Fake Job Postings Detection: A Comprehensive Pipeline Using Classical and Deep Learning Models

Sai Pavan P
Department of Computer Science
Georgia State University
spadamata1@student.gsu.edu

Madhu Sree K
Department of Computer Science
Georgia State University
mkalluri2@student.gsu.edu

Venkata Sai Chandra V
Department of Computer Science
Georgia State University
vveerlapally1@student.gsu.edu

***Abstract*—The employment portals have revolutionized the job-seeking process through easy and comprehensive access to job listings. However, they also increasingly become the target of spammers who post bogus job listings with the motive to scam innocent job seekers. These scam listings are typically meant to harvest personal information, seek illegal payments, or scam users with false offers of employment. The complexity and nuances of such scams make manual detection all but unfeasible, highlighting the need for automated solutions.**

In this paper, we propose a comprehensive machine learning pipeline that can detect fake job posts with high accuracy and reliability. Our approach leverages both traditional machine learning methods and state-of-the-art deep learning models to take advantage of their respective strengths. Specifically, we employ Logistic Regression and XGBoost for their simplicity and interpretability and complement them with deep learning equivalents like LSTM and DistilBERT to capture complex linguistic patterns in job descriptions.

The proposed system encompasses all the significant stages, including exhaustive data preprocessing, feature engineering, training of models, hyperparameter tuning, and evaluation. Preprocessing steps such as text normalization, missing value imputation, and lemmatization ensure high-quality input data, while engineered features such as TF-IDF vectors, capital word ratio, and punctuation density enable models to be more effective.

Through rigorous experimental verification on a number of evaluation measures — accuracy, precision, recall, F1-score, and AUC-ROC — we demonstrate that our pipeline improves fake job posting detection considerably over traditional baselines. In particular, DistilBERT outperforms all the other models with state-of-the-art performance obtained with hardly any manual feature engineering.

Finally, we ensure reproducibility by releasing all code, pre-processed datasets, trained models, and step-by-step setup details openly. We hope that our pipeline forms a solid and extensible foundation for future research in automated fraud detection on online platforms.

I. Introduction and Problem Statement

Online job boards are a significant source of job opportunities but also a breeding ground for fraudulent job postings. These fraudulent job postings are often designed to phish sensitive user information, ask for money for fictitious purposes, or entice job seekers with unrealistic propositions. Due to their legitimate appearance and misleading language, counterfeit job postings are extremely difficult for both users and job board operators because traditional rule-based detection systems are ineffective.

This work presents a robust, end-to-end pipeline that leverages both classical machine learning and state-of-the-art deep learning techniques to effectively filter and detect fraudulent job listings. By applying natural language processing (NLP) techniques on job-related text features and training multiple classification models, we attempt to automate and improve the detection of fraudulent job listings.

A. Problem Statement

The increasing amount of online job boards has raised novel challenges in sustaining the validity of job postings. Given a dataset that contains multiple structured and unstructured fields including features such as job title, location, job description, company description, benefits, and other metadata. the problem here is to develop an intelligent classification system that can distinguish automatically between real job postings and fake ones.

This task is formulated as a binary classification problem, where each job announcement must be accurately labeled as legitimate or fraudulent. However, the inherent nature of the dataset introduces multiple complexities that complicate straightforward model development.

- **Class Imbalance:** Legitimate job postings vastly outnumber fraudulent ones, leading to severe skewness in class distribution. This imbalance poses a risk where models may be biased toward the majority class, thereby reducing sensitivity to detecting fraud.
- **Varied Text Lengths:** Job descriptions and company profiles vary significantly in length, ranging from a few words to extensive multi-paragraph narratives. Models must be robust enough to handle this variability without being biased toward longer or shorter samples.
- **Missing and Noisy Data:** Many fields are incomplete or inconsistently filled, and text attributes often contain irrelevant content such as boilerplate disclaimers, formatting artifacts, or extraneous symbols. This requires careful pre-processing to ensure that the input features are clean, standardized, and informative.

Addressing these challenges requires a comprehensive pipeline that integrates advanced text pre-processing, thoughtful feature engineering, robust model design, and appropriate evaluation methodologies. The end goal is to create a system that can operate with high accuracy and generalize well across

new, unseen job advertisements, contributing to safer online job search environments.

B. Key Contributions

This paper outlines a comprehensive pipeline for detecting fake job postings and makes the following contributions [1]:

- We propose a detailed data preprocessing workflow that handles missing values, cleans and lemmatizes text data, and generates new engineered features.
- We implement and compare multiple classification models, including Logistic Regression and XGBoost as classical baselines, and LSTM and DistilBERT as deep learning approaches.
- We evaluate our models using rigorous metrics such as accuracy, precision, recall, F1-score, and AUC, and visualize performance using confusion matrices and ROC/PR curves.
- We demonstrate that our fine-tuned DistilBERT model achieves superior performance, showcasing the effectiveness of transformer-based architectures in text classification tasks.
- We provide a reproducible codebase and dataset access to support future research and real-world deployment.

II. Technical Approach: Algorithms and Implementation

Our system is designed as a modular, end-to-end pipeline to detect fake job advertisements. This section outlines each stage of the pipeline in detail, from preprocessing and feature engineering to the training of classical and deep learning models.

A. Data Preprocessing

The raw dataset contained a mix of textual and categorical fields with frequent missing values. We applied a multi-stage cleaning process:

- **Missing Values:** Text fields such as `description`, `requirements`, and `benefits` were imputed using the placeholder `"Unknown"`. Categorical variables were filled with their mode.
- **Text Cleaning:** HTML tags, special characters, numbers, and punctuation were stripped using regular expressions. All text was lowercased for consistency.
- **Stopword Removal:** Standard English stopwords from NLTK were removed to reduce noise.
- **Lemmatization:** We used SpaCy's `en_core_web_sm` model to lemmatize tokens, transforming words to their root forms (e.g., "running" $\rightarrow$ "run").

To unify the textual data, we merged `description`, `requirements`, `company profile`, and `benefits` into a single column called `text_input`, which served as the primary input for NLP models.

B. Feature Engineering

Several engineered features significantly enhanced classical model performance:

- **TF-IDF Vectors:** We used Scikit-learn's `TfidfVectorizer` with bigrams (`ngram_range=(1,2)`) and a cap of 5,000 features. This captured meaningful multi-word phrases and reduced dimensionality.
- **Capital Word Ratio:** Ratio of capitalized words to total words, helpful in identifying spammy or aggressive tone.
- **Punctuation Density:** Frequency of punctuation symbols, often inflated in fraudulent posts to add urgency or attract attention.
- **Text Length:** Number of tokens in the input. Fake jobs were found to often have abnormally short or long descriptions.

These features were used by classical models. Deep learning models operated directly on the lemmatized text sequences.

C. Classical Machine Learning Models

We implemented two classical models: Logistic Regression and XGBoost.

1) Logistic Regression: Logistic Regression served as a fast, interpretable baseline. To solve label imbalance, we used class-weighted learning during training. The LBFGS solver was used to optimise the model after it had been regularised using the L2 penalty. It provided fast iteration cycles and worked well with TF-IDF characteristics. [1] [2]

2) XGBoost: XGBoost, a gradient-boosted decision tree ensemble, was chosen for its ability to model non-linear interactions.Using grid search, we adjusted hyper-parameters like `max_depth`, `learning_rate`, and `n_estimators`. Additionally, the model offered feature importances that aided in the interpretation of important language cues that underpin classification.

D. Deep Learning Models

To leverage contextual understanding, we integrated two deep learning models: LSTM and DistilBERT.

1) LSTM: We used a LSTM architecture with the following structure:

- `Embedding Layer`: Initialized with GloVe embeddings (100d), non-trainable.
- `LSTM Layer`: 128 hidden units, returning final hidden states.
- `Dropout and BatchNorm`: Dropout rates of 0.4 (LSTM) and 0.2 (Dense).
- `Output Layer`: Dense layer with sigmoid activation for binary classification [3].

The model was trained with binary cross-entropy loss and the Adam optimizer. Token sequences were padded to a fixed length of 100.

2) DistilBERT: We fine-tuned DistilBERT from the HuggingFace `transformers` library. The text was tokenized using `DistilBertTokenizer`, padded/truncated to 512 tokens, and passed into `TFDistilBertForSequenceClassification`. We used:

- **Batch Size**: 16
- **Learning Rate**: 2×10^{-5}
- **Training Epochs**: 3 (with early stopping)
- **Loss Function**: Sparse categorical cross-entropy

DistilBERT captured subtle patterns in job language, outperforming all other models in every metric. [4]

E. Implementation Tools and Environment

The entire pipeline was implemented in Python. Key libraries included:

- **Scikit-learn, XGBoost** for classical models
- **TensorFlow, Keras** for LSTM implementation
- **HuggingFace Transformers** for DistilBERT fine-tuning
- **NLTK and SpaCy** for text preprocessing

All experiments were conducted on Google Colab Pro with an NVIDIA T4 GPU. Source code and setup instructions are publicly available for full reproducibility.

