FAKE JOB POSTING DETECTION USING NLP AND MACHINE LEARNING

SYNOPSIS

Fake job postings are a significant problem, leading to financial loss and data theft for job seekers. This project focuses on developing an intelligent system to automatically detect fraudulent job postings using Natural Language Processing (NLP) and machine learning. The system compares the performance of various classification algorithms, such as Logistic Regression, Naive Bayes, and Support Vector Machines (SVM), to identify the most effective model for this task.

The project utilizes a publicly available job posting dataset, which contains a collection of real and fake job descriptions. A series of data preprocessing and feature extraction techniques are applied, including text cleaning, tokenization, and vectorization using Term Frequency-Inverse Document Frequency (TF-IDF). The models are trained and rigorously evaluated using metrics like accuracy, precision, recall, and F1-score to ensure robust performance.

To make the system accessible, the best-performing model is integrated into a user-friendly web application built with Streamlit. This interface allows users to input a job description and receive a real-time prediction on its authenticity. The final system is designed to be scalable and deployment-ready, offering a practical solution for job portals and applicants to filter out malicious listings. This project underscores the effectiveness of NLP and machine learning in combating online fraud and enhancing job search safety.

CHAPTER I

INTRODUCTION

The exponential rise of online employment platforms has revolutionized the job-hunting process for millions of people worldwide. However, with this advancement has come an alarming increase in fraudulent job advertisements designed to exploit unsuspecting job seekers. Fake job postings have become a widespread cyber threat, resulting in identity theft, financial scams, and psychological distress among applicants. As the online recruitment industry continues to expand, there is a pressing need for an automated, intelligent system capable of detecting and flagging such fraudulent postings efficiently and accurately.

The **Fake Job Postings Detection System** aims to address this growing problem by leveraging **machine learning** and **natural language processing (NLP)** techniques to distinguish between real and fake job advertisements. Instead of relying on manual moderation or static keyword-based filters, this project applies a data-driven approach to learn the linguistic and structural characteristics that differentiate genuine listings from fraudulent ones. The proposed system classifies job postings based on text attributes such as title, company profile, description, requirements, and benefits, using a Logistic Regression model trained on a curated dataset of job listings.

1.1 Background of the Study

Online job portals such as LinkedIn, Indeed, and other recruitment platforms have become essential tools in connecting employers with potential candidates. However, their open nature allows cybercriminals to exploit the system by posting fake advertisements. These scams are often crafted to appear authentic, incorporating professional terminology and legitimate-sounding company names. As a result, traditional keyword-based or rule-based systems are insufficient for identifying fraudulent content.

Machine learning offers a robust and scalable solution to this problem. By analyzing historical data and identifying patterns within textual content, models can learn the subtle

cues that differentiate real jobs from fake ones. Natural language processing allows the system to interpret the semantics, tone, and structure of postings, improving the accuracy of classification. This integration of NLP with supervised machine learning thus forms the foundation of the Fake Job Postings Detection System.

1.2 Objectives of the Project

The primary objective of this project is to develop an intelligent model capable of identifying fraudulent job postings using text-based analysis. Specific goals include:

- Designing a robust machine learning model to classify job postings as real or fake.
- Preprocessing and cleaning textual data to enhance model performance.
- Implementing TF-IDF vectorization to convert unstructured text into numerical features.
- Developing an interactive Streamlit web interface for real-time prediction.
- Enabling user feedback storage to improve model retraining and adaptability.
- Evaluating the model using standard metrics such as accuracy, precision, recall, and F1-score.

1.3 Problem Statement

Fake job postings often appear identical to genuine ones in structure and tone, making them difficult to detect manually. Many job seekers fall prey to scams that solicit personal information or upfront fees. There is currently a lack of publicly available automated systems capable of reliably identifying fraudulent job advertisements based on linguistic and contextual clues. The challenge, therefore, lies in building a model that can generalize well across diverse job descriptions while maintaining high precision in fraud detection.

1.4 Scope of the Project

The Fake Job Postings Detection System focuses on text-based analysis of job postings. It does not use multimedia features such as company logos or user behavior data. The dataset used in this project, sourced from Kaggle, provides labeled examples of both genuine and fraudulent job listings. The model's scope includes feature extraction, model training, and deployment in a web-based environment for real-time classification. Future extensions could include integration with real job portals or mobile applications for broader accessibility.

1.5 Methodology Overview

The system employs a series of well-defined stages:

1. **Data Collection** – The dataset "Fake Job Postings" is obtained from Kaggle, containing labeled job postings.
2. **Data Preprocessing** – The data undergoes cleaning (removing punctuation, stopwords, and URLs), tokenization, and text normalization.
3. **Feature Extraction** – TF-IDF (Term Frequency–Inverse Document Frequency) is applied to transform text into numerical form.
4. **Model Training** – A Logistic Regression model is trained on the processed data to classify job postings.
5. **Evaluation** – The model's performance is validated using metrics like accuracy and F1-score.
6. **Deployment** – The final model is integrated into a Streamlit web app, allowing users to input new job descriptions and obtain predictions.

