Job Posting: Fake or Real

Julianna Cybrynski
Tiara Mathur
Jeffrey Mei

Problem of Interest

- Online job searching is the most popular medium in modern times
- Fake job postings are common
  - Waste of time
  - Ability to gain access to personal information
- Companies post fake jobs allowing for nepotism and other unfair hiring practices to occur

Dataset and Initial Cleaning

- 18,000 rows and 18 columns
  - Some redundant columns were removed
    - Department, Function, Industry equated

- Location - Filtered out values outside of the US
- Salary Range - Created a Mean Salary column

Textual Data Cleaning

- Getting rid of stop words such as “the”, “and”, etc.
- Converting conjunctions
- Removing punctuation
- Converting non-ASCII to ASCII
- Word stemming and word leminization- reducing words to their simplest form

Textual Data Cleaning: Numeric Conversions

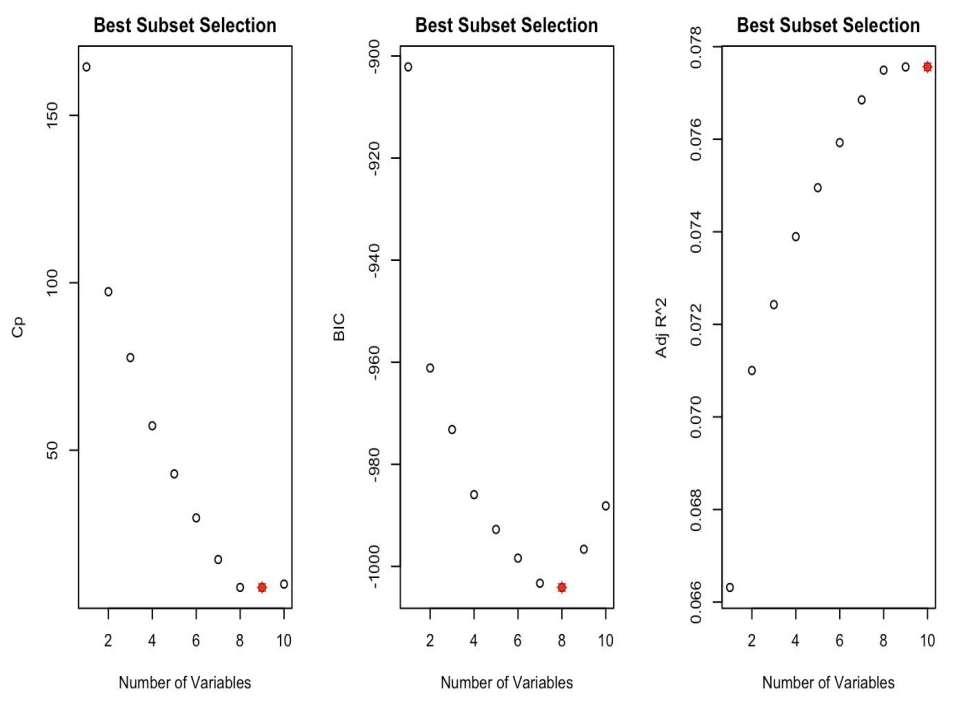
- Giving each body of text an emotional score using sentiment analysis
- Finding the length of each text

Numerical and Categorical Analysis

Sentiment Analysis

- Four main predictors for textual data
  - Company Profile, Description, Requirements, Benefits
- Majority of textual data is wordy
  - Converted into numerical data through sentiment analysis and counting word lengths - Python: TextBlob
- The sentiment score calculated by analyzing each specific word and determining if the connotation was negative or positive
  - Other more complex measures - grammar and sentence structure were also taken into account.
- A sentiment score of above 0.05 -> overall positive piece
- A sentiment score of 0.05 to 0 -> neutral
- Anything below 0 -> negative.
- To count word lengths ->
  - Removed all the stop words: words without value
    - Done by filtering the text through a list of stop words generated by Python library NLTK
  - Then lengths were calculated by counting the words in a body of text.

Best Subset Selection, Forward Stepwise Selection, Backward Selection



Created a least squares linear regression model -> test error of 0.04981329

Implemented Best Subset Selection in order to use most commonly selected features -> test error of 0.04982897

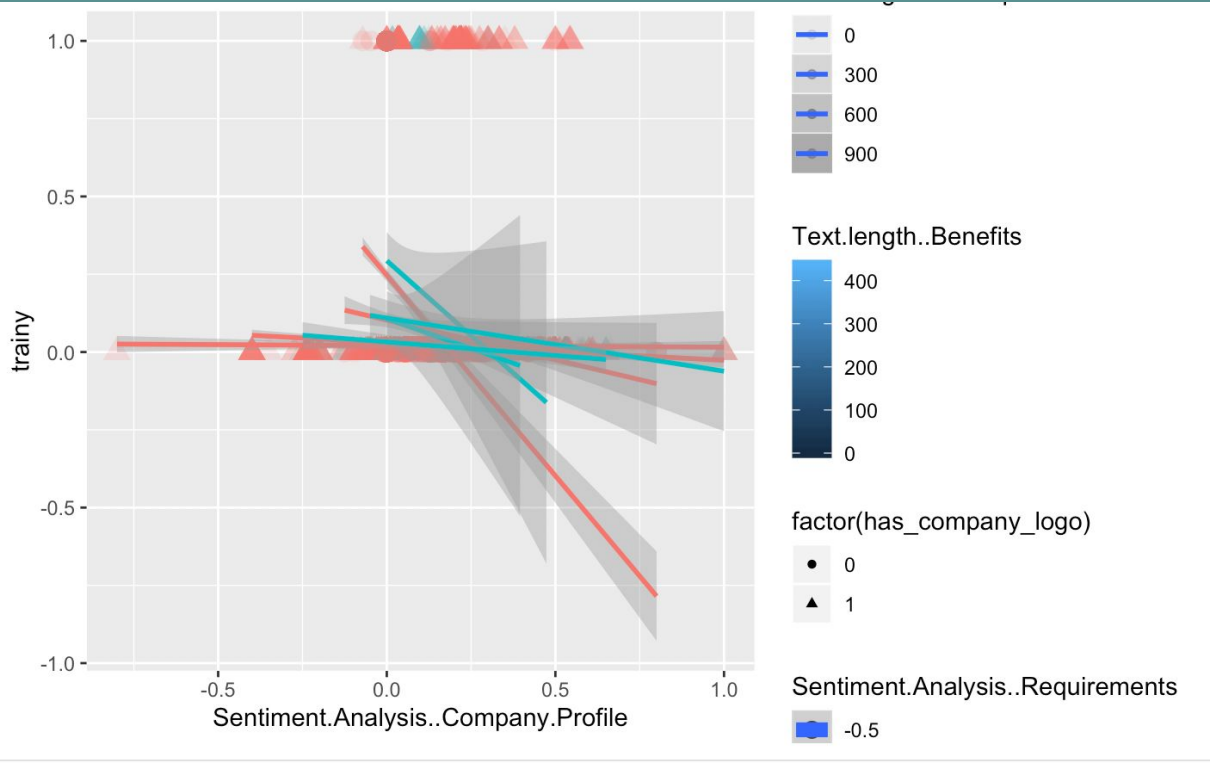
Partial least squares regression - best number of components: 3 -> test error of 0.04981387

Higher error estimates

Issues with using linear regression to solve a classification problem

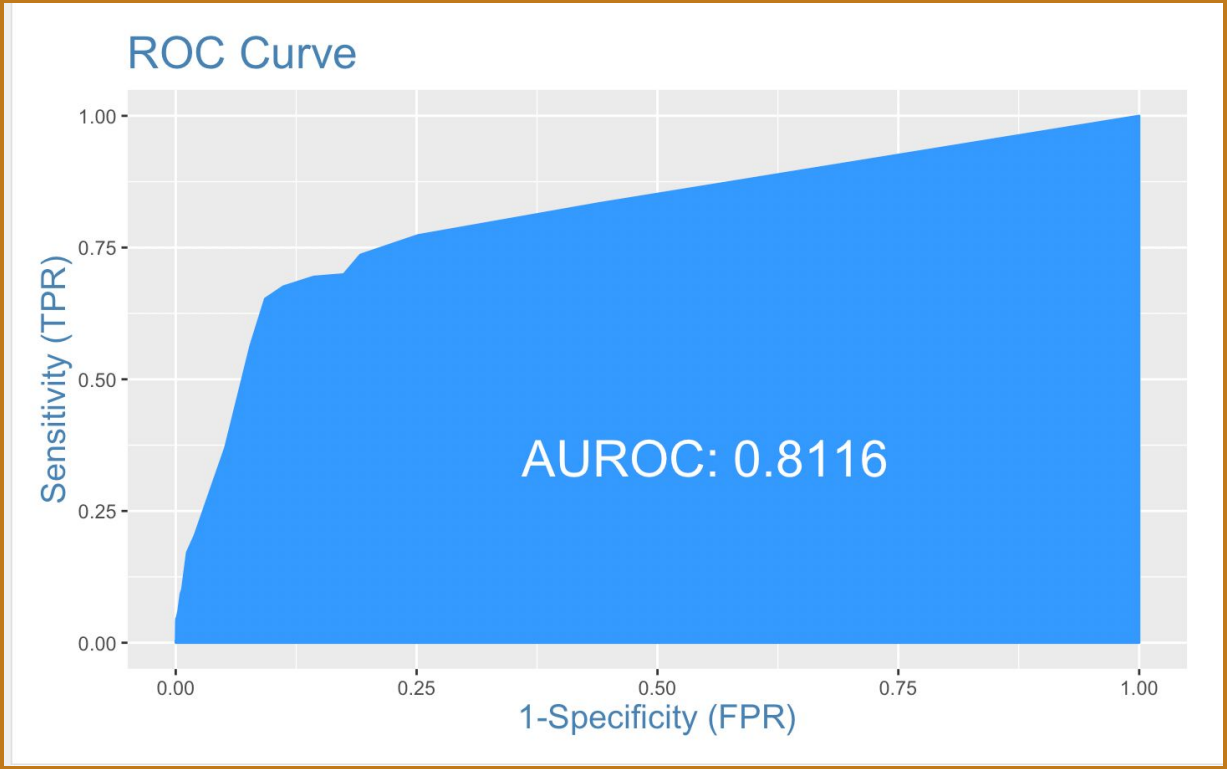
1. Predicts continuous values instead of a number 0: real or 1: fraudulent
2. Sensitive to imbalanced data, so it is inclined to give a worse prediction than a classification model
3. Choosing the best subset of features caused 9 variables to be selected - a quite large number, since only 2 were excluded

Best Subset Selection



Created from the best subset of variables selected, we can see that linear regression creates straight lines that do not visually fit the shape of the data, which is limited to values on the y-axis of 0 and 1, and the shadow of the lines shows standard error, which is quite large.

Logistic Regression



Error 4.65%
False positive rate of 0.047%
False negative rate of 95.37%

Logistic regression is a better model, since it is more well suited for classification

Model's high false negative rate is concerning

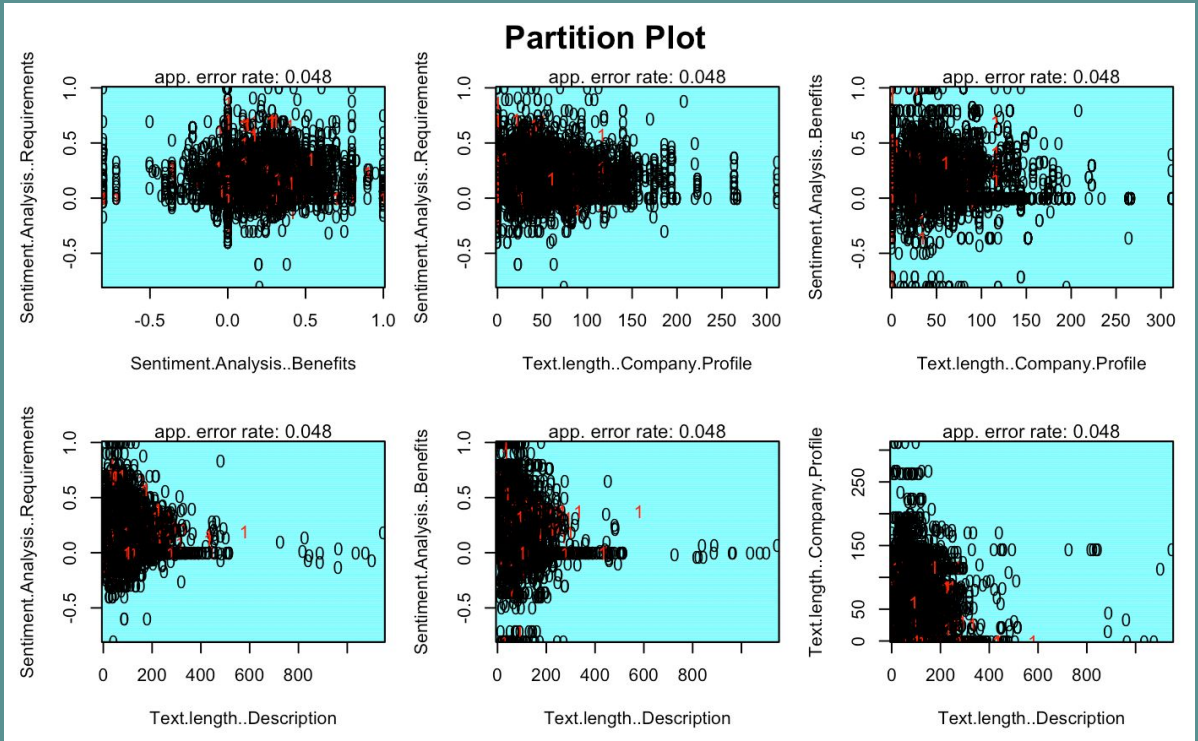
This could cause someone seeking employment serious harm by applying to an alarming number of fake jobs that they believed to be real.

	Actually 0 (legitimate)	Actually 1 (fraudulent)
Classified as 0 (“real”)	4252	206
Classified as 1 (“fake)	2	10

LDA

For linear discriminant analysis, the test error is 4.7%, which is higher than other models.

False positive rate is 0.07%, which is quite low, but false negative rate is 96.3% which is very high.



QDA

Quadratic discriminant analysis higher test error of 8.77%

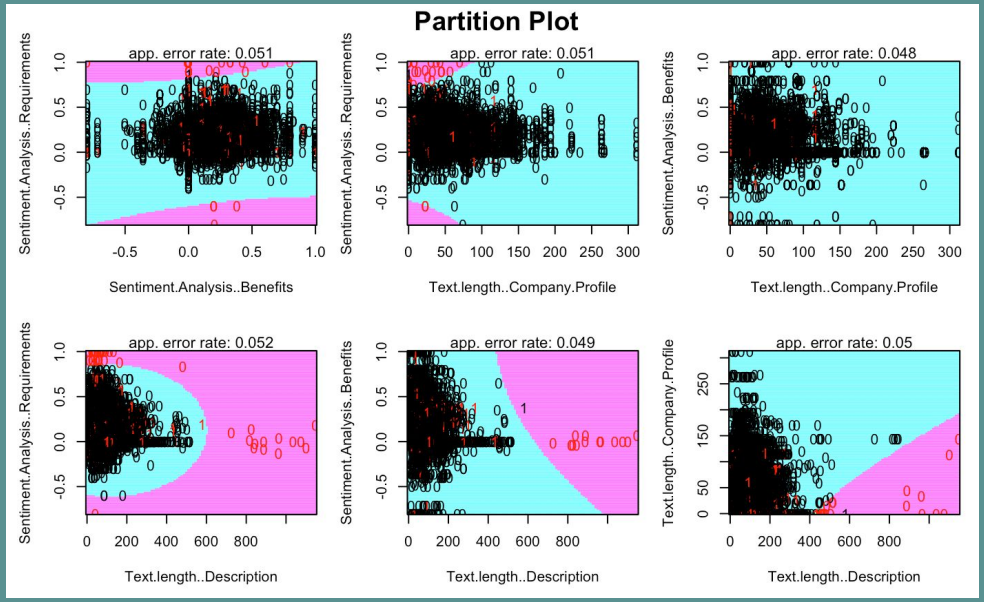
False positive rate is also quite higher at 6.5%

False negative rate is quite lower at 54.16%

False negatives should be considered more important than false positives in our case.

Skipping an application because 0 (real) was classified as 1 (fake) would not cause harm

Giving personal information to even one illegitimate company because 1 (fake) was classified as 0 (real) can cause harm to the applicant and there is a high chance of fraud and identity theft.



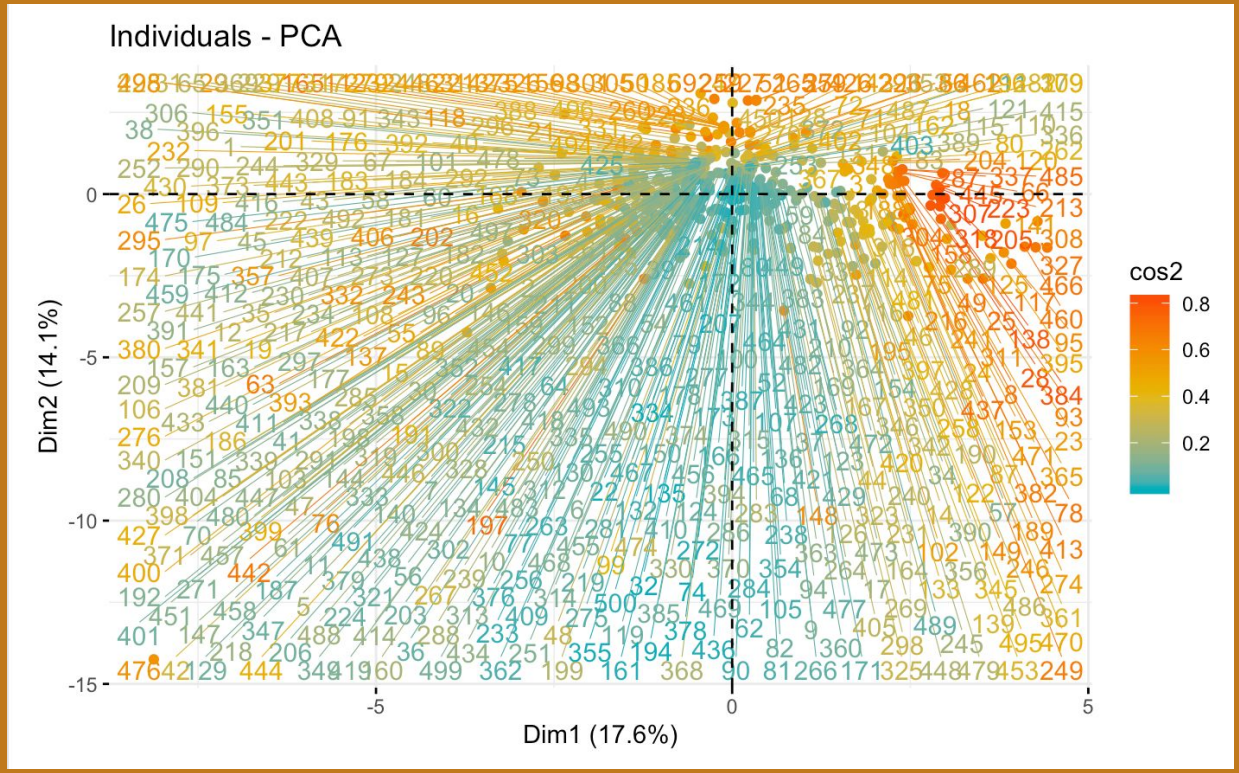
Plots show the partition from LDA and QDA on a subset of the features in the dataset.

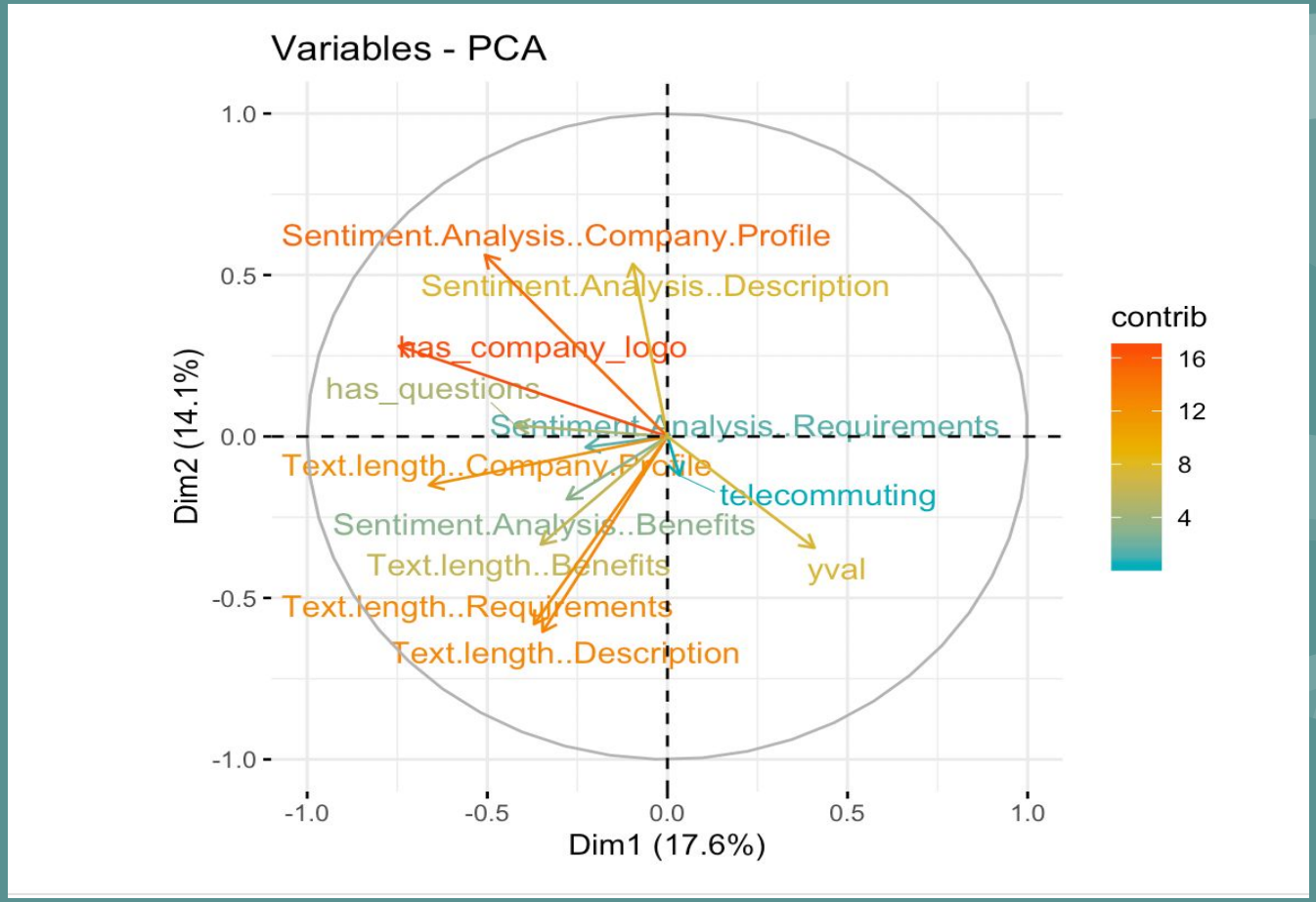
QDA partitions are more evident and clearly separate a group of non-fraudulent job points.

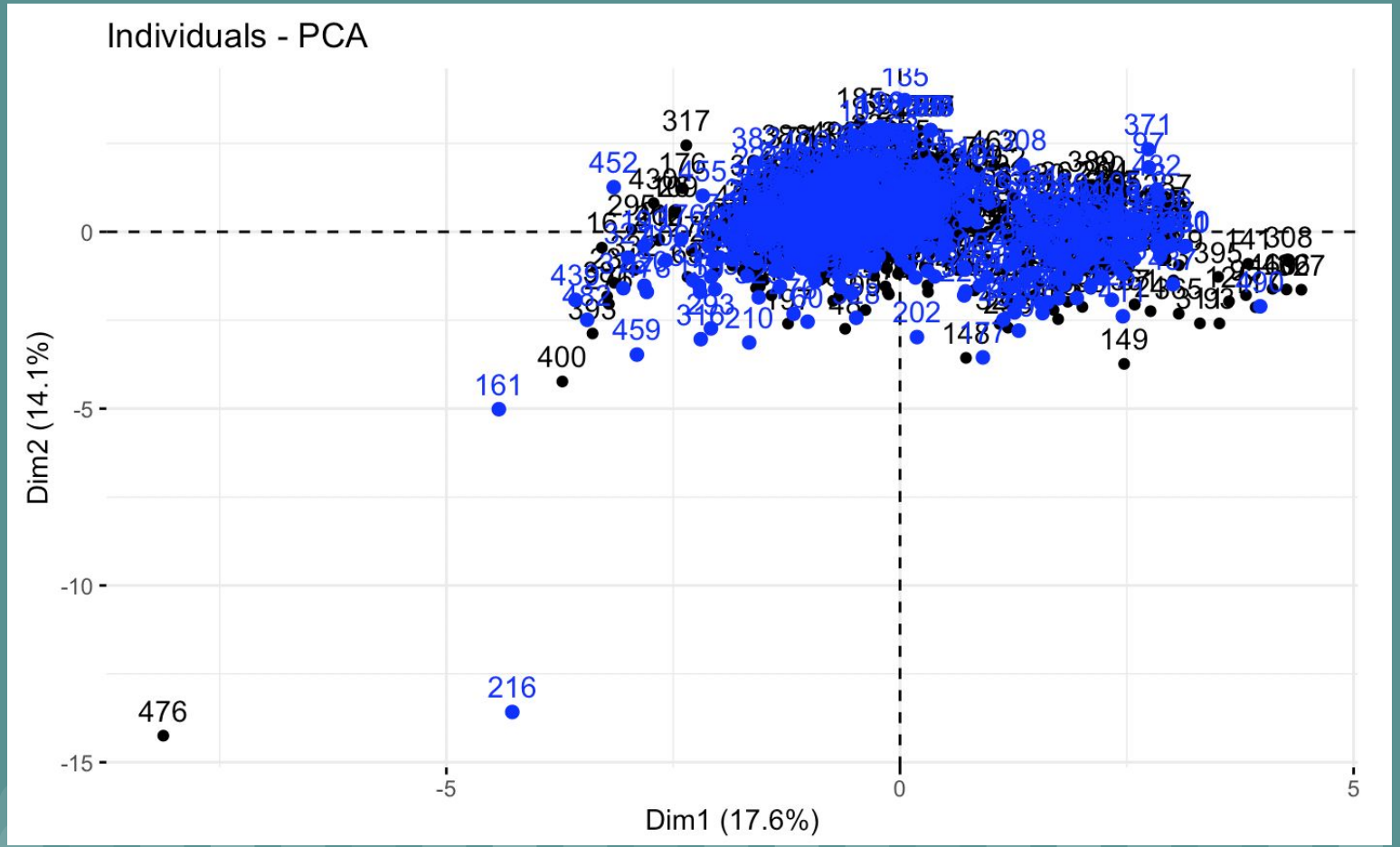
SVMs With Radial Basis Kernels

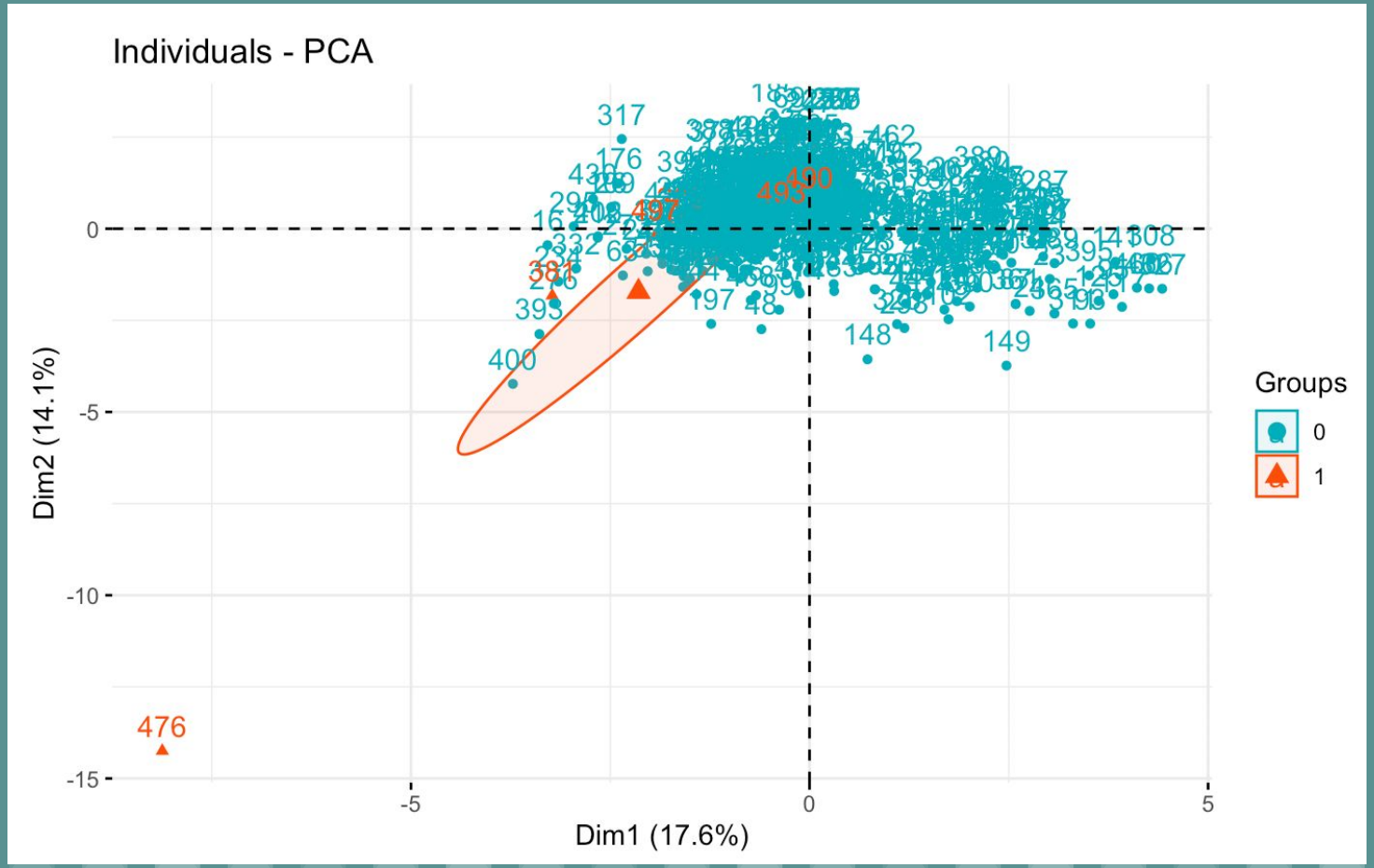
- Support vector machines using a linear and radial basis kernel with optimal values of degree, gamma, and cost selected from {0.001, 0.1, 1, 10, 100}
  - Error of 5% and 5.5% respectively.
- However, these models were very computationally intense
  - Required a smaller training set to avoid running out of memory
    - Limits the amount of information the model has to learn from
    - Limits how accurate these models have the capability to be.

PCA




Variables - PCA
Sentiment.Analysis..Company.Profile
Sentiment.Analysis..Description
has_company_logo
has_questions
Sentiment.Analysis..Requirements
Text.length..Company.Profile
telecommuting
Sentiment.Analysis..Benefits
Text.length..Benefits
yval
Text.length..Requirements
Text.length..Description
contrib
16
12
8
4
1.0
0.5
0.0
-0.5
-1.0
Dim2 (14.1%)
Dim1 (17.6%)


Individuals - PCA
Dim2 (14.1%)
Dim1 (17.6%)
0
-5
-10
-15
-5
0
5
317
452
176
371
308
395
459
400
161
202
149
210
216
476


Individuals - PCA
Dim2 (14.1%)
Dim1 (17.6%)
0
-5
-10
-15
-5
0
5
317
176
490
497
381
393
400
476
197
48
148
149
Groups
0
1

Principal Component Analysis does a decent job in certain areas separating the fake jobs from real jobs

- **Too much overlap in the predicted groups to be effective**

Textual Analysis

Naive Bayes

- Classifies a job posting by looking at its features individually
  - Features are the four main columns of textual data (Company Profile, Description, Requirements, Benefits).
  -
- The posterior probability is calculated for each feature and the posterior probabilities are all multiplied together to get the final probability
- The greatest probability determines the class

Implementation

- Training and test sets created
  - Over 8,000 rows used
  - X-Value: count vectorizer array
- Y-Value: 0 or 1
- Multinomial Naive Bayes used
  - Focus on counting the occurence of each feature
  - Uses prior fit
  - Alpha = 1.0 -> There is a smoothing parameter

Naive Bayes Results

- Company Profile - Successful
  - 99.16% accuracy rate
- The confusion matrix generated ->

	Actually 0 (legitimate)	Actually 1 (fraudulent)
Classified as 0 (“real”)	3384	27
Classified as 1 (“fake)	3	162

- Description - Successful
  - 98.35% accuracy rate
- The confusion matrix generated ->

	Actually 0 (legitimate)	Actually 1 (fraudulent)
Classified as 0 (“real”)	3384	27
Classified as 1 (“fake)	3	162

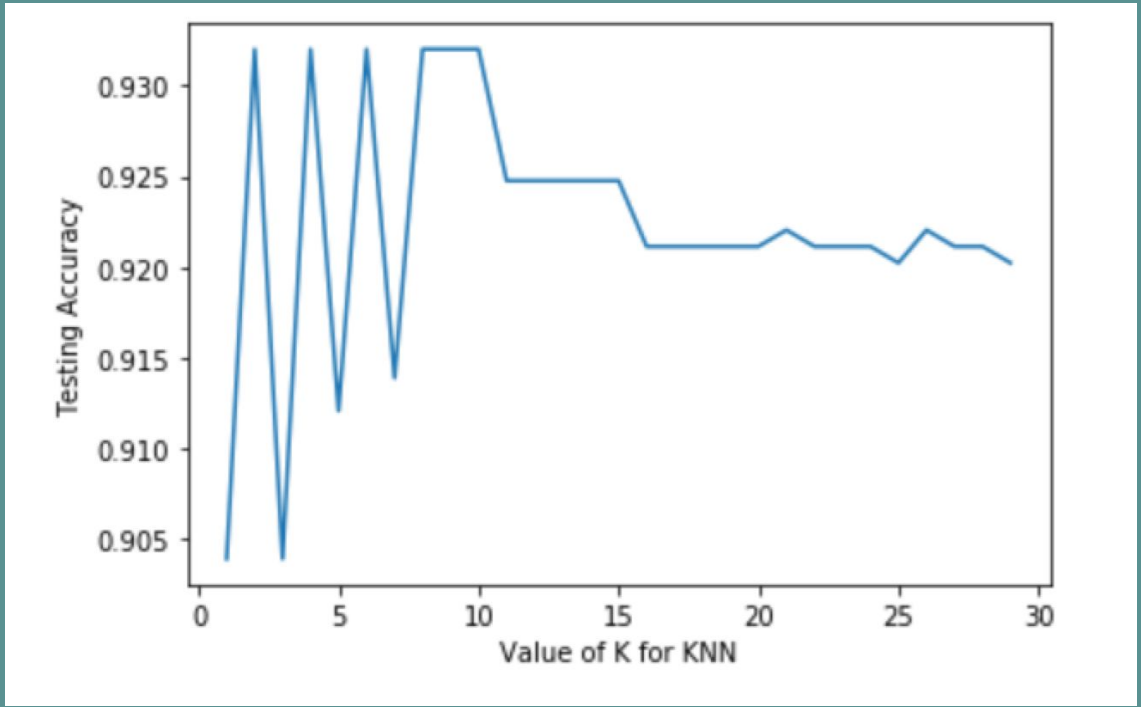
- In this application, it is better to misclassify a real job as fake, type 1 error, as it avoids the issues that would arise with a type 2 error

KNN

- Data cleaning showed that one of the columns initially deemed as useless was job location
  - Only used US locations with city and state given -> 10,000 rows
  - Implemented KNN to see if given a job's location, the job can be classified as fake or real.
- KNN can be used via majority voting to see if a job's location is near other job's locations that are of the same kind.
- The Geocode package of the Google Maps API used to translate the city to exact geographic coordinates
  - Much harder to effectively implement KNN with textual data
    - The slow speed of the API only allowed for 3300 rows
- Training and test set generated
  - 70% training and the rest test set
- The Python code KNeighborsClassifier of the sklearn library was implemented. The metric used was Manhattan distance.

KNN Results

- Experimented with a wide range of different numbers of neighbors to see which one would result in the most accuracy
- K value of 6 is appropriate.
- Confusion matrix generated with the k-value of 6.
  - 1023 jobs were correctly classified as real.
  - 1 falsely classified as fake
  - 5 correctly classified as fake
  - 74 falsely classified as real
- The accuracy of this model was 0.932.



KNN Conclusions

While the accuracy was fairly high, 0.932, KNN analysis of geographic coordinates is not the best model for detecting fake jobs.

The reason for the high accuracy of the model is the higher proportion of real jobs to fake jobs.

In addition, it is very worrisome that out of 79 fake job postings, only 5 were correctly classified. The rest were misclassified as real jobs.

Decision Trees

For the decision tree, bagging, boosting and random forest, the four textual features of each observation were combined to form an aggregate textual feature.

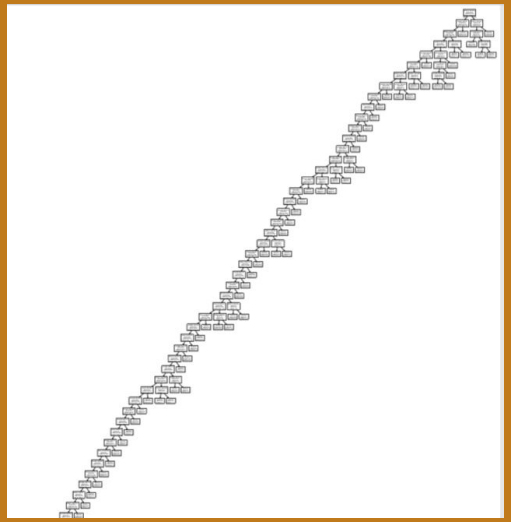
Decision Tree: Accuracy of 97.6%

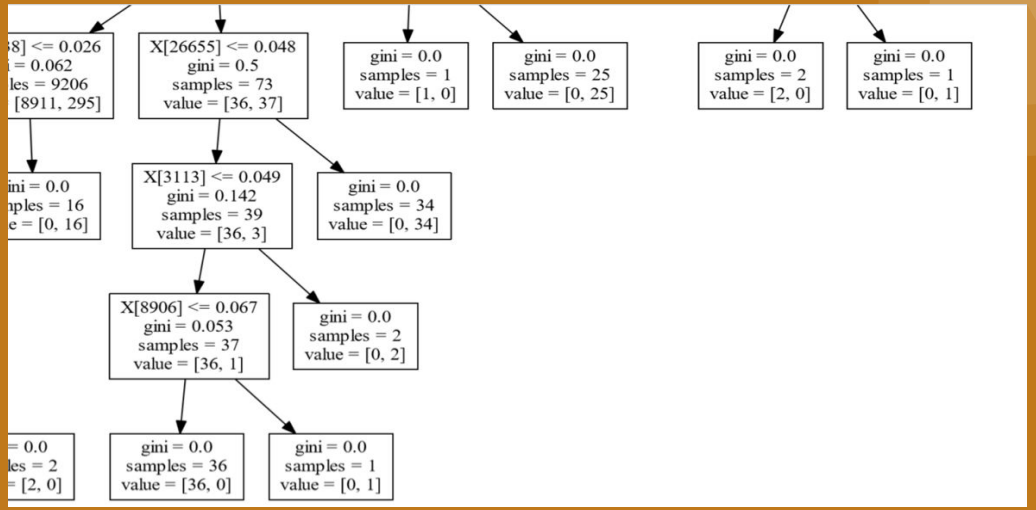
	Predicted Real Job	Predicted Fake Job
Real Job	3776	30
Fake Job	66	151

Random Forest: Accuracy of 97.36%

	Predicted Real Job	Predicted Fake Job
Real Job	3805	1
Fake Job	105	112

Decision Trees





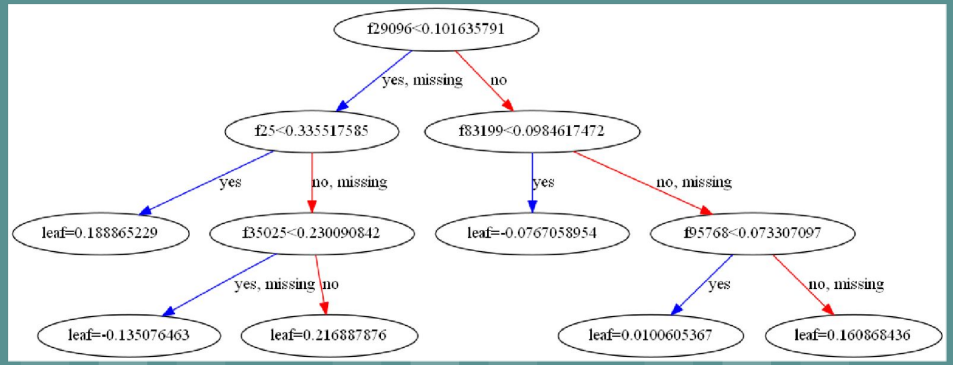
Each X[array] value represents a single word. Each branch, then, extends from that word with similar characteristics, such as sentiment score, length, etc.

Boosting (XgBoost)

Accuracy of 97.6%

XgBoost is a version of boosting, where the model is built sequentially by minimizing errors from previous models. XgBoost adds onto this by using gradient descent and tree pruning

	Predicted Real Job	Predicted Fake Job
Real Job	3802	4
Fake Job	91	126



Bagging

The base estimator for bagging is a SVC (support vector classification), where the base estimator is from which the ensemble is grown. An SVM is used as the base estimator. SVMs used with bagging help to reduce variance, hence avoiding overfitting.

Decision trees, despite being the simplest of the models used, was actually the most accurate.

- Bagging - worst result
- Random forest - second worst result
- Boosting - improved model slightly, although not as much as decision trees

	Predicted Real Job	Predicted Fake Job
Real Job	3805	1
Fake Job	120	97

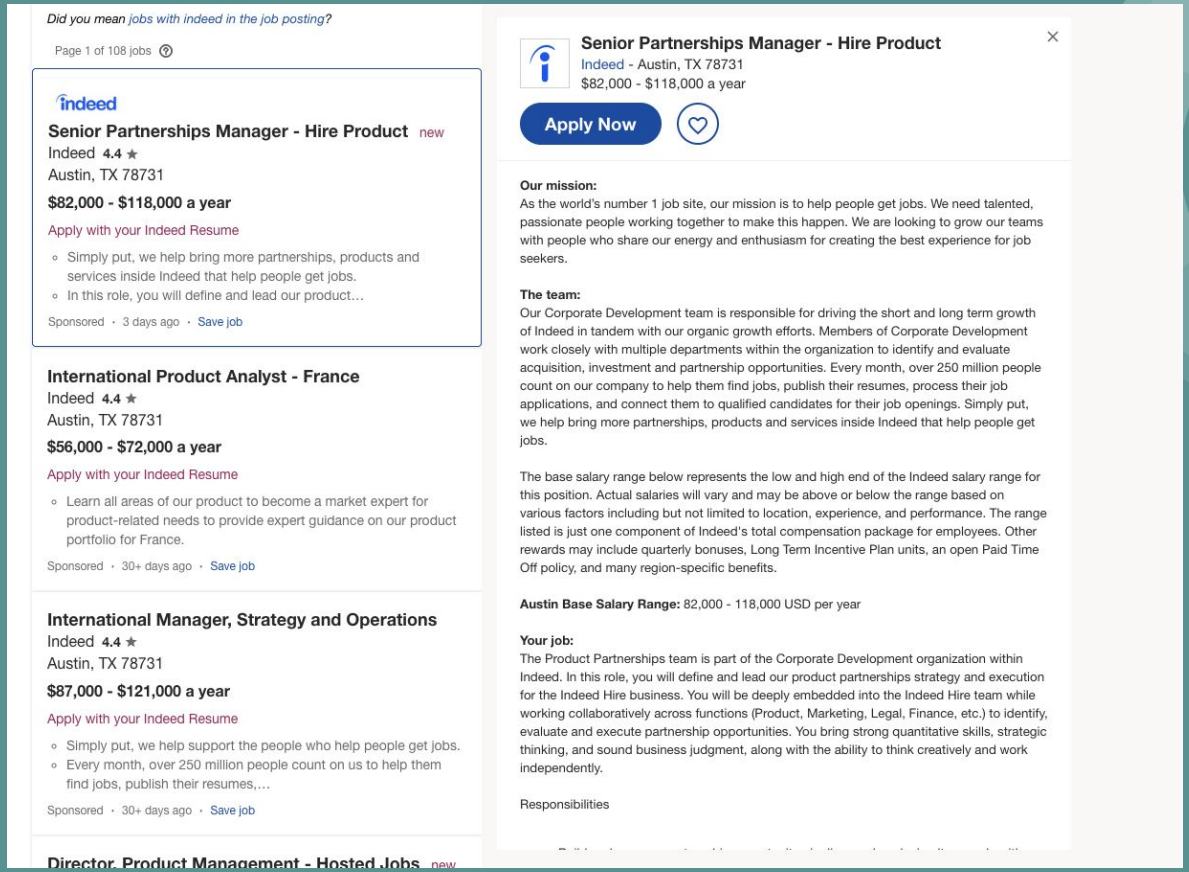
Decision Trees most useful when - explainability between variable is prioritised over accuracy

- Easy to compute and explain why a particular variable is having higher importance
- The tree can be visualized and hence, for non-technical users, it is easier to explain model implementation
- When the data is more non-parametric in nature

Conclusions

- **Numerical and Categorical Analysis**
  - Overall no outstanding models
    - SVM with radial basis kernels best
- **Textual Analysis**
  - Company profile and Description were important textual features. Requirements and Benefits were not so much.
  - Decision trees - The simplest of the textual models used
    - Most accurate and best model
  - Bagging - worst result
  - Random forest - second worst result
  - Boosting improved it slightly
    - Not as much as decision trees.

How are our findings useful in the real world?



"Fake" jobs like this one for the company Indeed, often look very real and can trick anybody.

Our decision tree model is very accurate and able to identify it as fake even though a human likely could not. This could save people from dangerous scams.

```
In [107]: decisiontree_test = clp.predict(x_realtest)
          print(decisiontree_test)

          [1]
```

Decision tree was able to correctly classify the fake job posting!