**Course Project Report for Undergraduate Students**

**Course Type：** Subject Elective

**Course Name：** Data Warehouse and Data Mining

数据仓库与数据挖掘(全英)

**Course Code：**

**《Featuring and Recognition of Real or Fake Job Posting》**

**Name**

**Student ID**

**College**

**Major**

**Email**

Submit Date：/2022

**Abstract:** In recent years, as finding a job by online posting becoming a convenient method, more and more fake job postings started springing up on the Internet. The detection of fake job posting has becoming increasingly necessary on the Internet epoch. In this course experiment, a dataset of real and fake job postings collected by The University of the Aegean is used to perform the featuring and recognition of real and fake job postings. The mainstream feature engineering methods and the traditional machine learning methods are performed. Since the lack of fake job postings are confirmed on the Internet, there are rare fake job postings in this dataset. Therefore, TextCNN and Transformer deep learning methods are performed. Meanwhile, an oversampling method is used for expansion of the fake job postings to improve the performance of the traditional machine learning methods.

**Keywords:** Fake Job Featuring, Fake Job Recognition, Machine Learning, Deep Learning.

Contents

[1. Introduction 5](#_Toc106574532)

[2. Method 5](#_Toc106574533)

[2.1. Exploratory Data Analysis & Preprocessing 5](#_Toc106574534)

[2.1.1. Helpless Elements Removal 6](#_Toc106574535)

[2.1.2. Unrelated Information Removal 7](#_Toc106574536)

[2.1.3. Tokenization & Stemming 8](#_Toc106574537)

[2.2. Feature Engineering 8](#_Toc106574538)

[2.2.1. Word Frequency (TF-IDF) 8](#_Toc106574539)

[2.2.2. Simple Sentimental Analysis 9](#_Toc106574540)

[2.2.3. Part of Speech 9](#_Toc106574541)

[2.2.4. Number of Words 10](#_Toc106574542)

[2.2.5. Repeated Words 10](#_Toc106574543)

[2.3. Oversampling 11](#_Toc106574544)

[2.4. Models 11](#_Toc106574545)

[2.4.1. Logical Regression 11](#_Toc106574546)

[2.4.2. Support Vector Machine 12](#_Toc106574547)

[2.4.3. Random Forest 12](#_Toc106574548)

[2.4.4. AdaBoost 12](#_Toc106574549)

[2.4.5. Word to Vector 13](#_Toc106574550)

[2.4.6. TextCNN 13](#_Toc106574551)

[2.4.7. Transformer 14](#_Toc106574552)

[3. Results 15](#_Toc106574553)

[3.1. Result of Traditional Machine Learning Models 15](#_Toc106574554)

[3.1.1. Original Data 15](#_Toc106574555)

[3.1.2. Oversampled Data 16](#_Toc106574556)

[3.2. Result of TextCNN 16](#_Toc106574557)

[3.3. Result of Transformer 17](#_Toc106574558)

[4. Conclusion 19](#_Toc106574559)

[4.1. Dataset 19](#_Toc106574560)

[4.2. Features 19](#_Toc106574561)

[4.3. Result Analysis 19](#_Toc106574562)

[4.4. Improvements 20](#_Toc106574563)

[References 21](#_Toc106574564)

# Introduction

In this project, the Employment Scam Aegean dataset of real and fake job postings collected by The University of the Aegean in 2012 to 2014 is used to perform the featuring and recognition. In this dataset, there are 17880 job postings and 17014 are real, while only 866 are fake postings. The dataset is widely used in fake job posting prediction in natural language processing (NLP) field. Dutta and Professor Bandyopadhyay used traditional machine learning models with traditional NLP features [1]. Ranparia, Kumari and Sahani used GloVe algorithm and Sequential Neural Network to anaylze the sentiments and patterns of the job postings [2]. Habiba, Islam and Tasnim used supervised and unsupervised machine learning method to perform the recognition [3]. Although they have got great result. However, for the lack of the fake job posting samples, they did not perform a suitable solution. In Vo, Nguyen, Sharma and Le’s paper [4], an oversampling method SVMSMOTE is proposed to expend the lacking fake job posting samples.

In this project, the mainstream feature engineering methods in NLP are performed. After that, Logical Regression, Support Vector Machine, Random Forest and AdaBoost, 5 traditional machine learning methods are used for recognition and the performance of models are analyzed and compared. The result turns out that there is no expected result appear for the fake job posting recognition. Therefore, an oversampling method implemented with K-Nearest Neighborhoods algorithm is performed to improve the performance. Meanwhile, TextCNN model and Transformer models are used for recognition for a remarkable performance on the fake job postings.

# Method

## Exploratory Data Analysis & Preprocessing

In this dataset, there are 17 elements that contains text descriptions and other auxiliary digital information with the label “fraudulent” with 17014 true job postings and 866 fake postings. However, there are a lot of missing values in some elements. The basic description of each element is shown in Table 2-1.

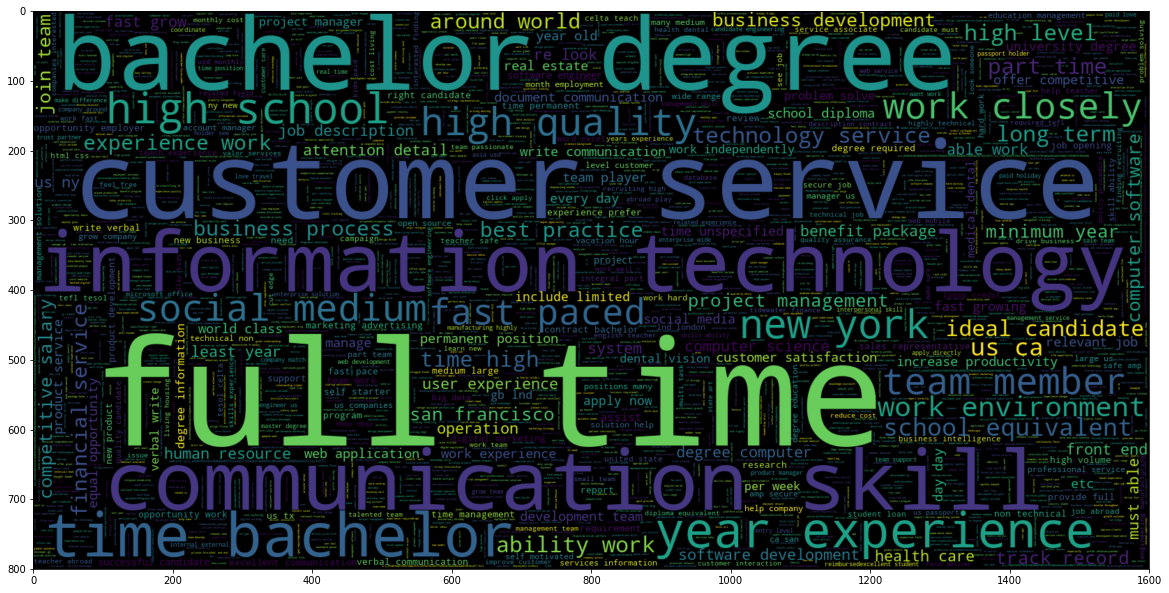
**Table 2-1:** Basic Description of the Dataset

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Element Name** | **job\_id** | **title** | **location** | **department** | **salary range** | **company profile** | **description** | **requirements** | **benefits** |
| **Description** | Unique Job ID | Title of Job | Geographical Location of the Job | Corporate Department | Indicative Salary Range | A brief company description | Short description of job | Enlisted Requirements for job | Benefits for the employers |
| **Missing Count** | 0 | 0 | 346 | 11547 | 15012 | 3308 | 1 | 2695 | 7210 |

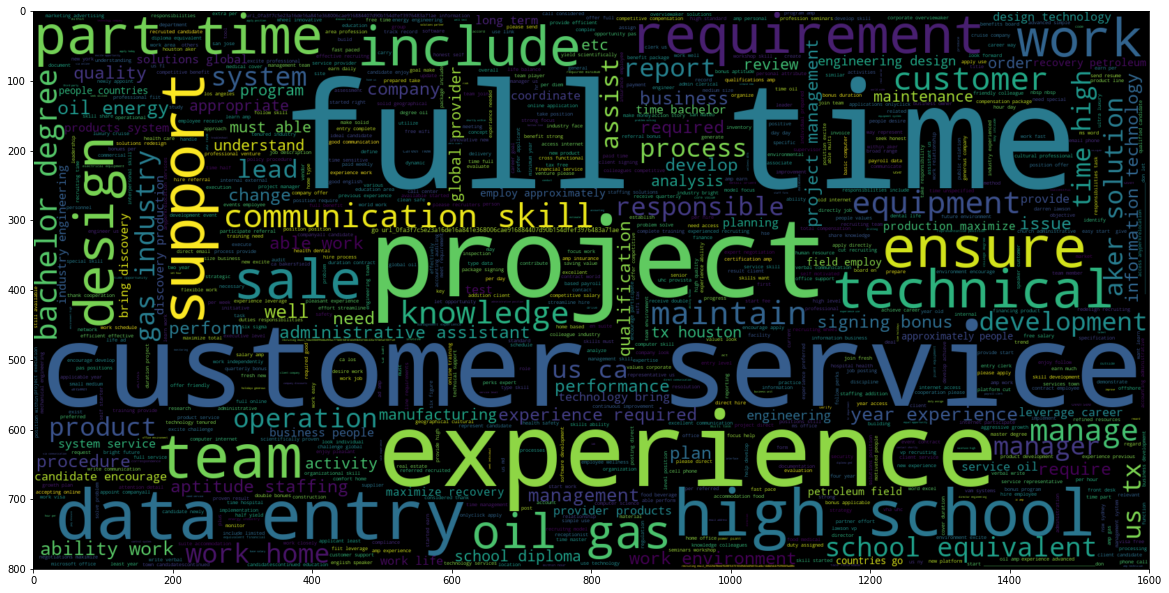
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Element Name** | **telecommuting** | **has company logo** | **has questions** | **employment type** | **required experience** | **required education** | **industry** | **function** | **fraudulent** |
| **Description** | If for telecommuting positions | If has company logo | If has more questions | Full time or part time job | Required working experience | The level of required education | Company industry | Employer position | If it is fake |
| **Missing Count** | 0 | 0 | 0 | 3471 | 7050 | 8105 | 4903 | 6455 | 0 |

### Helpless Elements Removal

For those elements that are digital information with large missing value, they are discarded. For those elements that are text description with missing value, they are replaced with blank space. Besides, to make the following feature engineering more convenient, all text description elements are merged as one element, and the extremely few descriptions that above 800 words are discarded. Figure 2-1 and 2-2 show the word cloud for real job postings and the fakes respectively.



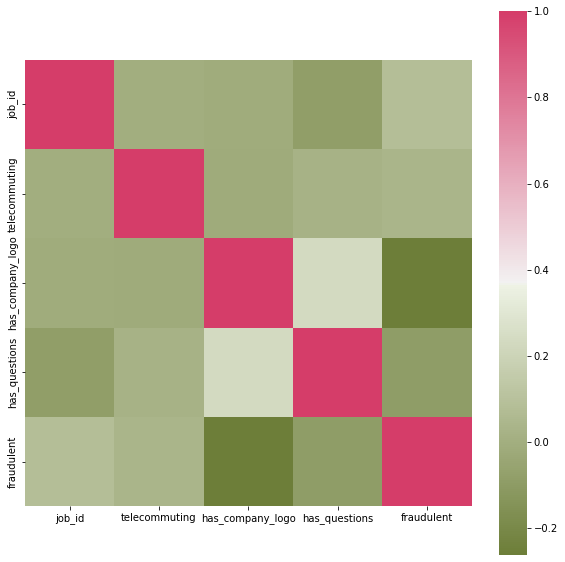
**Figure 2-1:** Word Cloud for Real Job Postings



**Figure 2-2:** Word Cloud for Fake Job Postings

As the above figures show, the most frequent words except the stop words for real job postings are “bachelor degree”, “information technology”, “full time”, “customer service”, “communication skill” and so on. For the fake job postings, the most frequent words are “full time”, “project”, “experience”, “high school” and so on.

To further explore the digital information elements of the dataset, the correlation among the digital information elements is explored and shown in the figure 2-3.



**Figure 2-3:** Correlation among digital information elements

As the above figure shows, only the element *“has\_company\_logo”* has a noticeable negative relationship with the label *“fraudulent”*. Hence, except the element *“has\_company\_logo”* and the label *“fraudulent”,* all other above elements in the figure are discarded.

### Unrelated Information Removal

Since there are a lot of brackets with unrelated information inside the text descriptions. Therefore, all the brackets and the contents inside are found, whose basic information are shown in the table 2-2.

**Table 2-2:** The Basic Information of Brackets and Contents Inside

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Round Brackets** | **Square Brackets** | **Curly Brackets** |
| **Example of Contents** | 'Akamai, Edgecast, etc.'  'or in progress'  'fails not more than twice per week' | ' Click to enlarge Image ' | ' Click to visit the website ' |
| **Number** | 35660 | 210 | 68 |

It is noticeable that the contents inside the square brackets and the curly brackets are unrelated with the recognition. Therefore, all brackets and the contents inside the square and the curly brackets are discarded.

### Tokenization & Stemming

To perform the further NLP feature engineering more conveniently, the text descriptions are tokenized using *nltk.tokenize.RegexpTokenizer* in the *nltk* library to separate the whole text description into single words. After the descriptions are tokenized. The stemming process is required to transform all forms of each word into its basic form, which is for avoiding the ambiguous of the multiform of the words using *nltk.stem.porter.PorterStemmer* in *nltk* library.

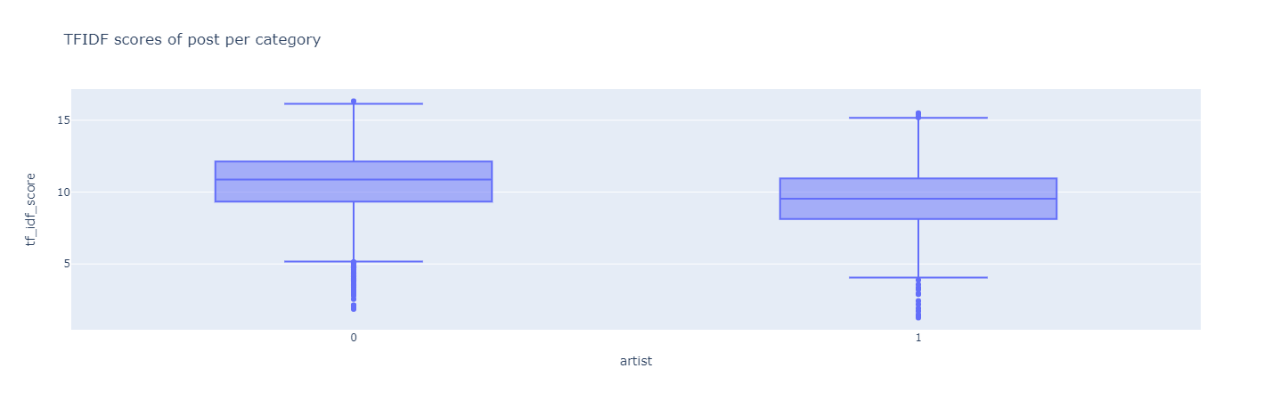
## Feature Engineering

### Word Frequency (TF-IDF)

In traditional NLP, Term Frequency – Inverse Document Frequency (TF-IDF) is one of the most important features in word frequency analysis. Term frequency measures how frequently a word appears in a specific text document, and inverse document frequency is the inverse frequency of the same word in all documents. The weights are calculated as the following formula:

where is the weight of term within document , is the frequency of in , is the total number of documents and is the number of documents containing . The words that appear frequently are assigned lower weights, and the words that are unusual are assigned higher weights. *TfidfTransformer* in *scikit-learn* is utilized to compute the TF-IDF vectors.

For each document, the TF-IDF vector is calculated. Each element of the vector represents one word, and its value is calculated by multiplying its frequency by the corresponding weight. The value will then be normalized by the total number of words in a document. The TF-IDF score is the sum of the elements in the TF-IDF vector. A higher TF-IDF score indicates a more unusual pattern of the word’s appearing in a document. Figure 2-4 is the box plot of the TF-IDF score for each of text description in real and fake job postings, where *“0”* represents the real and *“1”* represents the fake. It is noticeable that the real postings tend to have a little higher TF-IDF score, which means that for the real postings, the text descriptions tend to use unusual and less repeated words.

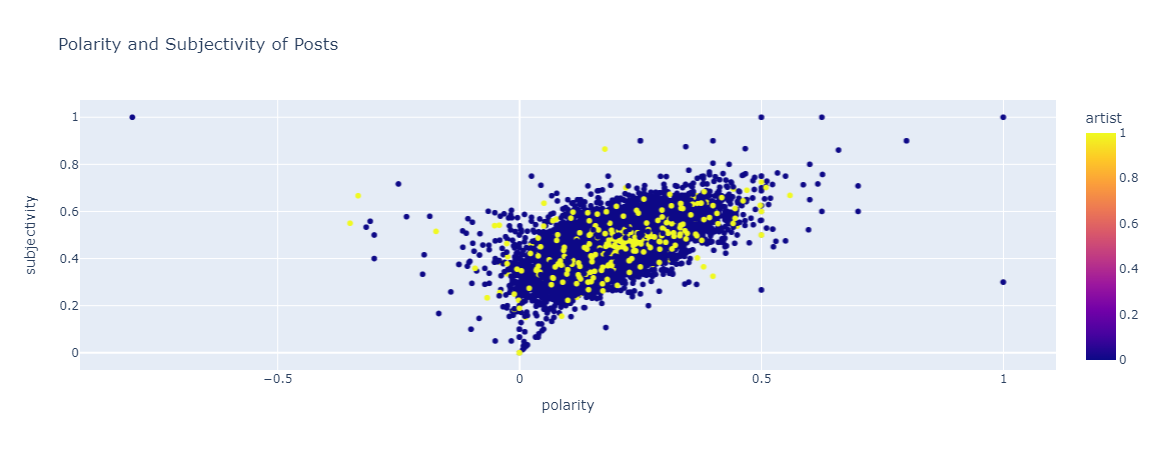


**Figure 2-4:** TF-IDF Score Box Plot of Postings per Categories

### Simple Sentimental Analysis

In recent years, sentimental analysis has been one of the most popular fields in NLP. One key aspect of sentimental analysis is to “understand” the emotions, contents, and opinions hidden in the texts, which is typically measured by what is called polarity. Subjectivity and emotional tendencies are the essential features for the text descriptions of job postings. Different from polarity, subjectivity is a value that measures the degree and strength of the author’s sentiment and feelings in the text.

In this experiment, the *TextBlob* library in Python is utilized to obtain the values of polarity and subjectivity in each text descriptions respectively. Figure 2-5 shows the polarity and subjectivity distribution of the real and the fake job postings, where the yellow points are the fake postings and the blue points represent the real postings. As the figure show, the real postings tend to have a wider distribution compared to the fake postings.

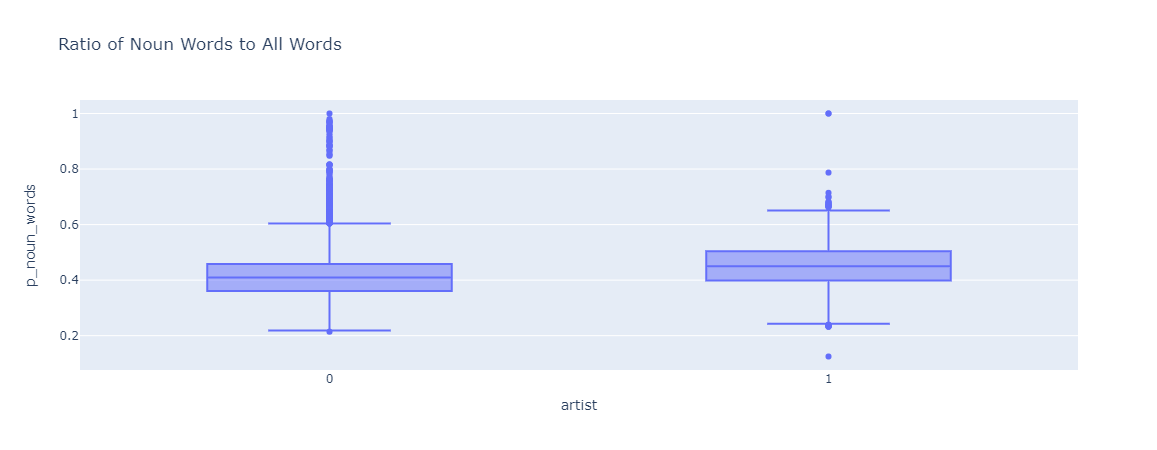


**Figure 2-5:** Polarity and Subjectivity Distribution of the Real & the Fake Postings

### Part of Speech

The part of speech structure is one of the most important features that reflects the writing style of each text description. In this project, the *nltk* library is used to check the part of speech for each word in every text description. The ratio of nouns, adjectives, verbs, and adverbs of each text description is calculated and used as the feature.

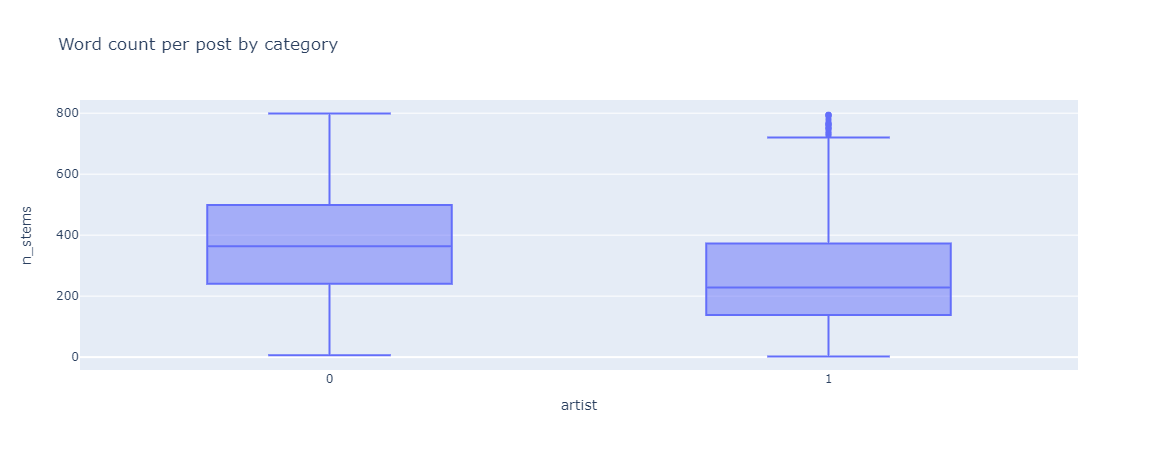
Based on the experiment, the ratio of nouns in the text descriptions is the most notable feature compared with the ratio of the other part of speech. Figure 2-6 is the box plot that illustrates the ratio of nouns in each text description for the real and the fake postings respectively, where *“0”* represents the real and *“1”* represents the fakes. It is noticeable that the fake postings tend to have a higher ratio of nouns in each text description compared with the reals. Furthermore, the real postings have much more outliers compared with the fakes. It also occurs in box plots for other part of speech comparisons. Similar as the polarity and subjectivity distribution, the real postings have a larger variance compared to the fake postings, which may be caused by the lack of fake postings.



**Figure 2-6:** Ratio of Nouns in Text Description Box Plot of Postings per Category

### Number of Words

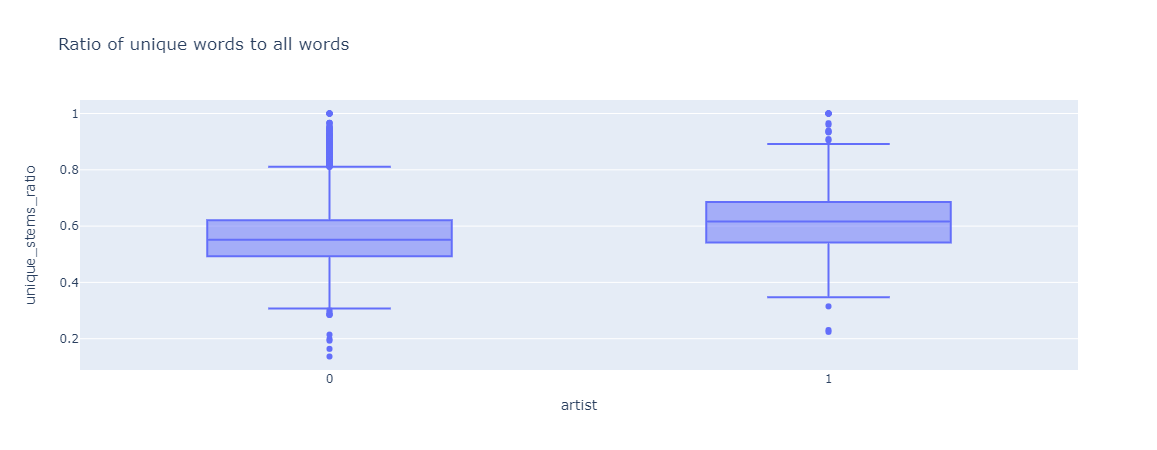
Number of words is a feature that reflects some part of writing style of the typical length of a type of job postings. Figure 2-7 shows the box plot of the number of words for each category. Similarly, *“0”* represents the real job postings, *“1”* represents the fakes. As the figure illustrates, the real postings tend to have more words in each text description, while the distribution of both categories is wide.



**Figure 2-7:** Word Count in Text Description Box Plot of Postings per Category

### Repeated Words

Repeated words are the way to express strong emotions or to put the emphasis on certain stories or events in traditional NLP feature engineering. In this experiment, the ratio of unique words is used as the feature for further recognition. Figure 2-8 is the boxplot that shows the ratio of unique words from two categories. Same as before, *“0”* represents the reals and *“1”* represents the fakes. Most of the fake job postings tend to have a higher ratio of unique words compared with the reals. However, as mentioned in the TF-IDF score, the fakes have a lower TF-IDF score, which means that the fake job postings tend to have a more usual expression pattern but with less repeated words compared with the reals.



**Figure 2-8:** Ratio of Unique Words Box Plot of Postings per Category

## Oversampling

Most of the results in feature engineering show that the real job postings tend to have a wider distribution, while it may be caused by the lack of the fake job postings. To improve the performance of the following traditional machine learning models, an oversampling method is used to expand the fake job postings.

The new samples of fake job postings are based on the features extracted in the last section. To perform the oversampling, the k-nearest neighbors algorithm is used to find the neighbors of the fake postings sample , where in this experiment, the number of neighbors selected is 5. After the neighbors are selected, the neighbors that belong to the fake postings are selected as . And the feature vector of the new sample for the fake postings is calculated using the following formula:

where is a random value in the range of . For each pair of and , there are 4 new samples generated by the above formula. The reason After oversampling, there are 16161 real job postings and 16652 fake job postings in the oversampled dataset. However, it can only be used in traditional models, since the deep learning models require the actual English words to be converted into vectors, which are not generated based on this simple oversampling method. The precision of the fake category by traditional machine learning models using oversampled dataset will be compared with the original dataset after preprocessing and feature engineering in the third section.

## Models

### Logical Regression

Logistic Regression (LR) is a supervised machine learning algorithm that models the discrete outcome. LR uses generalized linear model for classification. While LR is mostly used for binary classification, multinomial LR can perform multi-classification.

Multinomial LR predicts the probability of each outcome via a linear function: , where is the regression coefficient vector of outcome , and is the feature vector of the -th input. Suppose there are possible outcomes, the probability estimation of the-th outcome is: {\displaystyle f(k,i)=\beta \_{0,k}+\beta \_{1,k}x\_{1,i}+\beta \_{2,k}x\_{2,i}+\cdots +\beta \_{M,k}x\_{M,i},}

*=*

The coefficient vector is usually estimated by maximizing the posteriori probability estimation, and log-likelihood function is a common choice to maximize the probability estimation.

### Support Vector Machine

SVM is a supervised machine learning algorithm that is believed to be one of the most robust prediction methods. SVM is a non-probabilistic classifier, and it performs linear classification by maximizing the distance between the boundaries and the samples**.**

In SVM, the training samples are correctly classified with a nonzero margin, which is the distance between the boundaries and the sample. For the -th sample , there is:

The support vector is then introduced to minimize the regular penalty subject to for each sample in the set. Apart from linear classification, SVM can also map features into higher dimensions via kernel functions, and the linear classifier is later applied to the mapped features.

### Random Forest

Random Forest (RF) is a machine learning model constructed by ensembling a multitude of decision trees. To perform classification, the output of RF is the class that is selected by most trees. Assume the number of trees is , the algorithm of RF is shown below:

**Algorithm 2-1: Random Forest**

|  |
| --- |
| *for=1 to:*  *Draw a bootstrap sample of size from the training data*  *= number of variables*    *= 1*  *Grow a random-forest tree to the bootstrapped data by recursively repeating the following steps on each terminal node until the minimum node size is reached.*  *(i) Select variables randomly out of the variavles*  *(ii) Choose the best variable splitting point out of the variables*  *(iii) Split the node into two daughter nodes* |

### AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble learning model that improves weak classifiers by combining the outcomes of weak classifiers into a weighted sum that represents the final output of the boosted classifier. The classifiers are learned sequentially, and the training data is reweighted for the next classifier.

Assume there are weak classifiers and given the training data , the algorithm of AdaBoost is described as follow:

**Algorithm 2-2: AdaBoost**

|  |
| --- |
| *for =1 to :*  *Train weak classifier with {}*  *Calculate the error rate*  *for =1 to :*    *If is misclassified by :*  *=*  *Else:*  *=*  *Obtain the set of functions {}*  *Aggregate the functions* |

There are multiple choices to aggregate the classifiers, including uniform weight aggregation and non-uniform weight aggregation.

### Word to Vector

The *glove.6B.300d* in *pytorch.vocab* dictionary is used for converting words into vector. The processing of text can be simplified into vector operations in the vector space, and the similarity in the vector space can be calculated to represent the semantic similarity of the text. The similarity between two vectors is measured by the cosine of the angle between them. During comparison, only the direction of the vector is considered.

### TextCNN

In this experiment, the TextCNN is implemented by *pytorch* library. The core idea of convolutional neural network is to capture local features, which are sliding windows composed of several words for text data. The advantage of convolutional neural network is that it can automatically combine and filter N-gram features to obtain semantic information at different levels of abstraction. The following table shows the structure of the neural network.

**Table 2-3:** Neural Network Structure of the TextCNN

|  |  |
| --- | --- |
| **(embed)** | Embedding (15707, 300) |
| **(convs)** | Conv2d (1, 100, kernel\_size = (3, 300), stride = (1, 1)) |
| Conv2d (1, 100, kernel\_size = (4, 300), stride = (1, 1)) |
| Conv2d (1, 100, kernel\_size = (5, 300), stride = (1, 1)) |
| **(dropout)** | Dropout (p=0.5, inplace = False) |
| **(fc)** | Linear (in\_features = 300, out\_features = 2, bias = True) |

The embedding layer is used to convert the words into the vectors defined in the dictionary. The above embedding layer used a dictionary with 105934 words and convert each word into a 300-dimension vector. After embedding, 3 convolution layers are used with kernel size (3, 300), (4, 300) and (5, 300) respectively to capture the local features.

### Transformer

In this experiment, the Transformer structure is implemented by *pytorch* library. Transformer is an encoder-decoder architecture that uses the Attention as primary mechanism, which allows the neural network to pay more attention to relevant information. The structure of Transformer is shown in Figure 2-9 by Vaswani A in the classic paper *Attention is all you need*.

图示

描述已自动生成

**Figure 2-9**: The Structure of Transformer

Transformer adopts the design of key-value Attention, using matrix (query), (Key), and (Value) for Attention, which is defined as:

where ，，. The multi-head Attention based on Attention is defined as:

where ，，.

# Results

## Result of Traditional Machine Learning Models

### Original Data

For the original data after preprocessing and feature engineering, table 3-1 demonstrates that the accuracy of the real postings category, the fake postings category and the total accuracy, where 80% of the dataset is randomly selected as the training set and the rest 20% is used as the test set. It is noticeable that the accuracy of the fake postings is terrible in most of the models except the Random Forest.

**Table 3-1:** The Accuracy of Models for the Original Dataset

|  |  |  |
| --- | --- | --- |
| **Machine Learning Models** | **norm** | **Accuracy** |
| **Logistic Regression** | **real postings** | 0.95 |
| **fake postings** | 0.00 |
| **total** | 0.95 |
| **Support Vector Machine** | **real postings** | 0.95 |
| **fake postings** | 0.00 |
| **total** | 0.95 |
| **Random Forest**  **(Number of estimators = 10)** | **real postings** | 0.97 |
| **fake postings** | 0.98 |
| **total** | 0.97 |
| **AdaBoost**  **(Number of estimators = 100)** | **real postings** | 0.95 |
| **fake postings** | 0.60 |
| **total** | 0.95 |

### Oversampled Data

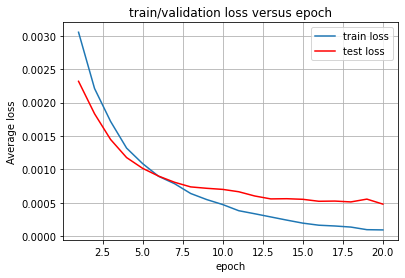
For the oversampled dataset that has a balance category structure, table 3-2 demonstrates that the accuracy of the real postings category, the fake postings category and the total accuracy, where 80% of the dataset is randomly selected as the training set and the rest 20% is used as the test set. It is noticeable that the accuracy of the fake postings has upgraded dramatically while the accuracy of the reals has declined in most of the models except the random forest.

**Table 3-2:** The Accuracy of Models for the Oversampled Dataset

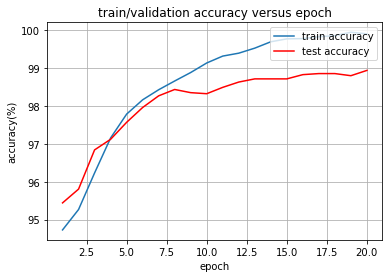
|  |  |  |
| --- | --- | --- |
| **Machine Learning Models** | **norm** | **Accuracy** |
| **Logistic Regression** | **real postings** | 0.75 |
| **fake postings** | 0.80 |
| **total** | 0.77 |
| **Support Vector Machine** | **real postings** | 0.73 |
| **fake postings** | 0.70 |
| **total** | 0.71 |
| **Random Forest**  **(Number of estimators = 10)** | **real postings** | 0.99 |
| **fake postings** | 0.99 |
| **total** | 0.99 |
| **AdaBoost**  **(Number of estimators = 100)** | **real postings** | 0.84 |
| **fake postings** | 0.85 |
| **total** | 0.84 |

## Result of TextCNN

For the TextCNN model, the dataset used is the original dataset, since the oversampling method can not produce new text descriptions. In this experiment, same as the metric in traditional machine learning models, 80% of the dataset is randomly selected as the training set, and the rest of the 20% is used for validation and test. Figure 3-1 demonstrates the train and the validation loss during the 20 epochs, the result turns out that the model converged successfully after 20 epochs. Figure 3-2 demonstrates the train and validation accuracy during the 20 epochs, and the final best accuracy for the validation/test set is 98.93%.



**Figure 3-1:** Train & Validation Loss versus Epoch

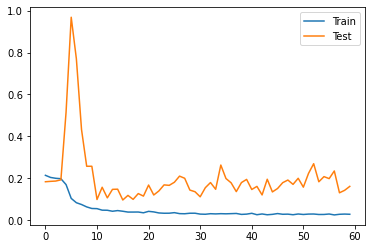


**Figure 3-2:** Train & Validation Accuracy versus Epoch

Based on the above result, TextCNN has converged very well, and it has a remarkable performance on the recognition. However, the test accuracy is a little lower than the train accuracy after 20 epochs.

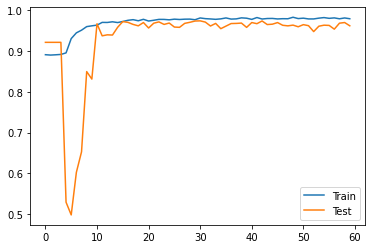
## Result of Transformer

For the Transformer model, the dataset used is the original dataset same as the dataset used in TextCNN model. In this experiment, same as the metric in traditional machine learning models, 80% of the dataset is randomly selected as the training set, and the rest of the 20% is used for validation and test. Figure 3-3 demonstrates the train and test loss versus 60 epochs.



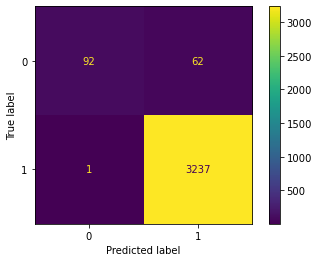
**Figure 3-3:** Train & Test Loss versus 60 epochs

As the above figure shows, the Transformer model did not converge as best as the TextCNN model. There is a huge fluctuation at about 5-10 epochs. Even after 60 epochs, the test loss is still fluctuating slightly. Figure 3-4 shows the train and test accuracy versus 60 epochs.



**Figure 3-4:** Train & Test Accuracy versus Epoch

Even there is a huge fluctuation at 5-10 epochs, the model still has a remarkable performance after 60 epochs. The model has the best test accuracy 98.14%. It is slightly lower compared with the TextCNN model. As for the precision of the fake job postings category, figure 3-5 shows the relation between the true label and the prediction label in the test set.



Specially, unlike the former marks, in the above figure, *“0”* represents the fake postings, and *“1”* represents the reals based on the encoding pattern of Transformer model. It is noticeable that for the real postings, the accuracy is up to 99.96%, while for the fake postings, the accuracy is only 59.74%. Therefore, for the fake category, Random Forest shows the best performance.

# Conclusion

## Dataset

The Employment Scam Aegean (ESA) dataset used in this experiment has an extremely unbalanced category distribution. Therefore, it is hard to classify the minority class, which is the fake job posting class. In this experiment, an oversampling method is used to expand the fake job posting samples. After the oversampled dataset is used for traditional machine learning models, the accuracy of recognition for the minority class has a dramatic promotion.

However, the oversampling method has bias, the feature vectors of the new samples generated are always in the range of its center sample and the neighbors of the center sample. Therefore, the distribution range of the fake job postings did not change, which can not simulate the realistic situation perfectly. Another limit is that the above oversampling method is difficult to generate new rational text description that can be used for the Deep Learning models.

## Features

In this experiment, mainstream features used in NLP and the original element *“has\_company\_logo”* are used for the traditional machine learning models. As the result shown in the Feature Engineering section, the distribution of both categories is very wide, and the difference of the feature vectors between two categories is not remarkable enough. In the pre-experiment, the result of the traditional machine learning models with only TF-IDF vectors feature shows a great performance too, although the feature vectors after the feature engineering is slightly better. That shows that the TF-IDF feature may be the most significant feature.

Considered about the huge number of missing values in lot of text elements, all text elements are merged into one general text description for each sample in feature engineering. However, the merging caused some significant features that lying in each element loss.

## Result Analysis

In this experiment, for the original dataset, the TextCNN model has achieved the best overall performance. And the Deep Learning models showed a slightly better result compared with the traditional machine learning models.

Specially, it is noticeable that for the Transformer model, after 60 epochs, the validation loss and the test accuracy still had a little fluctuation, while it did not happen on the TextCNN model after 20 model. It is barely possibly caused by without vector normalization. This is mainly caused by the reduction of the embedding size and the forward dimension, since my laptop cannot run the model with the original embedding size and forward dimension setting. Hence, the model cannot converge to the data more perfectly.

Moreover, in this experiment, Random Forest model has shown a remarkable performance compared with other traditional machine learning models. The reason is that Random Forest is able to handle high-dimensional (large number of features) data without feature selection, since the feature subsets are randomly selected and all the randomly chosen decision trees will be evaluated. Therefore, Random Forest is able to realize the interaction among features and which features are more significant. It is also remarkable that Random Forest showed a great performance on the extremely unbalanced dataset. That is because the resampling mechanism of Random Forest, which makes it strong enough when encounter the unbalanced data.

## Improvements

In the feature engineering section of the ESA dataset, it is possible to find more specific feature from the merged text description. That is, more feature engineering methods can be used but not limited in tackling the problem as NLP problem.

For this extremely unbalanced dataset, although some models can still get the great performance, the more objective and pseudo-realistic oversampling method should be performed to get an expended dataset which can be used in deep learning models. Furthermore, a higher quality dataset with balanced categories and fewer missing values is required to perform the recognition of fake job postings better.

# References

[1] Dutta, S. and Bandyopadhyay, S.K., 2020. Fake job recruitment detection using machine learning approach. International Journal of Engineering Trends and Technology, 68(4), pp.48-53.

[2] Ranparia, D., Kumari, S. and Sahani, A., 2020, November. Fake job prediction using sequential network. In *2020 IEEE 15th international conference on industrial and information systems (ICIIS)* (pp. 339-343). IEEE.

[3] Habiba, S.U., Islam, M.K. and Tasnim, F., 2021, January. A comparative study on fake job post prediction using different data mining techniques. In *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)* (pp. 543-546). IEEE.

[4] Vo, M.T., Vo, A.H., Nguyen, T., Sharma, R. and Le, T., 2021. Dealing with the class imbalance problem in the detection of fake job descriptions. *Computers, Materials & Continua*, *68*(1), pp.521-535.

[5] Engel, J. (1988). "Polytomous logistic regression". Statistica Neerlandica. 42 (4): 233–252. doi:10.1111/j.1467-9574.1988.tb01238.x.

[6] Bennett, Kristin P.; Campbell, Colin (2000). "Support Vector Machines: Hype or Hallelujah?". SIGKDD Explorations. 2 (2): 1–13. doi:10.1145/380995.380999. S2CID 207753020.

[7] Ho, Tin Kam (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[8] Freund, Yoav; Schapire, Robert E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". Journal of Computer and System Sciences. **55**: 119–139.

[9] Vaswani A., et al. 2017. Attention is all you need. In NIPS, 5998–6008.