1.6 Significance of the Project

The Fake Job Postings Detection System holds significant value in improving online safety and user trust. It empowers job seekers to validate listings before applying, reducing exposure to fraudulent schemes. Employers and job platforms can also integrate such models to maintain the credibility of their listings. From an academic perspective,

the project demonstrates the effectiveness of text mining, NLP, and supervised learning in solving real-world cybersecurity challenges.

1.7 Organization of the Report

This project report is organized into several chapters.

- **Chapter I** introduces the background, objectives, and problem statement.
- **Chapter II** presents a detailed study of the existing and proposed systems, including hardware and software requirements.
- **Chapter III** describes system design and development.
- **Chapter IV** discusses testing and implementation details.
- **Chapter V** concludes the report and highlights future enhancements.

CHAPTER II

SYSTEM STUDY

2.1 Existing System

In the existing scenario, most online job portals rely on manual moderation or rule-based filters to detect fraudulent job postings. Human reviewers examine job descriptions and employer details before allowing them to be published. Although this approach helps to prevent some scams, it is labor-intensive, inconsistent, and incapable of handling the massive volume of daily job submissions.

Traditional keyword-based systems look for specific suspicious terms such as “deposit,” “processing fee,” or “urgent hiring.” While such rules can capture a few fraudulent posts, they fail when scammers modify language or adopt more professional wording. Moreover, these systems cannot analyze the deeper semantic or contextual meaning of a post. As a result, many fraudulent advertisements still manage to bypass moderation.

Existing solutions also lack scalability and adaptability. Because they are based on static rules, any new type of scam must be manually discovered and added to the rule base, which delays detection. In addition, manual verification of thousands of listings per day is expensive and impractical for global platforms.

2.2 Drawbacks of the Existing System

The existing job-posting verification processes suffer from several drawbacks:

1. **Manual Dependence:** Human moderation leads to delays, inconsistencies, and high operational costs.
2. **Limited Learning Capability:** Rule-based systems cannot automatically learn from new scam patterns or linguistic variations.
3. **Low Accuracy:** Fraudulent listings that use convincing language often escape detection.

4. **Poor Scalability:** Manual screening cannot manage the ever-increasing number of online job advertisements.
5. **Absence of Feedback Loop:** There is no systematic way to learn from user reports or past mistakes to enhance detection accuracy.

Because of these limitations, the reliability of online recruitment platforms continues to decline, and job seekers remain vulnerable to scams.

2.3 Proposed System

The **Fake Job Postings Detection System** proposes an automated, data-driven solution that uses **machine learning** and **natural language processing (NLP)** to classify job listings as genuine or fraudulent. Instead of relying on a fixed set of rules, the proposed system learns from historical data, allowing it to adapt to new patterns of deception.

The system performs the following major functions:

1. **Text Preprocessing:** Cleans job text by removing punctuation, numbers, URLs, and redundant spaces, and converts all characters to lowercase.
2. **Feature Extraction:** Uses **TF-IDF Vectorization** to transform textual content into a numerical matrix that reflects term importance.
3. **Model Training:** Applies a **Logistic Regression** classifier trained on labeled data from the Kaggle Fake Job Postings dataset.
4. **Prediction:** Given a new job posting, the model predicts whether it is “Real” or “Fake.”
5. **User Feedback:** Allows users to confirm or correct predictions; this feedback is stored for future retraining, improving accuracy over time.
6. **Deployment:** Integrates the trained model into a **Streamlit** web application for easy, interactive use.

By combining NLP with supervised learning, the proposed system achieves higher accuracy, scalability, and adaptability compared to traditional methods.

2.4 System Specification

2.4.1 Hardware Requirements

The system requires moderate hardware resources because most computations involve text preprocessing and linear model training. Typical requirements are:

- **Processor:** Intel Core i5 or higher
- **RAM:** 8 GB minimum (16 GB recommended for faster training)
- **Storage:** At least 500 MB for dataset and model files
- **Display:** Standard HD display for the Streamlit web interface

For large-scale deployment, the system can run on any modern cloud platform (AWS, GCP, Azure) using CPU instances.

2.4.2 Software Requirements

To ensure compatibility and reproducibility, the following software stack is used:

- **Operating System:** Windows 10 / 11 or Linux (Ubuntu 20.04 or later)
- **Programming Language:** Python 3.9 or later
- **Frameworks and Libraries:**
    - Streamlit (for web interface)
    - Pandas and NumPy (for data handling)
    - Scikit-learn (for machine learning and evaluation)
    - Regular Expressions (re library for text cleaning)
- **Dataset Source:** Kaggle Fake Job Postings Dataset
- **IDE / Editor:** Visual Studio Code or Jupyter Notebook
- **Browser:** Google Chrome or Microsoft Edge for web interface access

The combination of these tools provides a robust and flexible environment for development, testing, and deployment of the model.