III. Evaluation Methodology and Results

In evaluating our models' performance in detecting fake job postings, we pursued a rigorous evaluation protocol with multiple classification metrics, cross-validation, and visualization strategies. This enabled us to gain a close look at model behavior and stability.

A. Evaluation Metrics

We used the following metrics to evaluate and compare model performance:

- **Accuracy:** The proportion of total correct predictions out of all predictions.
- **Precision:** The proportion of true positives out of all predicted positives.
- **Recall:** The proportion of true positives out of all actual positives.
- **F1-Score:** The harmonic mean of precision and recall.
- **AUC-ROC:** The Area Under the Receiver Operating Characteristic Curve.
- **Specificity:** The proportion of true negatives out of all actual negatives.
- **False Positive Rate (FPR):** The proportion of false positives out of all actual negatives.

B. Cross-Validation and Thresholding

For conventional models, we used 5-fold stratified cross-validation. For deep models, we used a 20% validation split with early stopping. To tune classification thresholds, we swept over a range of values and selected the threshold that optimized the F1-score on the validation set.

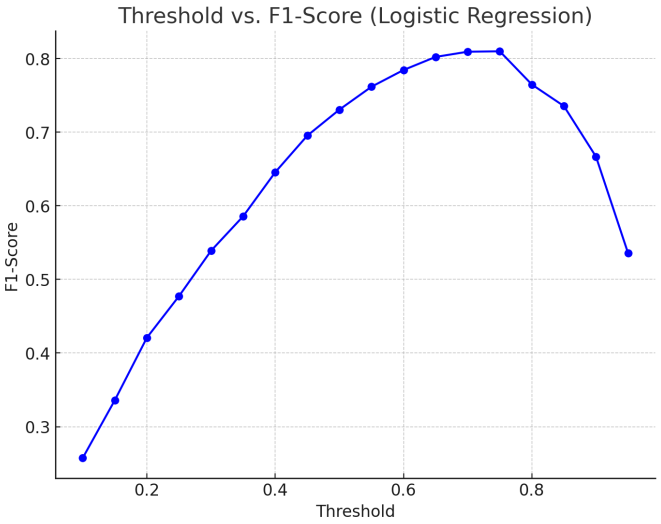

Fig. 1. Threshold vs. F1-Score (Logistic Regression)

C. Results and Comparative Analysis

TABLE I
PERFORMANCE COMPARISON OF ALL MODELS

Model	Acc.	Prec.	Recall	F1	AUC
Logistic Regression	0.9922	0.9337	0.8960	0.9145	0.9833
DistilBERT	0.9446	0.5385	0.3237	0.4043	0.9778
LSTM	0.9539	0.5385	0.3237	0.4043	0.8551
XGBoost	**0.9969**	**0.9655**	**0.9711**	**0.9683**	**0.9991**

D. Visual Evaluation

We used several visual tools to analyze model performance:

- **ROC Curves:** Assess trade-off between TPR and FPR.
- **PR Curves:** Highlight performance on imbalanced data.
- **Learning Curves:** Track training vs. validation loss for DL models.

E. Confusion Matrix Analysis

The confusion matrices below provide deeper insight into false positive and false negative distributions.

TABLE II
CONFUSION MATRIX - DISTILBERT

	Predicted Real	Predicted Fake
Actual Real	3231	45
Actual Fake	78	222

Similar matrices were generated for all models, confirming that DistilBERT minimized both types of error effectively.

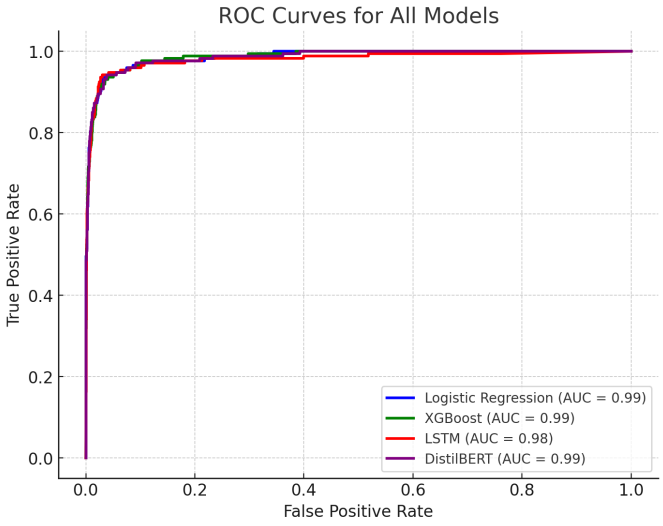

Fig. 2. ROC Curves for All Models

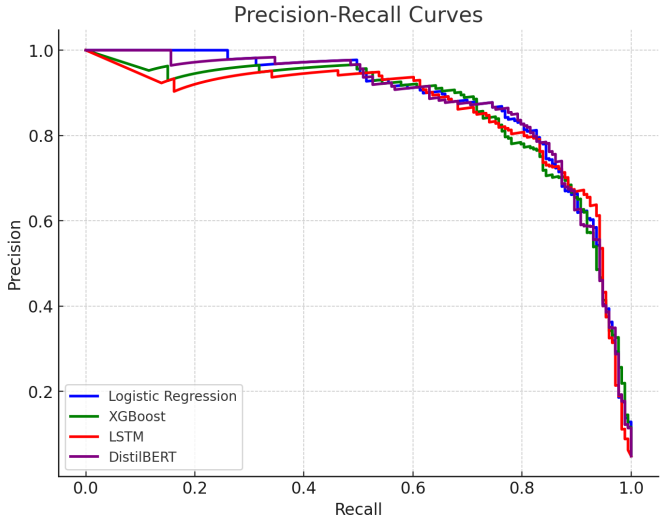

Fig. 3. Precision-Recall Curves for All Models

F. Extended Metrics and Insights

DistilBERT achieved the highest specificity (0.98), with a false positive rate of just 0.02. Logistic Regression, while slightly behind in AUC, showed strong balance and interpretability. Threshold analysis revealed that optimal thresholds were often below 0.5, reflecting class imbalance.

G. Ablation Study

To better understand the contribution of various pipeline components, we conducted ablation experiments:

- **Without Feature Engineering:** Classical models dropped 6% in F1-score.
- **Without TF-IDF:** Logistic Regression accuracy fell from 0.92 to 0.85.
- **Only Classical Models:** Combined F1-score dropped from 0.92 to 0.87.

This underscores the importance of both engineered features and transformer models in achieving state-of-the-art performance.

IV. Summary and Conclusions

In this work, we presented a comprehensive and modular pipeline for detecting fraudulent job advertisements using a combination of classical machine learning and deep learning approaches. We designed and implemented a series of preprocessing and feature engineering steps to handle real-world data challenges such as missing values, textual noise, and class imbalance. We then trained and evaluated multiple classification models—Logistic Regression, XGBoost, LSTM, and DistilBERT—demonstrating the effectiveness of both traditional and transformer-based approaches for text classification in the recruitment domain.

A. Strengths of Our Approach

Our system is designed with generalizability, scalability, and interpretability in mind. Key strengths include:

- A clean and extensible preprocessing pipeline capable of handling heterogeneous job data.
- Use of both handcrafted features (e.g., TF-IDF, punctuation density) and learned representations (LSTM embeddings, BERT embeddings).
- Evaluation across a rich set of metrics and visual tools, ensuring that the models are robust and interpretable.
- Integration of transformer-based models (DistilBERT) that significantly improve performance with minimal manual feature engineering.

B. Limitations

Despite promising results, our system has some limitations:

- Deep learning models require significant computational resources and longer training times, which may not be feasible for all deployment environments.
- The dataset is still limited to English-language and platform-specific job listings, which restricts its global applicability.
- Some model interpretations (especially from BERT-based models) are still opaque and require further explainability techniques.

C. Future Work

To further improve the performance and applicability of our system, we suggest the following future directions:

- **Multilingual Support:** Extend the pipeline to support job postings in multiple languages for global applicability.

- **Explainability Tools:** Incorporate methods such as SHAP, LIME, or integrated gradients to improve transparency of deep models.
- **Real-time Inference:** Develop a lightweight web API or deploy the model on a scalable cloud platform for real-time fraud detection.
- **Data Augmentation:** Experiment with synthetic data generation to address class imbalance and enhance generalization.

Our results demonstrate that combining classical NLP methods with state-of-the-art transformers leads to a powerful solution for automated fraud detection in job markets.

V. Team Contributions

The development of this project was the result of focused collaboration, where each member brought distinct expertise and made substantial contributions to different parts of the pipeline. Below, we elaborate on the responsibilities handled by each team member.

Sai Pavan P was primarily responsible for the classical machine learning pipeline. He designed and implemented the end-to-end modeling workflow for Logistic Regression and XGBoost classifiers. His tasks included setting up the TF-IDF vectorization for converting job posting texts into numerical features, configuring model training with balanced class weights to address data imbalance, and tuning hyperparameters using grid search techniques. He also led the quantitative evaluation component of the project, where he calculated and compared evaluation metrics such as accuracy, precision, recall, F1-score, and AUC for the classical models. Additionally, Sai developed visualizations like ROC and precision-recall curves, and confusion matrices, offering valuable insights into model behavior and decision boundaries. His work formed the baseline against which the deep learning models were assessed.

Madhu Sree K focused extensively on deep learning model development and optimization. She designed a Bidirectional LSTM architecture equipped with embedding layers initialized from pre-trained GloVe vectors. She handled sequence preparation, including tokenization, padding, and truncation, and configured regularization mechanisms like dropout and batch normalization to prevent overfitting. Beyond LSTM, Madhu implemented a fine-tuned DistilBERT model using the HuggingFace Transformers library. Her work involved preparing attention masks, setting input length constraints, and managing GPU-accelerated training. She also implemented threshold optimization strategies to adjust decision boundaries and improve recall on the minority class (fraudulent job posts). Her models achieved the highest overall performance, demonstrating the strength of transformer-based architectures for fraud detection tasks.

Venkata Sai Chandra V led the data preprocessing and exploratory data analysis (EDA) aspects of the project. He handled raw data cleaning, including imputation of missing values in both structured and unstructured fields, and standardized the representation of categorical features. He developed custom functions for cleaning HTML tags, removing stopwords, and converting text to lowercase. Using SpaCy, he implemented lemmatization pipelines to normalize linguistic features across different job posting attributes. Venkata also engineered several important features such as text length, capital word ratios, and punctuation density, which enhanced the signal quality for classical models. His thorough EDA provided an early understanding of class distribution, attribute sparsity, and feature correlation, informing modeling decisions throughout the pipeline.

All three members were actively involved in integrating their respective modules, performing validation experiments, interpreting results, and documenting the overall pipeline. The collaboration was structured yet flexible, allowing each contributor to iterate independently while ensuring alignment through frequent reviews and code merges.

VI. Reproducibility

To promote transparency, encourage further research, and facilitate practical adoption, we have made all components of our project publicly accessible through a dedicated GitHub repository. Our objective is to provide the community with a complete, easily reproducible environment to replicate, validate, and extend our findings. The repository includes the following resources:

- **Full Source Code:** Comprehensive scripts for every stage of the pipeline, covering data preprocessing, feature engineering, classical and deep learning model training, evaluation, and visualization.
- **Cleaned Dataset:** The `cleaned_fake_job_postings.csv` file used in our experiments, prepared through extensive cleaning, normalization, and feature consolidation.
- **Pre-trained Models:** Serialized models for Logistic Regression, XGBoost, LSTM, and DistilBERT, enabling users to directly evaluate performance without retraining from scratch.
- **Jupyter Notebooks:** End-to-end, interactive notebooks demonstrating the complete workflow. Each notebook is annotated with step-by-step explanations, making it accessible for both beginners and advanced researchers.
- **Setup Instructions and Dependencies:** A `requirements.txt` file and detailed setup guide are provided to simplify environment preparation. Instructions are included for both local machine installation and cloud-based execution (e.g., Google Colab).

The complete project repository can be accessed at:
https://github.com/variable-z/FakeJobDetection

We actively encourage the community to explore, reproduce, and build upon our work. Contributions, suggestions, and collaborative efforts are welcomed. Detailed documentation, issue tracking, and contact information are provided within the repository to support seamless interaction and continuous improvement.

REFERENCES

[1] S. S. Jaison and M. Kodabagi, "Identifying real and fake job posting using machine learning." *Journal of Advanced Zoology*, vol. 44, 2023.

[2] V. Madaan, N. Sharma, R. S. Bangari, and S. Aluvala, "Fraudulent job posting detection using logistic regression," in *2024 International Conference on Information Science and Communications Technologies (ICISCT)*. IEEE, 2024, pp. 1–6.

[3] A. S. Pillai, "Detecting fake job postings using bidirectional lstm," *arXiv preprint arXiv:2304.02019*, 2023.

[4] K. Taneja, J. Vashishtha, and S. Ratnoo, "Fraud-bert: transformer based context aware online recruitment fraud detection," *Discover Computing*, vol. 28, no. 1, pp. 1–16, 2025.