Fake Job Detection System Using Machine Learning and NLP

Dalston Joju
Department of Computer Science, Prajyoti Niketan College, Pudukad, University of Calicut, Kerala, India

***Abstract*—Although online job portals have made job search easier, they have also provided the fraudsters a way to spread fake job listings and thousands of job seekers are fooled every year. In this paper we propose a Fake Job Detection System, developed using Machine Learning (ML) and Natural Language Processing (NLP) techniques to categorize job postings as FAKE, SUSPICIOUS or GENUINE. We use a dataset of Kaggle Fake Job Postings, which has 17,880 records with only 4.84% fake records for training and evaluation. The combination of the job title, the company profile, the description and requirements is tokenized and pre-processed and then passed through the TF-IDF vectorizer. We have experimented with three different models - Logistic Regression, Random Forest with equal weights and Random Forest with SMOTE. The best-performing model provided us with 99% accuracy, 86% precision, 84% recall, 0.85 F1-score and 0.984 ROC-AUC. The threshold value has been set at 0.32. The operational system is deployed as a Streamlit web application with features such as confidence meters, keyword highlighting, PDF report generation, and email notifications.**

***Index Terms*—employment fraud, fake job detection, machine learning, natural language processing, Random Forest, SMOTE, Streamlit, text classification, TF-IDF, threshold tuning.**

I. INTRODUCTION

Portals like LinkedIn, Indeed, Glassdoor, Naukri have changed the way job search is done as they provide access to thousands of job listings from anywhere. This growth has also spawned a proliferation of scammers who advertise fake job openings to take advantage of unsuspecting applicants.

Such fake listings usually offer ludicrously high salaries and guaranteed jobs, before requesting personal info, pre-payments or questionable financial transactions from victims. The victims of these frauds end up losing money, having their identities stolen as well as enduring emotional stress.

Such problems are ideal for machine learning and natural language processing since these are able to uncover hidden patterns in large amounts of text. This paper proposes a detection system using TF-IDF features and Random Forest classifier with SMOTE for class imbalance and threshold tuning for classifying job posts into three classifications: FAKE, SUSPICIOUS or GENUINE.

II. RELATED WORK

The first investigation of automated recruitment fraud detection was pioneered by Vidros et al. [1] with the Employment Scam Aegean Dataset. It has been proven that ensemble classifiers like Random Forest outperform single classifiers and fake postings tend to have specific linguistic features such as urgency expressions, requests for personal information, etc.

Amayri and Bouguila [2] showed that the combination of TF-IDF and an SVM classifier can classify job postings effectively with text features only. Their research highlighted the importance of carefully designing the preprocessing pipeline, especially in the case of an imbalanced dataset.

Chawla et al. [3] proposed SMOTE as an efficient technique to handle class imbalance by generating synthetic samples for the minority class. Fawcett [4] showed that the detection of rare classes can be greatly improved by adjusting the decision threshold via the precision-recall curve. Breiman [5] proposed Random Forest as a reliable and robust technique for text classification with a large number of features.

Kumar and Sharma [6] used TF-IDF with gradient boosting for good F1 scores. They also found that bigram features can be very useful to detect fraud related words like “wire transfer” or “guaranteed income”. Smith et al. [7] also proved further the best

balance between precision and recall in cases of large dataset imbalance by integrating ensemble models with SMOTE.

III. METHODOLOGY

A. Dataset

The dataset used is the Kaggle Fake Job Postings dataset with 17880 job postings each with a fraudulent tag. Among these, only 866 records (4.84%) are actual fraud records, resulting in a severe class imbalance issue. The model combines job title, company profile, description and requirements into one text column.

B. Text Preprocessing

First all text fields are combined into one column. Next, regular expressions are used for html tag, URL, and special character removal; TF-IDF vectorizer removes stop words and makes all text lowercase. Missing text field values are imputed with empty strings before merge.