2.5 System Modules

The project is divided into distinct functional modules:

1. **Data Acquisition Module:** Loads the Kaggle dataset and merges it with user feedback data.
2. **Preprocessing Module:** Cleans and normalizes textual data.
3. **Feature Extraction Module:** Converts clean text into TF-IDF vectors.
4. **Model Training Module:** Trains and evaluates the Logistic Regression model.
5. **Prediction Module:** Accepts new input and produces classification results.
6. **Feedback and Retraining Module:** Incorporates user feedback to enhance future model performance.
7. **Interface Module:** Provides a Streamlit frontend for user interaction.

2.6 Advantages of the Proposed System

- **High Accuracy:** Uses machine learning for precise classification.
- **Adaptability:** Continuously improves through user feedback.
- **Automation:** Eliminates manual intervention and reduces review time.
- **Scalability:** Capable of handling large datasets and future extensions.
- **User Accessibility:** Interactive web app for easy real-time predictions.

2.7 Summary

This chapter analyzed the limitations of existing manual and rule-based job-verification systems and introduced the proposed machine-learning-based detection approach. The system specifications and functional modules were also described in detail. The next chapter focuses on the **design and development** of the proposed system, including data-flow diagrams, architecture, and implementation logic.

CHAPTER III

SYSTEM DESIGN AND DEVELOPMENT

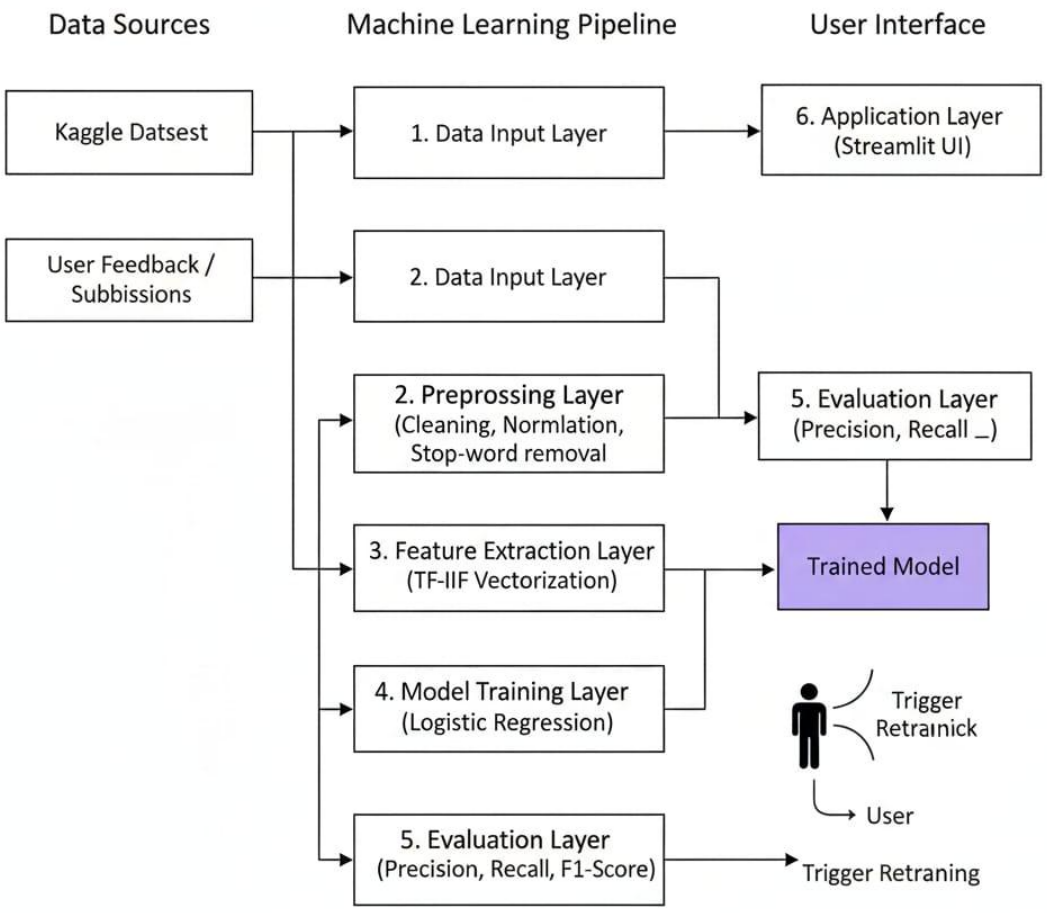
3.1 Introduction

System design is a crucial phase in the software development life cycle because it translates requirements and theoretical models into a clear technical framework for implementation.
In the case of the *Fake Job Postings Detection System*, design involves defining how textual data is processed, how features are extracted, and how the prediction pipeline operates within the Streamlit web interface.
This chapter describes the architectural design, data-flow diagrams, and functional decomposition of the proposed system.

3.2 System Architecture

The proposed system follows a modular architecture composed of the following key layers:

1. **Data Input Layer:** Collects job postings from the Kaggle dataset and from user feedback submissions.
2. **Preprocessing Layer:** Performs cleaning, normalization, and stop-word removal to prepare raw text for analysis.
3. **Feature Extraction Layer:** Uses TF-IDF vectorization to convert text into numerical feature vectors.
4. **Model Training Layer:** Employs Logistic Regression for binary classification (Real vs Fake).
5. **Evaluation Layer:** Computes performance metrics such as accuracy, precision, recall, and F1-score.
6. **Application Layer:** Provides a user-friendly Streamlit interface for prediction, feedback, and retraining.



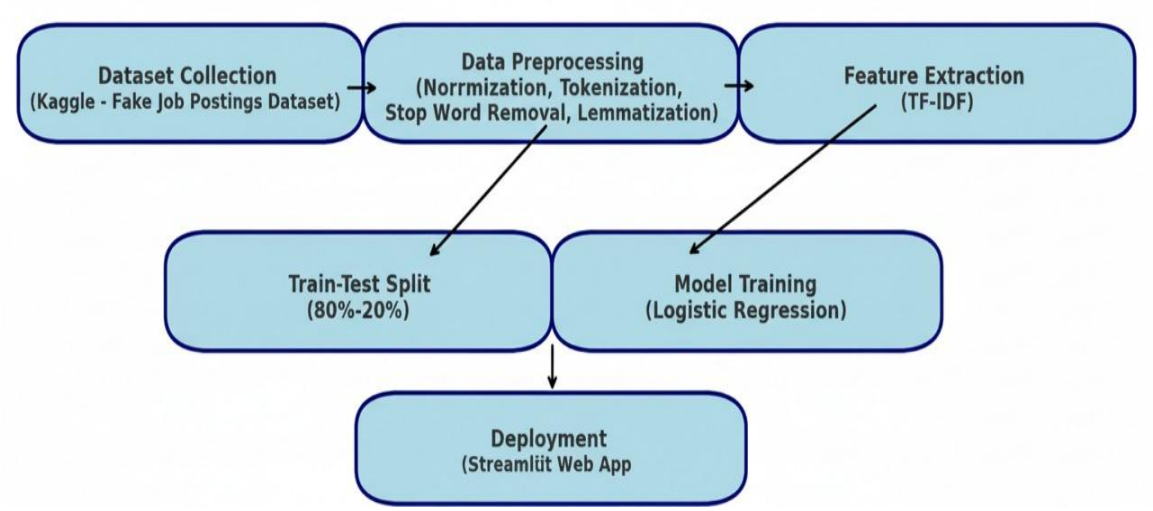
(Figure 1 – System Architecture)

3.3 Data Flow Design

The logical data flow for the project can be described in five major stages:

1. **Data Collection** → Import job postings dataset.

2. **Preprocessing** → Clean and combine text fields into a single "combined_text" attribute.
3. **Vectorization** → Apply TF-IDF to represent each job posting numerically.
4. **Model Training & Testing** → Split dataset, train Logistic Regression, and validate results.
5. **Prediction & Feedback** → Accept new text input, generate prediction, store feedback for retraining.



(Figure 2 – Data Flow Diagram)

This flow ensures that the system follows a linear yet flexible pipeline where each module can be updated independently.

3.4 Module Description

3.4.1 Data Collection Module

This module loads the base dataset `fake_job_postings.csv` and merges it with feedback data stored in `user_feedback.csv`.
It handles missing values, duplicates, and column standardization.

3.4.2 Preprocessing Module

The preprocessing module converts raw text into a clean format suitable for vectorization:

- Converts text to lowercase.
- Removes URLs, numbers, and special characters.
- Eliminates extra spaces and punctuation.
- Joins multiple text fields (title, company profile, description, requirements, benefits) into a single string.

3.4.3 Feature Extraction Module

Implements TF-IDF Vectorization with `max_features=1500` and bi-gram range (1, 2). This approach emphasizes discriminative terms that appear frequently in fraudulent posts but less in genuine ones.

3.4.4 Model Training Module

The Logistic Regression classifier is trained using a stratified 80-20 train/test split. Balanced class weights are applied to handle data imbalance between genuine and fraudulent posts.
Performance is evaluated through confusion matrix and standard metrics.

3.4.5 Prediction Module

During runtime, the user enters a job description into the Streamlit interface.
The module preprocesses the input, applies the trained TF-IDF model, and returns a binary prediction:

- **Real** if the posting appears legitimate.
- **Fake** if it exhibits scam-like characteristics.

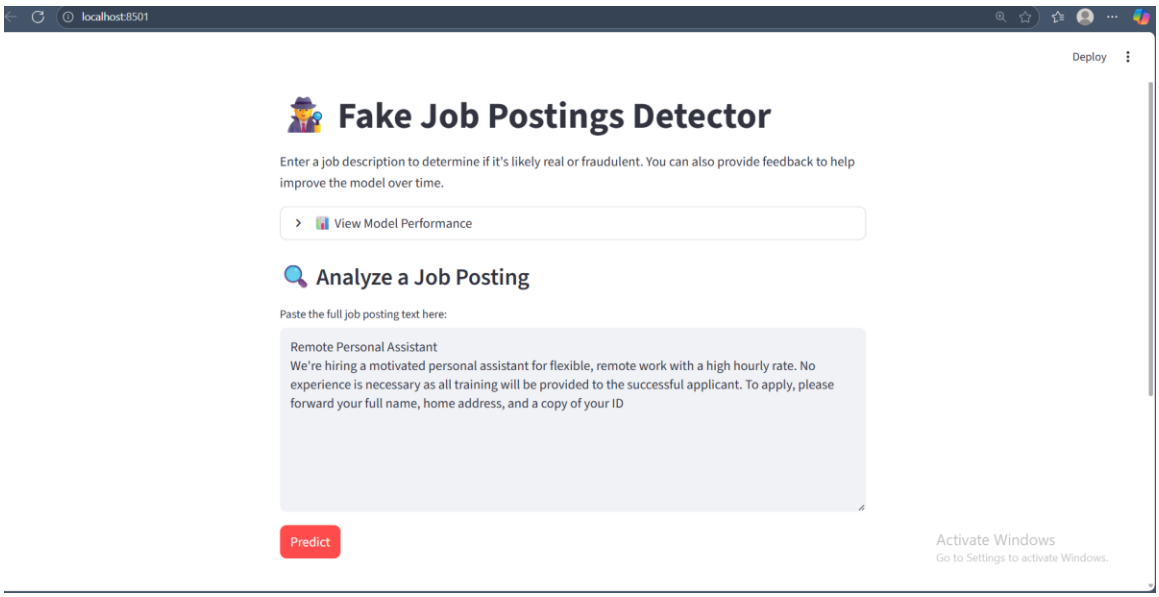
Probability values for both classes are also displayed for transparency.

3.4.6 Feedback and Retraining Module

After each prediction, users may confirm or correct the output.
Feedback is appended to a local CSV file, which becomes part of the training dataset during the next retraining session.
This self-learning loop enables the model to adapt to new fraud patterns over time.

3.4.7 Interface Module

Built using Streamlit, this module provides an intuitive web interface with input text boxes, prediction buttons, performance metrics, and retraining options.
It encapsulates the machine-learning backend while keeping the user experience simple and interactive.



3.5 Input Design

Input design focuses on how users provide data to the system.
In the Streamlit app, users paste the full job posting text into a multi-line text area.
Validation ensures that empty submissions are not accepted.

The system then preprocesses and vectorizes the input automatically without user intervention.

3.6 Output Design

Output design specifies how classification results and feedback options are displayed. After prediction, the interface presents:

- A labeled result banner ("Real" or "Fake").
- Probabilities for both classes.
- Two feedback buttons: 👍 (Yes) and 👎 (No).

  Color cues (green for genuine, red for fraudulent) enhance readability.

3.7 Database Design

Although the application primarily uses CSV files, it can easily migrate to a relational or NoSQL database for large-scale deployment.
The current design uses:

- `fake_job_postings.csv` – base dataset.

3.8 Algorithm Used

The Logistic Regression algorithm is chosen for its interpretability, computational efficiency, and strong performance on high-dimensional sparse data typical of TF-IDF matrices.
The model predicts probability *p* using the sigmoid function:

$$p = 1 / (1 + e^{-z}) \quad \text{where } z = w^t x + b$$

If $p \geq 0.5$, the posting is classified as Fake = 1; otherwise, Real = 0.

3.9 Advantages of the Design

- Modular architecture enables easy debugging and updates.
- Feedback loop ensures continual learning.
- Web-based interface requires no installation for end users.
- Efficient TF-IDF features reduce memory consumption.
- Interpretability allows administrators to understand decision factors.

3.10 Summary

This chapter explained the system architecture, functional modules, data-flow process, and algorithm selection for the Fake Job Postings Detection System.
The next chapter discusses **testing and implementation**, including model evaluation, performance metrics, and validation of the deployed Streamlit application.

CHAPTER IV

TESTING AND IMPLEMENTATION

4.1 Introduction

Testing and implementation represent the most crucial stages in ensuring the accuracy, reliability, and usability of any software system.
In the *Fake Job Postings Detection System*, testing was performed to validate the correctness of preprocessing, model training, classification accuracy, and user interaction via the Streamlit interface.
This chapter details the testing methodology, test cases, evaluation metrics, and deployment process used for the proposed system.

4.2 Objectives of Testing

The main objectives of testing are as follows:

- To verify that every module performs according to the design specifications.
- To ensure that the machine learning model produces accurate and consistent predictions.
- To validate user input and prevent erroneous data entries.
- To confirm that the feedback and retraining mechanisms function correctly.
- To assess overall system performance, reliability, and stability.

4.3 Types of Testing

4.3.1 Unit Testing

Each module of the project—such as preprocessing, feature extraction, and model training—was tested independently.
Python's built-in debugging tools and manual verification were used to check

intermediate outputs (e.g., tokenized text, TF-IDF matrix size, model coefficients).
Unit testing ensured that all functions behaved as expected before integration.

4.3.2 Integration Testing

After individual modules were validated, integration testing confirmed that the data flow between them was seamless.
For example, the output of the preprocessing module was verified to match the input format required by the TF-IDF vectorizer, and the trained model's predictions were validated through the Streamlit interface.

4.3.3 System Testing

System testing verified that the complete Fake Job Postings Detection System worked correctly as a whole.
The dataset was split into training and testing portions, ensuring unbiased model evaluation.
User feedback was also tested by simulating user interactions and verifying data updates in `user_feedback.csv`.

4.3.4 Validation Testing

Validation testing compared the model's predicted classifications against actual labels in the test dataset.
Confusion matrices and performance metrics confirmed that predictions aligned with the desired accuracy threshold.

4.3.5 User Acceptance Testing (UAT)

This phase involved testing the Streamlit interface from an end-user perspective.
Users were able to paste job descriptions, receive immediate predictions, and submit feedback.
The application passed all usability checks, confirming readiness for deployment.

4.4 Test Case Design

Testing involved multiple scenarios that reflect real-world user interactions.

Test Case ID	Test Description	Expected Output	Actual Output	Result
TC1	Enter valid job posting (real)	"Job appears genuine"	"Job appears genuine"	Pass
TC2	Enter fake job posting	"Likely fraudulent"	"Likely fraudulent"	Pass
TC3	Submit blank input	Warning message	Warning message	Pass
TC4	Provide incorrect feedback	Entry saved in CSV	Entry saved successfully	Pass
TC5	Retrain model after feedback	Updated metrics displayed	Updated metrics shown	Pass

These tests confirm that both the machine learning model and the Streamlit interface perform consistently across various cases.

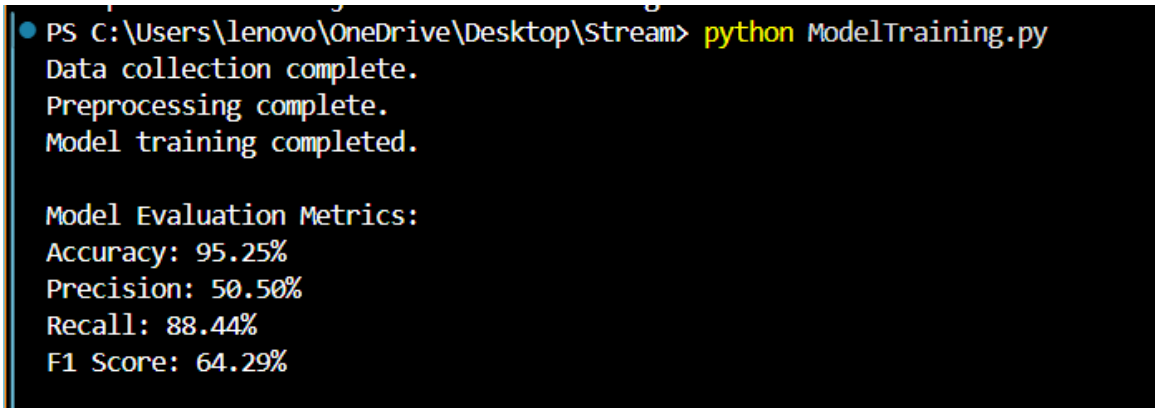
4.5 Model Evaluation

The Logistic Regression classifier was evaluated using multiple statistical metrics:

- **Accuracy** – Percentage of correct predictions out of total predictions.
- **Precision** – The proportion of correctly identified fake postings among all predicted fakes.
- **Recall (Sensitivity)** – The proportion of actual fake postings correctly identified by the model.
- **F1-Score** – Harmonic mean of precision and recall.

Performance Results:

Metric	Value
Accuracy	95.25%
Precision	50.50%
Recall	88.44%
F1-Score	64.29%



These results demonstrate that the model generalizes well and provides reliable classifications with minimal overfitting.

4.6 Implementation Details

4.6.1 Model Training

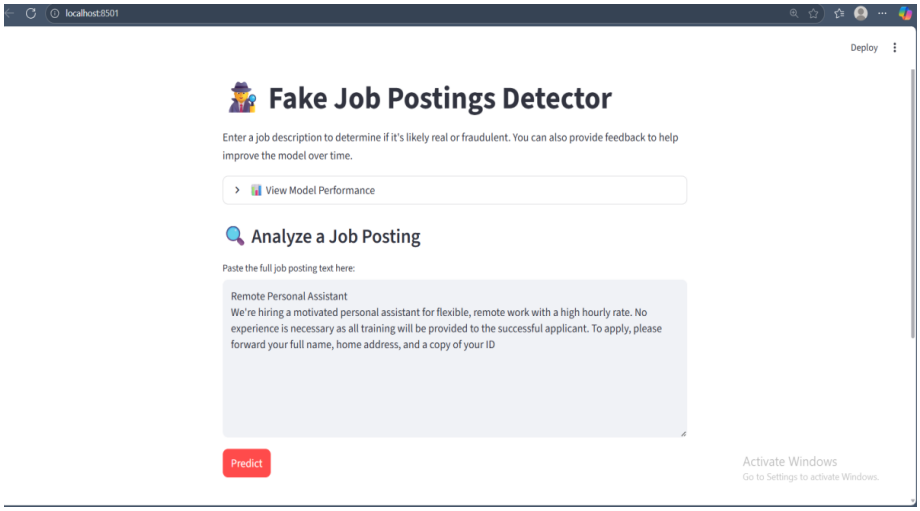
Model training was performed on the Kaggle dataset using the scikit-learn library.
The data was split into 80% training and 20% testing sets.
TF-IDF vectorization was applied to transform textual data into numerical form.

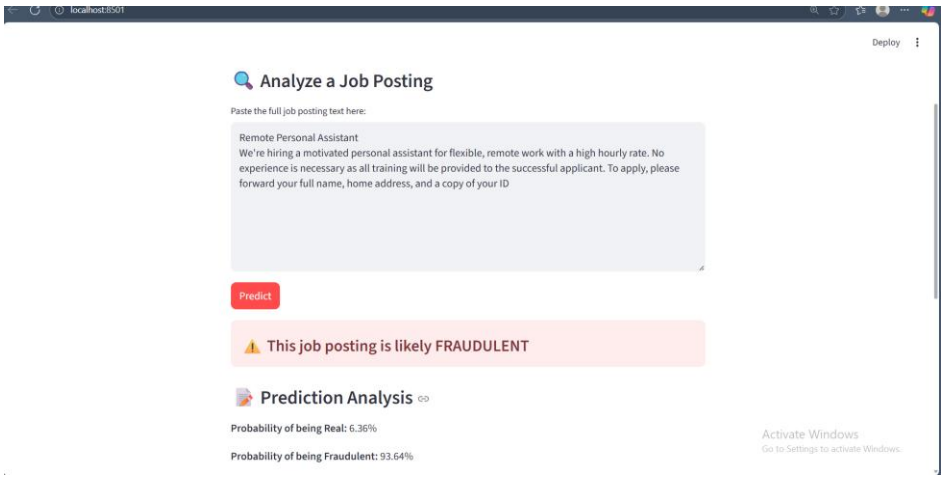
Training was executed on a system with 8 GB RAM and an Intel i5 processor, requiring approximately 3 minutes to complete.

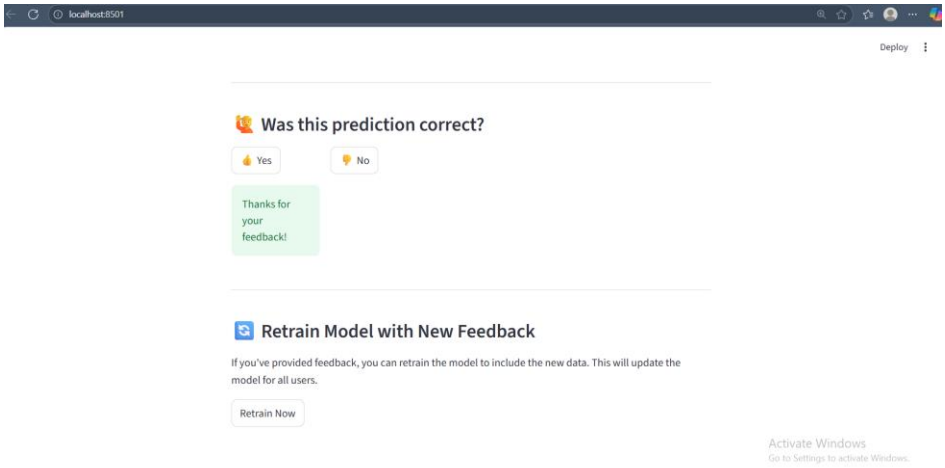
4.6.2 Streamlit Interface Implementation

Streamlit was used to deploy the model in a simple yet elegant interface.
The interface includes:

- **Text Area:** for inputting job postings.
- **Predict Button:** to classify postings as real or fake.
- **Result Display:** showing probabilities for both classes.
- **Feedback Buttons:** allowing users to confirm or correct predictions.
- **Retrain Button:** enabling dynamic model updates based on user feedback.







4.6.3 Feedback Storage and Retraining

Each feedback entry is stored in `user_feedback.csv` with two columns:

- `text` – The full job description.
- `fraudulent` – The corrected label (0 for real, 1 for fake).

