DEEP LEARNING–BASED FAKE JOB POST DETECTION ON SOCIAL MEDIA

Name: Hira Arif

Date: Novemeber 2025

Program: Buildables DataScience Fellowship 1

Abstract

Online job platforms and social media recruitment have become increasingly popular, but they also expose users to fraudulent job postings. These fake listings often attempt to extract personal information or scam applicants financially. This project develops a deep learning–based system to automatically detect fake job postings using both textual and metadata features. The approach combines traditional machine learning with deep learning techniques, including **TF-IDF vectorization** with **Logistic Regression** and a **Bi-LSTM network**. The dataset consists of approximately **17,880** job postings labelled as real or fake, containing rich textual content in fields such as job description, company profile, requirements, and benefits. After thorough data cleaning, preprocessing, and feature engineering, both models achieved over **97% accuracy**, demonstrating their effectiveness. A user-friendly **Streamlit web application** was deployed to provide real-time predictions, allowing users to input job postings and immediately determine their authenticity. This system not only assists job seekers and recruitment platforms but also offers insights into patterns of fraudulent postings. The project highlights the potential of **NLP** and AI in enhancing trust in online recruitment.

1. Introduction & Problem Statement

The rise of online recruitment has led to an increase in fraudulent job postings, resulting in significant financial losses, wasted time, and risks to personal information. Manual review of postings is impractical due to scale and volume. The project aims to build an AI-powered system that automatically detects fake job listings by analysing textual and metadata features.

Objective:

- Develop and compare ML and DL models for fake job post detection.
- Deploy the model via a web interface for public use.

Problem Statement:

- Fake job posts are difficult to identify manually.
- Automated detection can reduce scams and improve online trust.

2. Data Collection / Dataset Description

Dataset: Fake Job Postings Dataset (Kaggle/GitHub)
Records: ~17,880
Attributes:

- Textual: title, description, requirements, company_profile, benefits
- Categorical: location, department, salary_range
- Boolean: telecommuting, has_company_logo, has_questions
- Target: fraudulent (1 = fake, 0 = real)

The dataset is rich in textual data, making it suitable for NLP-based classification.

3. Tools & Technologies

- Language: Python 3

- Libraries: Pandas, NumPy, Scikit-learn, TensorFlow/Keras, Matplotlib, Seaborn
- Model Deployment: Streamlit Cloud
- Version Control: GitHub
- Documentation: Medium + Final Report (PDF)