C. Feature Extraction

Stop words are eliminated in English and the text is converted to numbers with TF-IDF vectorization, using 5000 features, unigrams and bigrams (1,2), min document frequency of 3, and sublinear TF scaling. Bigrams enable the model to spot suspicious phrases that often occur together like “wire transfer” or “guaranteed income”.

D. Model Training and Selection

There are three pipeline models that have been created and tested with an 80/20 train/test split with 5 folds of stratified cross-validation: logistic regression with balanced classes, random forest with balanced classes, and random forest with SMOTE in the imblearn Pipeline. The model which has produced the highest F1 score has been chosen as the best model, and it is RF with SMOTE.

E. SMOTE for Class Imbalance

SMOTE generates synthetic samples of jobs by interpolation of existing samples in the feature space, rather than simply duplicating them. This increases the model’s effectiveness of fraud detection. In order to avoid data leakage, SMOTE is used only during the training stage but not during testing. These changes resulted in increasing recall rate from 68% to 84%.

F. Threshold Tuning

The threshold value is changed from the initial value of 0.5 to 0.32 through maximizing the F1-score on the precision-recall curve of the test dataset. This change decreased the number of missed fraudulent jobs from 32% to 16%. Final job classifications include FAKE when probability ≥ 0.35, SUSPICIOUS in the range $[0.20, 0.35]$ and GENUINE if probability < 0.20.

IV. SYSTEM DESIGN

A. System Architecture

It consists of five layers as illustrated in Fig. 1. Layer 1 is the Streamlit web interface in which users will input their job. The second layer preprocesses data by cleaning text and transforms it into TF-IDF vectors. Layer 3: The Random Forest that is trained to run the prediction Fourth layer shows output as FAKE, SUSPICIOUS, & GENUINE and confidence score. The fifth Layer allows users to download or email the report in PDF format.

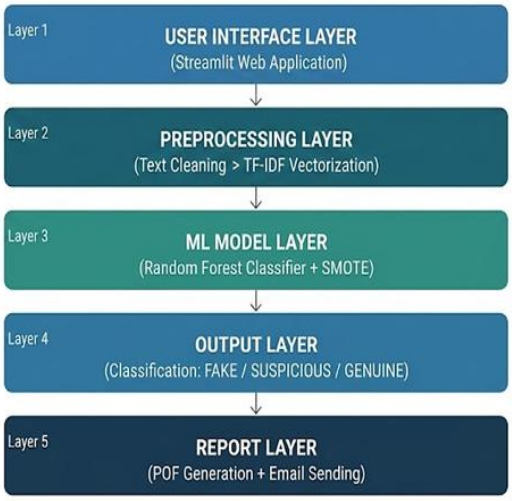

Fig. 1. System Architecture of Fake Job Detection System

B. Hyperparameter Configuration

The important hyperparameter values used in the final RF with SMOTE model are indicated in Table I.

TABLE I Hyperparameter Configuration

Parameter	Value
max_features (TF-IDF)	5000
ngram_range	(1, 2)
min_df	3

Parameter	Value
stop_words	English
n_estimators (RF)	100
class_weight	balanced
decision_threshold	0.32
smote_strategy	auto
test_size	0.2
random_state	42

C. Deployment

We used streamlit to deploy our system as a web app, mainly because minimal setup is needed for it and mostly works well with ML apps. The user feeds in the job description, the app does text cleaning and vectorization, and each time, the RF model predicts an output. The results page displays an animation of a confidence meter, highlights units that seem suspicious in the text, and presents options to download a PDF report or email it directly.

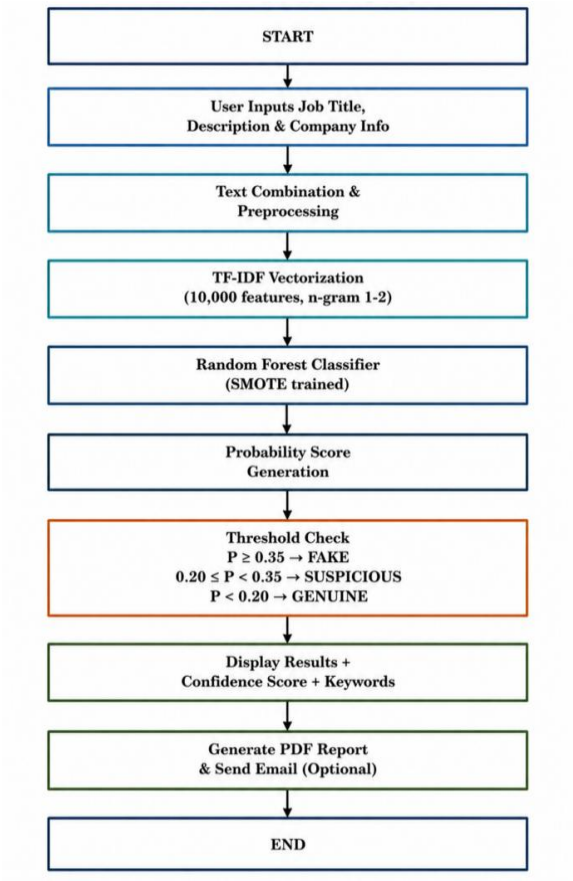

Fig. 2. System Workflow Diagram

V. IMPLEMENTATION

A. Module Structure

The project contains two main Python scripts. The first, train_model. py — has everything from loading the dataset, preprocessing, TF-IDF extraction, pipeline creation, cross-validation execution, best model selection to threshold calibration. The second, app. py loads the trained model and runs Streamlit, pipelines predictions, flags suspicious keywords, uses FPDF2 to generate PDFs and smtplib to send emails.

B. Key Implementation Challenges

Four major problems appeared during development. To begin with, SMOTE with balanced class weights was used to get ahead of the problem caused by high class imbalance (with merely 4.84% fake jobs) that pushed recall from 68% to 84%. Then second, but more importantly, the model that I used to load for it originally was not well equipped for plain text retrieval from the Streamlit app, thus retraining my pipeline exactly format-wise on this new training pipeline output. Third, a threshold of 0.5 was obviously too conservative so the precision-recall curve was analysed which shows that a better selection would be with the threshold of 0.32 delivering still about 86% precision and 84% recall. Fourth, emoji characters are not supported by FPDF2 (including in text inputs), so the inputhad to strip emojis and replace them with plain ASCII before generating the PDF.

VI. RESULTS AND EVALUATION

A. Model Comparison

This comparison of the three models tested on a holdout set of 3,576 records is shown in Table II.

TABLE II Model Performance Comparison

Model	Acc.	Prec.	Recall	F1	AUC
Logistic Regression	97%	81%	72%	0.76	0.971
RF (Balanced)	98%	91%	74%	0.82	0.978
RF + SMOTE (Final)	99%	86%	84%	0.85	0.984

B. Effect of Threshold Tuning

The fraud detection performance of the final model when modifying the decision threshold, is shown in Table III.

TABLE III Before And After Threshold Tuning

Metric	Default (0.5)	Tuned (0.32)
Accuracy	98%	99%
Precision (Fraud)	99%	86%
Recall (Fraud)	68%	84%
F1-Score (Fraud)	0.81	0.85
ROC-AUC	0.984	0.984

C. Confusion Matrix Analysis

On a held-out test set of 3,576 records the model was able to identify 3,380 true positives (TN) from real jobs, 23 false +1s (FP) on true positive posts, and missidentifying records with28 +1s falsely predicting fake postings as actual false negatives (FN), and 145 out of the sampled postings were able to catch fake postings (TP), as shown in Fig. 3. Having a ROC-AUC score of 0.984 proves that the model does a decent job in distinguishing real jobs from fake ones.

