iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** IV **Month of publication:** April 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.80867

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

# Fake Job Recruitment Detection Using a Supervised Machine Learning Approach

Prof. Sudhanshu Tripathi[1], Prof. Rohan B Kokate[2], Vishal Jain[3]

***Abstract:*** ***The rapid rise of online recruitment platforms in the current digital era has led to a significant increase in fake job postings that put job seekers at risk of financial losses, identity theft, and mental distress. This research paper offers a detailed study on how to automatically detect fraudulent job advertisements by employing supervised machine learning methods and a web application based on Flask. The study uses the Employment Scam Aegean Dataset (EMSCAD), which contains 17,880 labelled job postings, to train and test various classification models including Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, Light GBM, and deep learning models like Multi-layer Perceptron (MLP), Bidirectional Long Short-Term Memory (Bi-LSTM), and Deep Neural Networks (DNN).***

***The web application that was developed, using Python, Flask, and scikit-learn, takes job posting inputs from users across multiple fields and utilizes a trained machine learning model to classify these postings in real time as either genuine or fraudulent. The text features are extracted using the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method. The experimental results indicate that the Light GBM classifier achieves the most balanced performance with an accuracy of 98.18% and a ROC-AUC score of 0.91, while the Bi-LSTM model achieves the highest raw accuracy of 98.71%. The integrated web system connects academic model development with practical application, creating a fraud detection tool accessible to users. The results show that natural language processing (NLP) and machine learning can serve as essential elements in developing safer and more reliable digital recruitment environments.***

***Keywords include: Fake Job Posting Detection, Supervised Machine Learning, Natural Language Processing, TF-IDF, Flask Web Application, Bi-LSTM, Light GBM, Fraud Detection, EMSCAD Dataset, Online Recruitment Security.***

## I. INTRODUCTION

The growth of digital employment platforms has changed the global job market, providing significant convenience and access for both employers and job seekers. Platforms such as LinkedIn, indeed, Glassdoor, Naukri, and Internshala manage millions of job listings each year, allowing organizations to find talent and individuals to explore job opportunities from various locations. However, this rapid expansion has created a serious vulnerability that allows job seekers to be targeted through fake job postings.

Fraudulent job advertisements are a type of social engineering where malicious individuals pretend to be legitimate employers in order to obtain sensitive personal and financial information from unsuspecting applicants. Victims often face demands for upfront fees to apply, requests for identity documents such as passports and national IDs, and phishing attempts aimed at stealing banking information. Cybersecurity reports indicate that employment scams result in losses amounting to hundreds of millions of dollars worldwide every year, with fresh graduates, unemployed individuals, and economically disadvantaged groups being disproportionately affected. Traditional methods for dealing with fraudulent postings have depended on manual reviews by platform moderators, user-reported flags, and rule-based filtering systems. These methods have significant limitations; they are time-consuming, inconsistent, can easily be bypassed by clever attackers who adapt their strategies, and cannot keep pace with the volume of postings on large platforms. There is a clear need for intelligent automated systems that can analyze job postings on a large scale and accurately identify deceptive content. This research aims to tackle this issue by creating and assessing a machine learning-based system for detecting fake job postings, which is implemented through a deployable Flask web application. The system accepts structured inputs for various job posting fields such as title, location, department, company profile, job requirements, benefits, employment type, required experience, required education, industry, function, and description. These inputs are combined, vectorized with TF-IDF, and classified by a pre-trained machine learning model to determine if they are genuine or fraudulent. The rest of this paper is structured as follows: Section 2 offers a detailed literature review; Section 3 details the dataset used; Section 4 explains the methodology including preprocessing, feature engineering, model training, and system architecture; Section 5 presents experimental results and discussions; Section 6 discusses the design and implementation of the web application; Section 7 outlines limitations and suggests future work; Section 8 concludes the paper.

©IJRASET: All Rights are Reserved 

## II. LITERATURE SURVEY

A considerable amount of research has been conducted on fraud and spam detection in digital communication, covering areas such as email spam filtering, review spam identification, fake news detection, and more recently, fraudulent job posting detection. This section reviews key contributions in these areas, leading to a summary of research gaps that have inspired the current study.

### *A. Review Spam Detection*

Online reviews of products and services have a significant impact on consumer buying behavior, making them a target for manipulation. Spammers create fake positive reviews for their products or negative reviews for competitors, altering market information. Initial research in this area used behavioral indicators, review patterns, and linguistic features obtained through NLP to classify reviews as real or fake. Machine learning classifiers like Support Vector Machines (SVM) and Naive Bayes showed promising results, yet challenges such as adversarial adaptation and the scarcity of labeled training data continued to exist.

### *B. Email Spam Detection*

Email spam detection is one of the most developed fields in applied machine learning. Content-based filtering methods utilize bag-of-words models, TF-IDF vectorization, and n-gram features to differentiate spam from legitimate emails. Major email services like Gmail, Yahoo Mail, and Microsoft Outlook use advanced neural network architectures to nearly eliminate spam. Heuristic and adaptive filtering techniques have also been combined with machine learning ensembles to enhance resilience against changing spam strategies.

### *C. Fake News Detection*

The misuse of misinformation on social media has led to extensive studies on automated fake news detection. Research in this field has focused on three main analytical aspects: content-based features like writing style, sentiment, and linguistic markers; propagation patterns that track how information spreads across social networks; and user behavior signals that indicate whether accounts sharing certain content show suspicious activity. Deep learning techniques, such as recurrent neural networks, attention mechanisms, and transformer models like BERT, have shown significant advancements compared to traditional machine learning methods.

### *D. Fraudulent Job Posting Detection*

Pillai (2023) introduced a detection framework that uses Bidirectional LSTM networks applied to the EMSCAD dataset, showing that sequential deep learning models are better at capturing contextual linguistic details in job descriptions. Boka (2024) performed a comparative analysis of various machine learning models including Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting, finding that ensemble methods generally perform better than single classifiers. Kumar (2025) conducted a thorough review of NLP-based techniques, highlighting the importance of integrating textual and structured feature representations. Gulshan et al. (2024) investigated hybrid feature extraction that combines TF-IDF with word embeddings, indicating better precision in conditions requiring high recall. Naudé and Kumar (2023) developed a machine learning pipeline for real-time detection on streaming job data, addressing scalability issues not present in earlier academic work. Bhatta (2025) showed the effectiveness of transformer models for understanding the language of job postings.

### *E. Identified Research Gaps*

The review of existing literature shows several ongoing deficiencies that the current study seeks to address.

*1)* The EMSCAD dataset has a significant class imbalance where genuine postings outnumber fraudulent ones by about 10:1. Many previous studies did not adequately deal with this imbalance, leading to models that have high accuracy but low recall for the minority class, which is the fraudulent postings.

*2)* Some studies only used structured metadata fields and did not take into account the detailed textual content found in job descriptions and requirements, which is the main way deception occurs.

*3)* There are few deployed systems stemming from academic contributions, and models have rarely been turned into functional applications that users can access. The process of converting trained models into deployable web interfaces has not been explored much.

*4)* Many studies focused mainly on classification accuracy without considering other important performance metrics relevant to fraud detection, such as precision, recall, F1-score, and ROC-AUC score.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

*5)* Previous architectures for real-time detection have not been tested at the scale needed by large job portals that handle thousands of new postings every hour.

## III. DATASET DESCRIPTION

The dataset used in this study is called the "Real or Fake Job Posting Prediction" dataset, which is publicly available on Kaggle and comes from the Employment Scam Aegean Dataset, known as EMSCAD. It was compiled by the Laboratory of Information and Communication Systems Security at the University of the Aegean and is widely used as the benchmark dataset for research on fraudulent job detection.

### *A. Dataset Composition*

The dataset includes 17,880 job posting records, each marked with a binary label indicating whether it is genuine or fraudulent, with 0 representing genuine and 1 representing fraudulent. The distribution of these labels illustrates the inherent imbalance in this problem domain:

Table 1: Distribution of Job Posting Labels in EMSCAD Dataset

| Category | Count | Percentage |
|---|---|---|
| Genuine Job Postings | 16,244 | 90.85% |
| Fraudulent Job Postings | 1,636 | 9.15% |
| Total | 17,880 | 100.00% |

### *B. Feature Description*

The dataset contains 17 features that include both structured and unstructured attributes of job postings. The features used as inputs in the developed web application match those available in the dataset and are described below:

Table 2: Feature Description of EMSCAD Dataset

| Feature Field | Type | Description |
|---|---|---|
| title | Text | Job title or position name |
| location | Text | Geographic location of the job |
| department | Text | Organisational department |
| company_profile | Text | Description of the hiring company |
| description | Text | Full job description text |
| requirements | Text | Required qualifications and skills |
| benefits | Text | Offered compensation and benefits |
| employment_type | Categorical | Full-time, part-time, contract, etc. |
| required_experience | Categorical | Years or level of experience required |
| required_education | Categorical | Educational qualification required |
| industry | Text | Industry sector of the role |
| function | Text | Functional area or job function |
| fraudulent | Binary | Target label: 0 = Real, 1 = Fake |

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

### C. Data Characteristics and Quality

The dataset has several characteristics that are important for preprocessing and model design. A large number of records have missing values in textual fields, especially in the company_profile, benefits, and department columns. These missing values occur because many fraudulent postings intentionally leave out these fields, making their absence itself a potentially informative signal. The textual fields as a whole are the main source of distinguishing information, as the language patterns used in fraudulent postings differ significantly from those in genuine postings, showing differences in urgency markers, grammatical mistakes, vague promises of benefits, and unrealistic claims about compensation.

## IV. METHODOLOGY

The methodology used in this study follows a structured pipeline that includes data acquisition, preprocessing, feature engineering, model training and evaluation, and system deployment. Each stage is detailed in the following subsections.

### A. Data Acquisition and Preprocessing

The raw EMSCAD dataset was obtained from the Kaggle repository and put into a pandas DataFrame for processing. The preprocessing pipeline included the following steps:

1) Handling Missing Values: Missing values in all textual columns were replaced with empty strings to allow for uniform concatenation during feature construction. This method avoids introducing imputation artifacts in text fields while maintaining the potential informativeness of the absence of fields.
2) Text Normalisation: All text content was converted to lowercase to remove case-based inconsistencies. Punctuation, special characters, and excess whitespace were eliminated using regular expression substitutions. Stop words were removed using the NLTK stop word corpus, and tokens were lemmatised with the WordNet Lemmatizer to reduce morphological variants to their base forms.
3) Feature Concatenation: As shown in the deployed Flask application, the system creates a single unified text representation by concatenating the values of all relevant input fields, including title, location, department, company profile, requirements, benefits, employment type, experience, education, industry, function, and description. This concatenation method captures both structured signals and free-text content in a single representation appropriate for TF-IDF vectorization.
4) Handling Class Imbalance: Due to the significant imbalance between genuine and fraudulent postings, the Synthetic Minority Over-sampling Technique, known as SMOTE, was applied during model training to synthetically increase the minority class. This helps prevent classifiers from becoming biased towards predicting the majority class, thereby improving recall for the fraudulent class.

### B. Feature Engineering

Text vectorization serves as the important link between raw text input and numerical feature representations that are suitable for machine learning classifiers. In this study, the TF-IDF technique, which stands for Term Frequency-Inverse Document Frequency, is used for this purpose. TF-IDF gives a weight to each term in a document based on two factors: how often the term appears in the document (TF) and the inverse of how often it appears across all documents in the corpus (IDF). This weighting system ensures that terms that are common throughout the whole corpus, which are less discriminative, receive lower weights, while terms that appear frequently in a specific document but rarely elsewhere get higher weights, making them stronger indicators of the document's content and class membership. The TF-IDF vectorizer was fitted on the training corpus and used to transform both training and test inputs into sparse feature matrices. A maximum vocabulary size of 10,000 features was set, and n-gram ranges were configured to unigrams and bigrams to capture contextual word pairs. The fitted vectorizer was saved to disk as vectorizer.pkl for use in the deployment environment.

### C. Model Development and Training

Multiple supervised classification models underwent training and evaluation using the preprocessed dataset. The dataset divided into three subsets: training set consisting of 80 percent, validation set consisting of 10 percent, and test set also consisting of 10 percent, with stratified sampling used to maintain class proportions in these splits. The models that were evaluated include Logistic Regression, which acts as a probabilistic linear classifier and serves as a baseline with L2 regularization; Decision Tree, which functions as a hierarchical non-parametric classifier that employs Gini impurity for node splitting; Random Forest, which is an ensemble of decision trees trained through bootstrap sampling and utilizes bagging to lower variance; Gradient Boosting,

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

specifically XGBoost, which is an iterative ensemble method that builds trees in a sequential manner with each tree correcting the residuals of the previous one; LightGBM, which is a gradient boosting framework that uses leaf-wise tree growth and histogram-based splitting to achieve computational efficiency and high performance with large datasets; Multi-layer Perceptron, or MLP, which is a feedforward neural network that includes two hidden layers and uses ReLU activations; and Bidirectional LSTM, or Bi-LSTM, which is a recurrent neural network architecture designed to process sequences in both forward and backward directions and is trained on word embedding representations. Hyperparameter tuning took place for the ensemble models using k-fold cross-validation with k equaling 5 and involved grid search over significant parameters including the number of estimators, maximum depth, learning rate, and minimum samples per leaf. The model that performed best on the validation set was chosen for deployment and saved as model.pkl.

### *D. Evaluation Metrics*

Due to the class imbalance that exists in the dataset, accuracy by itself does not serve as a sufficient evaluation measure. A set of comprehensive metrics was used to describe model performance across all relevant aspects. These metrics include Accuracy, which is the proportion of instances correctly classified across both classes; Precision, which is the fraction of predicted fraudulent postings that are actually fraudulent and thus measures the false positive rate; Recall, which is the fraction of actual fraudulent postings that the model correctly identifies and thus measures sensitivity to the positive class; F1-Score, which is the harmonic mean of precision and recall and provides a single balanced metric for classification performance in the context of imbalance; ROC-AUC Score, which represents the area under the receiver operating characteristic curve and assesses the model's ability to differentiate between classes at all classification thresholds; and Confusion Matrix, which offers a detailed breakdown of true positives, true negatives, false positives, and false negatives.

## V. RESULTS AND DISCUSSION

### *A. Model Performance Comparison*

The table below provides a summary of the classification performance for all models that were evaluated on the held-out test set after the hyperparameter tuning and SMOTE-based resampling of the training data:

Table 3: Performance Comparison of Machine Learning Models on EMSCAD Test Set

| Model | Accuracy | Precision | Recall | F1-Score | ROC-AUC |
|---|---|---|---|---|---|
| Logistic Regression | 95.41% | 0.78 | 0.61 | 0.68 | 0.8 |
| Decision Tree | 96.02% | 0.85 | 0.7 | 0.77 | 0.85 |
| Random Forest | 97.04% | 0.94 | 0.66 | 0.77 | 0.83 |
| Gradient Boosting (XGBoost) | 96.67% | 0.97 | 0.4 | 0.57 | 0.7 |
| LightGBM | 98.18% | 0.89 | 0.83 | 0.86 | 0.91 |
| Multi-layer Perceptron | 97.52% | 0.91 | 0.72 | 0.8 | 0.86 |
| Bidirectional LSTM | 98.71% | 0.96 | 0.37 | 0.62 | 0.68 |

### *B. Discussion of Results*

The results from the experiments provide several key insights that are relevant for both choosing a deployment model and understanding the problem more broadly.

#### *1) LightGBM: Best Overall Performance*

LightGBM shows the best overall balance when looking at all the evaluation metrics, achieving a high accuracy of 98.18% and also the highest ROC-AUC score of 0.91. Its recall of 0.83 means that it correctly identifies 83% of fraudulent job postings in the test set, which is significantly better than many other models. The F1-score of 0.86 indicates that it performs
well even with class imbalance. These features make LightGBM the most suitable option for the system being deployed, where both false positives and false negatives have real-world implications.

*2) Bidirectional LSTM: Highest Accuracy, Lower Recall*

The Bi-LSTM model reaches the highest raw accuracy at 98.71%, but it has a very low recall of 0.37, which means it does not detect 63% of fraudulent job postings. This situation arises because of the class imbalance, as a model can classify most genuine postings correctly while still missing many fraudulent ones and still get a high overall accuracy. For a fraud detection purpose, this kind of performance is not ideal, because the undetected fraudulent postings are the main issue for users.

*3) Gradient Boosting: Precision-Recall Trade-off*

Gradient Boosting has the highest precision at 0.97 among all the models evaluated, but it only has a recall of 0.40, showing that it is very conservative in its classifications. This model only marks those postings that it is quite sure are fraudulent, leading to few false positives but a large number of false negatives. This trade-off might be suitable in situations where false fraud alerts would damage user trust, but it is not appropriate for platforms that want to thoroughly remove fraudulent postings.

*4) Random Forest: Reliable Ensemble Performance*

Random Forest has an accuracy of 97.04% and strong precision at 0.94, along with a ROC-AUC score of 0.83. However, its recall of 0.66 and F1-score of 0.77 show that a considerable number of fraudulent postings are not detected. As a model that is computationally manageable and interpretable, Random Forest serves as a practical option when there are deployment limitations that prevent the use of more complex models.

*5) General Observations*

Every model evaluated has an accuracy of over 95%, which confirms that TF-IDF features derived from the concatenated job posting text have strong discriminative power. The results highlight the precision-recall trade-off that comes with imbalanced classification and point out that it is important to choose deployment models based on the costs associated with false negatives versus false positives in the specific context of application the specific application context.

## VI. WEB APPLICATION DESIGN AND IMPLEMENTATION

A key contribution of this research is the creation and deployment of a working web application that uses the trained machine learning model, allowing end users to access it without needing technical knowledge in machine learning or data science. The application is created in Python using the Flask micro-framework and is made to allow real-time detection of fraudulent job postings.

### *A. Technology Stack*

The application is built on the following technology stack:

Table 4: Technology Stack of the Deployed Web Application

| Component | Technology | Purpose |
|---|---|---|
| Backend Framework | Flask (Python) | Web server and request routing |
| Machine Learning | scikit-learn | Model training and inference |
| Data Processing | pandas, NumPy | Data manipulation and preprocessing |
| Feature Extraction | TF-IDF Vectoriser | Text-to-feature transformation |
| Model Persistence | pickle (.pkl files) | Serialisation of trained model and vectoriser |
| Frontend | HTML/CSS (Jinja2) | User interface and form rendering |
| Session Management | Flask flash messages | Result communication to the user |

*B. System Architecture*

The system utilizes a traditional Model-View-Controller (MVC) architecture modified for a Flask web application. The operational process goes through the following stages:

• Stage 1 – User Input: The user goes to the home page of the application (/) and fills in job posting details in twelve structured form fields that correspond to the features outlined in Section 3.2.

• Stage 2 – Form Submission: The form is submitted through HTTP POST to the /submit endpoint, which activates the prediction pipeline.

• Stage 3 – Feature Construction: The application combines all submitted field values into one text string, maintaining the same feature engineering method used during model training.

• Stage 4 – TF-IDF Transformation: The combined text is converted into a sparse feature matrix using the pre-fitted TF-IDF vectoriser that is loaded from vectorizer.pkl.

• Stage 5 – Model Prediction: The feature matrix is provided to the trained classifier loaded from model.pkl, which produces a binary prediction (0 for genuine, 1 for fraudulent).

• Stage 6 – Result Display: The prediction result is shared with the user via Flask's flash messaging system, showing either "REAL JOB" or "FRAUDULENT JOB" on the interface.

*C. Application Code Analysis*

*1)* Model Loading: The model and vectoriser are loaded when the application starts, using Python's pickle module, which makes sure that the serialized artifacts are available in memory for all future prediction requests. This setup avoids the delay of loading model files for each request and allows for efficient real-time inference.

*2)* Route Definitions: The application has two routes. The root route (/) displays the input form template index.html. The /submit route handles POST requests from the form, runs the prediction pipeline, and re-displays the template with the prediction result as a flash message.

*3)* Input Feature Construction: The submit handler gathers twelve input fields using Flask's request.form.get() method, with safe default values being empty strings to manage missing or incomplete form submissions properly. The gathered values are combined with space separators to create the unified input representation, which includes title, location, department, profile, requirements, benefits, employment type, experience level, education, industry, function, and description.

*4)* Error Handling: The prediction pipeline is wrapped in a try-except block that captures any runtime errors and sends error information to the user through the flash messaging system. This design maintains application stability even when faced with unexpected inputs or issues in loading the model.

*D. User Interface Design*

The frontend interface uses Jinja2 templates to provide a structured data entry form with labelled input fields that correspond to each feature. When a user submits the form, the prediction result is displayed prominently through a flash message that is colour-coded to show the classification outcome, with red indicating fraudulent and green indicating genuine. The design of the interface aims to be intuitive for non-technical users which includes HR professionals, job seekers, and platform administrators.

## VII. LIMITATIONS AND FUTURE WORK

*A. Current Limitations*

The system that has been developed shows strong performance and practical utility but has several limitations that should be acknowledged.

*1)* The model is static as it is trained on a single snapshot of the EMSCAD dataset and lacks mechanisms for continual learning. As tactics for fraudulent postings evolve, the performance of the model may decrease without regular retraining on new data.

*2)* The current implementation only supports job postings in the English language. Detecting fraudulent postings in other languages would require capabilities in multilingual natural language processing.

*3)* The system does not analyze images or attachments. Many fraudulent job postings contain misleading logos, forged documents, or harmful attachments. The current system only processes text and cannot handle visual or binary content.

*4)* The best-performing model, which is LightGBM, still misses about 17% of fraudulent postings. In contexts where stakes are high, this false negative rate may be considered too high.

5) The web application does not include user authentication, session management, or logging, which would be necessary for a production environment.
6) The current Flask development server is not optimized for handling high-concurrency production workloads. For scaling up, a WSGI server such as Gunicorn or uWSGI would be needed, along with load balancing infrastructure.

### B. *Future Directions*

For future research and development, several extensions are proposed based on the current work.

1) The integration of transformer-based models like BERT, RoBERTa, or DeBERTa that are fine-tuned on the EMSCAD dataset is expected to improve both precision and recall by utilizing deep contextual language understanding.
2) The implementation of a continual learning framework that updates the model incrementally with newly identified fraudulent postings is necessary to adapt to changing deception strategies without needing full retraining.
3) Expanding the feature space to include visual elements like company logos, document scans, and website screenshots through convolutional neural networks for image analysis is important for multimodal detection.
4) Integrating model interpretability frameworks such as SHAP or LIME would provide users with explanations of fraud predictions, which could enhance trust in the system.
5) Developing a RESTful API endpoint is needed to allow seamless integration with third-party job boards and Applicant Tracking Systems, enabling programmatic screening of postings before publication.
6) Extending the preprocessing and feature extraction pipeline to include multilingual inputs using language-agnostic embedding models like LaBSE or multilingual BERT is essential for multilingual support.
7) Deploying an active learning strategy that identifies uncertain predictions and routes them for human review would progressively improve the model with examples labelled by experts.

## VIII. CONCLUSION

This research paper presents a comprehensive study on detecting fraudulent job postings using supervised machine learning techniques, culminating in the deployment of a Flask-based web application that is fully functional. The study shows that integrating natural language processing, specifically TF-IDF vectorization applied to concatenated multi-field job posting text, with ensemble and deep learning classifiers leads to effective fraud detection systems.

Among the evaluated models, LightGBM is the optimal choice for deployment, achieving an accuracy of 98.18% and an ROC-AUC score of 0.91 with a recall of 0.83. This represents the best balance between minimizing false positives and detecting fraudulent postings. The Bidirectional LSTM model achieves the highest raw accuracy of 98.71% but has significantly lower recall, highlighting the need for multi-metric evaluation in cases of imbalanced classification.

The web application developed translates academic model performance into a practical tool that users can access. It accepts structured job posting inputs, applies the trained pipeline in real time, and communicates binary classification results through an intuitive interface. The system effectively connects machine learning research with deployable fraud prevention solutions. The architecture of the Flask application shows that integrating scikit-learn models into web-based inference systems is feasible with minimal infrastructure overhead.

The findings affirm that AI-driven methods are viable and superior to manual or rule-based approaches for detecting fraudulent job postings, providing scalability, consistency, and potential for continuous improvement. Deploying such systems by online recruitment platforms can significantly reduce employment scams, protect job seekers, and enhance the integrity of digital recruitment ecosystems.

As fraudulent actors advance their deception strategies, developing adaptive, continually-learning, and multimodal detection systems is a critical area for future research. This work establishes a foundational framework for building such advanced systems.

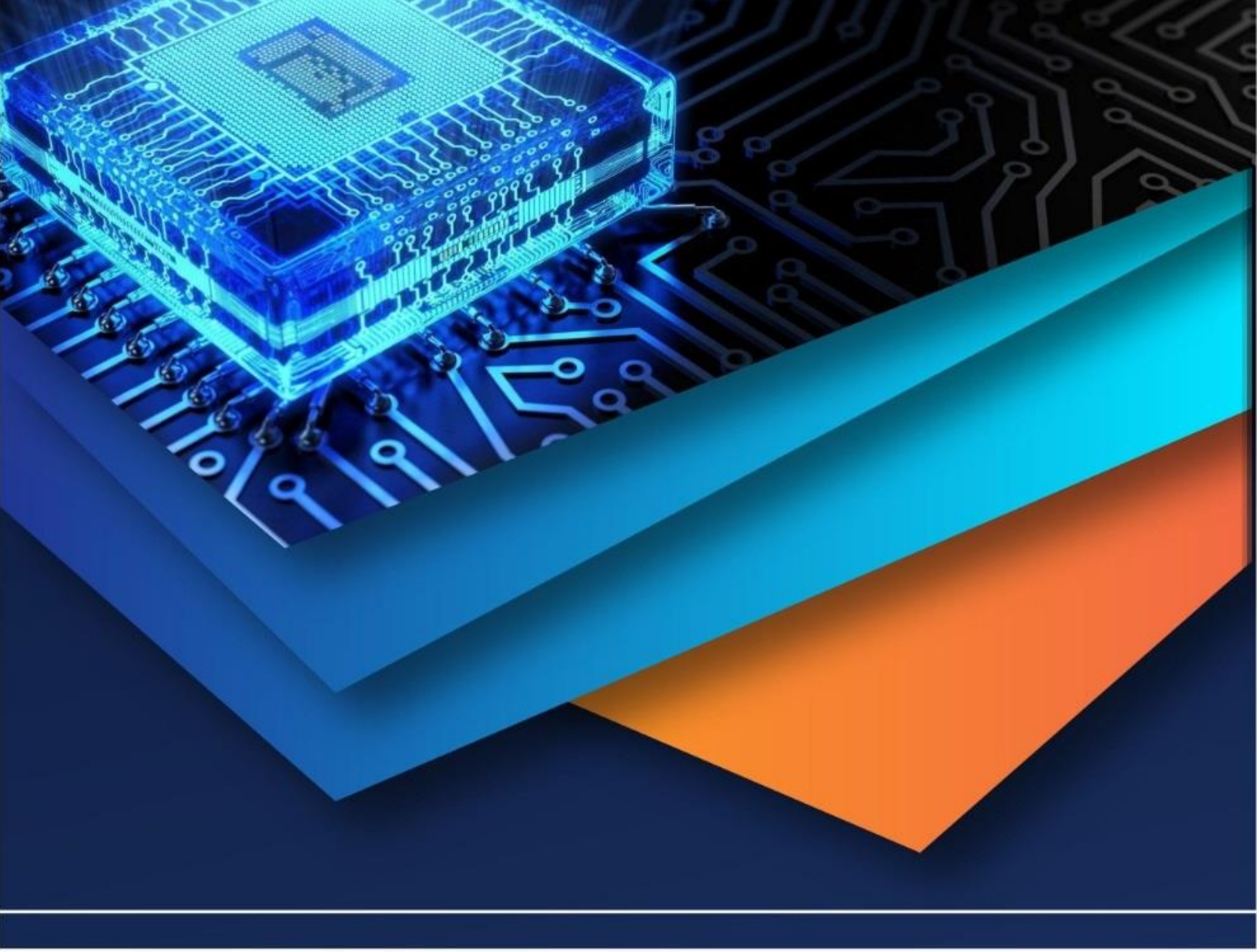
## REFERENCES

[1] Pillai, A. S. (2023). Detecting Fake Job Postings Using Bidirectional LSTM. arXiv preprint.
[2] Boka, M. (2024). Predicting Fake Job Posts Using Machine Learning Models. SSRN Electronic Journal.
[3] Kumar, S. (2025). A Review of Machine Learning and NLP-Based Detection of Fake Job Posts. Digital Manuscript Pedia.
[4] Naudé, M., & Kumar, S. (2023). A Machine Learning Approach for Detecting Fraudulent Job Postings. SpringerLink.
[5] Gulshan, P., et al. (2024). Fraudulent Online Job Advertisement Detection using Machine Learning Techniques. IJIRSET.
[6] Bhatta, S. (2025). Detecting Fake Job Postings using NLP and Machine Learning. GitHub Repository.
[7] Boka, M., & Gulshan, P. (2024). Fake Job Post Detection using Machine Learning and Deep Learning. IJAEM.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

[8] Sasidharan, A. (2023). Detecting Fake Job Postings Using Bidirectional LSTM. ResearchGate / IJRPR.
[9] Kumar, S., et al. (2025). Fake Job Post Detection using Machine Learning and Deep Learning. IJRPR.
[10] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
[11] Ke, G., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems (NeurIPS).
[12] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
[13] Chawla, N. V., et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.
[14] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.
[15] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python (2nd ed.). O'Reilly Media.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)