Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 2 (2024)

Detecting Fraudulent Job Postings: A Machine Learning Approach

[1]Ridam, [2]Riya Singh, [3]Tanuj Kumar Dhakery, [4]Prerna Chaudhary
[1,2,3]Student of Computer Science and Engineering
Meerut Institute of Engineering and Technology
Meerut, India
[4]Asst. Professor of Computer Science and Engineering
Meerut Institute of Engineering and Technology
Meerut, India

***Abstract*—**The advent of fraudulent job advertising poses a serious danger to job searchers' confidence and safety in the age of online employment marketplaces. Our research project develops an advanced machine learning-based system to predict the legitimacy of job profiles, which tackles this urgent problem. Utilizing state-of-the-art Python libraries like scikit-learn, NLTK, and spaCy, together with knowledge from earlier research, our project uses a thorough preprocessing pipeline to extract useful features from textual input. By combining several machine learning methods, such as Random Forest, Multinomial Naive Bayes, and Logistic Regression, with a Voting Classifier framework, we develop an ensemble model that can reliably distinguish between real and fake job listings. Our study strives to give job searchers a trustworthy tool to navigate the internet job market safely and confidently by methodically assessing and improving our models.

***Keywords*—**SpaCy, NLTK, Voting Classifier, Job, Accuracy

Introduction

The growth of internet job boards has completely changed the hiring process in today's digital world, providing never-before-seen access to a wide range of career options. But in the middle of all this convenience, there's a problem that lurks: a lot of fake job listings. These posts put job searchers at serious risk and compromise the recruiting ecosystem. To tackle this urgent problem, sophisticated machine learning methods must be used to differentiate between real and phony employment profiles.

Our research uses a variety of machine learning algorithms and advanced preprocessing approaches to create a strong prediction model for detecting bogus job posts.With the help of cutting-edge libraries like scikit-learn, NLTK, and spaCy as well as knowledge from earlier research, our project methodically preprocesses textual input, extracts relevant features, and trains ensemble models to reliably categorize job profiles.

Through the incorporation of algorithms like Random Forest, Multinomial Naive Bayes, and Logistic Regression into a Voting Classifier framework, we are able to better utilize the predictive capacity of these models in order to achieve improved accuracy in differentiating between genuine and fraudulent job posts. Our research enhances trust and security in online job markets by rigorous analysis and empirical assessment, equipping job seekers with the necessary skills to confidently navigate the digital employment environment.

Literature Review

Several deep neural network models, such as Text CNN, Bi-GRU-LSTM CNN, and Bi-GRU CNN, which are pre-trained using text datasets, were suggested to be used by Huynh [1] et al. They classified the IT job dataset as they worked. Using a Text CNN model with convolution, pooling, and fully connected layers, they trained a dataset of IT jobs. This model used pooling and convolution layers to train its data. The training weights were then transferred to the fully linked layer after being flattened. The softmax function was employed in this model as a classification method.

Numerous studies were conducted to determine the authenticity of job postings. Numerous studies are conducted to verify online job advertisements that are fraudulent. Job fraudsters were recognized by Videos [2] et al. as phony internet job advertisers. They discovered data on several actual, well-known businesses and organizations who created fictitious job postings or vacancy ads with bad intentions. They used a variety of classification methods, including the naive bayes classifier, random forest classifier, Zero R, One R, and others, in their experiments on the EMSCAD dataset. On the dataset, the Random Forest Classifier performed the best, with an accuracy of 89.5% in classification. They discovered that logistic regression did not perform well at all on the dataset. When the dataset was balanced and tested, one R classifier performed well. In their study, they attempted to identify the issues with the Online Recruitment Fraud (ORF) model and to address those issues with a variety of dominant classifiers.

Zhang [3] et al. suggested an autonomous fake detector model that uses text processing to differentiate between genuine and fraudulent news (containing articles, producers, and subjects). A bespoke dataset of news or articles from the PolitiFact website's Twitter account was utilized by them. The GDU diffusive unit model that was suggested was trained using this dataset. When information was received from several sources at once, the trained model outperformed an automated false detecting model.

According to P. Wang et al. [4], the model's tenets are the building blocks of neural networks that function similarly to a human brain. This enables a computer to compare two patterns and decide whether they are similar or distinct. A neuron is the structure that has certain properties and group categories.

A methodology to identify fraud risk in an online recruiting system was presented by Alghamdi [5] et al. They tested using a machine learning technique on the EMSCAD dataset. They worked on this dataset in three stages: selecting features, pre-processing the data, and applying a classifier to detect fraud. In order to maintain the overall text pattern, they eliminated html elements and noise from the data during the preprocessing stage. They successfully and efficiently decreased the amount of characteristics by using the feature selection approach. To identify phony job listings from the test data, an ensemble classifier with random forest was employed, and support vector machines were utilized for feature selection. With the aid of the majority voting approach, the Random Forest classifier—which appeared to be a tree-structured classifier—performed as an ensemble classifier. The accuracy of this classifier's classification in identifying phony job postings was 97.4%.

A novel algorithm has been developed by Sudhakar et al. [6] to distinguish between false information and real news. This study explores Support Vector Machine (SVM), logistic regression, and an innovative ensemble technique based on machine learning methods.

Methodology

A. About the Data

About 18,000 rows of job descriptions make up the Kaggle dataset, of which 800 are marked as fraudulent jobs. It provides a wealth of textual and meta-data on the employment vacancies that are posted. Along with textual descriptors like "title," "location," "salary_range," "company_profile," "description," "requirements," "benefits," "employment_type," "required_experience," "required_education," "industry," "function," and "department," important fields are "telecommuting," "has_company_logo," and "has_questions." This dataset is a useful tool for developing classification models that are intended to identify phony job postings. Online job marketplaces may be made more secure and trustworthy by using machine learning algorithms that are taught to identify patterns suggestive of fraudulent job profiles by utilizing both textual content and meta-information. The dataset's wide range of characteristics presents many chances for feature engineering and model building, which in turn helps to build reliable and precise classification models for spotting phony job postings.

B. Necessary Modules

1. Scikit-Learn:

Our project makes extensive use of the scikit-learn library, sometimes frequently referred to as sklearn, which is a complete Python machine learning toolkit. It provides a wide range of techniques for dimensionality reduction,

clustering, regression, and classification. We easily create a variety of machine learning models, including logistic regression, decision trees, random forests, and support vector machines, by utilizing sklearn's user-friendly API. Moreover, sklearn streamlines our workflow by offering tools for feature selection, data preprocessing, and model validation. For both inexperienced and seasoned practitioners, its comprehensive documentation and uniform interface make it an invaluable tool. Furthermore, smooth data manipulation and analysis are made possible by sklearn's interface with other Python libraries like NumPy and Pandas. Sklearn is essential to our research because it helps us train, adjust, and assess predictive models, which improves the precision and resilience of our system for classifying job profiles.

2. SpaCy:

Our research would not be possible without the advanced natural language processing (NLP) capabilities provided by the Python spacy module. We can effectively tokenize, lemmatize, and tag textual input with part-of-speech (POS) using spacy, which makes it easier to extract reliable features for our job profile classification system. Our NLP pipeline performs better thanks to its high accuracy and high performance pre-trained models for multiple languages. Furthermore, spacy provides smooth interface with other Python libraries, such NLTK and scikit-learn, allowing for more efficient workflows for text processing jobs. By utilizing spacy's entity recognition capabilities, we can enhance our feature set for model training by extracting significant entities from job descriptions and other textual data. The total effectiveness and precision of our machine learning models in differentiating between fictitious and authentic job profiles is enhanced by its proficient tokenization and dependency parsing methods.

3. NLTK:

Our project's main component, the Natural Language Toolkit (NLTK), offers a comprehensive set of resources and tools for natural language processing (NLP) applications. With the help of NLTK, we can efficiently preprocess textual input by providing features for tokenization, stemming, lemmatization, POS tagging, and syntactic parsing. By utilizing NLTK's vast corpus and lexicons, we augment our NLP pipeline with linguistic resources and domain-specific expertise. Our development process is streamlined by its user-friendly interface and extensive documentation, which allows for quick experimentation and iteration. Moreover, NLTK is a flexible option for multilingual NLP applications because to its strong support for a wide range of languages and vibrant community. By including NLTK into our project, we take use of its potent features extraction capabilities from job descriptions and other textual data, improving our machine learning models' interpretability and accuracy for classifying job profiles.

C. Preprocessing Steps

1. Removing useless columns

During our project's first preparation stage, we methodically remove the unnecessary "telecommuting," "has_company_logo," and "has_questions" entries from the dataset. These columns do not directly help distinguish between genuine and fraudulent job posts, hence we have decided that they are unimportant for our job profile prediction task. We simplify the dataset and concentrate only on the textual and categorical qualities that are more instructive for our machine learning models by eliminating these features. This feature selection procedure guarantees that our models are trained on the most relevant data and improves the effectiveness of our later data processing stages. Therefore, we maximize the computing resources and enhance the overall functionality of our job profile classification system by eliminating these superfluous data.

2. Merging Columns

During the preprocessing stage of our research, we combine multiple dataset columns to combine important textual and categorical variables. Title, location, income range, company profile, description, requirements, benefits, employment type, etc. We generate a single feature set by merging these fields, which records detailed information about every job posting. This consolidation guarantees that our machine learning models have access

to all pertinent data for precise prediction and streamlines later stages of the data processing process. Moreover, combining these columns makes it easier to comprehend the job profiles holistically, which improves our models' ability to identify trends and differentiate between authentic and fraudulent listings. All things considered, this step improves our dataset's coherence and completeness and provides a strong basis for further research and model training.

3. Cleaning text

To improve the quality of textual data, we apply text cleaning techniques throughout the data preprocessing phase of our research. To do this, the text must be stripped of any punctuation, special characters, and superfluous features like HTTP links. Our machine learning models are guaranteed to concentrate only on significant textual material by removing these uninformative elements. By removing noise and unnecessary information from the dataset, this phase is essential to enhancing the accuracy and interpretability of our models. To further improve the effectiveness of later NLP activities like tokenization and lemmatization, text cleaning expedites the processing of textual data. Preprocessing helps to improve our dataset overall, which makes our job profile classification system more capable of producing accurate and strong predictions.

4. Lemmatization

Lemmatization is the process of converting words into their lemma, or basic or root form, during the preparation phase of our project. This procedure improves the consistency and interpretability of our dataset by returning inflected terms to their common base form, so helping to normalize the textual data. By lemmatizing the text, we alleviate problems with repetition and word variation, allowing our machine learning models to concentrate on the semantic meaning beyond the surface variations in word forms. This is a critical step in enhancing the efficiency of later tasks in natural language processing, like sentiment analysis and feature extraction. Lemmatization, in general, results in a more logical and instructive textual representation, which eventually improves the functionality and precision of our job profile classification system.

5. Tokenization

In our research, tokenization is a crucial preprocessing step where we divide the textual input into discrete words or tokens. Tokenization helps with activities like feature extraction and further analysis by dividing the text into discrete parts. In this stage, we separated the unprocessed text into meaningful tokens by defining the boundaries between punctuation and words. Because this approach produces an organized representation of the data, it improves the ability of our machine learning models to read and evaluate the textual data. In applications like sentiment analysis and document classification, where precise prediction depends on a comprehension of the underlying language units, tokenization is vital. All things considered, through the process of tokenizing the text, we establish a basic framework that permits additional processing and examination of the textual data within our job profile classification system.

6. POS Tagging

Words in the textual data are given grammatical categories during the preprocessing stage of our study using part-of-speech (POS) tagging. Each word in this procedure is labeled with the appropriate part of speech, such as noun, verb, adjective, etc. Understanding the text's syntactic structure and semantic context is aided by the insightful language provided by POS tagging. More complex analysis and feature extraction methods, such locating important phrases or extracting pertinent entities from the text, are made possible by appending POS tags to the words. By improving the granularity and complexity of our textual data representation, this step makes machine learning model predictions more precise and nuanced. In the end, POS tagging adds linguistic information to our dataset that is critical to enhancing the functionality and readability of our job profile classification system.

7. Count Vectorizer

Count Vectorizer is a key component in our preparation workflow that converts textual input into numerical feature vectors. Using this method, a corpus of text documents is transformed into a matrix representation, with each row denoting a document and each column a distinct word from the corpus. With efficiency, Count Vectorizer tokenizes the text, eliminates stop words, and creates a vocabulary of terms found in the dataset. The next step is to count how often each word appears in each text, producing a sparse matrix representation in the process. In order to feed text data into machine learning algorithms—which need numerical inputs for model training—this numerical representation is essential. Through the use of Count Vectorizer, we are able to efficiently transform the textual characteristics that are retrieved from job ads into a format that is comprehensible for our classification models. This allows us to accurately forecast both genuine and fraudulent job profiles..

Models Used

1. Logistic Regression

Logistic Regression is a key machine learning technique in our project, and it performs remarkably well with an accuracy of 96.77%. This technique is ideally suited for jobs involving binary classification, such as differentiating between authentic and fraudulent job profiles. By using a sigmoid function to transfer input data to a probability score, Logistic Regression models the likelihood that a specific job posting is part of a given class. Iteratively determining the ideal coefficients to minimize the logistic loss function and attain better classification performance is accomplished by utilizing gradient descent optimization. Logistic regression is an excellent tool for capturing linear correlations between variables and class labels because of its interpretability, simplicity, and efficiency. As a result, it plays a vital role in enhancing the overall accuracy and dependability of our job profile prediction system.

2. Multinomial Naïve Bayes

Our research depends on the Multinomial Naive Bayes algorithm, which has an astounding accuracy of 96.77%. Since this algorithm performs exceptionally well on text classification tasks, it was a no-brainer for our job profile prediction system. Using the Bayes theorem, Multinomial Naive Bayes calculates the probability of each class given the input features, assuming that features are independent of each other. Despite its simple assumptions, Multinomial Naive Bayes often produces surprisingly impressive results in practice, especially when working with large, high-dimensional datasets like ours. By storing text data quickly and utilizing phrase frequency, Multinomial Naive Bayes effectively distinguishes between genuine and fraudulent job profiles, significantly enhancing the overall accuracy and resilience of our classification system.

3. Decision Tree

With an astounding accuracy of 94.47%, the Decision Tree algorithm is a key component in our project's ability to forecast the veracity of job profiles. With the goal of maximizing the purity of each resulting subset, Decision Trees are strong and intuitive models that recursively split the feature space based on the most informative qualities. Decision Trees provide a hierarchical structure that makes the classification process easier to understand and interpret by repeatedly dividing the dataset into homogeneous groups. Though they are prone to overfitting, methods like pruning and restricting tree depth serve to lessen this problem and guarantee strong generalization to unknown data. Decision Trees are a valuable tool for differentiating between genuine and fraudulent job ads due to their capacity to process both numerical and categorical variables. This characteristic enhances the overall performance and dependability of our classification system.

4. Random Forest

The Random Forest method proves to be an effective tool for predicting job profiles in our project, with an accuracy of 96.31%. Using the power of several decision trees, the ensemble learning technique Random Forest creates forecasts that are reliable. Random Forest reduces overfitting and improves generalization performance by training several decision trees on arbitrary subsets of the data and aggregates their predictions by voting or

average. Our objective of differentiating between authentic and fraudulent job profiles is a good fit for this algorithm because it is proficient in managing high-dimensional data and identifying intricate relationships between elements. Furthermore, our classification method is scalable and effective because of its built-in parallelism, which makes training on big datasets efficient. Our project's foundation method, Random Forest, stands out for its remarkable accuracy and resilience, which support the dependability and functionality of our job profile prediction model.

5. Support Vector Classification

The Support Vector Classification (SVC) algorithm plays a crucial role in our project and has an impressive accuracy of 93.08%. SVC is a potent supervised learning technique that seeks to maximize the margin between classes while identifying the ideal hyperplane for dividing data points into distinct classes. SVC is appropriate for difficult classification applications because it can handle non-linear decision boundaries by translating the input data into a higher-dimensional space using kernel functions. SVC is resistant to overfitting and, when properly adjusted, can generalize effectively to previously unseen data despite its computational complexity. Our job profile prediction system is more accurate and reliable because of SVC's capacity to handle high-dimensional data and nonlinear correlations. It also offers important insights into differentiating between genuine and fraudulent job ads.

6. AdaBoost Classifier

With an accuracy of 94.00% in our project, the AdaBoost Classifier proves to be a potent machine learning algorithm. Adaptive Boosting, or AdaBoost, is an ensemble technique that builds a powerful predictive model by combining several weak learners, usually decision trees. AdaBoost focuses on samples that are challenging to classify and enhances the overall performance of the model by iteratively modifying the weights of misclassified cases. Every weak learner is taught one after the other, with each new model focusing on the data points that the earlier models misclassified. AdaBoost builds a strong ensemble classifier that successfully captures intricate correlations in the data through this iterative process. AdaBoost considerably improves our job profile prediction system's accuracy and robustness by handling imbalanced and noisy datasets. This enhances the system's overall performance and dependability.

7. XGBoost Classifier

With an accuracy of 95.39%, the XGBoost Classifier is a machine learning method that performs quite well in our project. XGBoost, also known as eXtreme Gradient Boosting, is a potent ensemble learning method that combines the benefits of gradient boosting with an extremely scalable and efficient implementation. With the help of XGBoost, decision trees are iteratively fitted to the residuals of earlier trees, producing powerful ensemble models that are excellent at capturing intricate relationships in the data. Its sophisticated regularization methods, like tree trimming and shrinkage, reduce overfitting and improve generalization. Furthermore, XGBoost is a flexible and adaptable option for a range of classification applications because to its capacity to accommodate missing variables and include custom loss functions. Our project demonstrates the usefulness and dependability of XGBoost Classifier in practical applications by showing a considerable increase in the accurate prediction of job profile authenticity.

8. KNN Classifier

Our study uses the K-Nearest Neighbors (KNN) Classifier, which delivers an 87.55% accuracy rate in predicting the veracity of job profiles. The majority vote of an instance's closest neighbors in the feature space is used by KNN, a straightforward yet powerful non-parametric technique, to classify it. KNN relies just on the closeness of instances in the feature space to produce predictions as it considers the similarities between data points, eliminating the need for explicit model training. Although KNN is straightforward, it can achieve good results in a variety of situations, especially if the dataset cannot be linearly separated or if the decision boundary is complicated. Overly complex data sets or unequal class distributions, however, could cause it to perform worse.

9. Voting Classifier

In our work, the Voting Classifier—which combines the collective intelligence of three excellent algorithms—emerges as a potent ensemble learning technique: Random Forest, Multinomial Naive Bayes (MultinomialNB), and Logistic Regression. The Voting Classifier, which combines the predictions of these many models using a "soft" voting mechanism, achieves an impressive accuracy of 97.23%. The final classification decision is based on the likelihood scores produced by each individual model. The Voting Classifier efficiently utilizes the advantages of each component model by combining the outputs of several classifiers, producing better prediction accuracy and robustness. By using an ensemble technique, we may reduce the possibility of individual model biases or weaknesses while simultaneously improving the accuracy of our job profile prediction system. All things considered, the Voting Classifier is a key component of our work and demonstrates how ensemble learning can be used to achieve very high classification accuracy and consistency.

Implementation

1. import the VotingClassifier class from the ensemble module of the sklearn library by running the command "from sklearn.ensemble import VotingClassifier".

2. Base Model Initialization: - The following three base models are initialized and added to the estimator list:A logistic regression model for binary classification problems is provided by LogisticRegression(). MultinomialNB(): An effective multinomial Naive Bayes classifier for text classification applications. A random forest classifier that can handle complicated datasets and identify non-linear correlations is called RandomForestClassifier().

3. Building a Classifier for Voting: VotingClassifier(voting='soft', estimators=estimator): Using the given base models estimator and the voting method set to 'soft', a Voting Classifier is constructed.

4. Training of Models:vot_model.fit(y_train, X_train): Using the training data X_train and y_train, where X_train stands for features and y_train for target labels, the Voting Classifier is trained.Essentially, the code segment builds a Voting Classifier using random forest, logistic regression, and multinomial Naive Bayes as the foundational models. The probabilistic predictions of these models are combined using the 'soft' voting approach to create an ensemble model vot_model that can accurately predict whether a job profile is legitimate or fraudulent..

Results Obtained

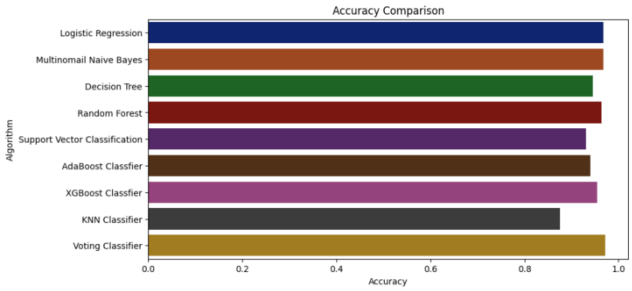

Figure 1 Accuracy Comparison Chart

```
Logistic Regression --> 0.967741935483871
Multinomail Naive Bayes --> 0.967741935483871
Decision Tree --> 0.9447004608294931
Random Forest --> 0.9631336405529954
Support Vector Classification --> 0.9308755760368663
AdaBoost Classfier --> 0.9400921658986175
XGBoost Classfier --> 0.9539170506912442
KNN Classifier --> 0.8755760368663594
Voting Classifier --> 0.9723502304147466
```

Figure 2 Accuracy values

In this study, we examine various machine learning models used to differentiate between authentic and fraudulent job profiles. With accuracies ranging from 93.08% to 96.77%, the models that were assessed include Support Vector Classification (SVC), Random Forest, Decision Tree, and Logistic Regression. Furthermore, ensemble methods with accuracies higher than 94%, including the AdaBoost Classifier and XGBoost Classifier, also produce encouraging results.

But with the maximum accuracy of 97.23%, the Voting Classifier proves to be the most successful. Using a "soft" voting mechanism, this ensemble method combines the predictions of Random Forest, Multinomial Naive Bayes, and Logistic Regression. Through the combination of the probability scores produced by each separate model, the Voting Classifier leverages the various capabilities of its constituent parts, leading to enhanced predictive performance and resilience.

Our results highlight how ensemble learning may significantly increase classification accuracy, especially in situations where individual models may have biases or restrictions. The Voting Classifier is the main model in our study article, and its remarkable accuracy of 97.23% highlights its importance in precisely predicting the legitimacy of job profiles in online marketplaces.

	precision	recall	f1-score	support
Real	0.9645	1.0000	0.9819	163
Fake	1.0000	0.8889	0.9412	54
accuracy			0.9724	217
macro avg	0.9822	0.9444	0.9616	217
weighted avg	0.9733	0.9724	0.9718	217

Figure 3 Classification Report

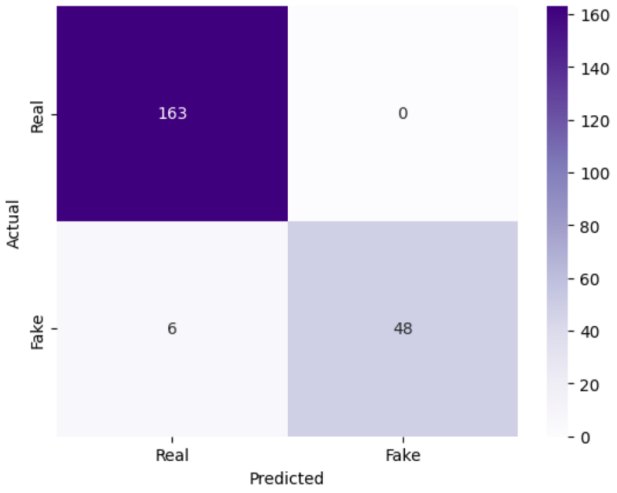

Figure 4 Confusion Matrix

Following the creation of the most realistic model, a web application is made that allows users to give details of the job and the application will predict whether the job is real or fake.

Figure 5 Web app predict page

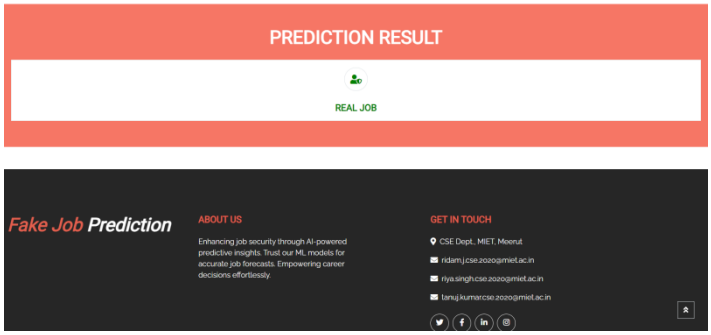

Figure 6 Result page

Algorithm	Accuracy
Logistic Regression	96.77%
Multinomial Naïve Bayes	96.77%
Decision Tree	94.47%
Random Forest	96.31%
SVC	93.08%
AdaBoost Classifier	94.01%
XGBoost Classifier	95.39%
KNN Classifier	87.55%
Voting Classifier	97.23%

Table 1 Accuracy Comparison in percentage

Conclusion

To conclusion, this research study offers a thorough examination of machine learning methods for spoofing employment profiles. By utilizing a wide range of algorithms, including Random Forest, Decision Trees, and Logistic Regression, as well as ensemble techniques like the Voting Classifier, we have created a reliable prediction model with excellent accuracy. We have carefully prepared the dataset for efficient model training by performing lemmatization, feature consolidation, and text cleaning. Our results show that ensemble learning can effectively improve classification accuracy; this is especially true for the Voting Classifier, which reached an astounding accuracy of 97.23%. Our effort contributes to protecting job seekers from potential risks and demonstrates the effectiveness of machine learning in preventing online deception by tackling the urgent problem of fake job ads. Subsequent investigations could examine the amalgamation of profound learning

frameworks with instantaneous data streams to enhance prognostic efficacy and flexibility in response to dynamic patterns in fraudulent job advertisements.

References

[1] Tin Van Huynh1, Kiet Van Nguyen, Ngan Luu-Thuy Nguyen1, and Anh Gia-Tuan Nguyen, "Job Prediction: From Deep Neural Network Models to Applications", RIVF International Conference on Computing and Communication Technologies (RIVF), 2020.

[2] S. Vidros, C. Kolias , G. Kambourakis ,and L. Akoglu, "Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset", Future Internet 2017.

[3] Jiawei Zhang, Bowen Dong, Philip S. Yu, "FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network", IEEE 36th International Conference on Data Engineering (ICDE), 2020.

[4] Thin Van Dang, Vu Duc Nguyen, Kiet Van Nguyen and Ngan Luu-Thuy Nguyen, "Deep learning for aspect detection on vietnamese reviews" in In Proceeding of the 2018 5th NAFOSTED Conference on Information and Computer Science (NICS), 2018.

[5] B. Alghamdi, F. Alharby, "An Intelligent Model for Online Recruitment Fraud Detection", Journal of Information Security, 2019.

[6] M. Sudhakar and K. P. Kaliyamurthie, "Efficient Prediction of Fake News Using Novel Ensemble Technique Based on Machine Learning Algorithm," in Information and Communication Technology for Competitive Strategies (ICTCS 2021).

[7] Source http://emscad.samos.aegean.gr/.

[8] Kaggle https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction.