Fake Job Posting Detection Using Different Supervised Learning Models

Haoran Zeng, Yao Wan, Seyoung Ahn, Biyun Jiang

Abstract

Due to the outbreak of the COVID-19 pandemic, the unemployment rate has increased dramatically in every part of the world, and so has the number of employment scams. It's not easy for people who are in desperate need of a job to know the warning signs of job postings as most of them are recognizable either when job seekers have clear minds to rationally assess the postings or when they start communicating with cybercriminals to play with fake job postings. In an effort to help job seekers stay alert, we propose a classifier that predicts whether a job posting is fraudulent or not. The paper evaluates the performance of different classification models such as logistic regression, K-Nearest Neighbor, Random Forest, and Gradient Boosted Trees. It also experiments on the effect of the three different types of text representations, including TF-IDF, CountVectorizer, and GloVe.

1 Introduction

Unlike the past when people used hardcopy newspapers to seek job opportunities, people nowadays use employment websites such as LinkedIn, Indeed, Handshake, and countless others. The authenticity of job postings has become more important than ever, but there has been a constant increase in the number of employment scams. In 2019, there were 3,434 employment scams reported in the U.S. according to the Better Business Bureau, and 2,499 scams were reported by the Australian Competition & Consumer Commission (ACCC) in the same period (Forbes, 2020). Since the outbreak of the COVID-19 pandemic in December 2019, the number has increased dramatically to more than 13,000 cases (CNBC, 2020) with a surge in the unemployment rate in the U.S.

Both job seekers and employers are victims of fake job postings. Cybercriminals use fake job postings to lure job seekers to provide their personally identifiable information (PII) or to send them money. The average reported loss from employment scams was about $3,000 per victim in 2019, in addition to the damage to the victim's credit scores (IC3, 2020). Employers suffer from damage to their employer brand and credibility as cybercriminals act as legitimate employers by creating a website similar in appearance to a legitimate company.

According to a Flexjob survey, 19% of people in the workforce have experienced employment scams, yet only 15% knew the warning signs (FlexJobs, 2020). Even with an abundance of blog posts introducing warning signs of employment scams, it is still hard for job seekers to avoid them. Many warning signs become noticeable, e.g. personal information or upfront fees are required, only after they start communicating with cybercriminals. This is why fake job posting detection which aims to give job seekers a warning sign in the first step of the job application process would be useful.

Machine learning approaches are commonly used to build a fake job posting detection model. Classification models such as logistic regression, Neural Networks, Random Forest, and Gradient Boosting Trees employ different features of a job posting, e.g., job description, salary information, and employment type, to predict the probability of a job posting being fraudulent.

2 Literature Review

Fake job posting detection shares similar characteristics with other problems in the domain of internet fraud (Vidros et al., 2017) such as email spam and fake news. Therefore, existing works on relevant internet fraud detection were first studied.

2.1 Email Spam

Email spam is an unwanted email that is sent in bulk to a list of indiscriminate recipients. Filtering of email spam is a well-studied field, and many scholars have proposed diverse methods of detecting unwanted emails. Spamcop, a Naive Bayes classifier (Pantel and Lin, 1998), was used as an early filter, and the use of n-grams was later proposed by several scholars (Androutsopoulos et al., 2004; Kanaris et al., 2006; Çıltık and Letters, 2008). Support Vector Machine (SVM) was also introduced to detect email spam (Drucker et al., 1999). The features used in trained classifiers are extracted from the message body and the message headers.

2.2 Fake News

Fake news is false or misleading information presented as news. Fake news has been a serious problem in the current era of social media networks such as Facebook, Twitter, and Youtube, and many studies have used multiple machine learning models to detect fake information spread via social media networks. Ahmed et al. included K-nearest neighbor (KNN), SVM, logistic regression, linear support vector machine (LSVM), decision tree, and stochastic gradient descent (SGD) in the experiment (Ahmed et al., 2017), while Wang (Wang, 2017) took a different approach of using Convolutional Neural Network (CNN).

Similarly, in the field of fake job postings, multiple supervised machine learning models are employed. Habiba et al. included SVM, KNN, Naive Bayes, Random Forest, Multilayer Perceptron (MLP), and Deep Neural Network models in the experiment and discovered that Random Forest classifier achieved the highest accuracy of 96.5% (Habiba et al., 2021). One thing to note about this study is that it included only the categorical attributes in the dataset; attributes with text such as job description and job requirement were not fitted to the classifiers.

On the other hand, Dutta and Bandyopadhyay included text attributes as input, but the type of text representation used was not specified in the paper (Bandyopadhyay et al., 2020). The models employed to classify job postings include Naive Bayes, Decision Tree, Multilayer Perceptron, KNN, AdaBoost, Gradient Boost Tree, and Random Tree. The best performing classifier was Random Forest, which achieved an accuracy of 98.27%.

Similarly, Ranparia et al. fitted text attributes to the classifier, and the type of text representation used was the Global Vector Model (GloVe) (Ranparia et al., 2020). Only a single classifier, the Sequential Neural Network, was used and achieved an accuracy of 97.94%. The interesting point about the study is that job postings from LinkedIn were used to test the performance of the classifier. 138 job postings were scrapped from LinkedIn using BeautifulSoup library and were manually labeled as “fake” or “not fake”. With the LinkedIn dataset, the classifier achieved 99% accuracy.

3 Our Experiment

From the study of preliminaries, we discovered that existing works vary in three different ways: (1) the selection of features, (2) the type of text representation if text attributes were included, and (3) classification models.

In this study, we aim to experiment with two of the three research topics that are mainly studied in the field of fake job postings. First, we want to uncover the effect of different text representations on the accuracy of detecting fake job postings. CountVectorizer, TF-IDF (Term Frequency - Inverse Document Frequency), and GloVe were studied. Second, we want to evaluate the performance of different supervised learning models, such as Logistic Regression, K-Nearest Neighbor (KNN), Random Forest, and Gradient Boosting Tree.

Among the three types of text representation, we expect GloVe to give us the best performance across different models, as GloVe captures the

semantic meanings of words. Within each text representation, the random forest classifier is expected to perform the best as it was observed in several preliminary works.

4 Dataset

The dataset used in the experiment was published by the Employment Scam Aegean Dataset (EMSCAD) and was retrieved from Kaggle. It contains 17,880 job postings, 17,014 of which are legitimate and 866 are fraudulent job postings published between 2012 and 2014 (EMSCAD, 2021). The attributes of a job posting included in the dataset are shown below:

Variable name	Description
job_id	identification number of a job posting
title	The title of the job
country, state, city	Geographical location of the job ad.
department	Corporate department (e.g. sales).
company_profile	A brief company description.
description	The details description of the job posting
requirements	Enlisted requirements for the job opening.
benefits	Enlisted offered benefits by the employer.
telecommuting	True for telecommuting positions.
has_company_logo	True if company logo is present.
has_questions	True if screening questions are present.
fraudulent	Ture if job posting is fradulent
employment_type	Full-type, Part-time, Contract, etc.
required_experience	Executive, Entry level, Intern, etc.
required_education	Doctorate, Master's Degree, Bachelor, etc.
industry	Automotive, IT, Health care, Real estate, etc.
function	Consulting, Engineering, Research, Sales etc.

Table 1. Description of variables in the dataset

4.1 Exploratory Data Analysis

Before building our models, we performed exploratory data analysis to understand the dataset.

Imbalanced data: The first thing we noticed is that the dataset is imbalanced. As you can observe in Figure 1, there are 17,014 cases of legitimate job postings, while the number of fraudulent job postings is 688. Considering that the imbalance ratio in datasets used in the real-life application of fraud detection ranges from 1:1000 to 1:5000 (Krawczyk, 2016), the imbalance in our dataset is not severe.

However, the imbalance in the class distribution leads to misleading classification accuracy as most of the machine learning models, e.g., Gradient Boosted Trees were designed around the assumption of balanced classification.

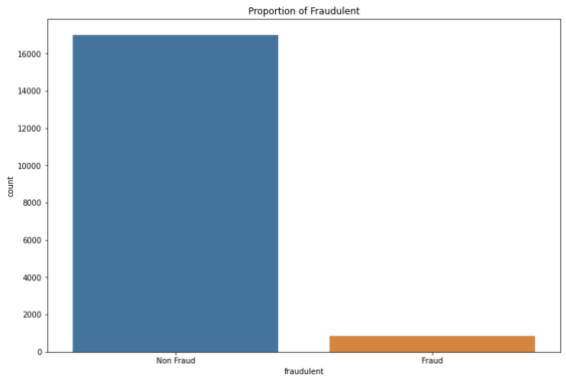

Figure 1. The number of non-fraudulent versus fraudulent job postings in the dataset.

In order to deal with the imbalanced data, we employed several tactics: (1) we used multiple performance metrics, e.g., F-1 and AUC, which are designed to give us a performance evaluation that are more truthful than accuracy score, (2) generated synthetic samples for the minority case, and (3) tried using different classification algorithms.

Keywords: We also generated word clouds for non-fraudulent and fraudulent job postings to determine whether there is a difference in important keywords and their frequency.

Figure 2. Word cloud of non-fraudulent job postings

It was interesting to discover that while many keywords in both word clouds overlap, keywords related to monetary rewards such as sign bonus, compensation package, referral bonus, perks were only observed in fraudulent job postings. We can deduce from this finding that cybercriminals include words related to compensation in fraudulent job postings in order to draw the attention of job seekers who are likely to be in desperate need of income.

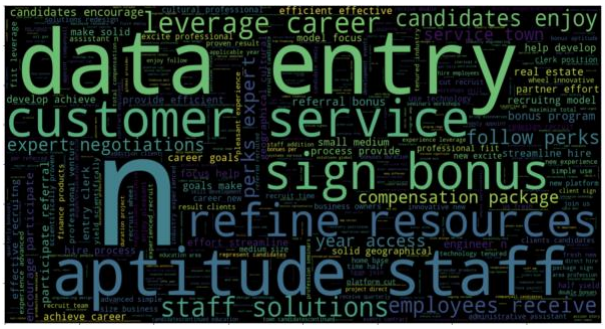

Figure 3. Word cloud of fraudulent job postings

5 Data Preprocessing

The dataset includes both categorical and text information. To better compare the methods and performances of each model, we decided to use a consistent data preprocessing method to handle missing values, categorical variables, and text information. After processing all variables, we combined them and made a new dataset with processed categorical and text variables.

5.1 Missing values

Most of the missing values are related to the company profile, benefits, salary information and position/title information. Missing values were very common in this dataset. Considering that missing values can be a powerful identifier of fake news/job postings, we decided to replace all missing values with “N/A”. By doing so, we were able to retain all non-null information instead of removing them.

5.2 Categorical Values:

Categorical variables include information about title, salary, experience level, and a few binary attributes. We first considered using One-Hot Encoding and Labelencoder to transform the categorical variables but decided to use Labelencoder. We did not use One-hot-encoding because many categorical variables are ordinal, for example, title, salary, and required experience level, and others are mostly binary features such as has_logo and has_questions.

5.3 Text Preprocessing

Before applying vectorizers and fitting models, text data was preprocessed using the nltk packages. We tokenized the text columns and filtered out stop words. In addition, we applied PorterStemmer to finish the stemming process.

5.4 Splitting the Dataset

For every model, we used an identical set of training, validation, and test data. We split the entire dataset into training-validation-test based on the ratio of 80%-10%-10% using the scikit-learn train_test_split function.

6 Experiment

We compared four supervised learning methods, each using three text representations respectively.

6.1 Logistic Regression

We believe that logistic regression is a great starter for this classification as it is a basic algorithm that is generally accepted in text-related classification. Besides, logistic regression can be very effective especially in binary classification based on our experiences.

We used the scikit-learn library for our logistic regression model building, categorical variables building, and standardizing, as it is a commonly used free library for Python programming language.

TF-IDF Vectorizer. Since the logistic regression model from the scikit-learn library is able to take sparse matrices as input, we decided to directly use the preprocessed TF-IDF sparse matrix combined with categorical variables as our explanatory data. To make sure we are handling this extremely unbalanced data set properly, we specifically set the argument class_weight to "balanced". By examining the importance of categorical variables and running the model with different variable combinations, we finalized our model with all the categorical variables. We also tuned our model with different C-values and found a better performance score with a 1 C-value.

The model generated 1963 true negative results on the test data set while having 8 false-positive predictions. This indicates that the model is doing a fairly good job of identifying the real job postings. On the other hand, the model identified 78 true positive results while having 9 false-negative results. This shows that the model was able to catch 78 out of 87 fake postings, but has failed to recognize 9 fake postings.

CountVectorizer. Similar to TF-IDF, we decided to use the preprocessed count matrix combined with categorical variables to feed this regression model. When examining the feature importances we noticed that some explanatory variables were not adding values to the accuracy of the model. After examining different combinations of categorical variables when retaining all text vectors, we noticed that the model performs the best with categorical variable industry and function, along with text vectors. Then we tuned our model by trying different C-values. Our final model included text vectors, industry, and function as explanatory variables with a C-value of 5.

The model generated 1693 true negative results and 8 false-positive results on the test data set. This shows that the model is again doing a good job of identifying the real job postings. On the other hand, the model identified 76 true positive results while having 11 false-negative results. This indicates that the model was able to catch 76 out of 87 fake postings, but has failed to recognize 11 fake postings.

Word Embedding using GloVe. We used pre-trained 100-dimensional word vectors to build this model. To process the text, we obtained the word vector for each of the words in one document, then sum them up into one 100-dimensional vector to calculate the average. Each text document will be converted into one 100-dimensional vector as its representation. We then combined the matrix of vectors for all text documents with other categorical variables and fed it to our logistic regression model. After examining the feature importance, dropping the unnecessary variables, and tuning hyperparameters, we have finalized our model using only the average text vector and country variable, with a C-value of 100.

The model predicted 1403 true negative results while having 298 false-positive results, which is noticeably different from the two previous logistic regression models. This indicates that while having the majority of our data as real job postings, the model was not good enough to identify all of the real postings. On the other hand, this model was able to obtain true positive results of 75 while having 12 false-negative results, meaning that the model was able to catch 75 out of 87 fake postings, which is similar to the previous two logistic regression models.

6.2 K-Nearest Neighbors

We applied a prototype selection method to achieve the goal of downsampling the majority (negative) cases. We also tested KNN on the original imbalance data set, because KNN classifies points based on Euclidean distance and the number of nearest neighbors, the algorithm is less sensitive to imbalanced data.

Prototype Method. We used the K-Means algorithm to achieve this goal. The motivation is that the KNN algorithm uses Euclidean distance as the decision rule and the K-Means algorithm also uses Euclidean distance as the decision criterion for updating each cluster's centroid (Kumar and Reddy, 2015). Since the loss functions of the KNN and K-Means algorithms are derived from Euclidean distances, we decided to use K-Means to do the prototype selection.

Among our training set, we applied K-Means on all majority cases and generated 693 centroids,

which equals the number of positive cases in the training set. Then, we stack the K-Means centroids and positive training data to make a new training set which consists of 693 positive cases and 693 negative cases.

TF-IDF Vectorizer. Vectorizing the text information transforms text information into numeric, which allows K-Means and KNN to be implemented. TF-IDF evaluates how relevant a word is to a document in a collection of documents. And using Euclidean distance provides a representation of similarity between data points. Since TF-IDF measures the relevant association, the document with similar words would have similar vectors. Thus, geometrically, similar documents would be clustered together, in our case, we expect to cluster fraudulent cases together so KNN can perform classification tasks easily.

The best F1 score achieved by KNN using TF-IDF vectorizer is 0.83 and the ROC AUC is 0.81.

CountVectorizer. CountVectorizer counts the frequency of words in a text cell. The word cloud in section [4] shows that there are some words that appear frequently in fraudulent job postings. Thus, we think the frequency of words should also be a strong indicator of fraudulent posts. Similar to TF-IDF, the feature matrix can be fitted to K-Means and KNN. Geometrically, we assumed that fraudulent posts share similar words, and those words appeared frequently. Thus, fraudulent posts' positions in the higher dimensional space are closer.

The best F1 score achieved by KNN using CountVectorizer is 0.89 and the ROC AUC is 0.92.

Word Embedding using GloVe. The underlying intuition behind Glove word embedding is word-word co-occurrence probability. Different from TF-IDF and CountVectorizer, GloVe embedding would consider the surrounding words of a particular word. We used the averaging method to represent the text features in vector spaces. Since the GloVe model was trained on aggregated global word-word co-occurrence statistics, the model is not specific to our job posting data and is usually used to find relations between words (e.g. cosine similarity and Euclidean distance). Thus, even though it is an advanced approach that considers word-word co-occurrence, we did not expect it outperforms other vectorizers.

The best F1 score achieved by KNN using GloVe is 0.83 and the ROC AUC is 0.80.

6.3 XGBoost

XGBoost is a type of gradient boosted trees algorithm. It uses boosting to improve prediction accuracy by combining a set of weaker and simpler models. Different from models using bagging (e.g. Random Forest), boosted trees are built sequentially and involve learning from weaker models.

To prevent overfitting issues, we set the gamma to 5, limit the max_depth to 5, and set the min_child_weight to 5. To handle the imbalanced data, we set the scale_pos_weight to 20, which roughly equals the ratio of the number of majority cases to the number of minority cases.

TF-IDF Vectorizer. We implemented the TF-IDF vectorizer to convert text columns to numeric values. We transformed all text information and used a sparse matrix to substitute the original text columns. We constructed the new training data frame by horizontally stacking the categorical variables.

XGBoost is a tree-based model, thus each model (aka. learner) was based on a sample of features and tried to find the optimal splitting methods to reduce impurity. We believe TF-IDF Vectorizer would represent fraudulent cases near each other in the vector space. Thus, we expect the model can successfully find the optimal splitting rules to identify fraudulent job postings and learn from the errors of weak learners to achieve high accuracy.

The best F1 score achieved by XGBoost using TF-IDF is 0.80 and the ROC AUC is 0.94.

CountVectorizer. Similar to TF-IDF Vectorizer's result, we expect the result of CountVectorizer could help the XGBoost model to identify the optimal splitting methods to identify fraudulent postings. As we discussed earlier, there are some words that appear very

frequently in fraudulent posts. CountVectorizer was able to give those words a higher frequency number. We hope these characteristics of our dataset and CountVectorizer would help XGBoost to quickly identify positive cases.

The best F1 score achieved by XGBoost using CountVectorizer vectorizer is 0.79 and the ROC AUC is 0.94.

Word Embedding using GloVe. The GloVe is able to show context-related information in vector form. Thus, we expect to see a great performance using GloVe if the context-related information or word-word co-occurrences of fraudulent cases are similar. Additionally, XGBoost is an ensemble model using boosting, we expect to see if XGBoost can capture unseen patterns underlying the vector space by carefully learning from weak learners.

The best F1 score achieved by XGBoost using GloVe is 0.82 and the ROC AUC is 0.89.

6.4 Random Forest

Another machine learning algorithm we used is Random Forest. Similar to XGBoost, the random forest is also a tree-based model. The ensembling method Random Forest algorithm used is bagging. Different from boosting (e.g. XGBoost), bagging does not involve the sequential-learning process. Boosting achieves higher performance by intentionally introducing bias to decrease variance. The algorithm starts with several weak and simple learners, meaning these trees would have a high bias, but when combining them using bagging, the final model would have low variance.

Random Forest also performs some level of feature selection and is less sensitive to overfitting issues.

TF-IDF Vectorizer. The TF-IDF Vectorizer shows the relevant importance of a particular word. The random forest algorithm would sample several features and build multiple simple trees. Thus, we hope that Random Forest can discover several powerful splitting rules when it builds simple trees. When it comes to the ensemble, those weak learners can form a strong ensembling model by combining those powerful splitting rules and effectively control the model's variance.

CountVectorizer. Similarly, we also implemented CountVectorizer. This gives the number of frequencies with respect to the index of vocabulary. Even though it is a simple way to vectorize text, it performs very well when some words appeared frequently in a specific label's feature space.

After training the model, we find it performs really well on identifying the real job postings with nearly 100% accuracy. However, it does poorly in recognizing fake job postings. This might be the result of the imbalance of our training data.

Word Embedding using GloVe. Finally, we imported the 100 dimension GloVe matrix and used the averaging method to calculate the representations for each document. Then we combined the categorical variables into the same data frame and used it for training the model. The trained model using this method is doing even better in identifying real postings but worse in recognizing the fake posts.

7 Results

One purpose of analyzing real and fake job postings is to build a sufficient enough tool that can filter out the fake job postings to avoid having people falling for job scams. Thus, we used accuracy and AUC to evaluate the effectiveness of models. As discussed, the dataset we used for training is highly imbalanced. Even though we have used relevant techniques to tackle the possible issues with the imbalanced dataset, it is necessary to use F1 scores, precision, and recall to evaluate the model's ability to identify both real and fake postings equally.

Accuracy score is one of the most commonly used metrics in machine learning where it represents the percentage of the model's predictions that are correct. The accuracy score of each model can tell us how well the model

predicts in general, without considering the effectiveness of predicting any class in particular. The accuracy score is a more commonly known metric that provides a general idea of the model performance. However, it has limited ability to explain how well the classification of the minority group is when the data is imbalanced. Using accuracy alone can give misleading results sometimes.

The AUC score, short for Area under the ROC Curve, is another important metric of classification model. It measures how well the model can distinguish between classifications. While it is helpful in describing the model, using the AUC score might not be good enough to deal with our extremely imbalanced data.

The precision score represents the percentage of positive predictions that are accurate. This metric gives us an idea about how well the model did within the positive predictions. In our models, predicting a fake post is a positive prediction. Therefore, this metric is important.

The recall score represents the percentage of positive results that have been classified correctly by the model. This metric tells us how well the model did in recognizing all of the positive results. It is extremely important for our models to catch as many fake postings as possible, thus this metric is very useful.

Finally, the F1 score, a harmonic mean of precision and recall, is another measurement of model accuracy. Although it is not as intuitive as the accuracy score, F1 is a good measure to imbalanced data. Due to this uniqueness, the F1 score is perhaps the most important metric for our model comparison.

Below is a summary of these metrics for each of the models we built.

Metric	Logistic	KNN	XGB	RF
Accuracy	0.986577	0.969799	0.947987	0.973154
Precision	0.985633	0.845361	0.739220	1.000000
Recall	0.867522	0.815078	0.939944	0.448276
F1	0.917548	0.829432	0.803645	0.802567
AUC	0.867522	0.815078	0.939944	0.724138

Table 2: TF-IDF Vectorizer Metrics Comparison

TF-IDF Vectorizer. Table 2 shows the metrics across all TF-IDF vectorizer models. The logistic Regression model appears to have the highest accuracy, precision, and F1 score across all models but a slightly lower recall score and AUC score than the XGBoost model. The KNN model has achieved acceptable scores on all metrics, but it did not outperform either the Logistic or XGB model. The XGB model achieved the best Recall score and AUC score, which seems to be a very strong model. While Random Forest performed especially well in precision metrics, it has the lowest recall, F1, and AUC score among all other models.

Metric	Logistic	KNN	XGB	RF
Accuracy	0.989374	0.977629	0.942953	0.978188
Precision	0.949153	0.862484	0.727497	0.980000
Recall	0.934430	0.917351	0.942752	0.563218
F1	0.941654	0.887714	0.792863	0.851994
AUC	0.934430	0.917350	0.942751	0.781315

Table 3: Count-Vectorizer Metrics Comparison

CountVectorizer. Table 3 shows the metrics for all CountVectorizer models. The logistic regression model has again achieved the highest accuracy, a relatively high precision, F1, and AUC score. Similar to TF-IDF, the KNN model for the CountVectorizer method has achieved acceptable good scores on all metrics. The XGB model has achieved the highest recall and AUC among all models but has a lower precision and F1. The Random Forest model has the highest precision but lowest recall once again.

Metric	Logistic	KNN	XGB	RF
Accuracy	0.826622	0.970917	0.960291	0.968121
Precision	0.596296	0.863338	0.776933	1.000000
Recall	0.843439	0.799307	0.891879	0.344828
F1	0.613300	0.827836	0.822777	0.748171
AUC	0.843439	0.799306	0.891879	0.672414

Table 4: GloVe Metrics Comparison

Word Embedding using GloVe. Table 4 shows the metrics for all GloVe models. The logistic regression model performed worse with GloVe compared to the previous models as it has a lower precision score and F1 score than all other models within the table. KNN has achieved the best accuracy, precision, and F1 score. XGB has achieved the highest recall and AUC than others while its precision is a little lower. The Random

Forest model has again achieved a perfect precision score but has the lowest recall and AUC.

8 Conclusion

Among all combinations of text representations and machine learning algorithms, we propose CountVectorizer combined with the logistic regression model to be the most effective option for this particular classification problem. This model has the best accuracy and F1 score while having other metrics ranked within 2nd - 4th place among the 12 models we created. Because of the fact that models were trained on an extremely imbalanced dataset, we weighted the F1 score more when evaluating.

Interestingly, as one of the basic text representation methods, CountVectorizer performed extremely well with all four algorithms experimented in this study. Although not the best, KNN and XGBoost had relatively more steady performance across different text representation methods.

This result is different from what we initially expected. We expected the combination of GloVe and Random Forest to give the best performance, but our experiment proved the simplest model to be the best performing one. This gave us a lesson that machine learning algorithms may not always be the most accurate. Logistic regression is just as powerful as other complicated classification, and sometimes it is enough to give us accurate predictions. Running machine learning models like KNN, Random Forest and XGBoost took us hours to run in our computing environment. If we had more time and computing resources, it would have been interesting to dive deeper into the classification models to discover where the differences were from.

To further improve our study, we could use an additional dataset to evaluate the performance of each model more accurately. Job postings could be scrapped from employment websites such as LinkedIn and Indeed, since it is difficult to find a similar dataset that is public and large enough. However, this process would be time-consuming as the scrapped job postings have to be manually labeled as fraudulent or not.

References

Hadeer Ahmed, Issa Traore, and Sherif Saad. 2017. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10618 LNCS, pages 127–138. Springer Verlag.

I Androutsopoulos, G Paliouras, and E Michelakis. 2004. Learning to Filter Unsolicited Commercial E-Mail. Technical report.

Samir Bandyopadhyay, Shawni Dutta, Prof Samir, and Kumar Bandyopadhyay. 2020. Fake Job Recruitment Detection Using Machine Learning Approach. *International Journal of Engineering Trends and Technology*, 68.

A Çıltık and T Güngör Letters. 2008. Time-efficient spam e-mail filtering using n-gram models. *Pattern Recognition Letters*:19–33.

CNBC. 2020. Job scams have increased during the Covid-19 crisis. How to avoid one.

H Drucker, D Wu, and V. N. Vapnik. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural networks 10.5*:1048–1054.

EMSCAD. 2021. Employment Scam Aegean Dataset.

FlexJobs. 2020. How to Avoid Work-from-Home Job Scams: 6 Tips | FlexJobs.

Forbes. 2020. Job Hunting Scams Amid COVID-19 Pandemic.

SU Habiba, MK Islam, and F Tasnim. 2021. A Comparative Study on Fake Job Post Prediction Using Different Data mining Techniques. *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*.

IC3. 2020. Internet Crime Complaint Center (IC3) | Cyber Criminals Use Fake Job Listings To Target Applicants' Personally Identifiable Information.

Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. 2006.

WORDS VS. CHARACTER N-GRAMS FOR ANTI-SPAM FILTERING. Technical Report X.

Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. November.

K.M. Kumar and A.R.M. Reddy. 2015. A fast k-

means clustering using prototypes for initial cluster center selection. *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*:1–4.

Patrick Pantel and Dekang Lin. 1998. SpamCop: A Spam Classification & Organization Program. Technical report.

D Ranparia, S Kumari, and A Sahani. 2020. Fake Job Prediction using Sequential Network. *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*.

Sokratis Vidros, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu. 2017. Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset. *Future Internet*, 9(1):6, March.

William Yang Wang. 2017. "Liar, liar pants on fire": A new benchmark dataset for fake news detection. In *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 2, pages 422–426. Association for Computational Linguistics (ACL).