4. Data Preprocessing & EDA

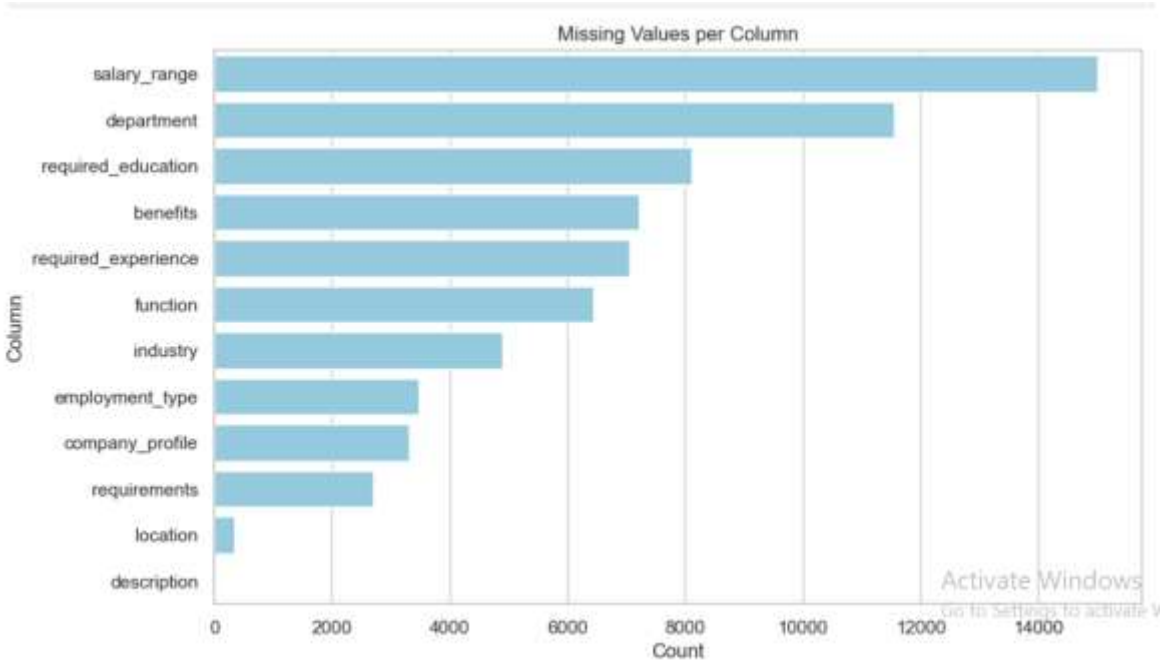
Data Exploration

1. Introduction

The first phase of the project involved performing an initial exploration of the dataset to understand its structure, content, and potential issues. The dataset, *fake_job_postings.csv*, contains textual and categorical information about job advertisements collected from online platforms. The objective of this step was to examine the data distribution, identify missing or inconsistent values, and develop an understanding of the key attributes that may help distinguish fake job postings from genuine ones.

2. Missing Value Analysis

A missing value assessment showed that some attributes—particularly salary_range, benefits, and department—contain a high proportion of null values.
A bar plot of missing values was generated to visualize which fields require cleaning or imputation in the next step. Handling these missing entries is essential to ensure that the model does not misinterpret absent information as meaningful data.



3. Class Distribution

An analysis of the fraudulent column revealed a **class imbalance**:

- Majority of postings are **real (class 0)**,
- While a smaller fraction is **fake (class 1)**.

This imbalance will be addressed later through resampling or class weighting techniques.
Visualizations using Seaborn confirmed that fake postings make up roughly **5–10%** of the total dataset, emphasizing the importance of robust evaluation metrics such as precision, recall, and F1-score in later stages.

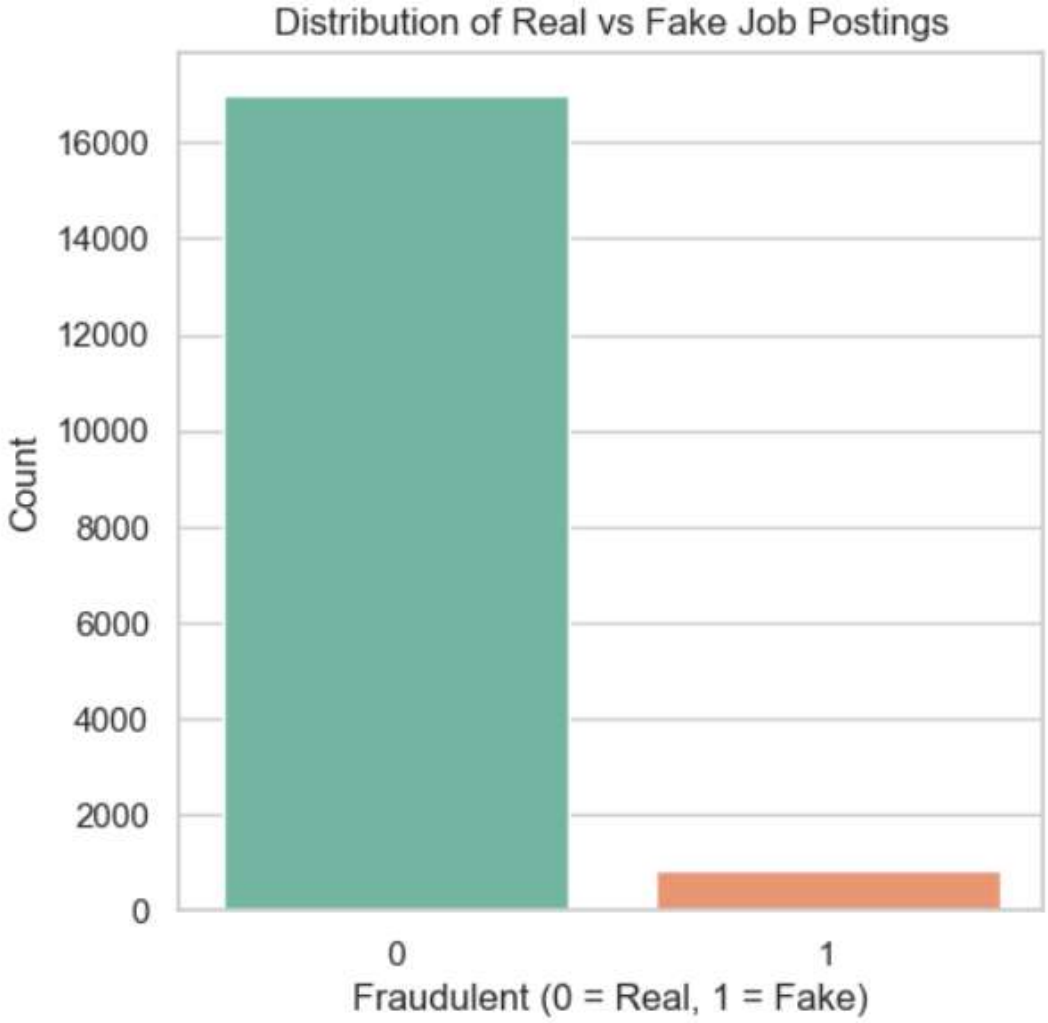
4. Textual Attributes

Columns such as company_profile, description, requirements, and benefits contain detailed free-text information that can provide strong semantic cues for detecting fraudulent behavior.
Sample inspection of these fields highlighted differences in language tone, structure, and content richness between genuine and fake postings.

5. Key Insights

- The dataset is predominantly textual, suitable for NLP modeling.
- Several columns contain missing or incomplete data.

- The target classes are imbalanced.
- Fake postings often exhibit shorter, vague, or repetitive descriptions compared to real ones.



Data Cleaning and Preprocessing

1. Handling Missing and Irrelevant Data

An initial examination revealed several columns with a high proportion of missing values, such as salary_range, employment_type, and industry. Since these fields provided minimal discriminative information for identifying fake job postings and contained excessive null entries, they were removed from the dataset.

For the remaining relevant textual columns — title, company_profile, description, requirements, and benefits — missing entries were replaced with empty strings. This preserved record count consistency while avoiding the introduction of artificial values.

2. Text Field Consolidation

To facilitate natural language processing, all text-based columns were concatenated into a single field named **text**. This unified field provides a comprehensive textual representation of each job posting, capturing both descriptive and contextual elements such as company details, job responsibilities, and offered benefits.

By merging text sources, we ensured that all linguistic cues contributing to fake job detection were consolidated in one place, simplifying downstream feature extraction and embedding generation.

3. Text Normalization and Cleaning

A custom cleaning function was applied to each entry in the text field to ensure linguistic uniformity. The preprocessing steps included:

- Removal of URLs and HTML tags
- Conversion of all characters to lowercase
- Removal of numerical characters and punctuation
- Elimination of extra whitespace

This produced a new, fully sanitized column named **clean_text**, which contains only meaningful natural language content. Such normalization reduces data noise and enhances tokenization quality during later modeling.

4. Target Label Preparation

The target variable fraudulent was verified and encoded as an integer (0 = real, 1 = fake). This encoding guarantees compatibility with classification algorithms used in deep learning pipelines.

5. Data Export

After cleaning, the refined dataset was saved as **cleaned_fake_jobs.csv** within the project's data directory. This version of the dataset will serve as the primary input for subsequent stages, including feature extraction, text vectorization, and model training.

7. Key Insights

- High missing-value columns were removed without affecting model utility.
- Textual attributes were successfully merged into a single feature-rich field.
- Cleaning functions removed noise, resulting in consistent and comparable text samples.
- The dataset is now fully ready for feature engineering and modelling.

5. Feature Engineering

1. Feature Extraction

For the first model, the cleaned text data was transformed using the **Term Frequency–Inverse Document Frequency (TF-IDF)** vectorizer.
This method assigns higher weights to unique, discriminative terms and reduces the influence of overly frequent words, allowing the model to identify key textual patterns associated with fake postings.

For the deep learning model, each word was converted into an integer sequence through the **Keras Tokenizer**, followed by zero-padding to maintain fixed input lengths. This structure is necessary for the sequential Bi-LSTM model.

6. Model Development

1. Model 1: TF-IDF + Logistic Regression

A Logistic Regression classifier was trained on 5,000-feature TF-IDF vectors.
The model quickly converged with outstanding accuracy and recall, highlighting its effectiveness for detecting textual deception patterns such as unrealistic salary claims, exaggerated benefits, or spam-like descriptions.

Performance Results:

- Accuracy: **97.26%**
- Precision/Recall/F1: High consistency across both classes

The model was serialized using the Pickle format (model_tfidf_lr.pkl) along with the trained vectorizer for later deployment.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

2. Model 2: Bidirectional LSTM (Bi-LSTM)

A Bidirectional Long Short-Term Memory network was implemented using TensorFlow/Keras.
It consisted of:

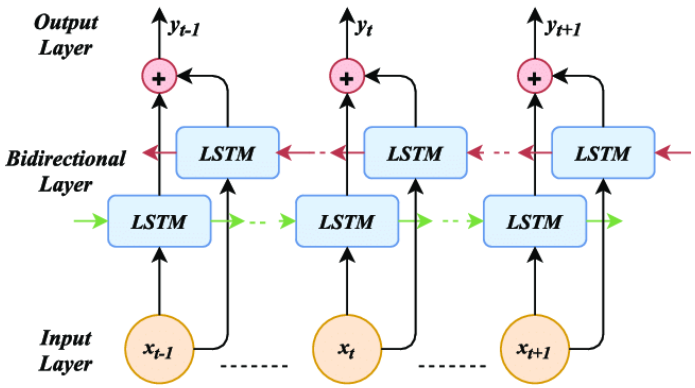
- **Embedding Layer:** 10,000 vocabulary words, 128-dimensional vectors
- **Bi-LSTM Layer:** 64 units with dropout regularization
- **Dense Layers:** ReLU + Sigmoid activations for binary classification

The Bi-LSTM model achieved comparable performance to the TF-IDF model, confirming that both traditional and deep learning approaches can effectively classify fake job postings.

Performance Results:

- Test Accuracy: **97.37%**
- Validation accuracy demonstrated stable convergence with minimal overfitting.

The model and tokenizer were saved as model_bilstm.h5 and tokenizer.pkl respectively for deployment.



7. Results & Evaluation Metrics

Model	Accuracy	Precision	Recall	F1-Score	Key Strength
TF-IDF + Logistic Regression	97.26%	High	High	High	Lightweight, fast inference
Bi-LSTM	97.37%	High	High	High	Captures context & word order

Both models achieved near-identical results, demonstrating that textual cues in fake job postings are highly learnable by both vectorized and sequential models.

8. Application Development

Overview

To provide an interactive and user-friendly interface for our *Fake Job Post Detection System*, we developed a web application using **Streamlit**, a Python framework designed for rapid deployment of machine-learning models. The goal of the application was to make the prediction model easily accessible to users, allowing them to input a job description and instantly determine whether the posting is real or fake.

Environment Setup

The application was implemented within a well-structured project directory in **Visual Studio Code**, using Python 3.13 and the following key libraries:

- **Streamlit** – for UI and real-time web app deployment
- **Scikit-learn** – for TF-IDF vectorization and Logistic Regression model
- **TensorFlow / Keras** – for the Bi-LSTM deep learning model
- **Pickle** – for model serialization and loading
- **Pandas, NumPy** – for data handling and preprocessing

Model Integration

Both trained models (TF-IDF + Logistic Regression and Bi-LSTM) were serialized and stored in the **model's** directory as:

- tfidf_vectorizer.pkl
- model_tfidf_lr.pkl
- tokenizer.pkl
- model_bilstm.h5

The application dynamically loads these models at runtime using Python's pickle and TensorFlow's load_model functions. Robust path-handling was implemented with the os module to ensure compatibility across environments.

Application Workflow

The Streamlit interface was designed for simplicity and efficiency. The workflow is as follows:

1. **User Input:** A text area accepts job descriptions or full postings.
2. **Preprocessing:** The input text is cleaned and tokenized using the same pipeline as during model training.
3. **Model Prediction:**
    - The TF-IDF + Logistic model provides fast, explainable predictions.
    - The Bi-LSTM model offers deeper semantic analysis for higher accuracy.
4. **Output Display:**
    - A probability score is shown, indicating the likelihood that a job post is fake.
    - Color-coded messages (green for *real*, red for *fake*) make the result intuitive.

User Interface

The final interface includes:

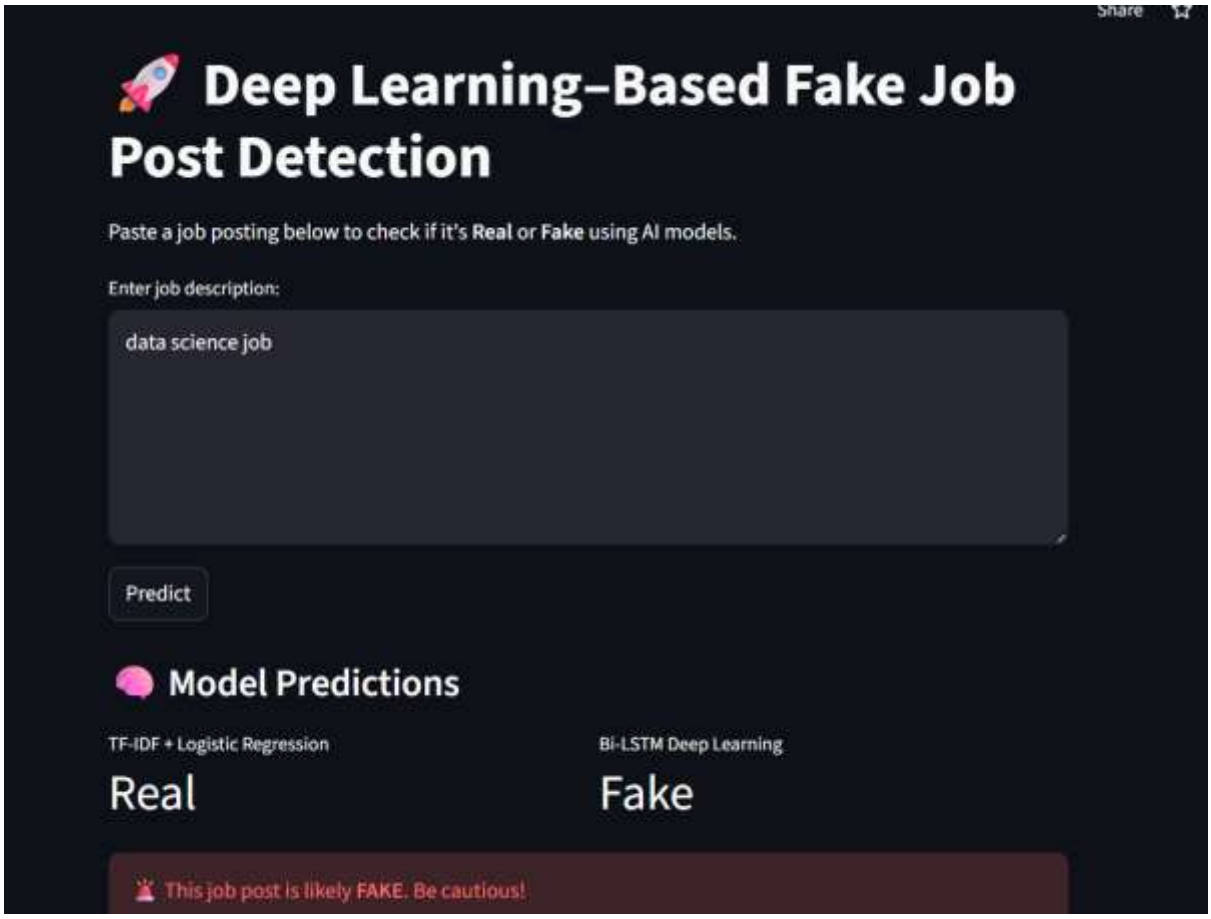
- A title banner and project description
- Text input area for job posting
- Buttons for running predictions

- Real-time result visualization

Streamlit automatically launches the app on a local server using the command:

streamlit run app.py

The interface is accessible in any browser and can be deployed on cloud platforms such as Streamlit Cloud or Hugging Face Spaces for public access.



9. Business / Real-world Impact

- Helps job seekers avoid scams.
- Assists recruitment platforms in automatic fraud detection.
- Enables research on patterns of fraudulent postings.
- Reduces time and financial loss associated with fake job posts.

10. Challenges & Limitations

- **Class Imbalance:** Fake postings are a minority, requiring careful evaluation.
- **Data Quality:** Missing or inconsistent textual entries needed extensive preprocessing.
- **Generalization:** Models trained on a specific dataset may require fine-tuning for other job platforms.
- **Deployment Limitations:** Bi-LSTM model requires cloud deployment due to computational resources.

11. Conclusion & Future Work

Conclusion:
This project successfully developed ML and DL models to detect fake job postings with high accuracy. A Streamlit web application makes the model accessible to users, translating AI research into practical solutions.

Future Work:

- Incorporate additional metadata features like company ratings.
- Implement ensemble models for higher accuracy.
- Expand the dataset with real-time scraping from multiple job platforms.
- Add explainability (SHAP/LIME) to interpret model predictions.

12. References

1. Fake Job Postings Dataset (Kaggle/GitHub)
2. TensorFlow Documentation: https://www.tensorflow.org/
3. Streamlit Documentation: https://docs.streamlit.io/
4. Scikit-learn Documentation: https://scikit-learn.org/

Links

GitHub repo: https://github.com/HiraArif666/FakeJobDetection-main

Medium Article: https://medium.com/@h.arif.kts4/deep-learning-based-fake-job-post-detection-on-social-media-c04410608b38?postPublishedType=initial