ISyE 6416 – Basic Statistical Methods – Spring 2020
Final Report

Team Member Names: Bo Yuan, Osman Dai, Steve Wong

Project Title: Fake Job Posting Prediction

Partition of roles:

Bo Yuan: research, implementation (coding) and explanation (in report) of bag of words feature extraction, implementation of classification methods
Osman Dai: implementation and explanation of categorical features and of orthographical features, organization of report
Steve Wong: research and explanation of classification methods, explanation of future work

Problem Statement:

For this project, we consider the problem of identifying fraudulent job postings among a large database of postings. This is a problem that is of great interest to career websites such as Linkedin.com or Indeed.com. We develop a system that detects a significant portion of false job postings while mistaking very few legitimate ones. As we explain in the discussion section of our report, since we imagine this system to be deployed with human supervision, the low false positive rate is critical for the deployment of such a system and is more important than achieving zero false negatives.

One key characteristic that makes the fake job description prediction unique from traditional classification problems is that it involves much textual information. However, the textual information alone may not be strong enough to show one description is fake. A more natural way, which is also commonly used in daily lives, is to find contradictions. For example, it says the required skills are quite simple but could offer a large amount of salary. In summary, to increase the prediction accuracy, one shall extract key information from the raw description, i.e., the required education background, the employment type or the industry background. Then combine these categorical features with remained documents to build the final prediction model. And this is exactly how the data file arranges the raw data.

Dataset:

The dataset we use contains 18k jobs descriptions out of which nearly 800 are fraudulent. Each posting in the dataset has 18 associated fields of various formats. These include numerical fields such as job id, boolean fields such as the identifier for fraudulent/true postings, fields with brief labels such as job title or location and finally lengthy descriptions such as company profiles or job requirements. An example of a posting is given below:

Label in the dataset	Explanation	Example
job_id	A unique id for each job description	1
title	The title of the job ad entry	Marketing Intern
location	Geographical location of the job ad	US, NY, New York
department	Corporate department	Marketing
salary_range	Indicative salary range	$50,000-$60,000
company_profile	A brief company description	We're Food52, and we've created a groundbreaking …
description	The details description of the job ad	Food52, a fast-growing, James Beard …
requirements	Enlisted requirements for the job opening	Experience with content management systems …
benefits	Enlisted offered benefits by the employer	What you will get from us Through being part of …
telecommuting	True for telecommuting positions	0
has_company_logo	True if company logo is present	0
has_questions	True if screening questions are present	0
employment_type	full-type, Part-time, Contract, etc.	Full-time
required_experience	Executive, Entry level, Intern, etc.	Internship
required_education	Doctorate, Master's Degree, Bachelor, etc.	Bachelor's Degree
industry	Automotive, Health care, Real estate, IT etc.	Computer Software
function	Consulting, Research, Sales etc.	Sales
fraudulent	target - Classification attribute	0

The data source is obtained in kaggle.com. One can attain this dataset by searching its title, 'Fake JobPosting Prediction'.

Feature extraction:

The processing step is to make use of the well-known bag-of-words approach for fields that include lengthy textual descriptions. Furthermore, considering the difficulty of the task, we define some additional features that characterize postings to support to the bag-of-words features. In the second step we are considering trying various machine learning techniques.

- **Bag-of-Words:**

Bag-of-words approach, as what it indicates in the name, assumes all the words are hold in bags where the order of words is ignored. This simplified representation has three steps. First, pre-process data. For a whole sentence, we should remove unnecessary punctuation, tags, stop words such as 'is',' the', etc. that do not have specific semantic. Then we shall reduce the remained words to their root forms, e.g. from 'extracting' to 'extract'. There are mainly two approaches to achieve it. One is called stemming where certain algorithms mechanically identity unnecessary suffixes and remove them. The other one, lemmatization, is based on known database storing words with similar meanings. One advantage of the lemmatization method is it can group words with different forms, which is very useful to identify traits existing in nature. Furthermore, grouping words in different physical formats gives fewer number of features. This advantage is highly valuable in the next step of training models. Although lemmatization has its advantages, the stemming approach is wider used in practice, since it is easier to implement without external database and more robust towards noisy data with spelling mistakes. Hence, we select stemming as our way to recover words' roots. More specifically, we utilize nltk library in the implement.

In the next step, we count the frequency of each word w_i occurring in one document d_j. Then adopt $c_{i,j}$ to denote this value. A more advanced representation is called 2-gram, which partly overcome the shortage of disregarding orders. A 2-gram term is a contiguous sequence of n items. i.e., for a sentence 'This is nice.', the collection of 2-gram terms is 'this is' and 'is nice'. This partition utilizes the context information and captures more textual structure from raw data. However, the cost is the increment of the number of features which is not good for prediction. With comparable advantages for both methods, we have implemented both frequency count and 2-gram and tested to analysis which works better in terms of prediction accuracy.

After simple counting, we convert the frequency into term frequency-inverse document frequency (TF-IDF) values. They are defined as follows:

Term Frequency (TF) = $\frac{c_{i,j}}{\sum_i c_{i,j}}$

Inverse Document Frequency (IDF) = log(N/n)

Here, N is the total number of documents, and n is the number of documents a term t has appeared. TF-IDF value is the product of TF and IDF. TF highlights the words frequently appearing in one document, and IDF can mark the terms that are distinct. Thus, this value TF- IDF cooperates the inter and intra information and widely used in natural language process.

The last phase of bag-of-words approach is feature selection. In most cases, even if the data were to be preprocessed, the huge number of features cannot be directly utilized in prediction. Typically, there may be thousands of words given by the whole

dataset. This magnitude of variables may dramatically reduce the computation efficiency and may even cause the calculation ill-defined. We implement this by so-called information gain criterion. We give each term’s information gain score and select the ones with highest scores.

- **Categorical Features**

In addition to the large number of lengthy descriptive fields contained in each posting, such as 'company_profile', 'benefits' and 'requirements', the dataset also includes many fields that contain brief categorical information, such as 'employment_type' (e.g. 'full-time'), 'required_experience' (e.g. 'internship'), required_education (e.g. 'bachelor's degree'), 'function' (e.g. 'marketing') and industry (e.g. 'telecommunications'). To extract features expressing information from these fields, we have decided that categorical features are the ideal approach.

As such, we firstly set identify a dictionary of categories for each of the fields. This dictionary contains the possible values the field can take. With the newly acquired dictionary of the field, we perform one hot encoding to encode the entry in every field: E.g. consider the field employment_type. This feature has dictionary ['contract', 'full-time', 'part-time', 'temporary', 'other'] and has size 5. We then encode a posting of type 'contact' with the binary vector [1,0,0,0,0], a posting of time 'full-time' with the binary vector [0,1,0,0,0] etc.

We briefly introduce the dictionaries of categories we have extracted for each field below:

- employment_type (size 5)
  ['contract', 'full-time', 'part-time', 'temporary', 'other']

- required_experience (size 6)
  ['director', 'associate', 'mid-senior', 'entry level', 'internship', 'not applicable']

- required_education (size 8)
  ['unspecified', 'high school or equivalent', 'bachelor's degree', 'master's degree', 'doctorate', 'certificate', 'vocational', 'some college coursework completed']

- function (size 32)
  ['accounting/auditing', 'administrative', 'advertising', 'art/creative', …, 'training', 'writing/editing']

- industry (size 143)
  ['accounting', 'airlines/aviation', 'alternative dispute resolution', …, 'wine and spirits', 'wireless']

In addition to these features, we have made the choice to simplify location information by considering the country where a job is located instead of also considering the city information. Countries are coded using a 2-letter system. E.g. US for the USA, CN for China, TR for Turkey. There are 89 such codes in the database.

In addition to these k-ary categories, the dataset includes three fields for binary categories:

- 'telecommuting'
- 'has_company_logo'
- 'has_questions'

We use these three fields to add one-hot encoded features. E.g. [0,1] for postings that have a company logo and [1,0] for companies that don't.

- **Special features (orthographic features)**

Through our preliminary observations on the dataset and specifically based on our observations of the fraudulent job postings, we have observed orthographic errors to be disproportionately frequent in fraudulent job postings. This is not surprising as one would reasonably expect legitimate posts to have been written with diligence.

Based on this observation we extract the 5+1 following features to characterize such anomalies in postings:

  - Being written in all lowercase letters,
  - Being written in all uppercase letters,
  - Not having a space after punctuation marks,
  - Words at the beginning of sentences not having their first letter capitalized,
  - Words starting with a consonant following the indefinite article 'an' or similarly words starting with a vowel following the words starting with the indefinite article 'a'.

Finally, we include one final feature to denote correct orthography. If none of the cases above hold, the correct orthography feature is 1, otherwise it is 0.

We extract these 6 features for all textual fields.

- **Finalized features**

The raw variables are divided into two groups: lengthy textual variables and categorical variables. The primary question for this project is how the two different types of predictors contribute to the learning model's accuracy. In the first section of result analysis, a series of classifiers were trained using only the textual variables such as 'benefits', 'requirements', etc. It is necessary to convert those textual features into numerical features so the machine to understand manipulate. The textual features were converted into statistic-based TF-IDF values. This technique was chosen due to ease of implementation while providing a basic metric of comparing documents. In total, there are 129 TF-IDF features after the conversion. In addition, to improve the computability of the model, PCA was adopted to reduce the dimension of variables from 129 to 113.

In the second part of the experiment, we include the categorical features and orthographic features in addition to the bag of words features. For the additional features, we perform PCA to reduce the dimension down to 120.

Experiment and classification:

1000 samples from the original dataset were selected for the experiment to be used as the test set. To negate the effect of different data distribution on the predication accuracy of the learning models, the same training set with the same data distribution were utilized for this part of the experiment. For the fake-to-true job posting sample distribution, 950 sample are true posters while the other 50 samples are fake. The small ratio of fake postings is not problematic, as we would expect the number of false postings to be very small in a real-life scenario.

We have made use of 5 classes of classifiers. We present the experimental results for each case in the form of confusion matrices, both in numbers of actual postings as well as in percentages. The percentage confusion matrix is color coded by intensity.

- **Random Forest**

A classification tree is grown by repeatedly splitting the dataset into two separate classes based on the features. Depending on how the tree is grown, the accuracy of the models changes with the optimal predicator in the middle of deeply and shallowly grown tree. A single tree is seldomly used for predication due to its inaccuracy. With multitude ways of bi-sect successively, it is highly improbable that one tree picked would be an exact reflection of the true hypothesis. To improve on such probability, a set of multiple trees – a forest – can be grown to improve the accuracy and reduce the variance by the virtue of law of large numbers.

	With BoW features				With all features			
	942	43	0.991579	0.86	945	39	0.994737	0.78
fine tree	8	7	0.008421	0.14	5	11	0.005263	0.22
	950	46	1	0.92	950	45	1	0.9
medium tree	0	4	0	0.08	0	5	0	0.1
	950	48	1	0.96	950	46	1	0.92
coarse tree	0	2	0	0.04	0	4	0	0.08
	950	47	1	0.94	950	45	1	0.9
bagged tree	0	3	0	0.06	0	5	0	0.1

- **Quadratic and Linear Discriminant Model**

A statistical method of classifying the dataset, the quadratic discriminate model is fitted to split the dataset into multiple groups based on the multivariate Gaussian. It is assumed that the two probability density functions are normally distributed. Under special circumstances where the variance for the two labels are equal, the model becomes linear discriminant with a linear bound in between.

	With BoW features				With all features			
linear discriminant	943	38	0.992632	0.76	944	35	0.993684	0.7
	7	12	0.007368	0.24	6	15	0.006316	0.3
quadratic discriminant	946	35	0.995789	0.7	945	35	0.994737	0.7
	4	15	0.004211	0.3	5	15	0.005263	0.3

- **Support Vector Machine**

With the data spread out in the dimensional space, Support Vector Machine (SVM) tries to find a hyperplane that spreads the two groups of data and maximize the space between the hyperplane and the closest data point for each class. The hyperplane need not be linear but in higher order such as cubic and quadratic. The order of the SVM is defined by the order of the kernel for the hyperplane.

	With BoW features				With all features			
linear svm	950	47	1	0.94	950	42	1	0.84
	0	3	0	0.06	0	8	0	0.16
quadratic svm	950	37	1	0.74	950	34	1	0.68
	0	13	0	0.26	0	16	0	0.32
cubic svm	947	35	0.996842	0.7	948	33	0.997895	0.66
	3	15	0.003158	0.3	2	17	0.002105	0.34

- **Logistic regression**

The purpose of the logistic regression is model binary dependent variable. Although the form of the function is logistic, the behavior is linear in natural. The logistic function is a sigmoid function with the dependent variable, z, being the dot product of the input feature vector, v, and weight coefficient, w. Since the output is not dependent on product of the features, this make logistic regression a linear model. With the given dataset, the algorithm will try to maximize the log likelihood function by optimizing w, the weight coefficient.

	With BoW features				With all features			
	946	38	0.995789	0.76	934	32	0.983158	0.64
logistic regression	4	12	0.004211	0.24	16	18	0.016842	0.36

- **k-Nearest Neighbors**

The core idea behind k-Nearest Neighbors is that a point in space can infer information about its own class by looking at its neighboring points within a proximity. Classification is based on how many total numbers of neighbors to look at – K, as defined by the user – and the majority determines the point's classification. The distance between points can be express as Norm of the number of features. By computing the norm of the point against the dataset and sorting the results afterwards, the algorithm can classify the point according to the above heuristics.

	With BoW features				With all features			
	942	31	0.991579	0.62	943	29	0.992632	0.58
weighted knn	8	19	0.008421	0.38	7	21	0.007368	0.42

Discussion and future work:

Based on the experimental results, we make the following observations

- In each case, the addition of categorical and orthographic features has resulted in an increased performance, both in terms of falsely identifying fewer legitimate postings as fraudulent as well as in terms of identifying more of the fraudulent postings. However, for no classifier has this difference been drastic. This suggests that the bag of words features obtained through textual data is indeed very strong in characterizing the postings in the context of this problem.

  One interesting observation is that adding orthographic features has often resulted in an increased number of legitimate postings being identified as fraudulent. This is most probably explained by the fact that some true job postings do have orthographic mistakes.

- Using a higher complexity variant of a given classifier has, with the exception of random forests, resulted consistently in a higher number of fraudulent posts being detected. However, we must also note that, this increase in true positives seems to be at the cost of a few extra false positives, such as in the case of cubic SVM.

- All classifiers we use have a strong bias towards marking posts as legitimate and never achieve more than 50% detection for the false posts. We argue that this is not a significant problem as we expect the system to be deployed with human supervision. Specifically, we consider a system that forwards all posts marked as fraudulent to some human observer to check any possible inconsistencies. Assuming there will be thousands of job posts on the website, having even a slightly increase rate of false negatives would be very costly as it would require many human supervisors. However, since we expect the number of true positives to be small anyway, the current system can be deployed without resulting in significant costs while still managing to clean up a considerable portion of false postings. Furthermore, we must acknowledge the difficulty of the problem, and that no matter how strong the system is, it would likely never be possible to detect a fraudulent job posting on based on the information on the post. We speculate that stronger detectors would need to perform fact checking through online resources to verify the identities of companies, employers etc.

Despite the satisfactory performance of this system, there are directions that could be studied to get better results:

- We have a very small number of fraudulent job postings compared to the legitimate ones in the training set. While we do acknowledge this imbalance reflects the case in real life, it is nevertheless reasonable to assume that, for training purposes, having more examples of fraudulent posts would help increase the capabilities of our classifier.

- This current model, despite its good performance, does not make use of all available information, such as specific location data or job salaries. We consider it of interest to investigate ways to implement this information into our feature extraction scheme and see whether this allows any measurable improvement in the classification capacity of our system.