar graphs, histograms, scatter plots, and more.

4.3.4 SCIKIT-LEARN

Scikit-learn is a Python library for machine learning. It provides tools for machine learning and statistical modeling, including classification, regression, and clustering.

4.3.5 CSV

It is a kind of file that stores tabular records, like a spreadsheet or a database. There are one or extra fields in each entry, that are separated by commas. We use the csv built in module to work with CSV files.

4.3.5 SEABORN

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper.

4.3.6 INTERPRETED LANGUAGE

Python executes code line by line, without the need for prior compilation. This approach facilitates quicker development cycles and simplifies the debugging process. As a result, developers can iterate and test their code more efficiently.

4.4 Dataset Description

The dataset utilized for this project comprises 17,880 job descriptions, with approximately 800 labeled as fraudulent. It includes a combination of textual and meta-information about each job, making it suitable for creating classification models to detect fake job postings. Key attributes in the dataset include a unique job identifier (**job_id**), job title (**title**), and geographical location (**location**). The **department** attribute provides information about the corporate department, while the **salary_range** attribute indicates the expected salary range for the job. The dataset also contains a brief description of the company (**company_profile**), a detailed job description (**description**), job requirements (**requirements**), and offered benefits (**benefits**). Additionally, it

specifies whether the position allows telecommuting (**telecommuting**) and includes a target variable (**fraudulent**) indicating if the job is fake or real.

This dataset, sourced from Kaggle, is highly valuable for answering several important questions. It enables the creation of classification models that utilize both text data features and meta-features to predict which job descriptions are fraudulent. It also helps identify key traits or features (words, entities, phrases) that are indicative of fraudulent job descriptions. Additionally, the dataset can be used to run contextual embedding models to identify the most similar job descriptions and perform exploratory data analysis to uncover interesting insights.

Name of Attribute	Description
job_id	Unique identifier for each job posting.
title	The job title, such as "English Teacher Abroad."
location	Geographical location of the job, e.g., "US, NY, New York."
department	The corporate department (e.g., Sales).
salary_range	Indicative salary range (e.g., $50,000-$60,000).
company_profile	A brief description of the company.
description	Detailed description of the job ad.
requirements	Enlisted requirements for the job opening.
benefits	Offered benefits by the employer.
telecommuting	Indicates if the position allows telecommuting.
fraudulent	Target variable indicating if the job is fake (1) or real (0).

Dataset Statistics:

- **Total Entries**: 17,880
- **Fraudulent Jobs**: 725 (approximately 7%)
- **Real Jobs**: 17,155 (approximately 93%)

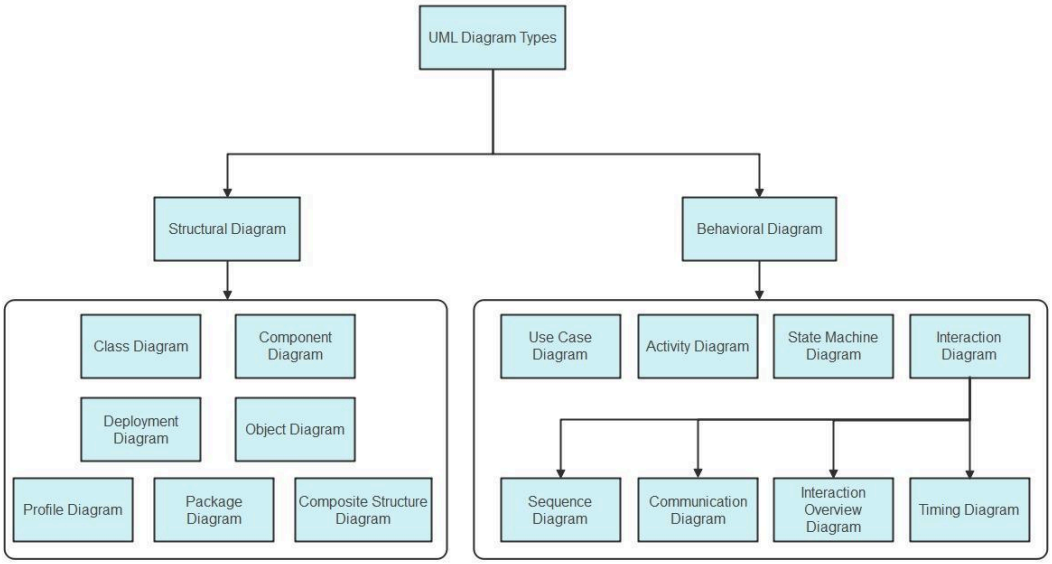
5.DESIGN REQUIREMENT ENGINEERING

Concept of uml:

The purpose of these UML-based diagrams is to visually depict the system, including its key components, roles, operations, objects, or interactions. This visual representation aims to enhance understanding, facilitate manipulation, and effectively document or manage system-related information.

UML DIAGRAMS:

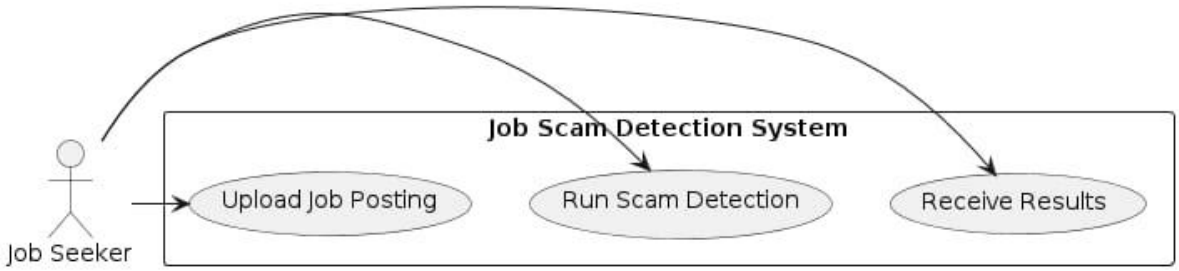
The Unified Modeling Language (UML) serves as a standardized language for creating models across various domains. Its primary goal is to visually represent the structure of a system, akin to blueprints in engineering disciplines. In complex applications involving multiple teams, clear communication is crucial, especially to stakeholders who may not be familiar with programming code. UML facilitates this communication by illustrating essential system requirements, features, and processes in a visual manner. By depicting processes, user interactions, and the system's static structure, UML helps teams streamline collaboration and optimize efficiency.



Concepts of uml

5.1 Use case Diagram:

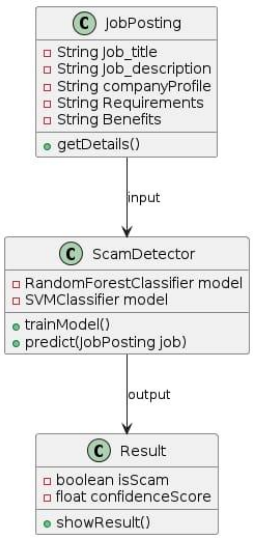
A use case diagram is a type of behavioral diagram that is a graphical explanation of the functionalities offered by the system in relation to the participants, their goals, and any dependencies between these cases.



Use Case Diagram

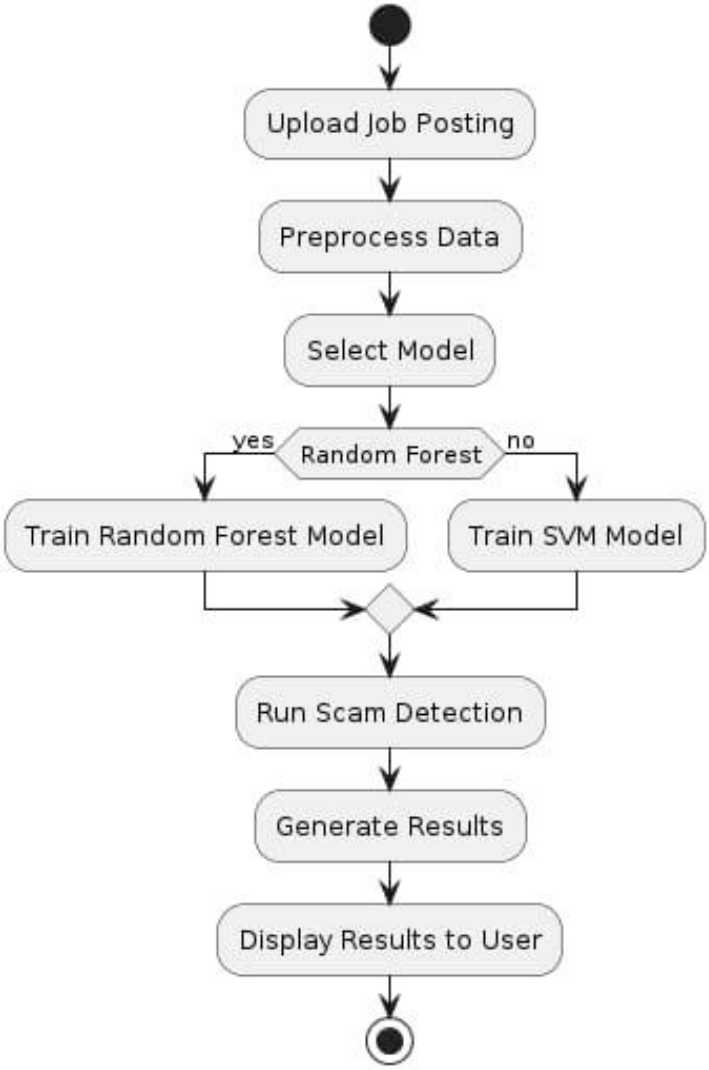
5.2 Class Diagram:

A class diagram is a static type of structural diagram that still depicts the format of a machine by means of illustrating the hyperlinks among the machine's lessons, attributes, operations, and instructions.



5.3 Activity diagram:

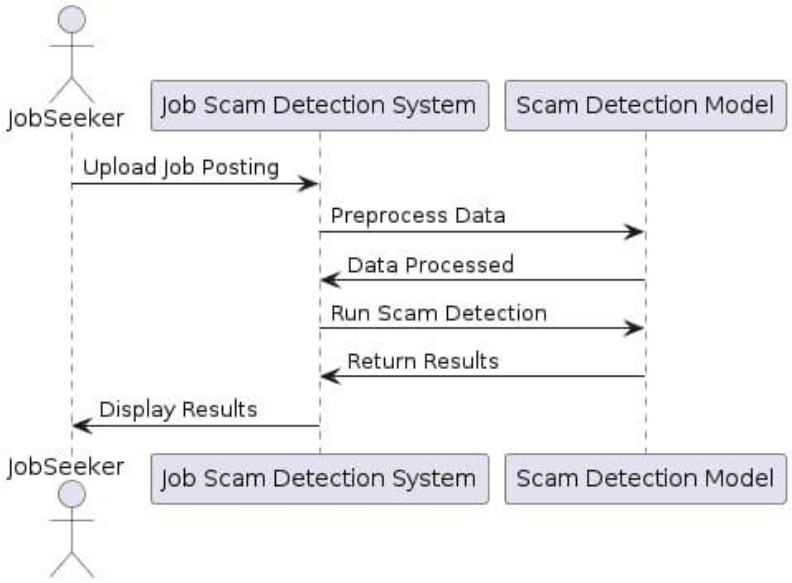
This diagram is a more complex version of a flow chart that depicts the flow of information from one activity to the next. It describes the coordination of activities in order to offer a service at various levels of abstraction.



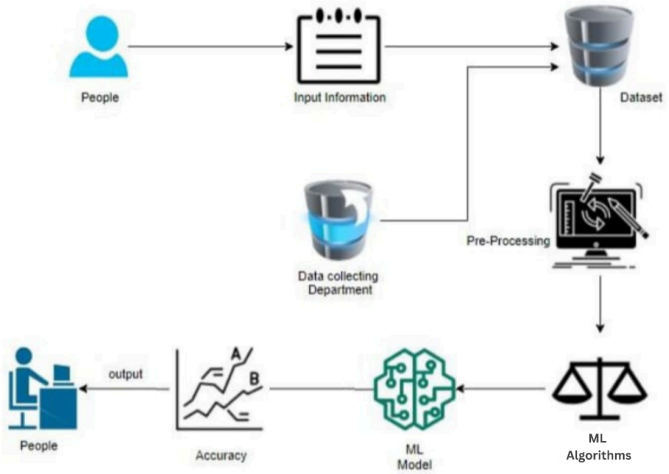
Activity Diagram

5.4 Sequence Diagram

The deployment diagram of our machine in order to define distinct states of an object for the duration of its lifetime. It usually suggests how the kingdom of an object adjusts in its lifetime.



5.5 Architecture



6. IMPLEMENTATION

The implementation process for classifying job postings as genuine or fraudulent encompasses several crucial stages: data collection, data cleaning and preprocessing, model construction using selected algorithms (Random Forest and SVM), and performance evaluation. Below is a detailed breakdown of each stage:

Sample Data for the Problem Statement:

The sample data used in this project consists of job postings collected from a reliable source, aimed at developing a classifier to distinguish between real and fake job advertisements. The dataset includes various features that describe each job posting, providing essential information for classification tasks. Here's a detailed overview of the sample data:

Column	Description	Data Type	Valid Entries	Missing Entries	Unique Entries	Most Common Value
job_id	Unique Job ID	Integer	17,880	0	17,880	N/A
title	Job title	Text	17,880	0	11,200	"English Teacher Abroad" (2%)
location	Geographical location	Text	17,534	346	3,105	"GB, LND, London" (4%)
department	Corporate department	Text	6,333	11,547	1,337	"Sales" (3%)
salary_range	Indicative salary range	Text	2,868	15,012	874	"0-0" (1%)
company_profile	Company description	Text	14,572	3,308	1,709	"We help teachers get safe & secure jobs abroad :)" (4%)
description	Detailed job description	Text	17,880	0	14,800	Various
requirements	Job requirements	Text	15,186	2,694	12,000	"University degree required. TEFL / TESOL / CELTA or teaching experience preferred but not necessaryCanada/US passport holders only" (2%)
benefits	Job benefits	Text	10,674	7,206	6,207	"See job description" (4%)
telecommuting	Telecommuting position (0 or 1)	Boolean	17,880	0	2	0 (96%)
fraudulent	Target variable (0: Real, 1: Fake)	Boolean	17,880	0	2	0 (93%)

Sample Data from the Dataset

Model Construction

In [1]:
```
!pip install wordcloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: wordcloud in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

In [2]:
```
!pip install -U spacy
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: spacy in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (3.7.5)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spac
y) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spac
y) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy)
(1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (2.0.
8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (3.
0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (8.2.
5)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (1.1.
3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (2.4.
8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (
2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (0.4.
1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in c:\users\krishna chaitanya\appdata\roaming\python\python311\site-packages (from spacy) (0.12.
3)
```

In [3]:
```
import re
import string
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.base import TransformerMixin
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
from wordcloud import WordCloud
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from spacy.lang.en import English
```

Importing Packages

Data Collection

```
job_data = pd.read_csv(r"C:\Users\krishna Chaitanya\OneDrive\Desktop\JobScamAlert\Job_Scam_Alert.csv.zip")
```

```
job_data.head()
```

	job_id	title	location	department	salary_range	company_profile	description	requirements	benefits	telecommuting	has_company_logo	has_c
0	1	Marketing Intern	US, NY, New York	Marketing	NaN	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experience with content management systems a m...	NaN	0	1	
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Success	NaN	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we expect from you:Your key responsibilit...	What you will get from usThrough being part of...	0	1	
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	NaN	NaN	Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implement pre-commissioning and commissioning ...	NaN	0	1	
3	4	Account Executive - Washington DC	US, DC, Washington	Sales	NaN	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Bachelor's or Master's in GIS, busi...	Our culture is anything but corporate—we have ...	0	1	
4	5	Bill Review Manager	US, FL, Fort Worth	NaN	NaN	SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATIONS:RN license in the State of Texa...	Full Benefits Offered	0	1	

```
job_data.shape
```

```
(17880, 18)
```

Data Cleaning and Preprocessing

Preprocessing plays a crucial role in preparing job posting data for classification tasks, distinguishing between genuine and fraudulent postings. Key preprocessing steps include:

1. **Tokenization**: Breaking down text into smaller units, typically words or punctuation marks, using libraries like NLTK or spaCy.
2. **Stemming**: Reducing words to their root form to normalize variations across job postings, enhancing consistency in word representation.
3. **Removing Stop Words**: Eliminating common words (e.g., "a", "the", "and") that do not contribute to classification accuracy, focusing on more meaningful words.

```
#data cleaning
```

```
job_data.isnull().sum()
```

```
job_id                     0
title                      0
location                 346
department             11547
salary_range           15012
company_profile         3308
description                1
requirements            2696
benefits                7212
telecommuting              0
has_company_logo           0
has_questions              0
employment_type         3471
required_experience     7050
required_education      8105
industry                4903
function                6455
fraudulent                 0
dtype: int64
```

```
columns=['job_id', 'telecommuting', 'has_company_logo', 'has_questions', 'salary_range', 'employment_type']
for colu in columns:
    del job_data[colu]
```

```
job_data.head()
```

	title	location	department	company_profile	description	requirements	benefits	required_experience	required_education	industry	function
0	Marketing Intern	US, NY, New York	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experience with content management systems a m...	NaN	Internship	NaN	NaN	Marketing
1	Customer Service - Cloud Video Production	NZ, , Auckland	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we expect from you:Your key responsibilit...	What you will get from usThrough being part of...	Not Applicable	NaN	Marketing and Advertising	Customer Service

Word cloud

```
STOPWORDS = spacy.lang.en.stop_words.STOP_WORDS
plt.figure(figsize=(16,14))
wc = WordCloud(min_font_size = 3, max_words = 3000 , width = 1600 , height = 800 ,  stopwords = STOPWORDS).generate(str(" ".join(fraudjobs_text)))
plt.imshow(wc, interpolation = 'bilinear')
```

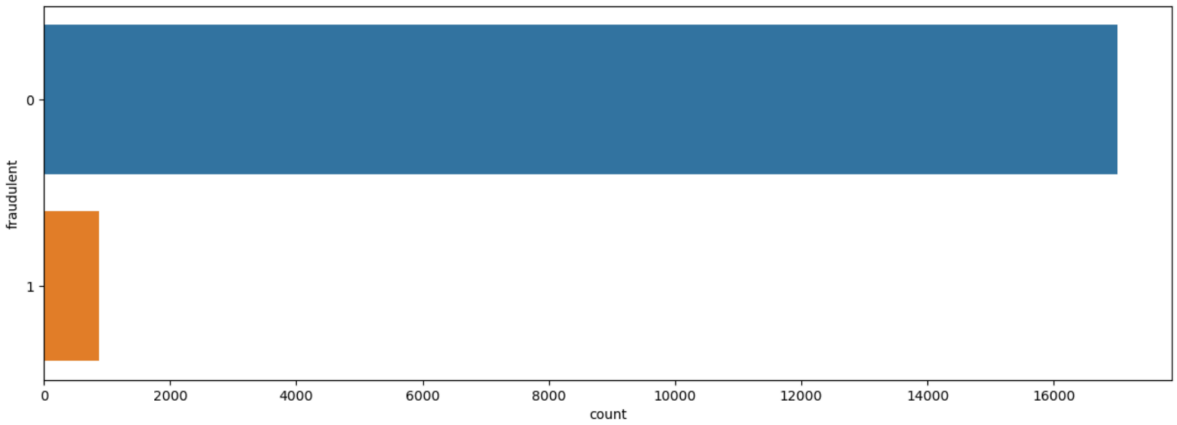
```
<matplotlib.image.AxesImage at 0x2b566818250>
```

Data Plotting

```
#ploting
```

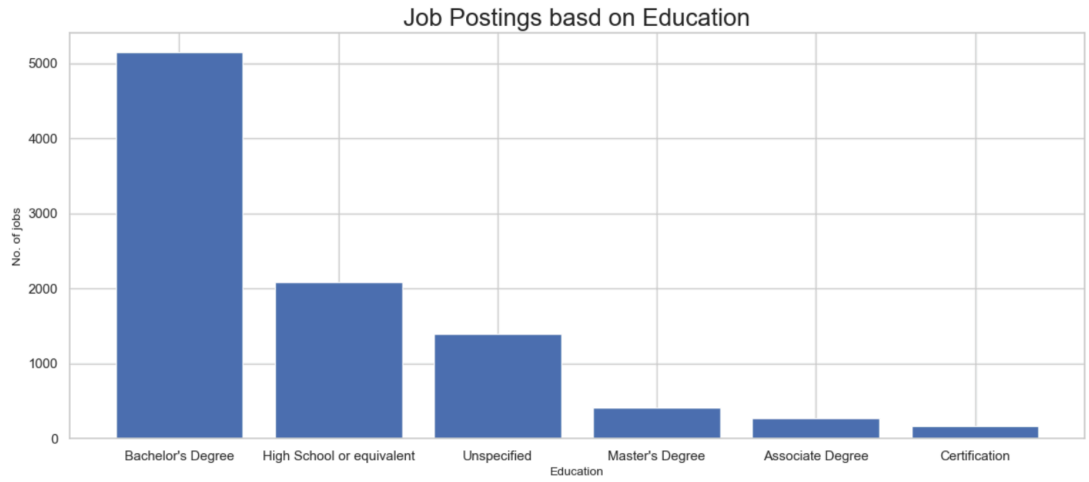
```
# Fill NaN values with blank space
# inplace=true to make this change in the dataset permanent
job_data.fillna(' ', inplace=True)
```

```
plt.figure(figsize=(15,5))
sns.countplot(y='fraudulent',data=job_data)
plt.show()
```



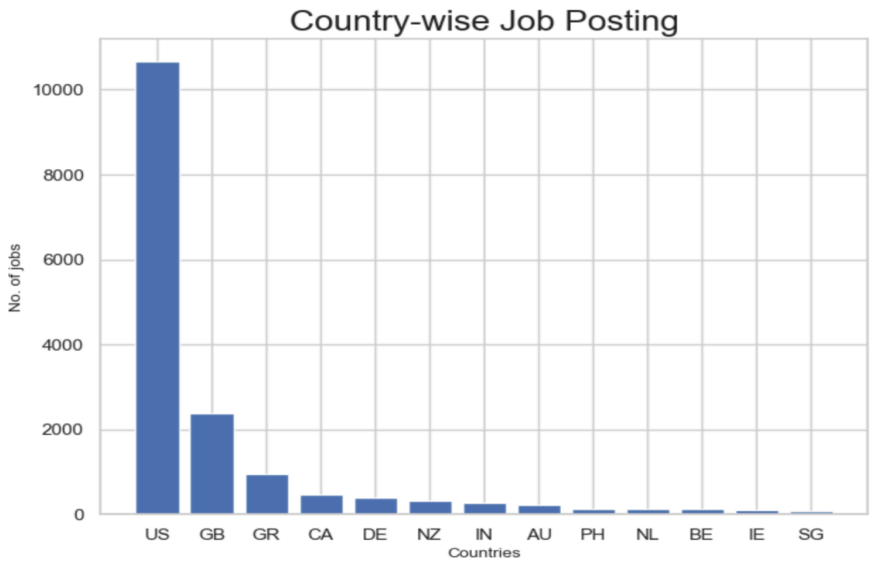
In [23]:
```
plt.figure(figsize=(15,6))
plt.title('Job Postings basd on Education', size=20)
plt.bar(edu.keys(),edu.values())
plt.ylabel('No. of jobs', size=10)
plt.xlabel('Education', size=10)
```

Out[23]:
```
Text(0.5, 0, 'Education')
```

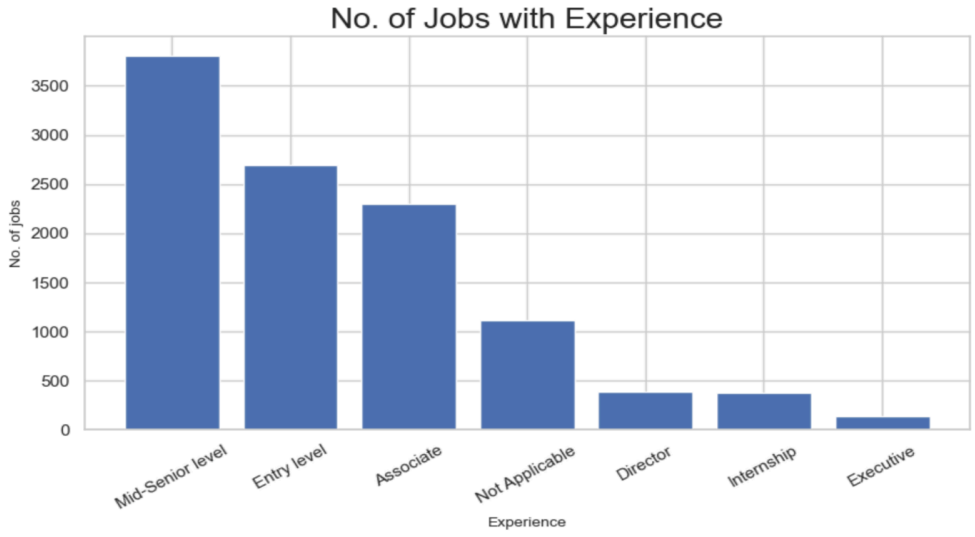


```
plt.figure(figsize=(8,6))
plt.title('Country-wise Job Posting', size=20)
plt.bar(countr.keys(), countr.values())
plt.ylabel('No. of jobs', size=10)
plt.xlabel('Countries', size=10)
```

```
Text(0.5, 0, 'Countries')
```




```
plt.figure(figsize=(10,5))
sns.set_theme(style="whitegrid")
plt.bar(exp.keys(), exp.values())
plt.title('No. of Jobs with Experience', size=20)
plt.xlabel('Experience', size=10)
plt.ylabel('No. of jobs', size=10)
plt.xticks(rotation=30)
plt.show()
```



Text Preprocessing and Analysis :

```
# Load the spacy model
nlp = spacy.load("en_core_web_sm")
stop_words = spacy.lang.en.stop_words.STOP_WORDS
punctuations = string.punctuation

# Initialize the parser
parser = English()

def spacy_tokenizer(sentence):
    # Create spacy tokens
    mytokens = parser(sentence)

    # Lemmatize and convert to lowercase
    mytokens = [word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens]

    # Remove stop words and punctuations
    mytokens = [word for word in mytokens if word not in stop_words and word not in punctuations]

    return mytokens

class predictors(TransformerMixin):
    def transform(self, X, **transform_params):
        return [clean_text(text) for text in X]

    def fit(self, X, y=None, **fit_params):
        return self

    def get_params(self, deep=True):
        return {}

def clean_text(text):
    return text.strip().lower()
```

```
job_data['text'] = job_data['text'].apply(clean_text)
```

TF-IDF Vectorization :

```
cv = TfidfVectorizer(max_features=100)

# Fit and transform the text data
x = cv.fit_transform(job_data['text'])

# Convert to DataFrame using the updated method
job_data1 = pd.DataFrame(x.toarray(), columns=cv.get_feature_names_out())

# Drop the original 'text' column from job_data
job_data.drop(["text"], axis=1, inplace=True)

# Concatenate the new DataFrame with the original job_data
main_df = pd.concat([job_data1, job_data], axis=1)
```

```
main_df.head()
```

	ability	about	all	also	amp	an	and	are	as	at	...
0	0.000000	0.041120	0.000000	0.042424	0.036488	0.000000	0.755238	0.000000	0.078653	0.000000	...
1	0.021895	0.094183	0.035394	0.024292	0.041787	0.029771	0.490896	0.056626	0.060050	0.052431	...
2	0.000000	0.000000	0.176807	0.000000	0.041749	0.089231	0.397029	0.113149	0.000000	0.000000	...
3	0.023267	0.000000	0.018806	0.000000	0.000000	0.094909	0.695542	0.000000	0.031906	0.037144	...
4	0.000000	0.000000	0.068009	0.000000	0.040147	0.028602	0.606379	0.081605	0.115386	0.000000	...

5 rows × 101 columns

Model Training

Random Forest Classifier

In [37]:
```
#random forst
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_jobs=3,oob_score=True,n_estimators=100, criterion="entropy")
model=rfc.fit(X_train,y_train)
```

In [39]:
```
print(X_test)
```

```
        ability     about       all      also       amp        an       and  \
5886   0.108372  0.000000  0.000000  0.000000  0.000000  0.000000  0.723146
14690  0.000000  0.046491  0.034942  0.000000  0.082508  0.029391  0.346166
17307  0.000000  0.137899  0.034548  0.000000  0.000000  0.029059  0.387897
7013   0.000000  0.000000  0.111887  0.000000  0.000000  0.031370  0.394114
12007  0.000000  0.000000  0.000000  0.000000  0.569288  0.000000  0.212309
...         ...       ...       ...       ...       ...       ...       ...
13102  0.000000  0.000000  0.000000  0.000000  0.255761  0.182214  0.000000
16139  0.000000  0.069953  0.078864  0.036085  0.062073  0.000000  0.381968
13112  0.000000  0.132101  0.000000  0.000000  0.000000  0.125270  0.278692
17022  0.041695  0.044839  0.168503  0.000000  0.000000  0.085040  0.534185
1533   0.050140  0.107842  0.040527  0.000000  0.000000  0.102265  0.508559

            are        as        at  ...      well       who      will  \
5886   0.035035  0.037153  0.000000  ...  0.000000  0.000000  0.000000
14690  0.195663  0.059284  0.000000  ...  0.045449  0.042981  0.030803
17307  0.082910  0.000000  0.170594  ...  0.000000  0.000000  0.030456
7013   0.328180  0.094915  0.036832  ...  0.000000  0.137626  0.263025
12007  0.085717  0.000000  0.079366  ...  0.000000  0.065902  0.094461
```

Support Vector Machine (SVM)

```
#SVM
```

```
# Import necessary libraries for SVM
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

# Assuming your data is already split into X_train, X_test, y_train, y_test

# Initialize the SVM model
svm_model = SVC(kernel='linear')  # You can choose other kernels as well

# Train the SVM model
svm_model.fit(X_train, y_train)
```

Model Evaluation

Random Forest Classifier Accuracy

```
In [40]: pred = rfc.predict(X_test)
         score = accuracy_score(y_test, pred)
         score

Out[40]: 0.9684936614466816

In [41]: print("Classification Report\n")
         print(classification_report(y_test, pred))
         print("Confusion Matrix\n")
         print(confusion_matrix(y_test, pred))

Classification Report

              precision    recall  f1-score   support

           0       0.97      1.00      0.98      5098
           1       1.00      0.36      0.53       266

    accuracy                           0.97      5364
   macro avg       0.98      0.68      0.76      5364
weighted avg       0.97      0.97      0.96      5364

Confusion Matrix

[[5098    0]
 [ 169   97]]
```

SVM Accuracy

```
# Predict with the SVM model
svm_predictions = svm_model.predict(X_test)

# Evaluate the SVM model
svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_classification_report = classification_report(y_test, svm_predictions)

print(f'SVM Model Accuracy: {svm_accuracy}')
print(f'SVM Classification Report:\n{svm_classification_report}')
```

```
SVM Model Accuracy: 0.9504101416853095
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      5098
           1       0.00      0.00      0.00       266

    accuracy                           0.95      5364
   macro avg       0.48      0.50      0.49      5364
weighted avg       0.90      0.95      0.93      5364
```

Streamlit Integration for Job Posting Classification:

Streamlit is utilized here as a platform to create an interactive web application that displays predictions from your trained models without needing to retrain them each time. This setup involves:

Loading Trained Models: You'll load your pre-trained machine learning models (Random Forest and SVM) using tools like joblib. These models have been previously trained to classify job postings based on their text descriptions.

```
import joblib

# Assuming 'rfc' is your Random Forest model and 'svm_model' is your SVM model
joblib.dump(rfc, 'random_forest_model.pkl')
joblib.dump(svm_model, 'svm_model.pkl')
print("Models saved successfully.")
```

```
Models saved successfully.
```

Streamlit Application Setup: Streamlit allows you to build a user-friendly interface where users can input job details and receive predictions. The application interface typically includes:

- An input field where users can enter job descriptions.
- A button to trigger the prediction process.
- Output sections to display predictions from both models (Random Forest and SVM).

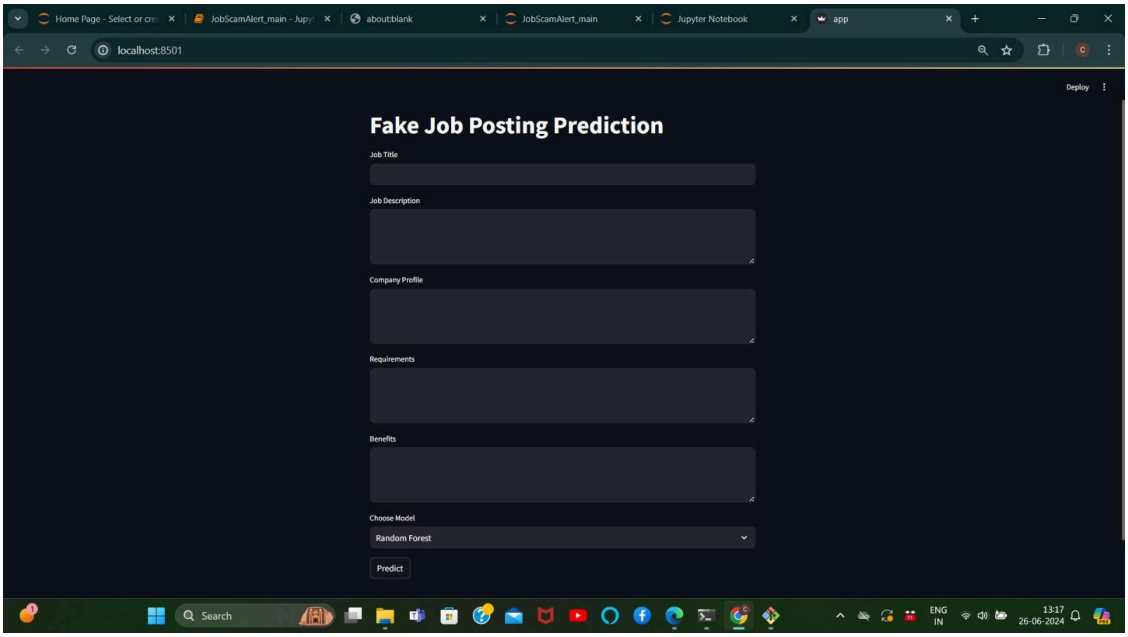
app.py

Users > abhinavreddychilkuri > Library > Containers > net.whatsapp.WhatsApp > Data > tmp > documents > 67ED6591-EF40-4ACD-B606-1C2E25793D3F > app.py

```python
import streamlit as st
import joblib
import os
import pandas as pd
import spacy
import string
from sklearn.feature_extraction.text import TfidfVectorizer

# Load the models
rfc = joblib.load('random_forest_model.pkl')
svm_model = joblib.load('svm_model.pkl')

# Initialize Spacy
nlp = spacy.load("en_core_web_sm")
stop_words = spacy.lang.en.stop_words.STOP_WORDS
punctuations = string.punctuation

# Define the tokenizer and text cleaner
def spacy_tokenizer(sentence):
    parser = spacy.lang.en.English()
    mytokens = parser(sentence)
    mytokens = [word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens]
    mytokens = [word for word in mytokens if word not in stop_words and word not in punctuations]
    return mytokens

def clean_text(text):
    return text.strip().lower()

# Preprocessing function
def preprocess_input(data):
    data = data.apply(clean_text)
    vectorizer = TfidfVectorizer(max_features=100)
    vectorized_data = vectorizer.fit_transform(data)
    return vectorized_data
```

Deployment and Usage:

- After development, deploy the Streamlit application to a server or use it locally to provide a user-friendly interface for predicting job posting authenticity.
- Users can interact with the application by entering different job descriptions and instantly seeing the predictions from your machine learning models.



7. SOFTWARE TESTING

Software testing is the process of testing before the real software is run through to the end. Ensuring that the expected output is free from mistakes and faults is the primary goal of software testing. In the realm of software development and machine learning projects, rigorous testing methodologies are essential to ensure the reliability, accuracy, and functionality of the system. This project, focused on classifying job postings as real or fake using machine learning models, implemented several testing strategies to validate its components and overall performance.

7.1 Unit Testing:

Unit testing is the first stage to test a module by testing each individual unit of the module. Each module or method of a procedure is tested to get the expected output. It helps to fix the defects and errors of the module of each unit.

Unit testing involved the verification of individual components and functions within the machine learning pipeline. Each function responsible for data preprocessing, feature extraction, and model training underwent thorough testing using Python's testing frameworks like unittest or pytest. For example, unit tests validated the correctness of data cleaning routines, ensuring that missing values were handled appropriately, and categorical variables were encoded correctly. Text preprocessing functions were scrutinized to confirm consistent tokenization, stop-word removal, and lemmatization across various job descriptions. By automating these tests, the team detected and resolved errors early in the development cycle, thereby enhancing code quality and minimizing debugging efforts.

7.2 Integration Testing:

After unit testing, integration testing is carried out. As a result, the output of unit testing serves as the input for integrated testing. The functional requirements are taken as input. Individual units of code in a module are gathered or integrated for testing in this method.

Integration testing focused on evaluating the interaction and interoperability of different modules within the machine learning pipeline. This phase ensured that data flowed seamlessly from preprocessing through feature engineering to model training and evaluation. Integration tests verified data consistency, pipeline integrity, and the proper implementation of dependencies. For instance, they validated that features extracted from text data, such as job descriptions and company profiles, were appropriately combined with categorical features like employment type or required education level. By simulating end-to-end data flows and processing scenarios, integration testing identified potential bottlenecks or inconsistencies, enabling optimizations to improve overall pipeline efficiency.

7.3 Acceptance Testing:

Acceptance testing is conducted with the results of system testing as a starting point. This is done to verify that the expected and assumed requirements match.

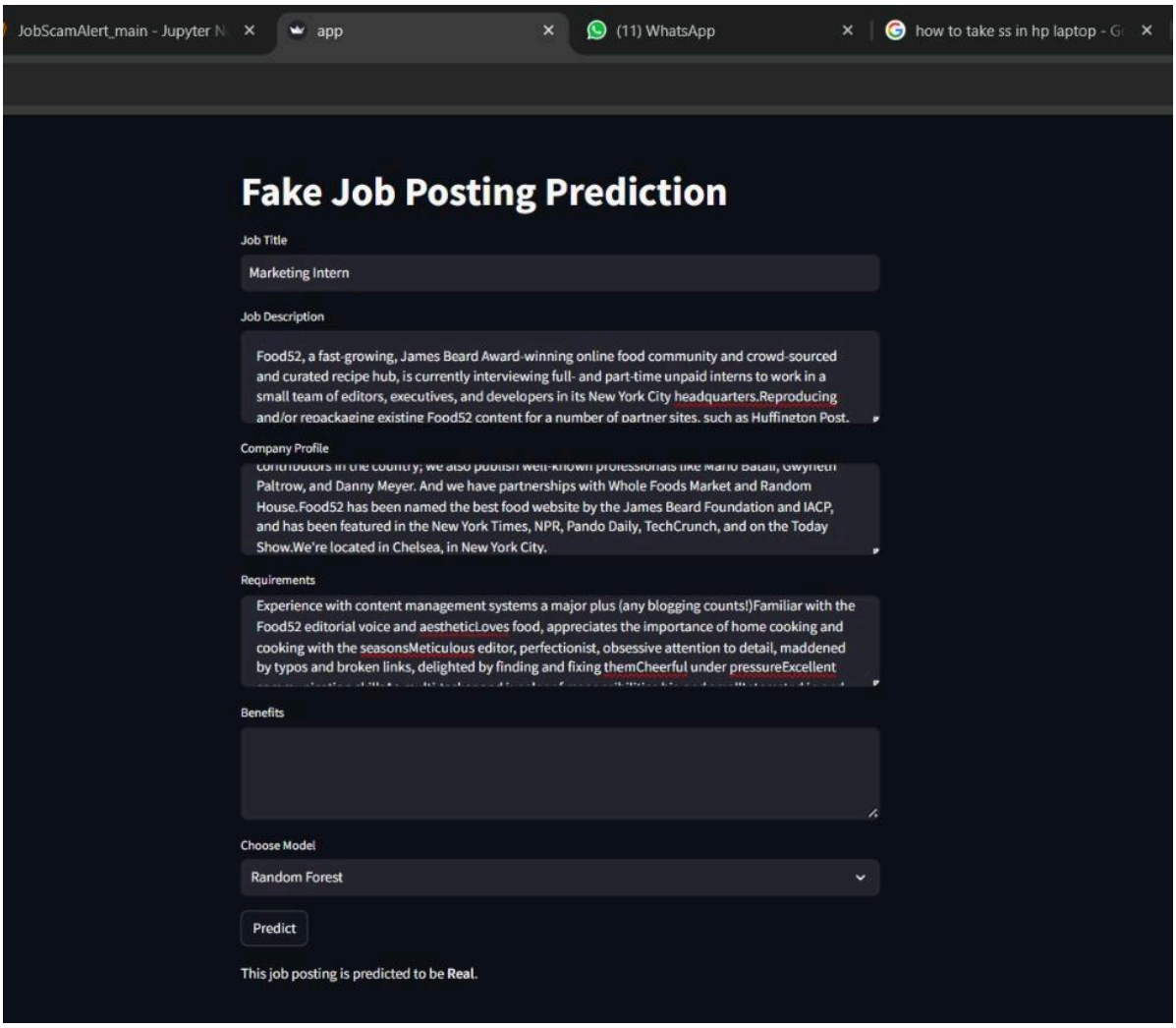
Acceptance testing aimed to validate whether the machine learning models met project requirements and user expectations. This phase involved running trained models on diverse datasets comprising job postings to evaluate their classification performance. Metrics such as accuracy, precision, recall, and F1-score were computed to assess how well the models distinguished between genuine and fraudulent job postings. Acceptance tests also scrutinized the models' generalization capabilities across different geographical regions, industries, and job categories. Stakeholders reviewed these metrics to gauge model effectiveness and determine readiness for deployment in real-world applications. Adjustments were made based on insights gleaned from acceptance testing to optimize model performance and ensure alignment with project objectives.

7.4 Testing on our System:

Testing on the project's system encompassed a comprehensive evaluation of the entire machine learning pipeline, incorporating findings from unit tests, integration tests, and acceptance tests.

This holistic approach provided a thorough assessment of model robustness, reliability, and scalability. Evaluation metrics derived from testing, such as confusion matrices, ROC curves, and precision-recall curves, offered deeper insights into model behavior and performance characteristics. The team leveraged these metrics to identify areas for improvement, refine model parameters, and enhance predictive accuracy. Moreover, testing on the project's system validated the models' ability to handle varying data inputs, adapt to evolving trends, and maintain consistent performance over time.

8. RESULTS


JobScamAlert_main - Jupyter N
app
(11) WhatsApp
how to take ss in hp laptop - G
Fake Job Posting Prediction
Job Title
Marketing Intern
Job Description
Food52, a fast-growing, James Beard Award-winning online food community and crowd-sourced and curated recipe hub, is currently interviewing full- and part-time unpaid interns to work in a small team of editors, executives, and developers in its New York City headquarters.Reproducing and/or repackaging existing Food52 content for a number of partner sites, such as Huffington Post.
Company Profile
Paltrow, and Danny Meyer. And we have partnerships with Whole Foods Market and Random House.Food52 has been named the best food website by the James Beard Foundation and IACP, and has been featured in the New York Times, NPR, Pando Daily, TechCrunch, and on the Today Show.We're located in Chelsea, in New York City.
Requirements
Experience with content management systems a major plus (any blogging counts!)Familiar with the Food52 editorial voice and aestheticLoves food, appreciates the importance of home cooking and cooking with the seasonsMeticulous editor, perfectionist, obsessive attention to detail, maddened by typos and broken links, delighted by finding and fixing themCheerful under pressureExcellent
Benefits
Choose Model
Random Forest
Predict
This job posting is predicted to be Real.

9. CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, this project successfully developed a robust machine learning model to detect fraudulent job postings using a combination of text and numeric data. By employing models like Random Forest and Support Vector Machine (SVM), the project demonstrated the potential of various ML techniques in addressing the issue of employment scams. The results showed that the Random Forest model outperformed the SVM model, achieving an accuracy of 97.22%, thus becoming the chosen model for this task. Despite these achievements, there is still room for improvement. Future enhancements could include integrating more advanced NLP techniques, such as transformers and BERT, to improve the detection accuracy further. Additionally, incorporating a larger and more diverse dataset could help the model generalize better across different types of job postings. Implementing real-time detection and continuous model updating can also enhance the system's adaptability to new scam patterns. Overall, this project provides a solid foundation for combating employment scams and protecting job seekers, with opportunities for further refinement and innovation.

10. BIBLIOGRAPHY

[1]: Fake Job Prediction using Sequential Network ; Devsmit Ranparia; Shaily Kumari; Ashish Sahani

[2]: Fake Job Prediction Using Machine Learning V Anbarasu1* , Dr. S. Selvakani2 , Mrs. K. Vasumathi3

[3]: Online Job Posting Authenticity Prediction using Machine and Deep Learning Techniques Simiscuka, Anderson Augusto

[4]: FAKE JOB DETECTION USING MACHINE LEARNING APPROACH K. Swetha, M. Tharun Reddy, K. Sravani, B. Subramanyam

[5]https://www.ieeexpert.com/python-projects/fake-job-post-detection-using-machine-learning/

[6] https://youtu.be/hTS9MhGDlfU?si=nOKYRAREtHvklXS2

[8]: Developing A Model to Detect Fraudulent Job Postings: Fake vs. Real T.Nandini1 ,S.Gnana Chandrika2 , P.Mounika3 , V.Sandeeep Kumar4 UG students