Divide to Conquer

Using SVC Models and Text Vectorization to Predict Fraudulent Job Postings

Lead Author: Kelli Sugai
Co-authors: William de Melo, Fan Yang, Prosperity Land

1. Introduction

Many popular job websites, such as LinkedIn and Handshake, have seen considerable growth in recent years and represent a transition of recruitment practices from traditional to virtual methods. Although these websites make the job-seeking process easier, they also provide a means for scammers to post fake job listings. Detection of these fraudulent postings is necessary to protect these websites' users and maintain their credibility.

Previous research has demonstrated the effectiveness of machine learning approaches in fraud detection. Due to its ability to work with high-dimensional feature spaces and non-linear relationships using kernels, the SVC model has shown promise in text classification tasks. Model interpretation and performance can be affected by whether an independent vectorization or combined vectorization is chosen to process the text. Independent vectorization allows for the semantic distinctiveness of each text field to remain intact since words may differ in meaning or carry different weights between contexts. Research in domain specific text analysis suggests that context-aware feature extraction can improve classification accuracy by preserving the nuanced meanings that words have in different contexts (Sebastiani, 2002). In contrast, combined vectorization may lose finer contextual distinction but is capable of recognizing broader patterns within the document.

In this study, we will analyze how effective Support Vector Classification (SVC) models are at identifying fake job postings from a dataset containing 18,000 job postings, 800 of which are fraudulent. We investigate not only the model's performance, but also how two different

approaches to text preprocessing affect its precision. We compare an approach that independently vectorizes individual fields of text, such as a job posting's description and requirements, and another that unifies and collectively vectorizes all text fields as one.

We hypothesize that an SVC model has the ability to effectively detect fraudulent job postings. We also hypothesize that using independent TF-IDF vectorizers for each descriptive category will result in greater predictive power than using one combined vectorizer for all fields, as the meaning of words can vary significantly across different fields. We hypothesize that independent vectorizers can capture contextual semantics specific to each field type since the same words can refer to different levels of legitimacy depending on which descriptive category they appear in. The separate vectorizers will learn field-specific vocabulary patterns, their associations with fraudulent behavior and better understand text as it relates to fraud detection across various domains. Our goal is to evaluate the suitability of SVCs for fraud detection and the impact of text preprocessing choices on the performance of the model.

2. Methods

2.1 Dataset

We use a dataset on job listings found on Kaggle, with 18,000 job descriptions (Bansal, 2020). The provenance for this data is the EMployment SCam Aegean Dataset (EMSCAD), a publicly available dataset on fraudulent job postings collected by the University of the Aegean (Laboratory of Information & Communication Systems Security, 2014). Although collected in Greece, the dataset prominently features job postings from the US and Great Britain, which together comprise 73% of the observations. Access to the original dataset is currently unavailable via the original link, but the dataset has been re-posted several times on Kaggle and Github.

800 of the descriptions (around 4.5%) are fraudulent, making this a rare event problem. We compensate for this by employing balanced class weights to emphasize the relatively few instances of fraudulent job descriptions during training. Certain columns within the dataset were deemed unusable; for instance, the salary variable, concerning the range of yearly salaries for the postings, suffered from high missingness (around 84%). We determined that imputation efforts would not work well given the objective dearth of data; that salary information from 16% of the data would be applied to the other 84%. Furthermore, transforming the column into a binary indicator - of having a salary range versus not having a salary range listed - would also be counter-intuitive.

Among the information in the dataset that we employed, three categories emerged: textual, categorical, and binary. The textual data denote the columns that contain scraped text information about a given role, including its title, the profile of the posting company, the job description, requirements from the applicants, and benefits. Categorical data included the

type of employment (full-time, part-time, etc.), the levels of experience and education recommended for the role, the location of the role, and terms denoting the job's industry and function. Lastly, binary variables were included for whether the application had a company logo, telecommuting support, and questions in its application.

2.2 SVCs and TF-IDF Vectorization

To answer our research questions, we constructed two SVC pipelines in Python using SciKit-Learn. SVC is an application of the broader Support Vector Machine (SVM) algorithm for classification tasks. It uses features passed to it to create a decision boundary that best separates data points of different classes. In our project, the SVC algorithm finds the hyperplane that keeps real and fraudulent postings as far apart as possible by maximizing the gap between the closest examples of each class. If perfect separation is not possible, it makes a trade-off controlled by a parameter C.

To translate the text columns into appropriate inputs for our SVC models, we employed SciKit-Learn's Term Frequency - Inverse Document Frequency (TF-IDF) vectorizer. TF-IDF is used to convert tokens – words or series of words from a body of text – into vectors that function like explanatory variables in the SVC models. In our case, we limit our tokens to unigrams and bigrams, or single words and pairs of words that appear in the text data.

We selected TF–IDF because it offers a straightforward and effective way to represent text. Compared to basic word counts or more complex neural embeddings, TF–IDF highlights terms that are more informative by reducing the weight of very common words. It also requires minimal parameter tuning and scales efficiently with larger datasets, making it a practical choice for establishing a strong performance baseline before experimenting with more advanced methods.

For a given instance of text in a tabular dataset, the value of a feature derived from a token using TF-IDF is the term frequency (the number of times it appears in that instance relative to the number of tokens in that instance) weighted by the inverse document frequency (the logarithm of the total number of entries divided by the amount of entries that contained that token). This produces values for tokens that balance how often those tokens appear in that specific instance of text data with how rare they are across the whole dataset. The downside to this and similar approaches, however, is a high-dimensional dataset, warranting the use of SVC models due to their ability to handle such data (SciKit-Learn).

2.3 Design

We compared two models with different tokenization strategies:

2.3.1 Separate-Vectorizer Model

In this model, each of the five text fields received their own TF-IDF transformations using separate vectorizers, resulting in five distinct vocabularies and weight assignments. Each TF-IDF vectorizer was identical, but upon application to a particular text field they created unique, separate vocabularies for those fields. This means that different weights are assigned to words specific to the context of their feature (e.g. the relevance of "paid leave" to fraud detection in the benefits field may differ from that of "paid leave" in the description field).

2.3.2 Combined-Text Model

In this approach, all five cleaned text fields were concatenated into one string per posting and vectorized once using TF-IDF, producing a unified feature space that treats each term identically across contexts. In contrast to the separate vectorizer approach, tokens that appear in multiple text fields for a given job posting have a greater term frequency once those fields are combined. This means that their values for each entry in the training data increase. This approach also removes the individual contexts of each field (e.g. "paid leave" no longer has separate, field-dependent relevancies to fraud detection).

2.4 Preparation and Hyperparameter Tuning

In both models, we fitted linear-kernel SVCs to detect the 4.5% of fraudulent job postings. To prepare the data for modelling, we first split the 17,800 cleaned observations into an 80:20 stratified training-testing set so that the testing set preserved the 4.5 % fraud rate. We also cleaned the text data by following standard preprocessing steps, such as turning all words lowercase and removing punctuation. We also employed stopword removal using the nltk library, where common prepositions and conjunctions are removed, including the, and, for, etc.

We also utilized Sci-Kit Learn's pipeline function to streamline the process of converting the raw data into a usable format before training the SVCs on it. Before being passed to the model, the data was passed to a preprocessor which contained TF-IDF blocks (either one per text field or one concatenated block) for the text data, and a one-hot-encoded matrix for the six categorical variables. The three binary variables were passed as-is to the model without any preprocessing.

Hyper-parameters were tuned with 5-fold cross validation, in which the training data was subdivided into five folds: four used to fit the entire pipeline and one reserved for validation. We searched over a grid of regularization values ($C = \{0.01, 0.1, 1\}$) and selected the value that maximized the mean F1 score across folds. Additionally, both approaches were tested using varying numbers of maximum features for the TF-IDF vectorizers. This hyperparameter denotes the maximum number of unigrams and bigrams to be transformed from the text.

Setting maximum features allows the vectorizer to establish a vocabulary for only the most commonly occurring unigrams/bigrams from the data.

For both models, the optimal C was 0.01, and the ideal number of maximum features was e^12 (roughly 33,000 per vectorizer, using 5 vectorizers) for the separate and combined approaches respectively. Although employing this many features means using more features than there are observations, the SVC model's approach to fitting its hyperplane allows it to handle high-dimensional data like this (Awad & Khanna, 2015).

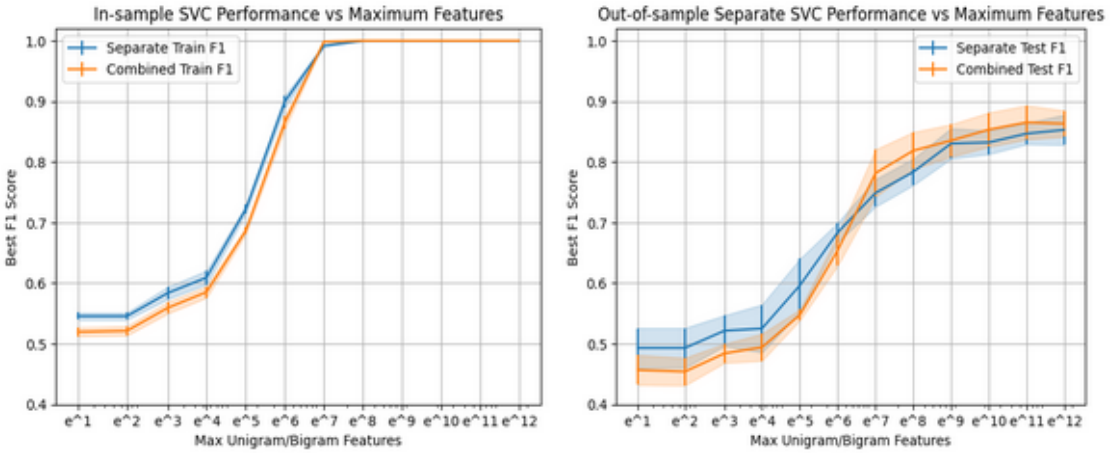

Figure 1: F1 Scores as a Function of Learned Features from Text Data

The in-sample (left) and out-of-sample (right) performances of the separated-vectorizer model and the combined-text model compared to the amount of unigram/bigram features converted from the text data by each vectorizer. Error bars are constructed using distributions from 5-fold cross validation. The separated model overfits the training data at around 1000 features learned, and the combined model overfits at around 3000 features. In terms of out-of-sample performance, the separated model overtakes the combined model at first, but this inverse occurs between 1000 and 3000 features learned. At no point in the out-of-sample testing do the error bars for each model fail to intersect.

We also employ balanced class weights in order to adjust for the lack of fraudulent job postings in the data. This slight change gives rare instances within the training data additional weight that is inversely proportional to their frequency, allowing the training process to prioritize their detection. After fixing the optimal C, we refit the full pipeline on all training data and generated class-probability estimates for the test set.

By default, if the raw estimate produced by the model for a given job posting exceeds 0.5, that posting would be predicted as a success, or in this case fraudulent. However, because the default threshold can be biased for imbalanced data, we further optimized the classification threshold. Among nearly 3,300 thresholds we compared, the threshold that maximized the F1 score was chosen and applied to the final fraud predictions. Testing set performance was

compared at the optimized parameters using precision, recall, F1, and ROC-AUC scores, enabling an objective comparison of the separate-vectorizer and combined-text models.

3. Results

3.1 Relevant Words

Using the combined-text model, we were able to extract the coefficients of each of the text features employed in the hyperplane calculation and arrange them into a word cloud. The size of each token is dependent on the size of the coefficient, meaning that the larger the token, the more indicative of a fraudulent posting it is. We used this word cloud in an attempt to see which "buzzwords" might function as strong indicators of a fraudulent job posting, allowing users of job websites to identify signs of risk on their own.

Certain salient words such as ux and executive appear to refer to specific competencies within the job descriptions, yet others, like ae and neededskillsqualifications appear to be artifacts created from the preprocessing steps or grammatically incorrect phrases from specific observations. It may also be the case, as evidenced by the prior signs of overfitting in the data (see Figure 1), that some of these terms may be overly specific to the training data and not generalizable elsewhere.

The myriad terms complicate matters of interpretation; it is difficult to say that a single word or phrase is a dead giveaway of fraud. For instance, there are roughly 1,900 non-fraudulent observations featuring the term ux and 52 fraudulent observations with the term. Therefore, the presence of ux on its own does not necessarily indicate a fraudulent posting. By this same logic, there are 12 non-fraudulent and 2 fraudulent postings specifically featuring girls, which could serve as a possible indicator but is highly dependent on context and limited by the lack of data. For instance, among the fraudulent postings, one was for a Boys and Girls Club location, and the other was for an adult website.

Yet, regardless of a word or phrase's importance, the maximum coefficient of a unigram/bigram in the combined-text model is .02, where the sum of all coefficients used by the model is 20.7. In other words, the term ux is worth only .01 percent of the combined-text approach's hyperplane calculation. This makes it clear that, ultimately, the model relies on the use of many words and phrases to determine fraud, and individual terms are unlikely to be of significant importance to users of job websites, accentuating the need for computational tools in predicting fraud.

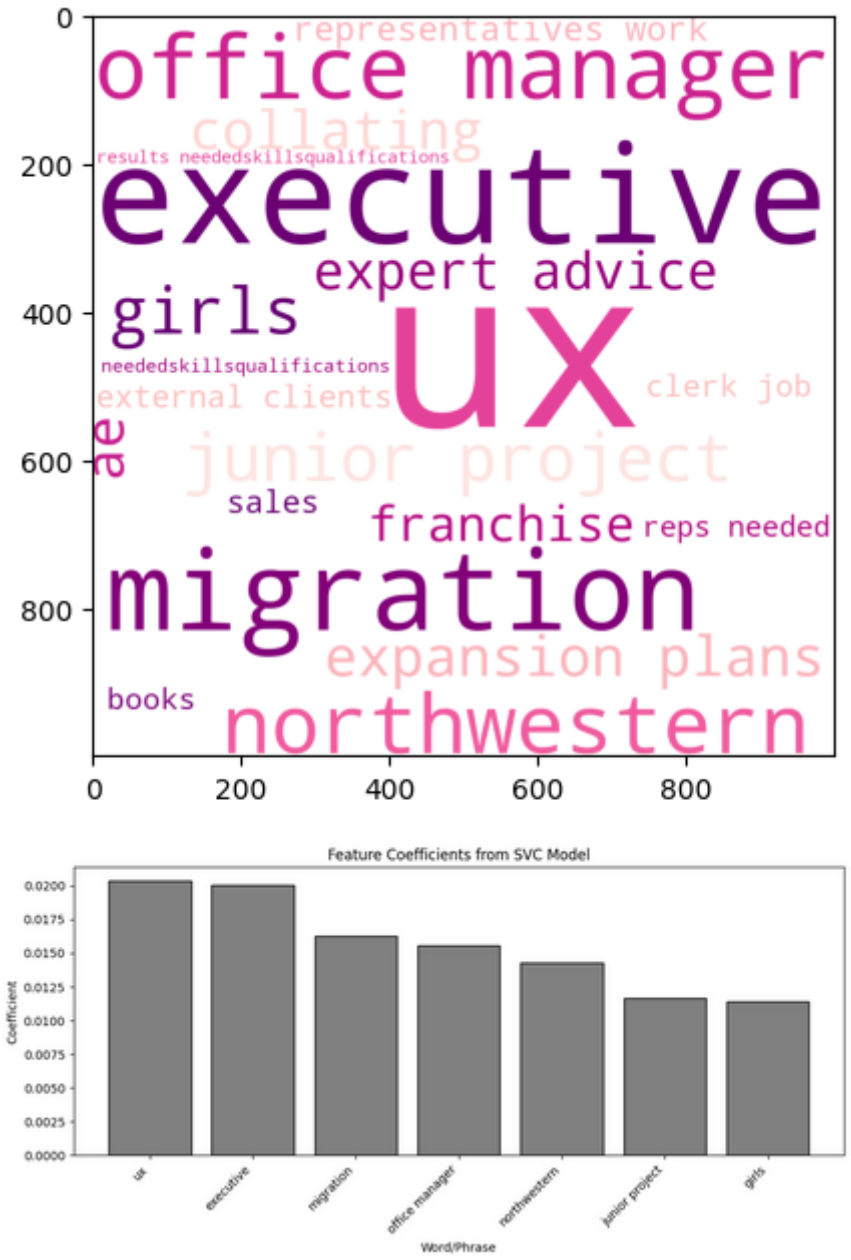

Figure 2: Word Cloud

A word cloud showing the top 20 phrases, in unigrams and bigrams, from job descriptions that were most strongly associated with the model's predictions of whether a job posting is fake or legitimate. The font size in the word cloud reflects the feature importance, or how large the coefficient assigned to that phrase when constructing the hyperplane by which to separate legitimate and fraudulent job

postings.

3.2 Model Comparison

3.2.1. Separate-Vectorizer Model

Using five independent TF-IDF vectorizers, the model reached a cross-validated F1 of 0.824 and an ROC AUC score of 0.993. Scanning 3,364 thresholds identified 0.688 as the optimized threshold, raising the test-set F1 score to 0.90. The confusion matrix after threshold tuning in Figure 3 (left) shows the classifier correctly detects 144 of the 173 fraudulent postings (recall = 0.83) and mislabels 29 real postings (precision = 0.98). The precision-and-recall curves (see Figure 3, right) intersect near the chosen cut-off, which demonstrates that changing the threshold from 0.50 to 0.688 improves recall of the minority class with only a minor loss in precision.

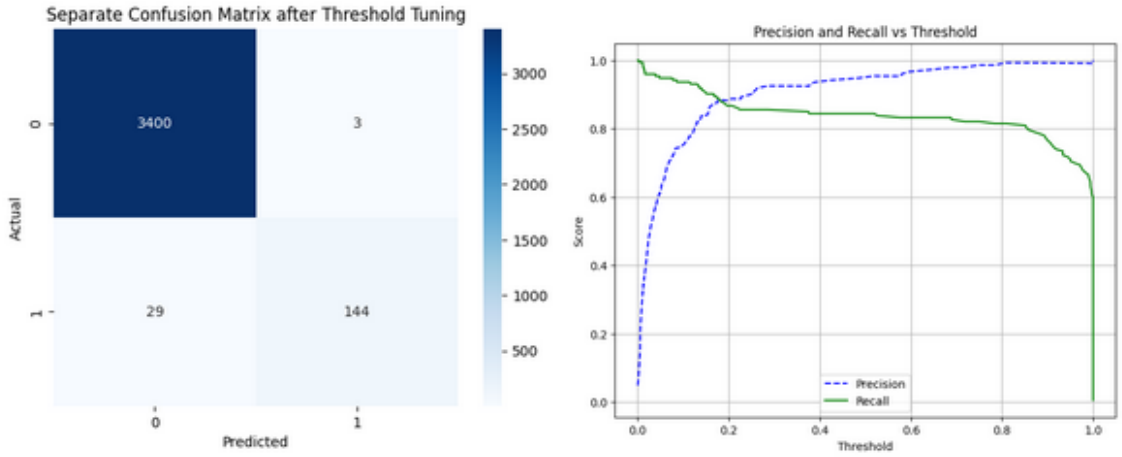

Figure 3: Separate Confusion Matrix after Threshold Tuning and Ideal Threshold

The performance of the SVC model on the test data when smaller vectorizers are used for distinct bodies of text. Illustrates how the precision (accuracy) and recall (true positives detected) changes as the classification threshold is adjusted (right).

3.2.2. Combined-Text Model

Concatenating all bodies of text into one produced a slightly higher cross-validated F1 score of 0.813 and an ROC AUC score of 0.994. Shifting the cut-off from 0.50 to 0.40 also results in better performance for the combined-text model. The confusion matrix in Figure 4 (left) shows 158 instances of fraud correctly predicted and just 5 false positives, improving recall to

0.91 and precision to 0.97. After threshold tuning the model also achieves the best F1 score of 0.941. The precision-and-recall curves in Figure 4 (right) show a similar overall shape to that of the separate model. However, the intersection shifts to the right, indicating that the optimal threshold improves precision while preserving high recall.

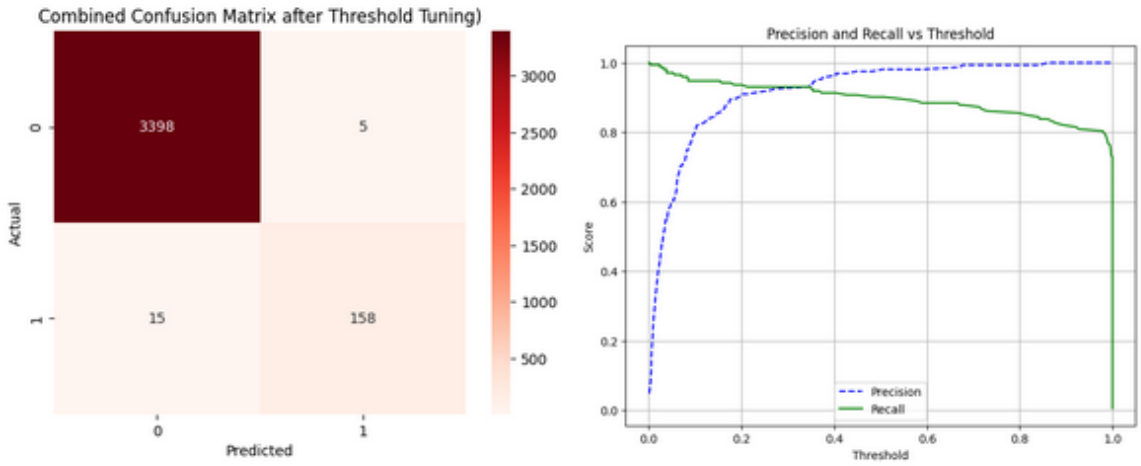

Figure 4: Combined Confusion Matrix after Threshold Tuning and Ideal Threshold

The performance of the SVC model on the test data when bodies of text are combined and vectorized as a whole. Illustrates how the precision (accuracy) and recall (true positives detected) changes as the classification threshold is adjusted (right).

Using 5-fold cross validation, we create a distribution of out-of-sample F1 scores for each model at e^12 (162,755) maximum learned features each. We then compare these scores using a paired t-test. The result is t = 1.665 and p = 0.171, asserting that there is no significant difference between model performances at their optimal amount of learned features.

On a more granular level, the models do significantly differ in how their predictions are made. A paired t-test comparing the models' raw predictions, as opposed to classifications, of all entries in the dataset returned t = 3.759 and p = 0.0002, where the combined-text model is shown to predict a higher average probability of fraudulent postings from the data.

4. Discussion

Our project used support vector machines to predict fraudulent job postings, comparing two different preprocessing strategies: one that treated each text section (title, posting company's

profile, job description, requirements, and benefits) as separate inputs, and one that concatenated all text into one input. The former provides better interpretability of specific job-posting sections, while the latter offers a cleaner and simpler workflow without sacrificing accuracy.

Our first hypothesis, that SVCs would succeed in predicting fraudulent job postings, was met. However, our findings did not lend support to our second hypothesis that the separate-vectorizer model would exhibit greater performance than the combined-text model. Prior to threshold optimization, each model achieved nearly identical F1 scores (0.824 vs 0.813). After threshold optimization, the models became more distinguishable in their performance (0.90 vs 0.941); however, a paired t-test confirmed that there was no statistically significant difference between their predictive power. They performed similarly well based on F1 score, asserting that the approaches are equivalent in performance.

From the results of the paired t-test on model predictions, it would seem that although the models are capable of performing at an equal level, their raw predictions are significantly different. This indicates that, on average, the combined-text model yields higher probabilities of fraudulence to job postings than the separate-vectorizer model. Because of this difference in prediction, the performance of the models on other datasets could significantly differ.

Despite the accuracy of our models, we encountered some limitations. First, the dataset was highly imbalanced, with only 4.5% of the postings being fraudulent. This is not ideal, as SVMs perform more effectively with a balanced dataset. An unbalanced dataset can lead to dissatisfactory classification on the minority class (see Figure 3 and Figure 4) (Batuwita & Palade, 2013). Balancing class weights during training helped ameliorate this problem, but having more examples of fraudulent postings would have been beneficial. Furthermore, there was high missingness in some of the columns, which led to their exclusion from the dataset. This could have removed potentially informative signals for detecting fraudulent postings. The insights offered by the range of salaries for each posting, for instance, could have helped the models in correctly classifying fraud.

Possible future directions could include artificially rebalancing the dataset so that there are more instances of fraudulent postings. This could then be compared to the imbalanced dataset to discern whether up-sampling or down-sampling improves minority classification (Provost, 2000). While we selected TF-IDF vectorization and SVCs for their interpretability, efficiency, and strong baseline performance, future works could integrate contextual embeddings (e.g. fine-tuned BERT or ELMo) to capture word sense and long-range dependencies.

5. References

1. Awad, M., Khanna, R. (2015). Support Vector Machines for Classification. In: Efficient Learning Machines. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-5990-9_3
2. Bansal, S. (2020, February 29). Real / fake job posting prediction. Kaggle. https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction?resource=download
3. Batuwita, R., & Palade, V. (2013). Class imbalance learning methods for support vector machines. Imbalanced learning: Foundations, algorithms, and applications, 83-99.
4. Laboratory of Information & Communication Systems Security. (2014). Employment Scam Aegean Dataset. University of the Aegean. http://emscad.samos.aegean.gr
5. Provost, F. (2000, July). Machine learning from imbalanced data sets 101. In Proceedings of the AAAI 2000 workshop on imbalanced data sets (Vol. 68, No. 2000, pp. 1-3). AAAI Press.
6. Scikit-Learn. (n.d.). 6.2. Feature extraction. Retrieved May 11, 2025, from https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
7. Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. ACM Comput. Surv. 34, 1 (March 2002), 1–47.https://doi.org/10.1145/505282.