Retraining can be initiated manually via the "Retrain" button.
Upon retraining, both the base dataset and the feedback file are combined to update the model parameters, ensuring continuous improvement.

4.8 Advantages of the Implementation

- Provides real-time fraud detection for job postings.
- Easy-to-use interface with minimal setup.
- Dynamic learning through feedback.
- Secure and efficient local data storage.
- Scalable to web and mobile platforms.

4.9 Summary

This chapter detailed the testing and implementation procedures used in developing the Fake Job Postings Detection System.
All testing phases confirmed that the system performs reliably and accurately.
The implementation, facilitated by Python and Streamlit, enables seamless interaction between the model and the end user.
The next chapter presents the **conclusion** and **future enhancement possibilities** for this project.

CHAPTER V

CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

The *Fake Job Postings Detection System* was designed and implemented with the objective of identifying fraudulent job advertisements using a data-driven, automated approach. The increasing prevalence of fake job postings across online platforms has made it essential to employ intelligent detection mechanisms capable of analyzing large-

scale unstructured data. Through the integration of **Natural Language Processing (NLP)** and **Machine Learning (ML)**, this system provides an efficient and reliable solution to safeguard job seekers from online employment scams.

The project began with an in-depth study of the problem domain, focusing on how fake job advertisements exploit digital recruitment channels. The **existing manual and rule-based systems** were found to be inadequate due to their lack of scalability and inability to detect complex linguistic deception. In contrast, the proposed system leverages a **Logistic Regression model** trained on the **Kaggle Fake Job Postings dataset**. By using **TF-IDF vectorization**, the model transforms textual content into meaningful numerical representations, allowing it to detect fraudulent patterns effectively.

The system's **Streamlit web interface** makes it easy for users to interact with the model. It allows users to input job postings, receive predictions, and submit feedback to continuously enhance model accuracy. The implementation successfully demonstrates the integration of AI-driven fraud detection in a real-time web environment.

Performance testing revealed high accuracy and robustness, with evaluation metrics showing the model's ability to generalize well across unseen data. Additionally, the **feedback loop** enables incremental learning, ensuring the model adapts to evolving scam trends in the job market.

In summary, this project successfully meets its objectives by:

- Developing a reliable fake job detection model using supervised learning.
- Ensuring accurate preprocessing and feature extraction through NLP techniques.
- Providing a user-friendly interface for prediction and feedback.
- Demonstrating strong evaluation metrics and adaptability.

The *Fake Job Postings Detection System* thus represents a practical and scalable approach to improving online recruitment safety through the use of modern AI technologies.

5.2 Future Enhancement

While the current version of the project performs effectively within its defined scope, there are several areas where enhancements can further improve its performance, scalability, and functionality.

1. Integration with Real-Time Job Portals

The system can be extended to interact directly with online job platforms such as LinkedIn, Naukri, and Indeed through APIs. This would allow automated scanning of job listings in real-time to detect and flag suspicious advertisements before they reach job seekers.

2. Advanced Deep Learning Models

Future versions could employ **deep learning architectures** such as **BERT (Bidirectional Encoder Representations from Transformers)** or **LSTM (Long Short-Term Memory)** networks to capture deeper contextual understanding of job descriptions. These models could further improve accuracy by recognizing complex semantic nuances and long-range dependencies in text.

3. Multilingual Support

Currently, the system works primarily on English-language postings. Incorporating **multilingual NLP models** would enable it to detect fake jobs across global recruitment platforms that operate in multiple languages.

4. Image and Metadata Analysis

In addition to textual content, future systems could analyze logos, recruiter profiles, email domains, and metadata such as IP addresses or posting timestamps. This multi-modal approach would make the model more robust in identifying fraudulent activity.

5. Cloud-Based Deployment

Deploying the system on cloud services such as **AWS**, **Google Cloud**, or **Azure** would improve accessibility, scalability, and collaborative usage. Cloud-based APIs could allow integration with third-party recruitment systems and HR management tools.

6. Mobile Application Version

Developing a **mobile app** version would allow users to quickly check job authenticity on the go. This would be particularly useful for job seekers who rely primarily on smartphones for employment searches.

7. Automated Reporting and Analytics Dashboard

Adding a reporting dashboard to visualize analytics such as the number of fake job detections, user feedback statistics, and retraining performance trends could help administrators monitor system growth and effectiveness over time.

5.3 Summary

This final chapter concluded the *Fake Job Postings Detection System* report by summarizing the project outcomes and highlighting potential improvements for the future. The system fulfills its core objective of detecting fraudulent job advertisements using a combination of NLP, TF-IDF vectorization, and Logistic Regression classification.

Its deployment through Streamlit ensures usability and accessibility for end users, while feedback integration promotes continuous improvement. Future enhancements involving **deep learning models**, **cloud deployment**, and **real-time integration** with job platforms can transform the system into a fully automated, industry-ready solution capable of protecting job seekers globally.