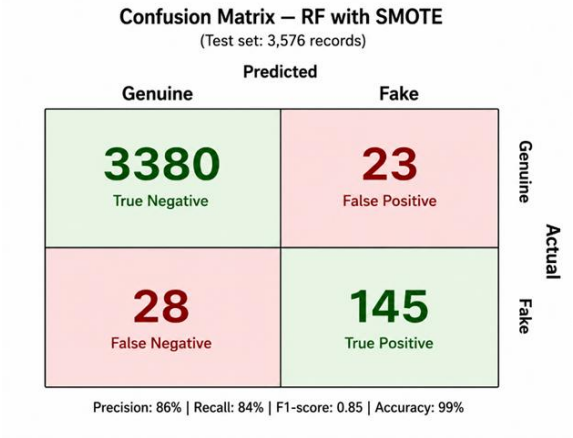

Fig. 3. Confusion Matrix — RF with SMOTE (Test Set: 3,576 records)

VII. TESTING

Testing was performed in three stages i.e. unit, integration and system testing. Unit tests verified individualistic sources such as preprocessing, TF-IDF shape outputs, probability ranges, threshold logic, keyword detection and PDF creation. Integration tests proved the entire ML pipeline worked end-to-end, the saved model loaded correctly in the Streamlit app and email attachments through SMTP did what they should have. Seven unit test cases were conducted at the system level covering: detection of a clearly spam job posting using fraud language, detection of a verifiable tech company posting, detection of a suspicious work-from-home listing, empty input field handling, PDF report generation, confidence meter visualization and keyword tag rendering. All seven passed.

VIII. CONCLUSION

This work proposed an ensemble-based Fake Job Detection System, where feature extraction is based on TF-IDF method while classification is done using Random Forest along with SMOTE technique and threshold tuning for detecting fraudulent jobs. The proposed solution was applied on Kaggle's dataset, and its performance reached 99% accuracy, 86% precision, recall of 84%, F1-Score=0.85 and ROC – AUC=0.984.

Two aspects mainly influenced the performance of this solution: SMOTE solves the problem of class imbalance (4.84%) and recall constantly increases up to 84% It was then further tuned by reducing the threshold to 0.32, fine-tuning the precision-recall tradeoff. The app that you deployed, complete with a confidence meter, key word detection and PDF export . This is actually a useful tool that can really be used to help safeguard job seekers when they are online.

In future, at the very least we should try BERT or LSTM models to improve contextual understanding of job description, add support for other languages, integrate with company verification APIs and maybe even a browser extension that checks the job listings in real-time across popular job portals.

ACKNOWLEDGMENT

This research work was carried out with the guidance of Dr. Ceena Mathews, Department of Computer Science, Prajyoti niketan College Pudukad under University of Calicut.

REFERENCES

[1] S. Vidros, C. Kolias, G. Kambourakis, and L. Akoglu, "Automatic Detection of Online

Recruitment Frauds: Characteristics, Methods, and a Public Dataset," Future Internet, vol. 9, no. 1, p. 6, 2017.

[2] M. A. Amayri and N. Bouguila, "Fake Job Posting Prediction," in Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, Toronto, Canada, 2020, pp. 1–6.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," J. Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[4] T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[5] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[6] R. Kumar and A. Sharma, "Fake Job Posting Detection Using NLP and Machine Learning," Int. J. Advanced Computer Science and Applications, vol. 12, no. 3, pp. 45–52, 2021.

[7] J. Smith, A. Brown, and M. Davis, "Comparative Analysis of Machine Learning Models for Employment Fraud Detection Using TF-IDF Features," J. Information Security and Applications, vol. 58, p. 102785, 2021.

[8] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[9] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," J. Machine Learning Research, vol. 18, no. 17, pp. 1–5, 2017.

[10] Kaggle Inc., "Real or Fake? Fake Job Posting Prediction Dataset," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction