Course number: CS 5593 – Online Sections 995-999

Course name: Data Mining

Semester, year: Fall, 2024

Title of your project: Detection of Fraudulent Job Listings

Names and email addresses of group members:

1. Sreeja Reddy Muchantula: Sreeja.Reddy.Muchantula-1@ou.edu

2. Sanjana Reddy Ella: sanjana.reddy@ou.edu

3. Tushar Jayendra Mhatre: Tushar.Jayendra.Mhatre-1@ou.edu

Detection of Fraudulent Job Listings

Sreeja Reddy Muchantula
Computer Science
University of Oklahoma
Norman, Oklahoma, USA
sreeja.reddy.muchantula-1@ou.edu

Sanjana Reddy Ella
Computer Science
University of Oklahoma
Norman, Oklahoma, USA
sanjana.reddy@ou.edu

Tushar Jayendra Mhatre
Data Science and Analytics
University of Oklahoma
Norman, Oklahoma, USA
tushar.jayendra.mhatre-1@ou.edu

ABSTRACT

In recent years, the rise of fraudulent job listings on online platforms has become a growing concern, leading to identity theft and significant risks for both job seekers and companies. With the rise of online job portals, it has become increasingly challenging for users to distinguish between legitimate and fraudulent postings. This project aims to address this issue by developing a system to detect fraudulent job listings through the application of machine learning techniques. It combines feature selection, data preprocessing and multiple classification models to accurately distinguish between legitimate and fraudulent job postings. The system is built using a publicly available "Fake Job Postings" dataset, which includes various job attributes such as title, description, salary range and company profile. Preprocessing steps, such as imputation of missing values, TF-IDF vectorization for text features and feature selection using algorithms like Random Forest and XGBoost, are employed to optimize model performance. Five classification models Logistic Regression, Decision Tree, Random Forest, XGBoost and Support Vector Machine (SVM) are trained and evaluated on the dataset with cross-validation used to prevent overfitting. Performance metrics such as accuracy, precision, recall, F1-score, specificity and AUC are computed to assess model effectiveness. The final system is a user-friendly web application where users can input job posting details and receive predictions of whether a job listing is fraudulent or legitimate.

KEYWORDS

Machine Learning, Classification, Feature Selection, TF-IDF, Logistic Regression, Decision Tree, Random Forest, XGBoost, Support Vector Machine (SVM), Model Evaluation, Cross-Validation, Web Application.

ACM REFERENCE FORMAT:

Sreeja Reddy Muchantula, Sanjana Reddy Ella and Tushar Jayendra Mhatre. 2024. Detection of Fraudulent Job Listings.

1 INTRODUCTION

The rise of online job portals has made it easier for job seekers to find opportunities, but it has also led to the increase in the number of fraudulent job postings. These fake listings not only waste time and resources but can also result in financial losses and harm to job seekers. Fraudulent job postings often feature misleading descriptions, fake company profiles and unrealistic salary expectations, with the intent to deceive applicants into providing personal information, paying fees or engaging in scams. The scale of this issue continues to grow, especially on large job boards and social media platforms where posts are not always carefully reviewed [2].

Detecting these fraudulent postings manually is an inefficient and labor-intensive task. The application of machine learning techniques automates this process. Many of the previous approaches suffer from issues such as overfitting due to static datasets or lack of robust feature engineering to notice the small details that differentiate fraudulent from legitimate postings [6].

Our project combines advanced preprocessing techniques, such as TF-IDF vectorization for text attributes, with feature selection methods and a variety of classification models to improve detection accuracy. By addressing the gaps in existing research, our work aims to offer a more reliable and scalable solution for identifying fraudulent job listings. We employ five different classification model Logistic Regression, Decision Tree, Random Forest, XGBoost, and Support Vector Machine (SVM) and evaluate them based on various performance metrics such as accuracy, precision, recall, F1-score, and AUC. The results are then integrated into a user-friendly web application, where users can input job details and receive predictions about the legitimacy of the posting in real-time.

This paper has six sections including this introduction. Section II provides a review of related work on detecting fraudulent job postings using machine learning techniques. Section III describes the dataset used, the data preprocessing steps, and the experiments carried out. Section IV details the methodology, including model selection, model training and model evaluation. In Section V, the implementation details are discussed. Followed by the user guide manual, results in next sections. Finally, Section VIII presents the conclusion and future work.

2 RELATED WORK

Various methods were proposed to identify false listings on online job portals. Early approaches primarily relied on manual verification and rule-based systems, which proved to be inefficient and unsustainable given the volume and complexity of job data [3].

One of the studies utilized a combination of textual analysis and metadata to identify patterns associated with fraudulent postings [3]. It explored the use of basic classifiers such as Naive Bayes and SVM to classify job postings based on features extracted from job descriptions and company information. While effective to an extent these models often struggled with high-dimensional data and the nuanced language used in fraudulent postings.

More recent studies have emphasized the importance of advanced feature engineering techniques. It demonstrated how feature selection could significantly impact the performance of machine learning models [2]. They used ensemble methods like Random Forest and Gradient Boosting to manage the feature space more effectively, reducing overfitting and improving model generalizability. However, these approaches did not address the issue of real-time data processing and were often limited to static datasets.

A comparative study of several machine learning algorithms, including newer deep learning models to detect fraudulent job postings highlighted the potential of convolutional neural networks (CNNs) in capturing textual patterns more effectively than traditional machine learning models [9]. Nonetheless, the real-time application of such models was not explored, which is critical for practical implementations in job portals.

Recent innovations have also explored the integration of user feedback into the detection process which introduced a dynamic learning system where user-reported data was used to continuously train the model, adapting to new fraud tactics [6]. This approach showed promise but required robust mechanisms to filter and verify user input to prevent data contamination.

Despite these advancements, there remains a significant gap in developing an efficient, scalable system that can process real-time data and adapt to evolving fraud tactics. This project aims to fill these gaps by employing a comprehensive data preprocessing strategy that includes TF-IDF vectorization to enhance the textual analysis capabilities of our models and utilizing a combination of traditional and advanced machine learning models to improve detection accuracy and handle both linear and non-linear patterns. Also, developing a user-friendly web application that allows for real-time job posting verification and incorporates mechanisms for continuous model training based on user feedback.

3 PROPOSED WORK

In this section the proposed approach for detecting fraudulent job postings using machine learning is described. This work involves several key components: dataset selection, data preprocessing, model selection, and performance evaluation. We aim to develop a robust and scalable system capable of detecting fraudulent job postings in real-time. The system integrates various preprocessing techniques, feature selection methods, and classification models to ensure high detection accuracy and generalization.

3.1 DATASET

The “Fake Job Postings” dataset from Kaggle serves as the primary data source for this project. This dataset contains job listings with labeled categories: fraudulent and legitimate postings. Each job posting includes textual information such as the job description, company profile, salary range, and other metadata. The dataset is balanced, containing both legitimate and fraudulent job postings, which is essential for effective model training. The dataset is divided into training and testing subsets using stratified sampling, ensuring that the proportions of fraudulent and legitimate postings remain consistent in both subsets.

3.1.1 Data Preprocessing Data preprocessing is essential to ensure that the data is clean, consistent and ready for feature extraction and model training. The below steps are to address missing values, transform textual data into numerical representations, handle high-cardinality categorical features and prepare the dataset for next steps.

- **Exploratory Data Analysis (EDA):** Exploratory Data Analysis (EDA) was performed to explore the structure and characteristics of the dataset. The goal of EDA is to uncover patterns in the data, detect anomalies, and gain insights to identify the necessary preprocessing steps to prepare our dataset [6]. It is observed that the dataset has a very high-class imbalance as shown in Figure 1, with legitimate job postings being more common than fraudulent ones. This can lead to models being biased toward the majority class, which would affect the model’s ability to detect fraudulent postings. To mitigate this issue, we decided to implement Stratified K-Fold Cross-Validation during model training [11].

  Stratified K-Fold cross-validation ensures that each fold in the cross-validation process contains approximately the same proportion of fraudulent and non-fraudulent job postings as the entire dataset. This approach helps in preventing the model from being biased ensuring a more balanced evaluation across different subsets of the data. It divides the dataset into K equal-sized subsets (or folds). In each iteration one-fold is used as the validation set and the remaining K-1 folds are used for

training the model. Stratified K-Fold ensures that each fold maintains the proportional distribution of classes found in the overall dataset.

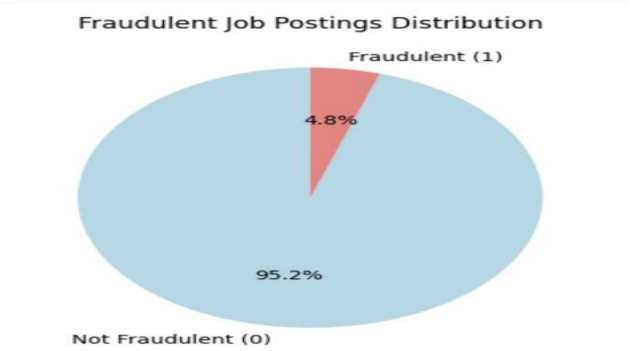

Figure 1: Pie Chart showing the distribution of legitimate and fraudulent jobs in the dataset

- **Dropping the ID Column:** The job_ID column in the dataset acts as a unique identifier for each job posting. Since this column provided no useful information for classification it was dropped during preprocessing. Removing unnecessary columns is an important step in reducing complexity and preventing overfitting, as the ID column does not contribute to distinguishing fraudulent from legitimate postings.
- **Imputing Missing Values:** Several features in the dataset such as salary range and department contained missing values. Rather than removing these rows which could lead to significant data loss, missing values were imputed with placeholders like "Unknown" or "Not Provided" [1]. This choice was based on the rationale that missing information in critical fields like salary and department could serve as an important indicator of a fraudulent job posting.
- **Vectorization of Text Attributes**: Many features such as job description and company profile are text based. Since machine learning models like Logistic Regression require numerical inputs and models like Decision Trees and Random Forest may face challenges, as they will need a huge amount of time to train the data and there will also be a problem of overfitting it is essential to convert the text data into a format that can be interpreted by classification algorithms [6].

  TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that represents the importance of a term in a particular paragraph, based on how often the term appears in that paragraph (TF) and how rare or common the term is across the entire collection of texts/paragraphs (IDF). The TF-IDF score is basically the product of these 2 terms. The words with highest TDF scores are selected as they are not only frequent in a single document but are also discriminative, meaning they appear in fewer texts across the dataset, making them more informative for classification or other tasks.

  'TfidfVectorizer' function from the sklearn package is used. It fits all the text columns to this function and builds a vocabulary (list of all unique words in the corpus) and assigns each word an index. Common words can be removed and most important words from the vocabulary can be selected based on term frequency or TF-IDF score. Based on the EDA results and analyzing the frequency of words in fraudulent and non-fraudulent postings, common words from each class can be identified. These frequently occurring words can be excluded from vectorization. So that the focus will only be on words with higher informational value.

- **Factor Collapsing of Highly Cardinal Attributes:** Features like job title, location and department had many unique values making them highly cardinal. It is important to handle categorical features with high cardinality as they can increase the dimensionality of the feature space, which in turn can result in overfitting. To address this, factor collapsing was applied, where the most frequent categories were retained, and less frequent categories were grouped into an "Other" label. This helps to ensure that the focus remains on the most informative categories.
- **One Hot encoding:** One-hot encoding is a method of representing categorical data as binary vectors to make it interpretable for machine learning algorithms [11]. Categorical variables such as job title, location and industry, are textual and cannot be directly interpreted by machine learning models like Logistic Regression, Decision Trees and XGBoost. These algorithms require numerical input.

  In one-hot encoding, each unique category in a categorical column is converted into a new binary feature. For each row in the dataset if the row belongs to a particular category, the corresponding binary feature is assigned a value of 1. If the row does not belong to the category, the binary feature is assigned a value of 0.
- **Outlier Detection:** Initially outlier detection was performed to identify and handle extreme values that may not represent legit job postings [7]. Both IQR method and Z score method was used. The IQR method detected many outliers, but it was difficult to establish a reliable connection between the outliers and fraudulent job postings. This method works well for symmetric distributions but is less effective if the data is skewed, which was the case with our dataset. The Z-score method measures how far data points are from the mean

in terms of standard deviations. A threshold is set to identify outliers. While the Z-score method detected fewer outliers compared to the IQR method, it still flagged a considerable number of non-fraudulent postings as outliers. This suggests that outlier detection alone is insufficient for distinguishing fraudulent job postings.

- **Feature Selection:** The final dataset had over 100 features after preprocessing, feature selection was performed to identify the most important features for classification [8]. Decision Tree, Random Forest, and XG Boost were used to calculate the feature importance scores. The importance factor in decision trees, random forests is measured by Gini importance which describes how much we are splitting on a feature to reduce uncertainty. The feature importance in XGBoost is also based on the number of times a feature is used to split across the tree and the information gain the feature provides in classification. High scores indicate that the feature has more say on the model's decision making. Based on the feature importance scores and scale of scores in all 3 models, we set a threshold of 0.0015. The features which have a score less than this threshold in all 3 models are eliminated, leaving the top 60 features for training. This step helps reduce overfitting by eliminating irrelevant or redundant features.

4 METHODOLOGY

After completing the data preprocessing steps the next task was to build machine learning models. It can be divided into 3 main tasks. The model selection, model training and model evaluation. Based on the characteristics of the dataset such as class imbalance, mix of categorical, numerical data we chose five classification models.

4.1 MODEL SELECTION

The five selected classification models are:

- Logistic Regression
- Decision Trees
- XG Boost
- Random Forest
- Support Vector Machine

4.1.1 Logistic Regression: It is a linear classification model and is used for binary classification tasks. Here our project is also based on classifying the job posting as fraudulent or legitimate. So, it works by estimating the probability that a given input belongs to a particular class using the sigmoid function. The logistic function outputs a value between 0 and 1, which is interpreted as the probability of the legit job and fraudulent job posting. If the probability exceeds a set threshold the instance is classified as fraudulent.

Logistic Regression was selected as one of the algorithms as it is effective when there is a linear relationship between the features and the target variable. Our preprocessed dataset contains binary and numerical features making them suitable for Logistic Regression [10][13].

$$\sigma(\mathrm{Z}) = \frac{1}{1 + \mathrm{e}^{-\mathrm{z}}} \tag{1}$$

4.1.2 Decision Trees: It splits the data based on feature values to make decisions. The tree is built by recursively selecting the feature that maximizes information gain or Gini impurity.

$$E(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i \tag{2}$$

$$Gini\ (S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{3}$$

After preprocessing steps like TF-IDF vectorization and one-hot encoding, categorical variables are transformed into numerical representations, which makes them directly usable in a Decision Tree. The model can split these features to make predictions [7][8][3].

4.1.3 XG Boost: It builds a series of decision trees in a sequential manner. Each tree attempts to correct the errors made by the previous tree and the loss function is used to control the complexity of the tree.

$$L_{xgb} = \sum_{i=1}^{N} L\left(y_i,\ F(x_i)\right) + \sum_{m=1}^{M} \Omega\,(h_m) \tag{4}$$

$$\Omega(\mathrm{h}) = \gamma T + \frac{1}{2}\lambda\|\mathrm{w}\|^2 \tag{5}$$

XGBoost works similarly to Decision Trees and Random Forest but with an advantage of boosting. We chose this model as it handles complex, high-dimensional datasets, where there is a mix of categorical, numerical, and textual data. XGBoost performs well in situations where there is a high degree of non-linearity, as it iteratively builds trees that reduce residual errors [1].

***4.1.4 Random Forest*:** It combines multiple Decision Trees to make predictions. It helps to mitigate overfitting and is suitable for our dataset which contains categorical and numerical features.

Like Decision Trees, Random Forest can handle both numerical and categorical features. Each tree in the forest is trained on a random subset of features and instances, reducing the likelihood of overfitting compared to a single decision tree. Random Forest also automatically computes feature importance, which was useful in the feature selection step of our project [10].

***4.1.5 Support Vector Machine*:** It finds the hyperplane to separate the classes in a high dimensional space. It is chosen for its effectiveness in handling high-dimensional feature spaces, which is typical of text data. It can find the optimal decision boundary between classes which is helpful in detecting fraudulent jobs when there are complex relationships between features [4].

Before proceeding to model training step, the class imbalance issue was addressed using SMOTE (Synthetic Minority Over-sampling Technique) [5]. SMOTE works by generating synthetic datapoints for the minority class by creating new examples that are like existing ones by selecting two or more similar instances from the minority class and generating new points along the line segments joining them. By oversampling the minority class, SMOTE helps to balance the class distribution and reduce the bias toward the majority class, allowing the model to focus more on the fraudulent postings.

It was performed after preprocessing as SMOTE works best with numerical data and the previous preprocessing steps have transformed all features into numerical values so now the dataset is numerical.

4.2 MODEL TRAINING

The models were trained using the training subset of the dataset, with an 80:20 split for training and testing. The hyperparameter tuning was performed using GridSearchCV function in python with 10-fold cross validation. The hyperparameters for each model are:

- For Logistic Regression hyperparameters such as regularization strength(C), solver and penalty are used to fine tune the model's performance. The inverse of regularization strength parameter controls the strength of the regularization with smaller values leading to underfitting and larger values leading to overfitting. The solver specifies the optimization algorithm used to minimize the loss function. We used 'liblinear' and 'saga' which are effective for small to medium sized datasets and supports L1, L2 regularization.
- For Decision Trees hyperparameters such as max_depth, min_samples_split, min_samples_leaf, criterion are used. The max_depth is used to control the complexity of the tree, min_samples_split is the minimum number of samples required to split an internal node and min_samples_leaf is the minimum number of samples required to be at leaf node. Criterion is used to measure the quality of a split and we have used gini and entropy criterion.
- For XG Boost hyperparameters such as n_estimators, learning_rate, max_depth and subsample are used. The n_estimators specifies the number of trees, learning_rate controls the step size during optimization, max_depth controls the maximum depth of the individual trees and subsample is the fraction of samples to use for each boosting round.
- For Random Forest the hyperparameters used are n_estimators, max_depth, min_samples_split and min_samples_leaf. The n_estimators specify the number of trees, max_depth to control the depth of each individual tree, min_samples_split and min_samples_leaf parameters control how nodes are split and how many samples are required at each leaf node.
- For Support Vector Machine the hyperparameters used are C, kernel and gamma. The 'C' parameter controls the trade-off between achieving low training error and low testing error, kernel is used to map the data into higher dimensional space and gamma defines how fair the influence of a single training point reaches.

Cross validation ensures that the model is trained on different subsets of the data and improves its generalization performance. GridSearch evaluates all combinations of hyperparameters and selects the best set based on the model's performance.

4.3 MODEL EVALUATION

For model evaluation, instead of using the results from grid search instance, we extracted the best performing model from the results. As GridSearchCV in python will automatically evaluate the best performing hyper-parameters from the grid search, we extracted the respective best performing model from each grid. We then used these models to perform further tests on the unseen testing data. We split this testing data for another 5 folds by making sure the class distribution is same as the original data with the help of Stratified sampling. Then all 5 models were tested on the testing Idea across 5 folds, all the performance like accuracy, precision, F1 score, Recall, Specificity and AUC were calculated across all 5 folds. These results were then used for the statistical analysis in-order to determine the best performing models

4.3.1 Visual Evaluation of model results:

- Logistic Regression shows moderate performance with accuracy around 84% but has a precision of 0.21-0.22 and F1 scores of 0.33-0.34 even with high recall of 0.73-0.83. The AUC was between 0.85 and 0.88.
- Decision Tree outperformed Logistic Regression in F1 scores of 0.44–0.60 and had higher recall of 0.69–0.8 with AUC values between 0.89 and 0.93.
- SVM had a similar performance to Decision Tree with F1 scores ranging from 0.47 to 0.61 and AUC values between 0.89 and 0.92.
- Random Forest and XGBoost consistently showed the best results, especially in F1 score of 0.55–0.61 and AUC of 0.92–0.94 with recall values up to 0.89.

XGBoost slightly outperformed Random Forest in terms of precision but both models showed strong performance across all metrics.

4.3.2 Statistical Analysis of model results:

For statistical analysis we used a combination of ANOVA Test to first determine if there is a significant difference between the groups of 5 models collectively and then we will use t-test to perform individual statistical test between model to determine which is best. ANOVA stands for Analysis of Variance. It's a statistical test used to compare the means of three or more groups (or models, in your case) to see if there are significant differences between them [12]. In simple terms, it checks if the performance of different models is statistically different from each other. ANOVA divides the total variation in the data into two parts:

- Variation between the groups (in your case, the models).
- Variation within the groups (the variability within the cross-validation results for each model).

It then compares these two types of variation. If the variation between the models is much larger than the variation within each model's results, then there is likely a significant difference between the models.

Metric	F-Statistic	P-Value
AUC	24.036	2.13E-07
Recall	1.14	0.3662
F1 Score	48.928	4.73E-10
Accuracy	186.571	1.58E-15
Precision	74.039	1.06E-11
Specificity	267.305	4.71E-17

Table 1: Results of Annova tests for all 6 metrics across all models

The above table gives us an overview of ANOVA tests performed on all the metrics. F-statistic determines the variance between the groups, higher the F-statistic higher the variance. p-value indicates if there is statistically significant difference among the groups. From the above table we can see that p-value is less than 0.05 for all metrics except for Recall, which indicates all model perform similarly on recall, but for other metrics there is at least one model which has a significant difference than rest of the group. From Visual inspection it was evident that XGBoost and Random Forest were the better performing models. Individual T-tests were performed by comparing the results of XGBoost and Random Forest against all the models. Since there was no significant difference between the model for Recall in the ANOVA test, we did not perform individual t-tests for that metric.

Comparison	t-statistic	p-value
XGBoost vs Logistic	8.09	0.0013
XGBoost vs Decision Tree	8.21	0.0012
XGBoost vs Random Forest	-1.19	0.298
XGBoost vs SVM	8.21	0.0012

Table 2: T-tests for AUC of XGBoost against all other models

Comparison	t-statistic	p-value
XGBoost vs Logistic	26.65	1.18E-05
XGBoost vs Decision Tree	2.42	0.0726
XGBoost vs Random Forest	0.18	0.865
XGBoost vs SVM	18.1	5.48E-05

Table 3: T-tests for F1 of XGBoost against all other models

Comparison	t-statistic	p-value
XGBoost vs Logistic	52.56	7.84E-07
XGBoost vs Decision Tree	3.88	0.0179
XGBoost vs Random Forest	1.3	0.264
XGBoost vs SVM	24.26	1.71E-05

Table 4: T-tests for Accuracy of XGBoost against all other models

Comparison	t-statistic	p-value
XGBoost vs Logistic	22.63	2.26E-05
XGBoost vs Decision Tree	3.84	0.0185
XGBoost vs Random Forest	1.23	0.221
XGBoost vs SVM	11.76	4.68E-05

Table 5: T-tests for Precision of XGBoost against all other models

Comparison	t-statistic	p-value
XGBoost vs Logistic	11.39	0.000249
XGBoost vs Decision Tree	2.86	0.0278
XGBoost vs Random Forest	-0.36	0.724
XGBoost vs SVM	10.63	0..000497

Table 6: T-tests for Specificity of XGBoost against all other models

We performed t-test with 95% confidence intervals for all the parings. The above tables suggest a similar pattern across all metrics where xgboost a statistically significant difference against all the models except Random Forest since the p-value is less than 0.05 against all the models. The T-statistic of indicates the magnitude of which entity is greater between X and Y, where, since it is positive, the entity on the left, which means the entity that was passed first in the function is greater, and when negative the second value is greater. From the results we can infer that Xgboost performs significantly better than Logistic regression, SVM and Decision tree, and performs on a similar level with Random Forest. We can also conclude that Random Forest also performs better than these 3 models as performs on a similar level with Xgboost with no significant difference. TO verify that we performed t-tests between Random Forest and all other models except xgboost.

Comparison	t-statistic	p-value
Random Forest vs Logistic	6.52	0.0029
Random Forest vs Decision Tree	7.56	0.0016
Random Forest vs SVM	6.66	0.0026

Table 7: T-tests for AUC of Random Forest against all other models

Comparison	t-statistic	p-value
Random Forest vs Logistic	11.36	0.00034
Random Forest vs Decision Tree	4.11	0.0148
Random Forest vs SVM	11.11	0.00037

Table 8: T-tests for F1 of Random Forest against all other models

Comparison	t-statistic	p-value
Random Forest vs Logistic	24.74	1.59E-05
Random Forest vs Decision Tree	3.89	0.0177
Random Forest vs SVM	20.05	3.65E-05

Table 9: T-tests for Accuracy of Random Forest against all other models

Comparison	t-statistic	p-value
Random Forest vs Logistic	15.13	0.00011
Random Forest vs Decision Tree	4.3	0.0126
Random Forest vs SVM	14.65	0.00013

Table 10: T-tests for Precision of Random Forest against all other models

Comparison	t-statistic	p-value
Random Forest vs Logistic	24.61	1.62E-05
Random Forest vs Decision Tree	3.95	0.0168
Random Forest vs SVM	19.69	3.93E-05

Table 11: T-tests for Specificity of Random Forest against all other models

5 IMPLEMENTATION DETAILS

To build this portal, we leveraged a combination of various technologies to create a user-friendly application. The technologies we used are python, flask, JavaScript, HTML, IDE, Juypter Notebook.

5.1 SYSTEM ARCHITECTURE:

Python is the primary programming language used in our backend development, and which helps in designing, training, feature analysis and developing the models. Python is also used for creating pipelines and preprocessing.

Flask is the web framework used for developing APIs and establishing a connection with the front-end. It helps in efficiently

managing the HTTP requests and responses between client and server.

JavaScript and HTML are the languages used to build the front-end i.e., UI of the application which helps us in building user friendly and interactive webpages.

Juypter notebook is used for initial analysis, preprocessing and data training. It provides flexibility for exploring data and performing experiments.

6 USER GUIDE MANUAL

6.1 REQUIREMENTS:

For a user to run the application, the basic software necessary is VS Code, Python. Along with python the user should have several libraries like scikit-learn, pandas, numpy which can be installed via pip and are useful for data preprocessing, visualization and data manipulation.

Once installations are completed and the directory path is setup, users can run the application locally. For this, the users need to execute the command python “file_name”.py in the terminal.

As a flask server is created which connects the backend, the application will be ready to be hosted on http://127.0.0.1:5000

6.2 UI Design:

The UI of the application features a user-friendly webpage that enables a user to determine if a job posted online is fraudulent or not by submitting a form which contains various fields about the job like the job title, description, salary range, benefits and other relevant details about the jobs which the user fills out. Figure 2 shows the primary GUI of the application which has the form and input fields.

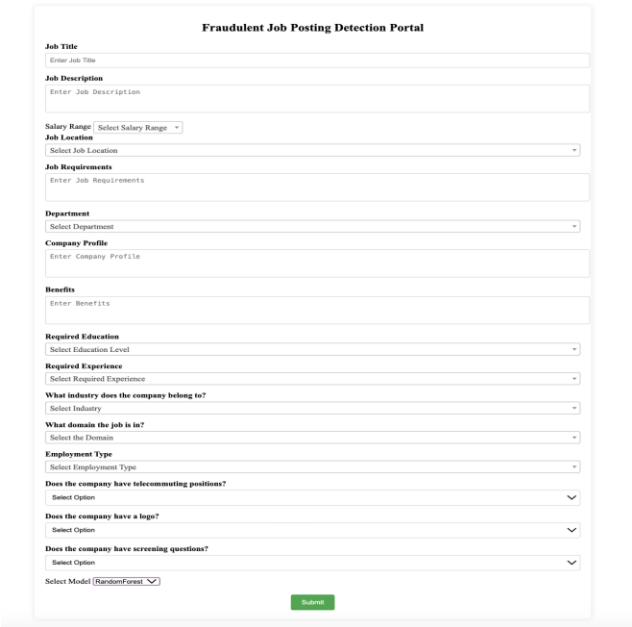

Figure 2: GUI of the Fraudulent Job Posting Detection Portal

Once the form is filled, the user will be able to select a machine learning model which they want to use for evaluation. After the model is selected, the user submits the form for analysis. The application processes the data then and provides a prediction on whether the job is fraudulent or not. The user at the end has an option to select one of the models from Random Forest of XGBoost.

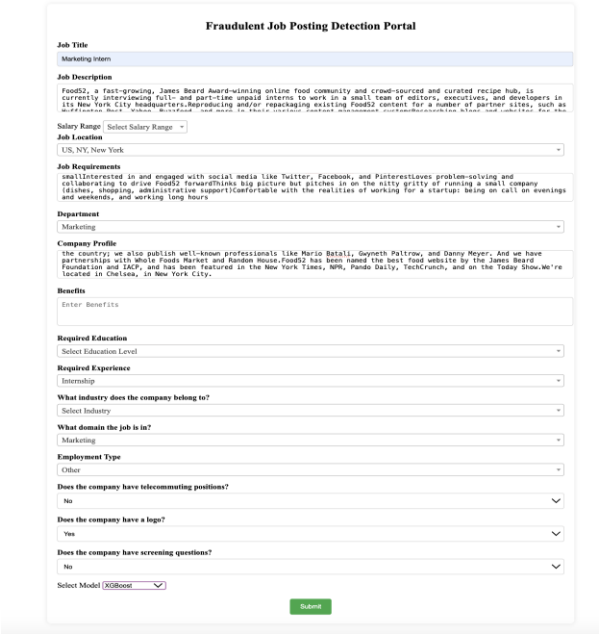

Figure 3: Selected model is XGBoost

The below screenshot shows the gui implementation when a job is determined to be legitimate. The output will contain the prediction, model name with which it is tested, job title, department, location.

Figure 4: The GUI screenshot which shows that the job submitted in Figure 3 is legitimate

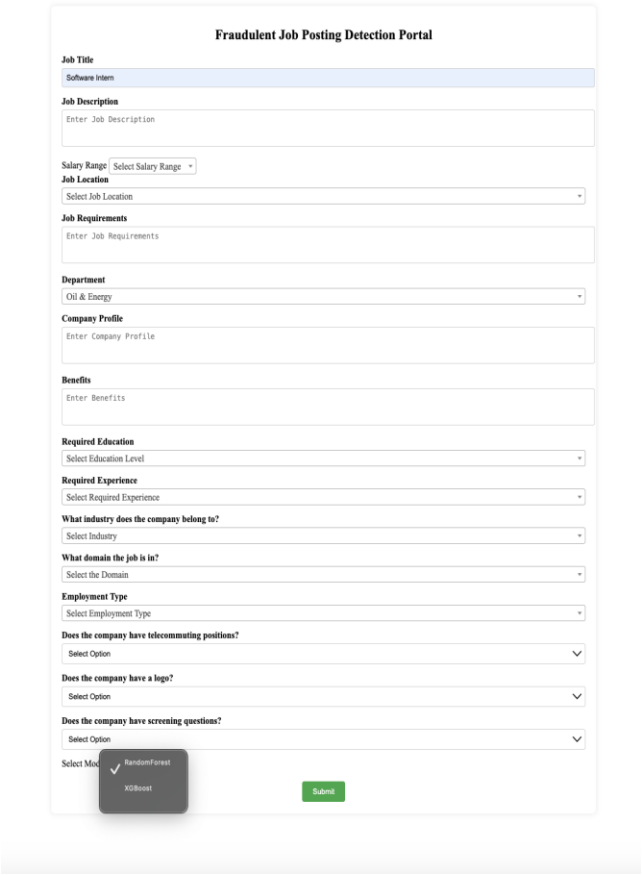

Figure 5: Entering the input fields and selecting the model as Random Forest

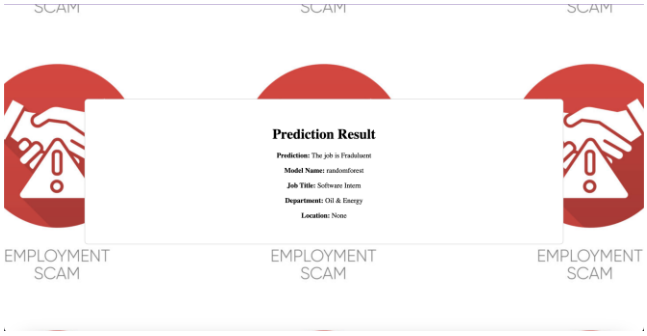

Figure 6: The GUI screenshot which shows that the job submitted in Figure 5 is Fraudulent

6.3 REASON FOR DESIGN AND IMPLEMENTATION

The design chosen for UI is user-friendly, interactive and helps user determine a fraudulent job website by entering the job details. The core objective of this application is to provide a simple and easy-to-use interface. The form that the user fills out is designed to collect and store all the data relevant to the job in a structured manner which ensures that the data is consistent and effectively processed.

The application is built using JS, Python and is connected using Flask which is highly scalable and maintainable. Flask provides a foundation for building restful APIs and these technologies ensure long-term sustainability, have a huge base and are widely supported.

7 RESULTS

The results of this application is finding whether the job is legitimate or not. Based on the user inputs and the job details a user can choose a model with which they want to determine the job status, whether it is fraudulent or legitimate. This helps the user determine if the job is legitimate and apply to it.

8 CONCLUSION AND FUTURE SCOPE

The web application developed in this project successfully identifies the fake job listings by leveraging machine learning and text analytics, providing users with a simple and easy to use interface and giving them an immediate result. It provides the interface.

In future, the application can be enhanced by implementing reinforcement learning to make the model gradually better after each iteration of prediction. Integration of Large language models can also be introduced better interpret the textual features of the postings.

REFERENCES

[1] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. “A Comparative Analysis of XGBoost” ResearchGate, November 2019.

[2] Anita C S, P. Nagarajan, G. Aditya Sairam, and P. Ganesh. “Fake Job Detection and Analysis Using Machine Learning and Deep Learning Algorithms.” Revista Gestão Inovaçãoe Tecnologias, vol. 11, no. 2, June 2021, pp. 642-650.

[3] Shawni Dutta and Samir Kumar Bandyopadhyay. “Fake Job Recruitment Detection Using Machine Learning Approach.” International Journal of Engineering Trends and Technology (IJETT), vol. 68, no. 4, April 2020, pp. 48.

[4] Sourish Ghosh, Anasuya Dasgupta, and Aleena Swetapadma. “A Study on Support Vector Machine Based Linear and Non-Linear Pattern Classification.” Proceedings of the 2019 International Conference on Intelligent Sustainable Systems (ICISS), IEEE, January 2019.

[5] Pradipta G. A., Sanjaya I. N. H., Wardoyo R., Musdholifah A, and Ismail M. 2024. SMOTE for Handling Imbalanced Data Problem: A Review. 2021 IEEE Sixth International Conference on Informatics and Computing (ICIC), November 2021.

[6] Jumana Jouhar, Anju Pratap, Neharin Tijo, and Meenakshi Mony. “Fake News Detection using Python and Machine Learning.” Procedia Computer Science, vol. 233, 2024, pp. 763- 771.

[7] Nikhil Kumar, Sanket Sonowal, and Nishant. “Email Spam Detection Using Machine Learning Algorithms.” Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), July 2020.

[8] Ibomoiye Domor Mienye and Nobert Jere. “A Survey of Decision Trees: Concepts, Algorithms, and Applications.” IEEE Access, vol. 12, June 2024.

[9] Pablo Quihui, Guillermo Pérez Espinosa, and Alberto Laines Vázquez. “Fake Job Detection with Machine Learning: A Comparison.” ResearchGate, June 2023.

[10] Matthias Schonlau and Rosie Yuyan Zou. “The Random Forest Algorithm for Statistical Learning.” The Stata Journal, vol. 20, no. 1, March 2020.

[11] Szeghalmy S., and Fazekas A. “A Comparative Study of the Use of Stratified Cross-Validation and Distribution-Balanced Stratified Cross-Validation in Imbalanced Learning.”, February 2023.

[12] Kim T. K. 2024. “Understanding one-way ANOVA using conceptual figures”, 2017.
[13] Xiaonan Zou, Kaiyuan Shen, Yong Hu, and Zhewen Tian. “Logistic Regression Model Optimization and Case Analysis.” Proceedings of the 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), October 2019.