Data Preparation and Analysis of Fraudulent Job Postings

CSP 571 Final Project

Alisha Khan
akhan92@hawk.iit.edu

Anette Volkova
avolkova@hawk.iit.edu

Tiffany Wong
twong10@hawk.iit.edu

Abstract

Due to Covid there has recently been a large surge of fraudulent job postings online and on popular career sites. We sought to analyze the dataset, *Real or Fake Job Posting Prediction*, and figure out which attributes were correlated with fraudulency. As well as, create and compare models based off of those attributes, which would be able to correctly predict fraudulent and real job postings. Prior to identifying which attributes most correlated with fraudulency we needed to deal with the large text columns which were present. They were cleaned and split up based on tokenization into a variety of other columns which then were used for modeling. Further data cleaning and engineering was necessary and can be seen within the report. The model which worked best ended up being stochastic gradient boosting, though there were other models which had high classification accuracies as well. Overall, there were a variety of steps and modeling techniques that were utilized to arrive at our final conclusions.

Overview

As the job market grows and shrinks, it is not safe from scams and cons. With more job applications being online, there is a rising concern on fake job postings. Some fake job postings exist because companies want to keep resumes on file, while some have sinister motives related to the personal and financial information the applicant is willing to provide. Due to the pandemic there was a drastic shift to online employment and a dependency on online job postings. This created a large spike in fraudulent job postings as scammers took advantage of this shift[1]. This has been reported by many popular job posting sites and the New York Times even stated, “online human resources schemes where criminals pose as potential employers have soared 295% from a year ago, while schemes used for money laundering have skyrocketed by 609%.” [2]. Due to this rising issue the following analysis overviews patterns in job postings to identify if there are certain patterns that can identify if a job posting is fraudulent.

1. Ravenelle, Alexandrea J, Erica Janko, and Ken Cai Kowalski. *Good Jobs, Scam Jobs*
2. Popper, Nathaniel., *A Job That Isn't Hard to Get in a Pandemic: Swindlers' Unwitting Helper.*

Utilizing *Job Postings Description*, a dataset from Kaggle, to predict if a job posting is real or fake. The analysis below utilizes a few modeling methods; Naive Bayes, logistic regression, stochastic gradient boosting, random forest, and KNN. The numerous different models gave insight into how different modeling techniques have different outputs and accuracies. Our goal is to predict the legitimacy of job postings based on certain features.

Data Processing

The dataset we are using for this project comes from Kaggle, where there are 18,000 job postings total and 800 fraudulent ones. There are both textual, categorical, and range attributes. The original dataset has 18 attributes, ranging from a unique job ID to employment type of the job listing.

Exploratory Data Analysis in Train Dataset

We split attributes into four groups: numerical/range, binary, text, and categorical.

- For binary attributes, these include has_company_logo, has_questions, telecommuting, and fraudulent.

- Most of our dataset include text attributes, which include title, location, department, company_profile, description, requirements, and benefits.

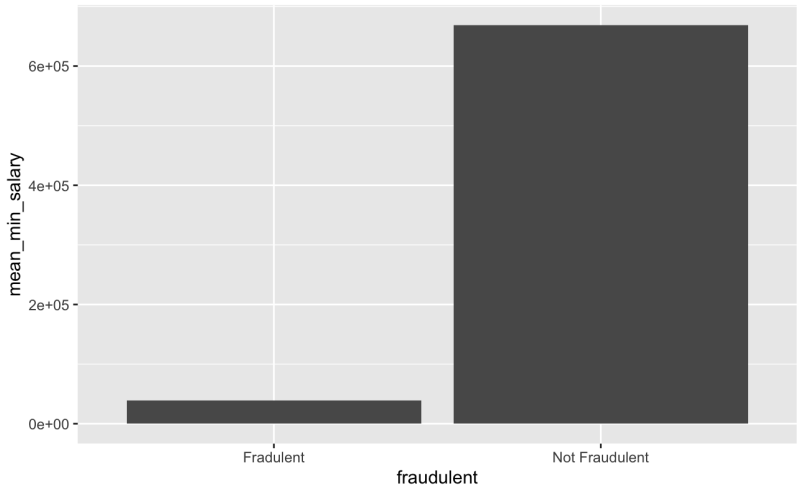
- For the categorical attributes, these include employment_type, required_experience, required_education, industry, and function

Numerical/Range Attributes

Salary Range

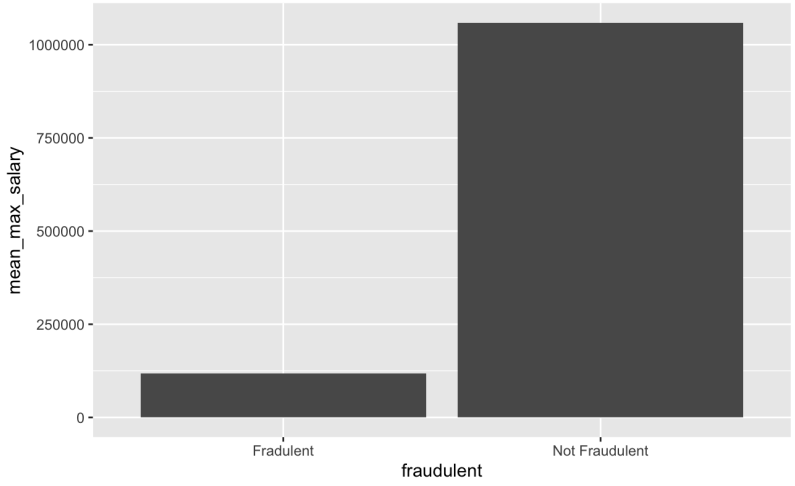
- A string that contains the salary range for the position in the format of min_salary - max_salary. There is no set currency for the salaries
- Broke this into numericals columns for min and max values.
- Distribution of min_salary
    -

```
   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
      0     18000     35000    621178     60000 800000000     12014
```



- Distribution of max_salary

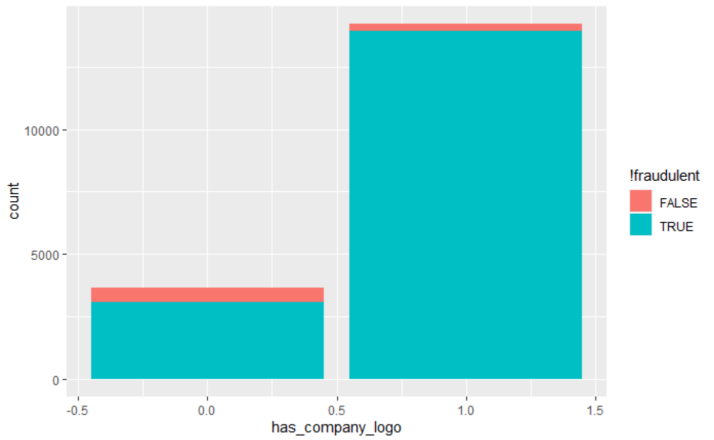
```
      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
0.000e+00 2.500e+04 5.000e+04 9.874e+05 9.000e+04 1.200e+09     12028
```

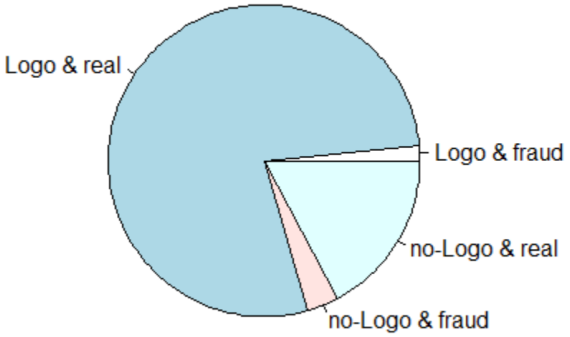


Binary Attributes

has_company_logo

- The feature has_company_logo had no missing values and like stated above is a binary attribute. The distribution of has_company_logo is displayed below (1 or 0). For the barplot the pink represents the fraudulent job postings while the blue represents the real job postings. The first column is the amount of entries which do not have the company logo and the second is the amount of entries that do have the company logo. The pie chart is a representation of this as well from the total amount of data.

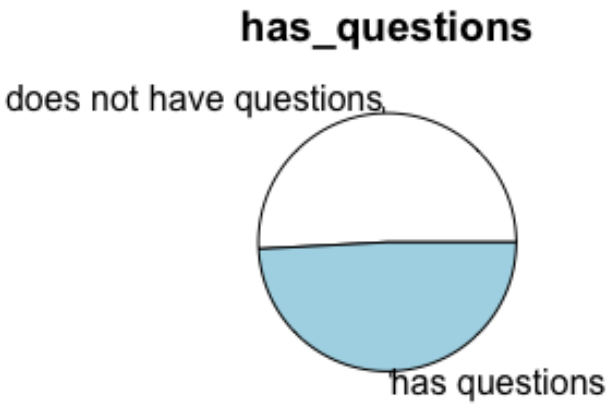




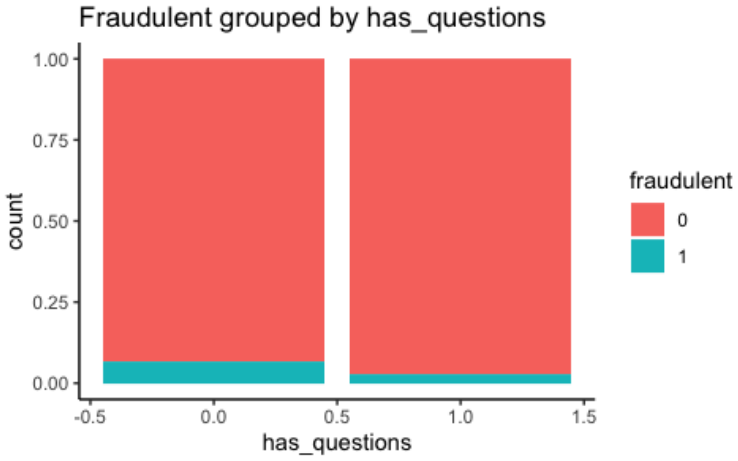
- Due to the lack of NA values it was not necessary to figure out substitutions for those values. In addition since it was not a categorical or text file there was no cleaning necessary as well. As displayed from the visuals above 79.53% of the data has the company logo while 20.47% does not have it.

Has_questions

- Given that has_questions is a binary attribute, there are no missing values. There are 8792 listings with questions, which is roughly 50% of the dataset.



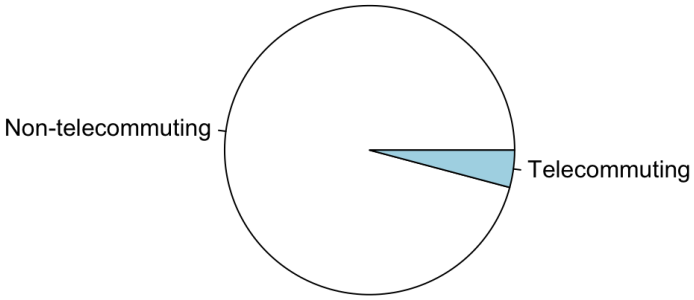
- With no unique values reducing necessary, I plotted the ratio of fraudulent versus non-fraudulent listings within the ratio of listings that have and don't have questions. For listings that do have questions, only 3% of the listings were fraudulent. For listings that do not have questions, 7% of the listings were fraudulent. This is important to take note of because it could mean that a listing not having questions can have a higher chance of being fraudulent.



-

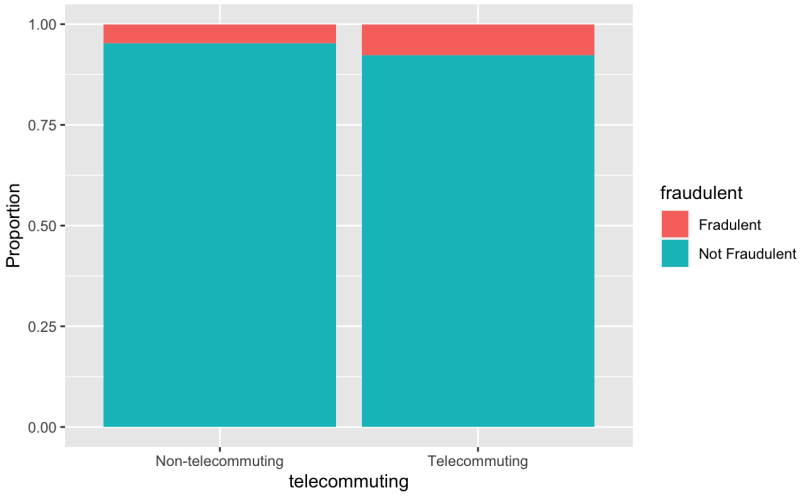
Telecommuting

- Telecommuting means work-from-home jobs. Most jobs are not telecommuting jobs.



-
-

	Non-telecommuting	Telecommuting
Counts	13709	595
Fraction	0.958	0.042



-

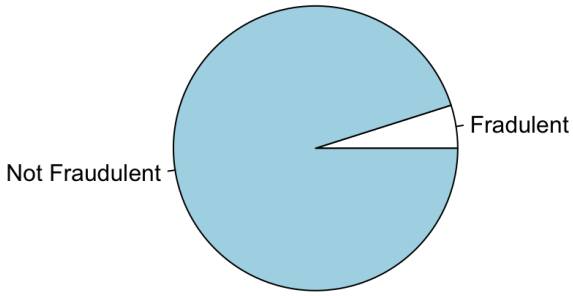
telecommuting <chr>	fraudulent <chr>	n <int>	pct <dbl>	lbl <chr>
Non-telecommuting	Fradulent	654	0.04770589	5%
Non-telecommuting	Not Fraudulent	13055	0.95229411	95%
Telecommuting	Fradulent	46	0.07731092	8%
Telecommuting	Not Fraudulent	549	0.92268908	92%

-

    - 8% of telecommuting jobs are fraudulent compared to 5% of non-telecommuting jobs.

Fraudulent

- Fraudulent is our target attribute and there were no missing values.



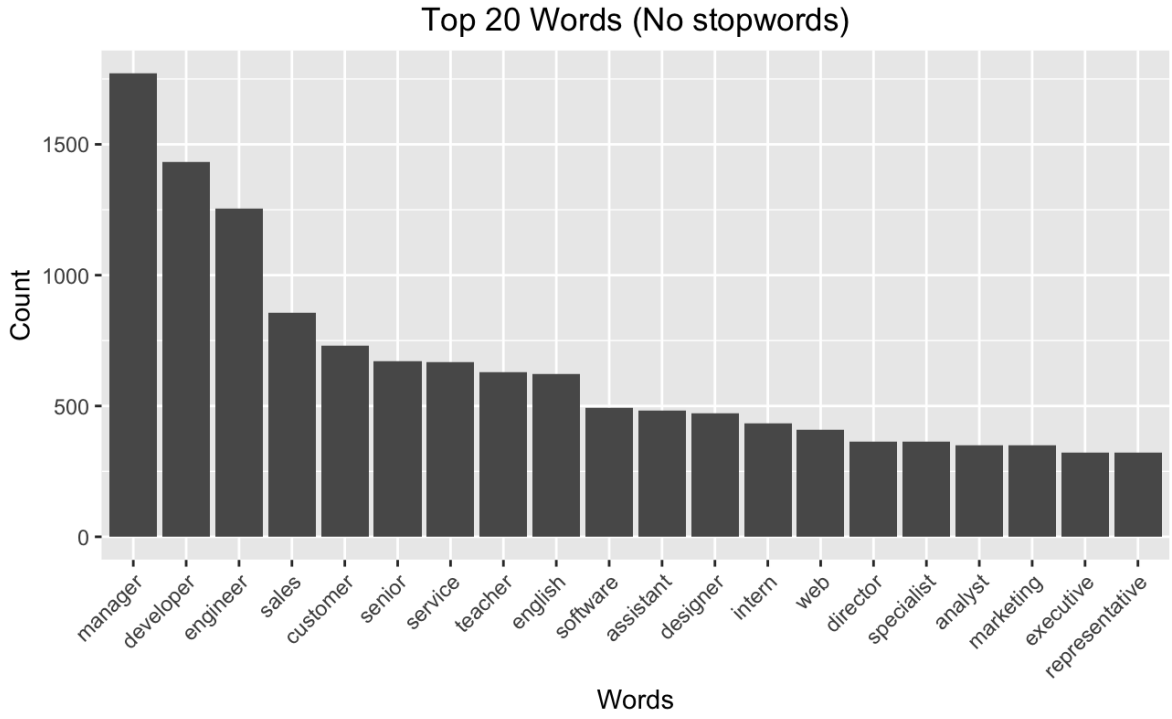
-
-

	Fraudulent	Not Fraudulent
Counts	700	13604
Fraction	0.049	0.951

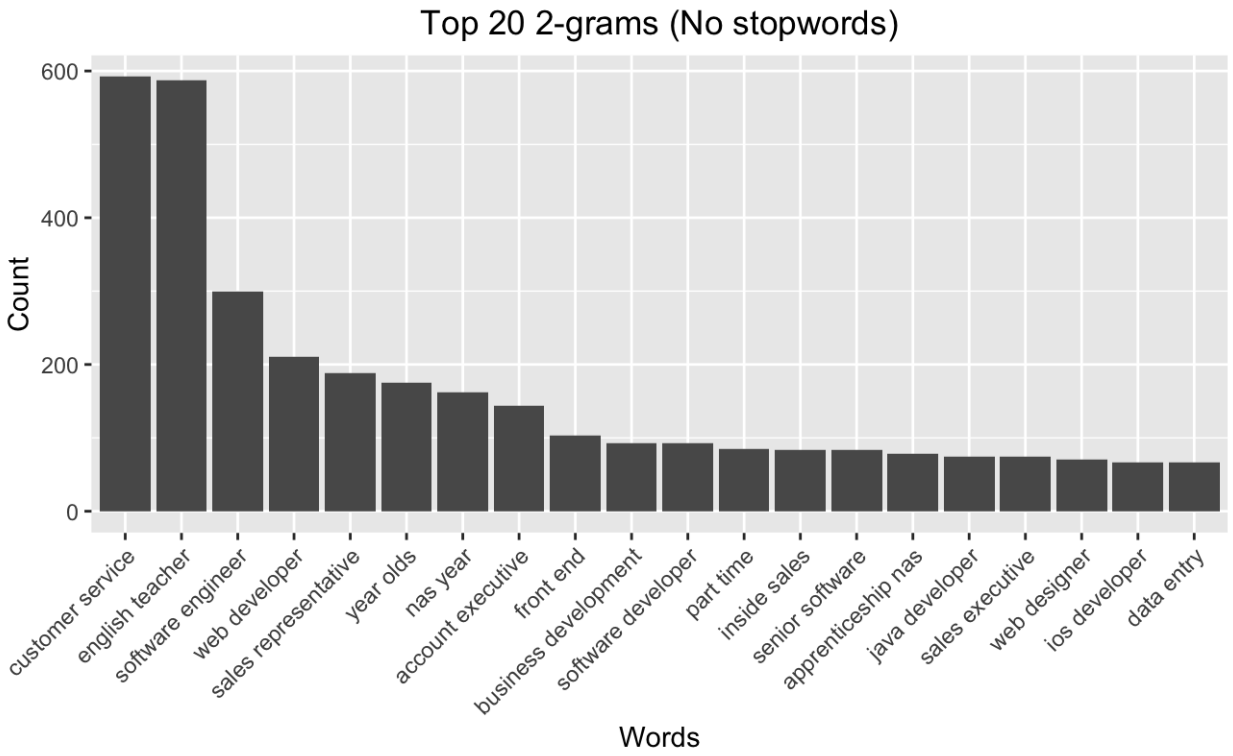
Text Attributes

Title

- The title feature contains the title of the job posting. This column contained no NA's. Upon observation of frequent unigrams and bigrams, I was able to determine some groups for job postings, like teachers, managers, customer service, assistants, and interns.



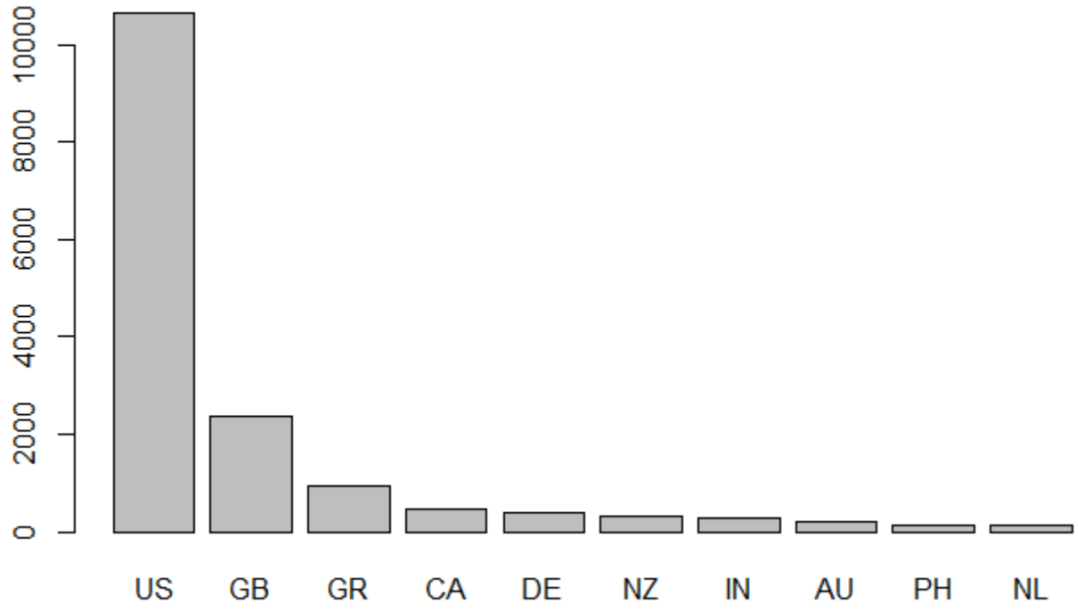
-



-

Location

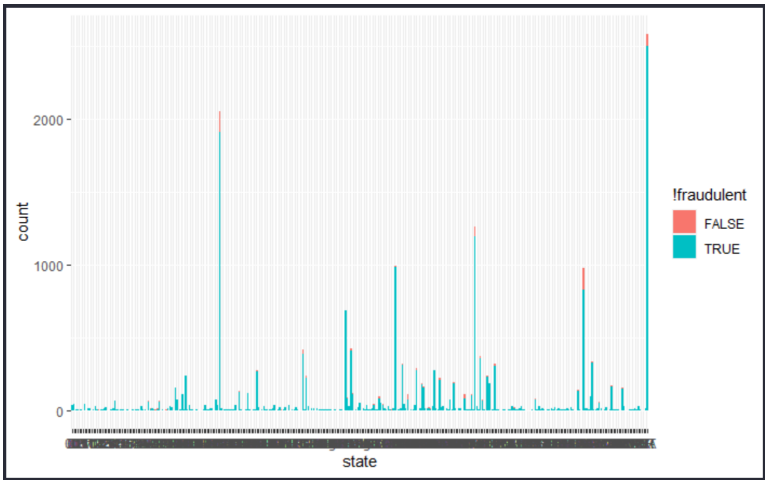
- The location feature held varying information of country, state, and city; and initially included 346 missing values and 3106 unique values. The first step in cleaning up this attribute was to split location based on its respective information. As stated above location was therefore split into country, state, city, and a new feature called region. The location feature itself was then omitted due to the split.
- The new country feature includes 346 missing and 91 unique values, because of this the attribute was created into a dummy variable since it had certain repeated values. The mode of country is US, which came in at 10656 values which is about 60 % of the countries listed.
- Missing values originally were being dealt with in two ways. The first was replacement with the mode. This seemed to create a skew within this feature due to the already large count of US countries. Additionally, due to the other three attributes that depended on this one there had to be careful consideration of those as well. After acknowledgement of dependency from state, city, and region on country and correlation on if one feature is present that fraudulency might be influenced on if the other three categories are present. Missing values were then dealt with by replacing them with MIS, which meant unspecified.
- Below is the distribution for the top 10 most common countries.



- Lastly, to understand comparisons of countries present a numeric version of the countries was created using dummy variables. Below is the distribution for different countries and their fraud to not fraud ratios.

	country	fraudulent
1	US	730
2	AU	40
3	GB	23
4	MIS	19
5	CA	12
6	MY	12
7	QA	6
8	BH	5
9	IN	4
10	PL	3

- The next attribute that was looked at was state. There are 325 unique values and 2580 missing, which is a larger amount than country. Below is a distribution of counties and their fraud/not fraud count. Pink is representative of fraudulent observations while blue is representative of not fraudulent observations.

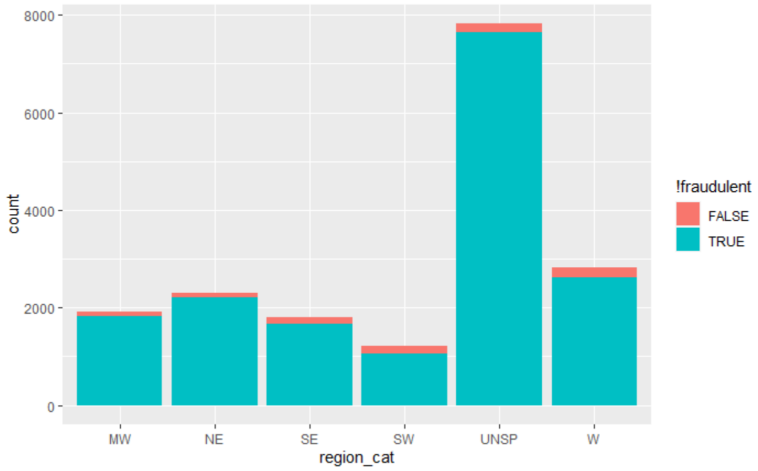


	Var1	Freq
1	CA	2051
2	NY	1259
3	LND	992
4	TX	975
5	I	688
6	IL	424
7	FL	415
8	OH	372
9	VA	332
10	MA	321

- The state with the largest amount of fraudulent reports is CA at 2051. To add, missing values were handled by created another variables called mis.
- The feature following that was city which had 2388 unique values and 2067 missing values. Originally due to the larger amount of missing data this feature was going to be ignored. Overall some analysis was done as can be seen below. The output below is grouped by city and displayed with country and state for city responses with the most fraudulent job postings.

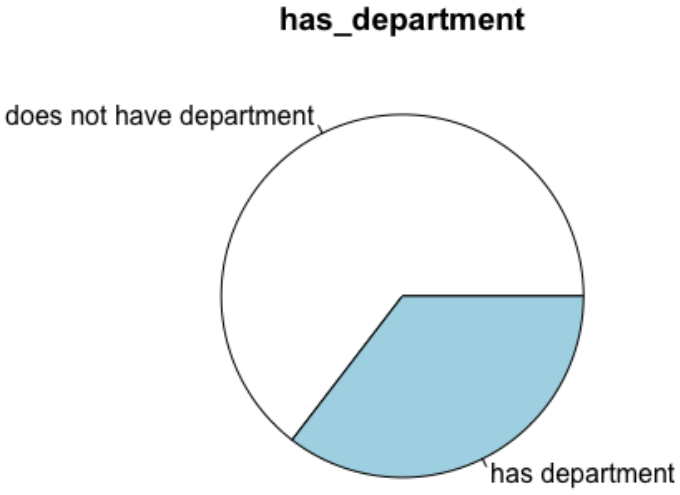
	country	state	city	sum_fraud
1	US	TX	Houston	92
2	US	*NA*	*NA*	33
3	AU	NSW	Sydney	31
4	US	CA	Bakersfield	24
5	US	CA	Los Angeles	23
6	US	CA	San Mateo	22
7	US	NY	New York	20
8	*NA*	*NA*	*NA*	19
9	US	CA	San Jose	14
10	US	TX	AUSTIN	14

- The last attribute that came out of location was region. Region targeted specifically areas of the United States. This was split into 6 regions; MW, NE, SE, SW, W, and UNSP. Anything that was not within the US was just labeled as unsp. or unspecified.

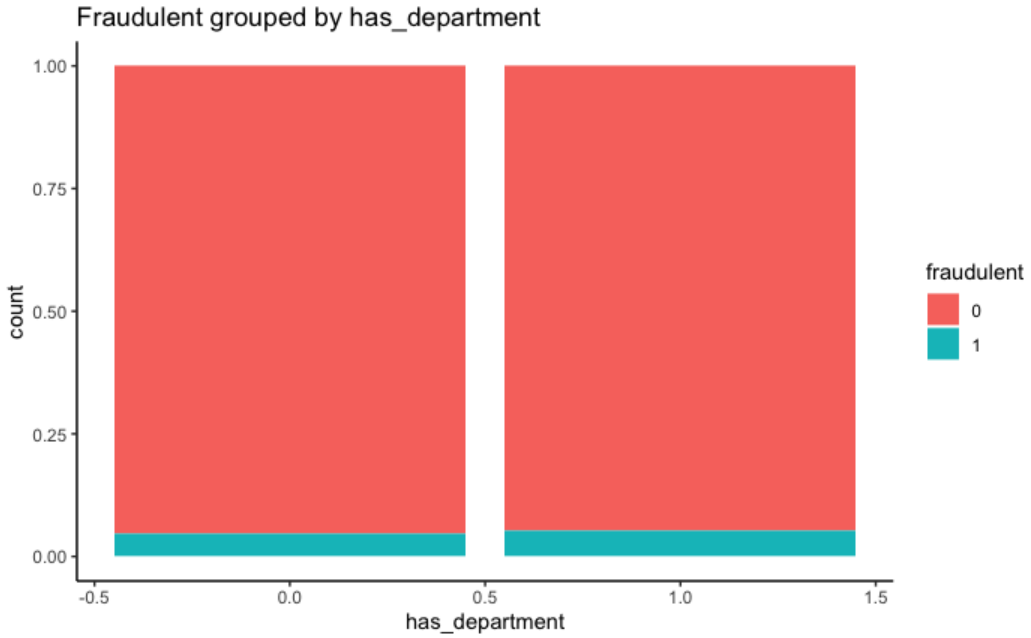
- 

Department

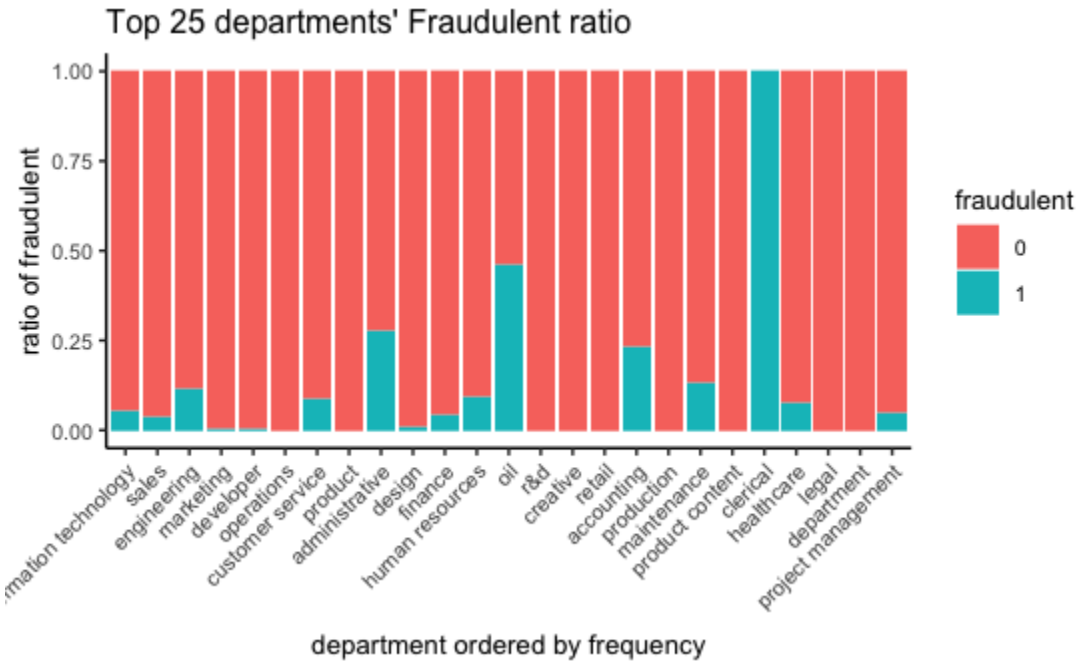
- For the department attribute, there are 11,547 missing values and 1284 unique values. To reduce this number, we grouped similar departments together and cleaned the text in order to group certain keywords into one department. For example, all listings with the word “engineer” will be the department “engineering”. This resulted in 813 distinct values in the department attribute. Since 64.58% of the listings do not have a department included, we create a binary attribute *“has_department”* to indicate whether the listing has a department or not.



- We looked into the ratio of fraudulent to non-fraudulent labels in listings that have and don’t have a department given. Proportion of fraudulent to non-fraudulent is the same whether or not the listing has the department listed or not, 95% of the listings are non-fraudulent and 5% of the listings are fraudulent.



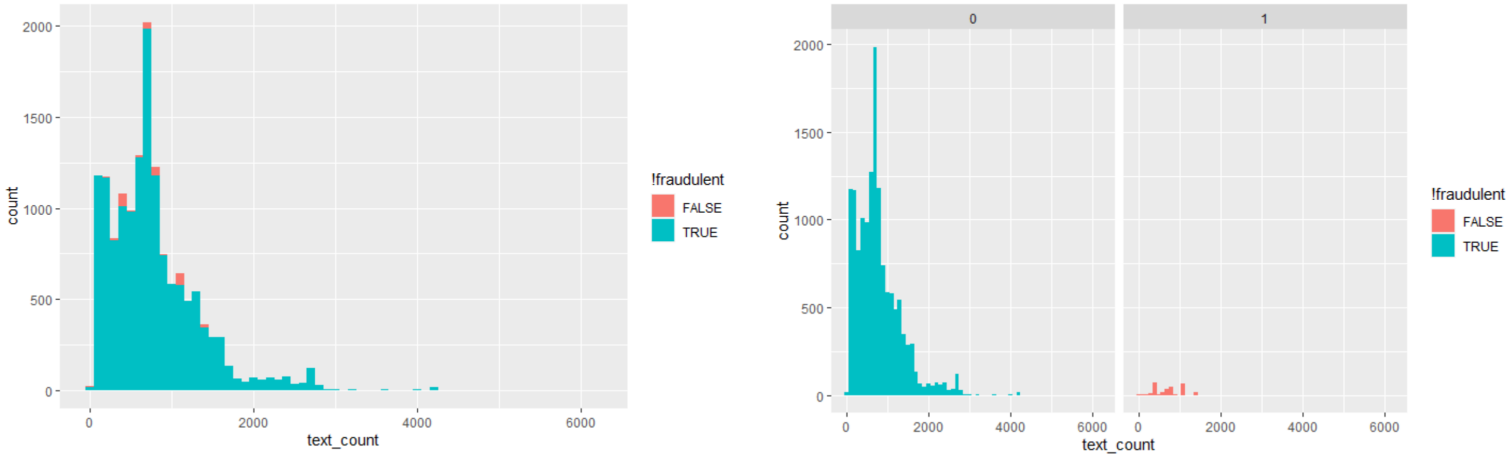
- First, we investigate the top 25 departments that appeared in the dataset ordered by frequency.



-
- While looking at this visualization, keep in mind that the x-axis is the top 25 departments ordered by highest frequency to lowest frequency. This means that the frequency of "project management" listings is lower than the frequency of "information technology" listings.
- The top three departments in this dataset are Information Technology, Sales, and Engineering. Of the three departments, Engineering had the highest ratio of fraudulent to non-fraudulent.
- Among the top 25 departments, the departments that had the highest ratio of fraudulent to non-fraudulent listings are Clerical, Oil, Administrative, and Accounting.

Company Profile

- Company profile is a text attribute which contains information on the companies which posted the job posting. There were 3308 missing values within this column.
- The way that missing values were dealt with since there was such a variety in the text was adding unspecified to the columns where it was missing.
- Displayed below are two plot displaying the amount of fraud and not fraudulent job postings in comparison to the word count. The pink is the fraudulent postings and the blue is the non-fraudulent one.



- Next the text within this column was created into a corpus and cleaned. The text was lowered; stop words, numbers, white spaces, and punctuation were removed and then it was transformed into a document term matrix. Based off of that this word cloud was created which shows the most frequent words within the matrix.

- The top 1000 and top 100 most frequent words were found and analyzed for their associations.
- Finally the column was converted to numerical attributes based on tokenization that occurred during the cleaning process. Some of the different columns that came out of this

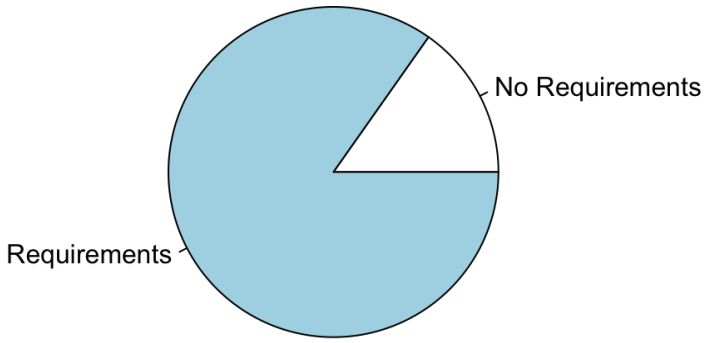
are: hashtags, chars, exclaims, words, uq_words, sent_affin, sent_bing, n_polite, n_third_person, n_first_person, n_prepostion.

Description

- Description is a text attribute, meaning that it doesn't provide any numerical information about the listing for us.
- There were 14,802 unique values and no missing values in this attribute.
- To convert this attribute into a numerical one, we computed the word count of the descriptions instead.

Requirements

- Requirements is a text attribute that is a paragraph talking about the skills, qualifications, and traits that employers expect successful candidates to have. It contains 2178 missing values. With this column we made a dummy variable if it is NA or not. We also created additional features to count elements in the requirements column.



-
-

	No requirements	Requirements
Counts	2178	12126
Fraction	0.152	0.848

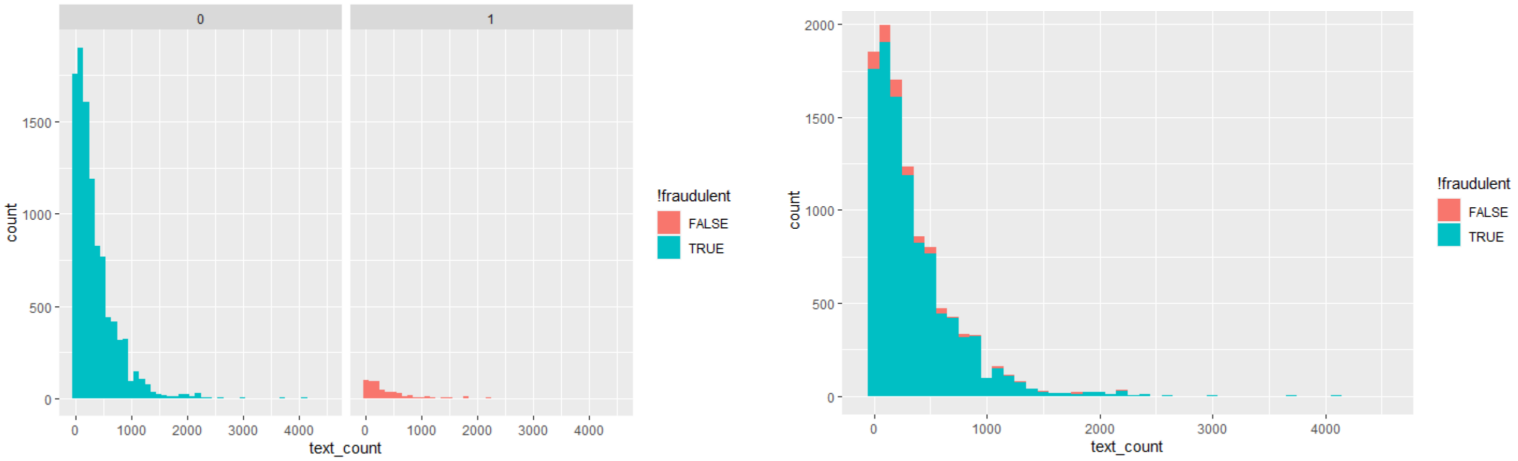
-

has_requirements <chr>	fraudulent <fctr>	n <int>	freq <dbl>
No Requirements	0	2058	0.94490358
No Requirements	1	120	0.05509642
Requirements	0	11546	0.95216889
Requirements	1	580	0.04783111

Benefits

- Benefits is another text attribute which describes types of benefits which could be received from the job. There were 7206 missing values and 6207 unique answers. Any missing values were replaced with unspecified. It is worth to note that within the column prior to cleaning this was a response and it added the amount of this response.

- Based on the text count of the responses in this column another column was created which was text_count. Based on this new column the following histograms were made comparing fraudulent and not fraudulent job postings. The pink represents the fraudulent job postings and the blue represents the non- fraudulent job postings.

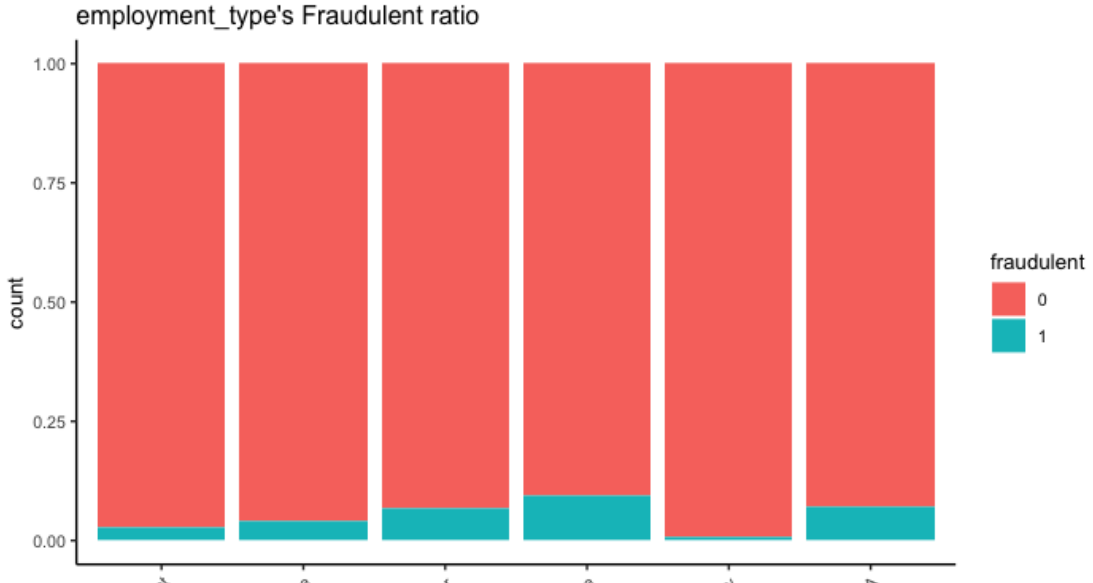


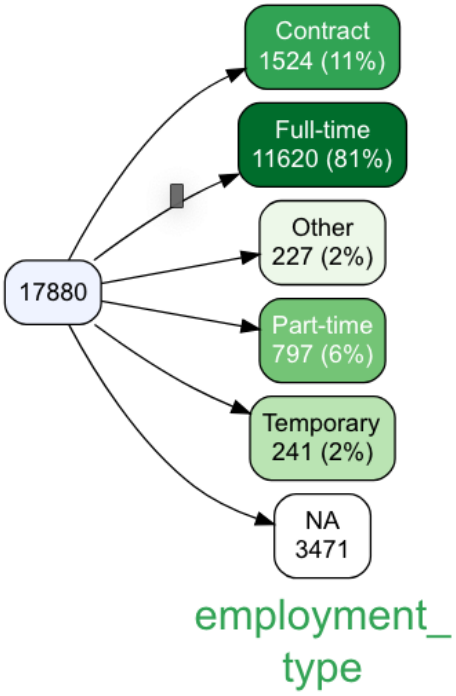
- Next from the text a corpus and document term matrix was created where stopwords, white-spaces, punctuation, and numbers were dropped. This was transformed into numeric features based on the tokenization that occurred from the cleaning process. The columns which were utilized from this are: benefits_n_hastags, benefits_n_caps, benefits_n_charsperword, benefits_sent_affin, benefits_sent_bing.

Categorical Attributes

Employment type

- Employment_type starts as a text attribute, with 6 unique values: contract, full time, part time, temporary, other, and NAs. There were also 3,471 missing values. With these missing values, we decide to make this attribute into 6 dummy variables instead, which we will discuss more in the Feature Engineering section.
- We looked into the ratio of fraudulent to non-fraudulent labels in listings across employment types. Proportion of fraudulent to non-fraudulent listings is relatively small regardless of employment type, but the highest fraudulent to non-fraudulent listings have the employment type of “part-time”. This could be an indication that if a listing is listed as part-time, then the listing has a higher chance of being a fraudulent listing.



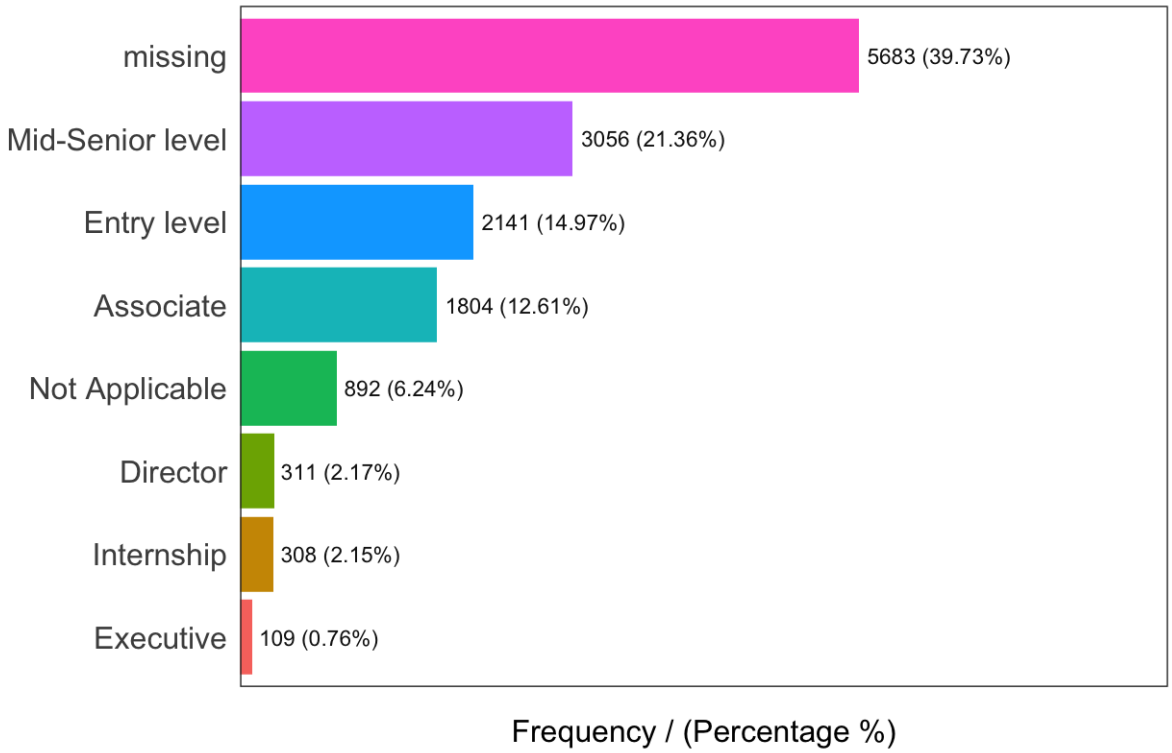


-

Employment_type	Fraction of Dataset
Full-time	65.4%
Contract	8.7%
Part-time	4.2%
Temporary	1.4%
Other	1.2%
NA	19%

Required Experience

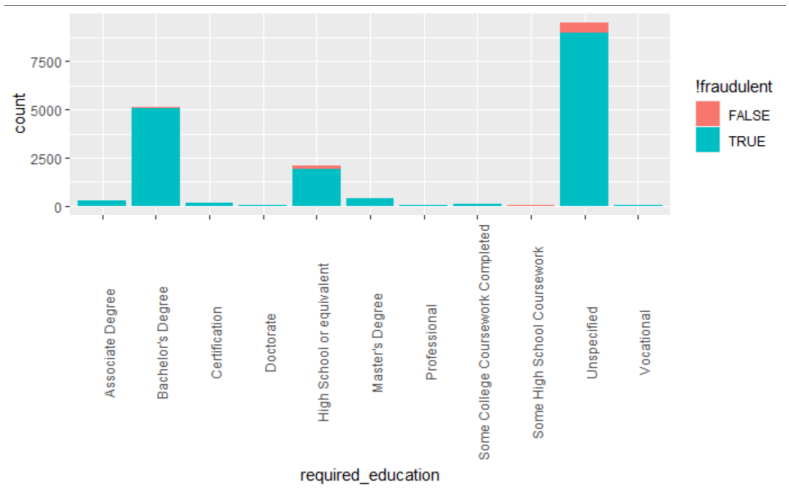
- Contains different categories such as Associate, Entry level, Mid-Senior level, and Director. Set NA's to missing. And made these into dummy variables.



-

Required Education

- Within the required_education feature there were 8105 missing values and 14 unique values. To begin clean up on this column "Vocational - Degree" and "Vocational - HS Diploma" were grouped into "Vocational" and NA's were grouped into an "unspecified" column. The most common required education was bachelor's degree other than unspecified and the category with the most fraudulent job postings was when required_education was unspecified or highschool or equivalent.

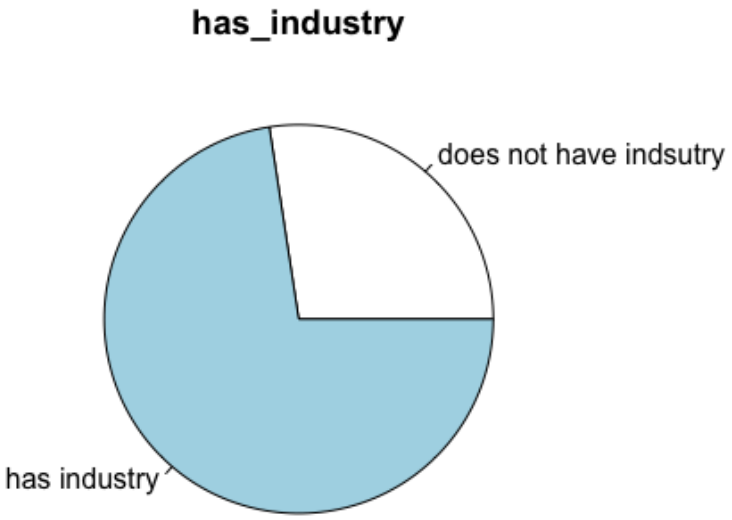


- The plot above shows the count of each of the different categories with fraudulent and non-fraudulent job postings. The pink represents the fraudulent while the blue represents the non-fraudulent.

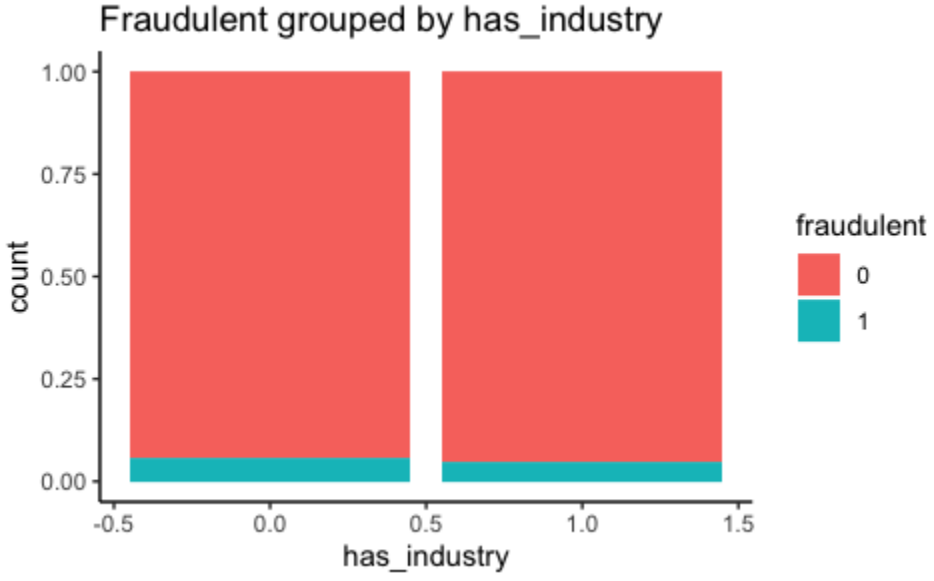
- Then using dummy variables this attribute was turned into a numerical feature to use for future modeling analysis.

Industry

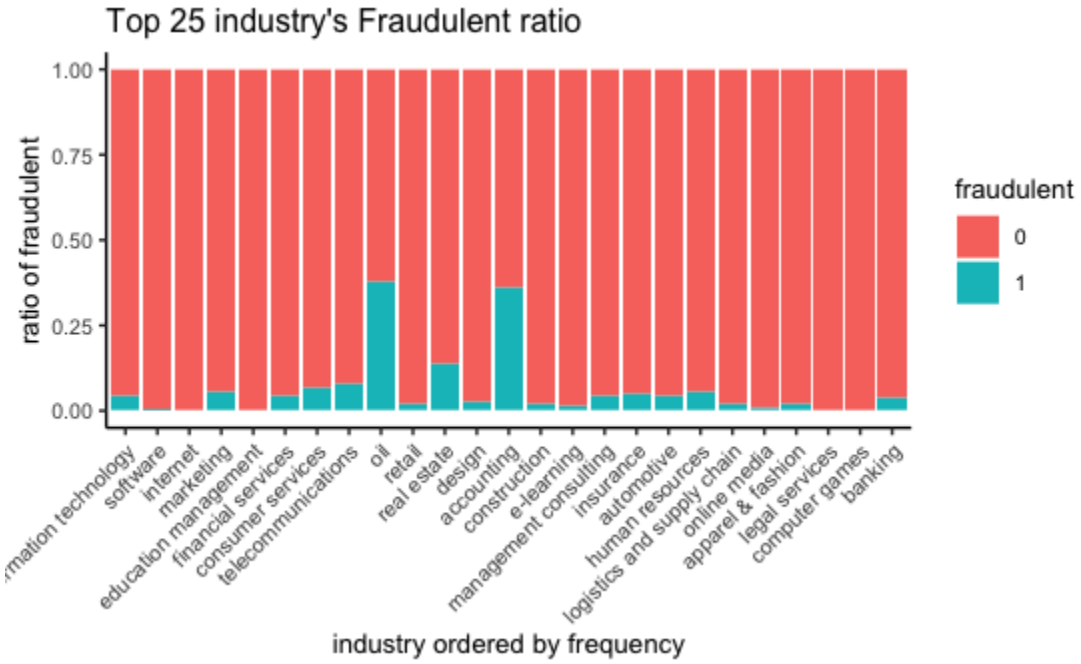
- For the industry attribute, there are 4,903 missing values and 132 unique values.
- We grouped similar industries together and cleaned the text in order to grep certain keywords into one industry. For example, all listings with the word “admin” will be the industry “administration”. This resulted in 104 distinct values in the industry attribute.



- Proportion of fraudulent to non-fraudulent is the similar whether or not the listing has the industry listed or not. Listings with the industry listed are 5% fraudulent and listings without industry listed are 6% fraudulent.



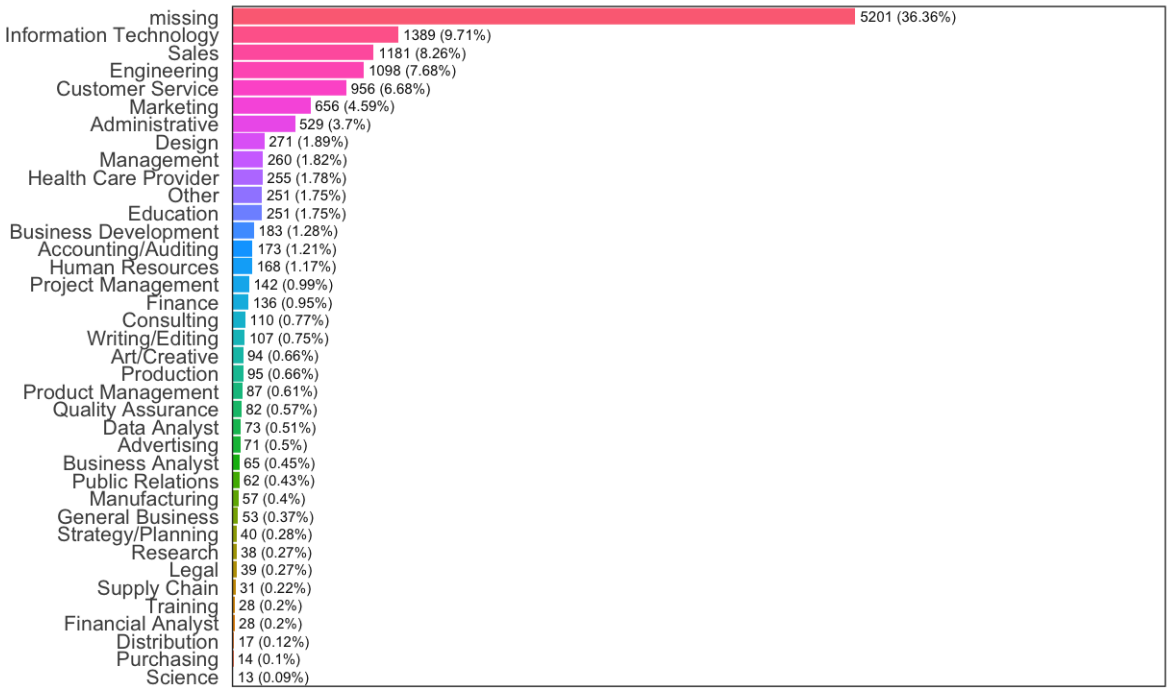
- Next, we look into the top 25 industries that appeared in the dataset ordered by frequency.



- While looking at this visualization, keep in mind that the x-axis is the top 25 industries ordered by highest frequency to lowest frequency. This means that the frequency of "information technology" listings is higher than the frequency of "banking" listings.
- We see that the top three frequent industries that appear in the dataset are Information Technology, Software, and Internet.
- The industries that have the highest ratio between fraudulent and non-fraudulent are Oil, Accounting, and Real Estate. So these industries could be potential indicators of a job listing being fraudulent.

Function

- The job functional area (science, consulting, administrative), etc. There were 38 distinct values.



-

Feature Engineering

Starting off, we deleted the listings' unique job ID since it would not provide us any insight on the actual job posting sentiment and we can just rely on the dataframe's index to access an individual listing.

Then for each factor column, we create a binary variable of whether the value is NA or not. For each text column, we applied the textfeatures library in R and created columns such as number of words, characters per word, number of lowercase characters, etc. from the text columns. This way we can use these numerical columns in the models we create.

For location, we split it into country, state, and city, then we saw the most important features we could derive from this is whether the country is US or not, as well as some important regions, so we made a binary column called YN_usa and then created a categorical variable for regions (midwest, northeast, etc). From this categorical variable, we made dummy variables 1/0 for each region.

For required_education, we made dummy columns for the different distinct values.

For department, we grouped similar departments together, for example if the department name contained support, we grouped it with customer service. We then created a variable yes/no if a department was in the top 25 departments (after text cleaning). We notice that the highest ratio of fraudulent to non-fraudulent listings belong to "administrative", "engineering", and "oil". Following this, we create three binary attributes: "*dep_admin*", "*dep_engineering*", and "*dep_oil*" to indicate if a listing is any of the respective departments. In the end, we dropped the

original department attribute and have 5 new binary attributes: “*has_department*”, “*dep_top*”, “*dep_admin*”, “*dep_engineering*”, and “*dep_oil*”.

We handle the industry attribute similar to how we cleaned the department attribute. To reduce this number, Since 27.42% of the listings do not have an industry included, we create a binary attribute *“has_industry”* to indicate whether the listing has an industry or not. We create a binary attribute to dictate whether a listing’s industry is within the top 25 frequent industries, called “*industry_top”*. On top of that, we notice that the highest ratio of fraudulent to non-fraudulent listings belong to “accounting” and “oil”. Following this, we create two binary attributes: “*industry_acc*” and “*industry_oil*” to indicate if a listing is any of the respective industries. In the end, we dropped the original industry attribute and have 4 new binary attributes: “*has_industry*”, “*industry_top*”, “*industry_acc*” and “*industry_oil*”.

Since description was a very lengthy string column, it was hard to group descriptions without creating a column already present (industry or department). We felt that the textfeatures column as well as the has_na column for description provided sufficient information that would allow this feature to assist in predicting whether a job description is fraudulent or not.

For employement_type, required_experience, and fn we made dummy variables for the unique values as there were a select amount of unique values.

For title, we cleaned the text and grouped similar titles together, for example: grouped all titles that contain "teacher", grouped all titles that contained "intern". Then we created dummy variables for these different categories of title.

For salary range, we parsed the string to split it into min_salary and max_salary. We also made a column for whether the salary range is present or not.

Model Training & Validation

Naive Bayes

A classifier derived from Bayes theorem. It is a very naive design with simplified assumptions. But it works very fast. Naive Bayes classifiers assume that the predictor variables are independent of each other. Naive Bayes can be used for multiclass classification problems as well.

	Observed Label (Fraudulent)	
Prediction Label (Fraudulent)	0	1
0	715	8
1	2695	158

Confusion Matrix	Percentage Rate
Accuracy	24.41%
Sensitivity (Recall)	99%
Specificity	5.5%

The performance is not great compared to other classification methods, as we can see with the results from this model compared to the rest of our models. We have a 24.4% accuracy rate. The model predicted many more fraudulent job postings (2853, 80%) than non fraudulent (723, 20%). There are few false negative results, as seen by our high sensitivity rate, but many false positives, as seen by our low specificity rate.

Logistic Regression

Logistic regression is a modeling technique used to fit a regression using $y = f(x)$ when y is a binary or categorical variable. This classification technique can be used for binary as well as multinomial classification. It is known for its easy implementation, interpretation, no assumptions about distributions of classes, and efficiency in training but can have drawbacks of possibly overfitting in certain instances and needing close to no multicollinearity.

	Observed Label	
Prediction Label	0	1

0	13518	476
1	86	224

Confusion Matrix	Percentage Rate
Accuracy	96.07%
Sensitivity (Recall)	99.37%
Specificity	32%

Test Results

0	1
3508	68

From these results one can observe that the accuracy rate is quite high for classification of the predictor. The results for sensitivity and specificity above are at a .5 threshold, we can see the sensitivity(true positives) is very high meaning our model has good prediction accuracy of true positives. The specificity is lower which means the correct classifications of negatives. If we lowered the threshold the specificity may go up, but the sensitivity could go down letting more fake job postings through. This is the trade-off for this type of model.

Additionally during the modeling process of the logistic regression, the issue of multicollinearity came up. There are a couple of columns which have identical if not near identical collinearity which then skewed the original logistic models that were performed. With further EDA by using a correlation matrix this was to be overcome.

Support Vector Machine

SVM algorithms are supervised learning algorithms that can be used for classification and regression. We will use it for classification. Both linear and non-linear classifiers can be done with SVM. For simplicity, we used a linear approach. Points are plotted in the space and the optimal hyperplane that differentiates fraudulent vs non-fraudulent job postings will be found. SVM differs from logistic regression in that it is less vulnerable to overfitting. This is because Logistic regression is able to have multiple decision boundaries with varying weights that are close to the optimal point, whereas SVM aims to identify the "best" margin that divides the classes. Logistic Regression is a more flexible model compared to SVM when the decision boundary is linear which makes it more prone to overfitting.

	Observed Label (Fraudulent)	
Prediction Label (Fraudulent)	0	1

0	3389	110
1	21	56

Confusion Matrix	Percentage Rate
Accuracy	96.34%
Sensitivity (Recall)	96.86%
Specificity	72.73%

However, we can observe that the difference in accuracy between our logistic regression and SVM model is not too large, with it being only 0.13% difference. This model shows great improvement compared to logistic regression though.

Classification Tree (CART)

Classification and regression trees are predictive models that try to explain how the response variable can be predicted based on the values of the predictor variables. The output of this algorithm is a decision tree with splits in predictor variables and at the end of each split, there are predictions on the response variable. The end goal is to partition the feature space so that each partition will indicate the decisions made to predict a certain value of the response variable.

For the Fraudulent Job Listings dataset, with many variables being binary, the goal of implementing this model is to see if the classification on a job listing's fraudulent status is better using a decision tree versus the linear algorithms we tried.

After running our classification tree model on our training dataset with the rpart() library, we get an accuracy of 96.03%.

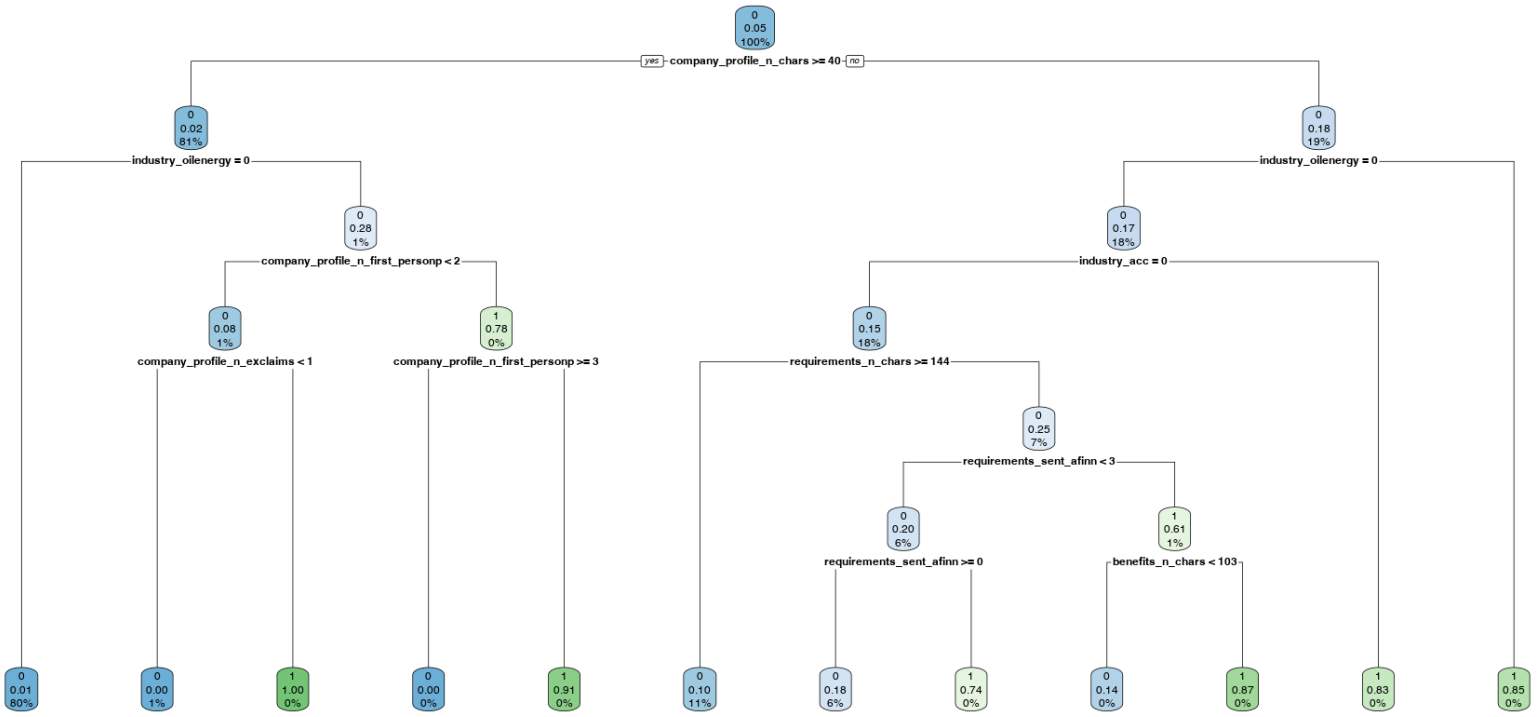
Looking at the confusion matrix generated, we see that there are 3,395 predictions that were true positives and 127 false positives. Meanwhile, there are 15 predictions that are false negatives and 39 predictions of true negative. In this case, we would rather have more false positives than false negatives as in the real world setting, there is more importance on over-predicting that a listing is fraudulent rather than under-predicting (DataTechNotes). This way, an applicant's vital information would not be given to a scammer or waste their time in applying. This means we want to optimize the true positive rate, which is sensitivity.

In the classification tree model, we got a sensitivity rate of 99.56%.

	Observed Label (Fraudulent)	
Prediction Label (Fraudulent)	0	1
0	3395	127
1	15	39

Confusion Matrix	Percentage Rate
Accuracy	96.03%
Sensitivity (Recall)	99.56%
Specificity	23.49%

Looking at the classification tree generated from our model, we see that the *company_profile_n_chars* attribute is the first split attribute and it's split between company profiles that have 40+ and <40 characters. The next split uses *industry_oilenergy* to determine whether the listing's industry is oil energy. Another split of importance is using the *requirements_n_chars* attribute and it's split between the listing's requirements having 144+ and <144 characters.



Random Forest

Random Forest is a supervised Machine Learning algorithm. It uses ensemble learning which is a technique that combines predictions from multiple machine learning models in order to make a more accurate prediction rather than a single model (Chaya). Random forest models can be used even when attributes have nonlinear relationships. Another advantage of random forest is that this algorithm has no formal distributional assumptions.

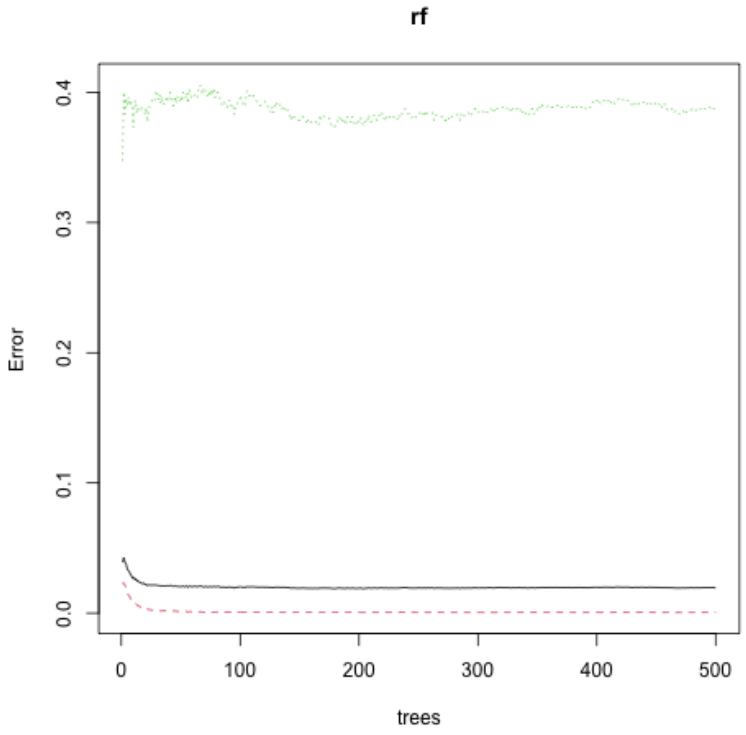
For our Fraudulent Job Listings data, we want to use Random Forest since it is one of the most accurate and efficient algorithms for regression and given the nature of our dataset, we cannot solely rely on linear regression.

After running our random forest model on our training dataset, we get an accuracy of 97.99%. As for sensitivity, we have 100%, meaning that in this random forest model, it accurately predicted 100% of the positive classes. It was able to classify all listings that are fraudulent as fraudulent.

	Observed Label (Fraudulent)	
Prediction Label (Fraudulent)	0	1
0	3410	72
1	0	94

Confusion Matrix	Percentage Rate
Accuracy	97.88%
Sensitivity (Recall)	100%
Specificity	56.63%

Looking at the error plot of the random forest model, it shows that the model predicted with comparably higher accuracy, which means there is no need for further tuning (Finnstats).



Stochastic Gradient Boosting

The purpose of ensemble methods and random forest was to allow decision trees to be created using the greedy algorithm on the subsamples from the training dataset. Gradient boosting models serve the same purpose and also help reduce the correlation between the trees created in the sequence (Kassambara.).

Stochastic gradient boosting is different, where at each iteration, a subsample of the full training dataset is randomly selected without replacement. That randomly selected subsample is then used to fit the base learner instead of the full training dataset. The benefit of this method is that subsampling by column helps reduce overfitting the model when compared to the original subsampling of rows (Dwivedi).

With gradient boosting, we can use hyper-parameter tuning as it will help the model from underfitting and overfitting.

With our Fraudulent Job Listings dataset, there are a lot of predictor variables that are binary data types so with stochastic gradient boosting, we hope to improve the performance of classification tree models and random forest models.

After running our random forest model on our training dataset, we get an accuracy of 97.82%. As for sensitivity, we have 99.59%, meaning that in this stochastic gradient boosting model, it accurately predicted 99.59% of the positive classes.

	Observed Label (Fraudulent)	
Prediction Label (Fraudulent)	0	1
0	3396	50
1	14	116

Confusion Matrix	Percentage Rate
Accuracy	98.21%
Sensitivity (Recall)	99.59%
Specificity	69.88%

Looking at the table of variables of importance, we see that the top attribute is *company_profile_n_chars*, which means that the number of characters in the company's profile is a huge indicator of whether a listing is fraudulent or not. Other attributes of importance are the word count of the listing's job description, *cleandescription_length*, and whether or not the listing has a company profile tied to it, *has_company_profile*.

Referring back to the random first model earlier, most of the deciding attributes for the splits are the same attributes that got the top importance ratings in this stochastic gradient

boosting model. A very important and recurring attribute is whether a job listing is in the oil energy industry or not, *industry_oilenergy*.

Variables	Importance Rate
company_profile_n_chars	100.00000
cleandescription_length	86.80829
description_n_charsperword	70.96889
has_company_profile	66.80317
industry_oilenergy	65.27648
description_n_polite	62.99229
has_company_logo	62.81475
requirements_n_chars	53.82249
company_profile_sent_syuzhet	52.87599
requirements_n_caps	43.43704

KNN

K-nearest neighbors is a supervised algorithm that can be used for both classification and regression. It computes the distance between an observation and the K closest points in the train data and uses the mode of its response variable as the predicted response. We used K=3 for this model as that is the parameter that gave us the best result. The most common distance used in KNN is Euclidean distance. We get an accuracy of 97.68%, but there is still room for improvement in specificity. Out of the models we created, this one had the highest specificity though.

	Observed Label (Fraudulent)	
Prediction Label (Fraudulent)	0	1
0	3375	48
1	35	118

Confusion Matrix	Percentage Rate
Accuracy	97.68%

Sensitivity (Recall)	98.6%
Specificity	77.12%

Conclusion

	Accuracy (%)	Sensitivity (Recall) (%)	Specificity (%)
Naive Bayes	24.41	99	5.5
Logistic Regression	96.21	99.37	32
Support Vector Machine	96.34	96.86	72.73
CART (Classification Tree)	96.03	99.56	23.49
Random Forest	97.88	100	56.63
KNN	97.68	98.6	77.12
Stochastic Gradient Boosting	98.21	99.59	69.88

This table summarizes the performance of each of our models.

In terms of our research issue, accuracy is the overall rate of correctly classifying a fraudulent job listing as fraudulent, also known as a positive case. Specificity is the overall rate of correctly classifying a non-fraudulent job listing as non-fraudulent, also known as a negative case. Sensitivity is the ratio of predicting a positive classification, which in this context means the number of job listings we classify as fraudulent divided by all actual fraudulent job listings.

For impact, sensitivity is more important than specificity as we would rather our models to over-classify fraudulent cases than to under-classify. This way, we would not have fraudulent job listings classified as non-fraudulent. In a business case, this would mean saving an applicant time from applying to any possible fraudulent job listings and prevent a loss of personal information to scammers behind the fraudulent listings.

The best performing model was Stochastic Gradient Boosting. The model that performed the worst was Naive Bayes, because of its oversimplified assumptions. Regarding sensitivity, the

model that had the highest was Random Forest, meaning that all fraudulent job postings were identified as fraudulent. All our models had relatively high sensitivity, >=96.8%. Our models struggled with Specificity, the ability to predict a job posting that is not fraudulent as not fraudulent, and that is a room for improvement in our project. The model that had the highest was KNN with a value of 77.12%, but we see a range of specificity values from 5.5% to 77.12% in our models.

In terms of future work, we believe that we could further increase our accuracy by further mining into our text columns. Columns such as requirements and job description could provide lots of information to help determine if a job posting is fraudulent. Although we did get many numerical features from these columns such as the number of first-person words, the number of words, etc. We could look into sentiment analysis features to determine the tone of the job posting.

Another idea would be to incorporate hyper-parameter tuning. With our dataset being very unbalanced, we could introduce weights to the attributes that hold higher importance and try for a slower learning model, where the up and down weights for each iteration of the training process are smaller so that we can use more iterations and train the model longer with the same total training data.

Data Sources

- Links, downloads, access information.

The dataset we are using for this project comes from Kaggle. It's a dataset named "Real or Fake Job Posting Prediction" (Bansal). It contains 18,000+ job postings, where 800+ are fraudulent. There were 18 attributes that came with it, most belonging to the text attribute category.

Link to dataset (also cited in Bibliography):
https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction?resource=download

Within the added zip files one can find four different datasets created from the overall dataset above. The four are, train_numeric.csv, train.csv, joint_test.csv, and joint_test_numeric.csv. This is a split training and test dataset, the one's labeled numeric just including the numeric features while the other include others as well. For most of the analysis and modeling above the numeric test and train dataset were utilized. The split for the train and test dataset is 80/20.

Source Code

- Listings, documentation, dependencies (open-source).

In this submission folder, we have a README file that contains information about the different files and code.

Bibliography

Chicago Style

"A Step by Step Guide to Implement Naive Bayes in R." Edureka, May 26, 2020. https://www.edureka.co/blog/naive-bayes-in-r/.

Bansal, Shivam. "Real / Fake Job Posting Prediction." Kaggle, February 29, 2020. https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction?resource=download.

Bassey, Patricia. "Logistic Regression vs Support Vector Machines (SVM)." Medium. Axum Labs, September 19, 2019. https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16.

Chaya. "Random Forest Regression." Medium. Level Up Coding, April 14, 2022. https://levelup.gitconnected.com/random-forest-regression-209c0f354c84.

DataTechNotes. "Precision, Recall, Specificity, Prevalence, Kappa, F1-Score Check with R." Precision, Recall, Specificity, Prevalence, Kappa, F1-score check with R, February 20, 2019. https://www.datatechnotes.com/2019/02/accuracy-metrics-in-classification.html.

Dwivedi, Rohit. "XGBoost Algorithm for Classification and Regression in Machine Learning: Analytics Steps." XGBoost Algorithm for Classification and Regression in Machine Learning | Analytics Steps. Accessed December 4, 2022. https://www.analyticssteps.com/blogs/introduction-xgboost-algorithm-classification-and-regression.

Finnstats. “Random Forest in R: R-Bloggers.” R-bloggers, April 13, 2021. https://www.r-bloggers.com/2021/04/random-forest-in-r/.

Kassambara. “Gradient Boosting Essentials in R Using XGBOOST.” STHDA, March 10, 2018. http://www.sthda.com/english/articles/35-statistical-machine-learning-essentials/139-gradient-boosting-essentials-in-r-using-xgboost/#loading-required-r-packages.

“K-NN Classifier in R Programming.” GeeksforGeeks. GeeksforGeeks, June 22, 2020. https://www.geeksforgeeks.org/k-nn-classifier-in-r-programming/.

“Naïve Bayes Classifier.” Naïve Bayes Classifier · UC Business Analytics R Programming Guide. UC Business Analytics. Accessed December 4, 2022. https://uc-r.github.io/naive_bayes.

Popper, Nathaniel. “A Job That Isn't Hard to Get in a Pandemic: Swindlers' Unwitting Helper.” The New York Times. The New York Times, September 15, 2020. https://www.nytimes.com/2020/09/15/technology/money-mules-fraud-pandemic.html

Ravenelle, Alexandrea J, Erica Janko, and Ken Cai Kowalski. “Good Jobs, Scam Jobs: Detecting, Normalizing, and Internalizing Online Job Scams during the COVID-19 Pandemic.” *New Media & Society* 24, no. 7 (2022): 1591–1610. https://doi.org/10.1177/14614448221099223

“What Is the K-Nearest Neighbors Algorithm?” IBM. IBM. Accessed December 4, 2022. https://www.ibm.com/topics/knn#anchor-480417839.