Enron Email Fraud Detection with Machine Learning

By Anirban Chatterjee

Summary

1.Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: “data exploration”, “outlier investigation”]

The goal of this project is to create a Person Of Interest (POI) identifier model for the Enron email and financial dataset, which will accurately predict if someone should be a person of interest for the Enron fraud case. Machine learning is useful in trying to accomplish this goal because of it's ability to find trends, categorize data and learn from information in order to apply these learnings to new datasets.

Dataset

Enron was one of the largest companies in the United States. By 2002, it collapsed into bankruptcy due to widespread corporate fraud, and during a federal investigation the Enron corpus, emails and financial information, which was retrieved/gathered together was released to the public. The dataset used for this project was preprocessed and stitched together by Katie Malone from Udacity.

Data Exploration

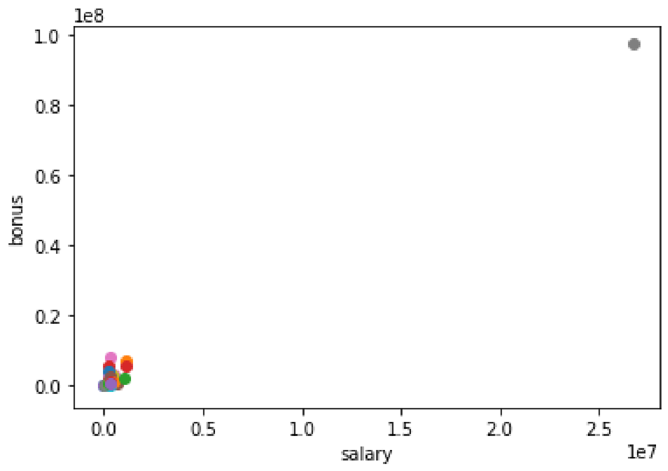
Data exploration was the first step, determining size of data set, number of features, etc. When exploring the data, a number of outliers were discovered, some were not relevant while others were. In order to determine next steps, more data analysis was required.

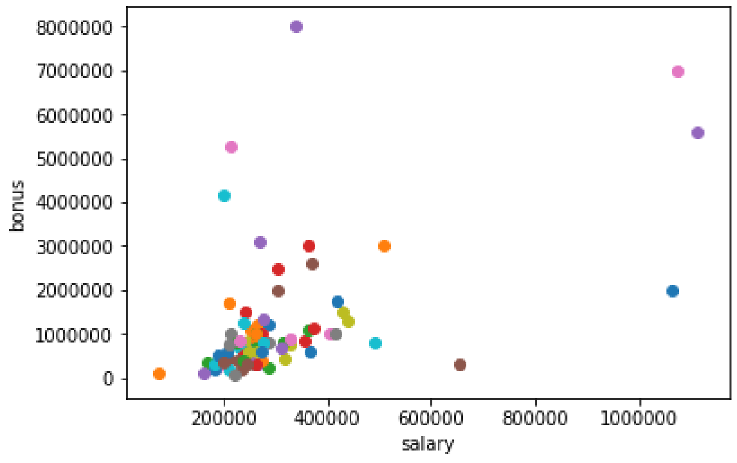
Extreme outliers were removed from the dataset. One was a data point representing the total values and not an actual observation, and the other was listed as "The Travel Agency In The Park". This outlier seems to be the account of a company. Other outliers were kept because although they were outliers, they represented important information about the data. Incompleteness was also a problem with the dataset. Below is an overview of the data, missing values and interesting data points.

Data Overview

- Number of people in dataset: 146
- Number of features for each person: 21
- Number of Persons of Interests (POIs) in dataset: 18 out of 34 total POIs
- Number of non-POIs in dataset: 128
- POIs with zero or missing to/from email messages in dataset: 4
    - Kopper Michael J
    - Fastow Andrew S
    - Yeager F Scott
    - Hirko Joseph

Salary vs Bonus (before and after) dropping outliers:





Employees receiving a salary of 1 Million +:

- Lay Kenneth L
- Skilling Jeffrey K
- Frevert Mark A

Incomplete data - NaN values in features(In percent):

```
[('loan_advances', 97.93103448275862 ),
 ('director_fees', 88.96551724137932 ),
 ('restricted_stock_deferred', 88.27586206896552),
 ('deferral_payments', 73.79310344827587),
 ('deferred_income', 66.89655172413794),
 ('long_term_incentive', 55.172413793103445),
 ('bonus', 44.13793103448276),
 ('to_messages', 40.689655172413794),
 ('shared_receipt_with_poi', 40.689655172413794),
 ('from_messages', 40.689655172413794),
 ('from_poi_to_this_person', 40.689655172413794),
 ('from_this_person_to_poi', 40.689655172413794),
 ('other', 36.55172413793103),
 ('salary', 35.172413793103445),
 ('expenses', 35.172413793103445),
 ('exercised_stock_options', 30.344827586206897),
 ('restricted_stock', 24.82758620689655),
 ('email_address', 23.448275862068964),
 ('total_payments', 14.482758620689657),
 ('total_stock_value', 13.793103448275861),
 ('poi', 0.0)]
```

Feature Selection & Engineering

2.What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

Both univariate feature selection and engineering were performed and used in tested when creating the final model. Feature scaling was also utilized as there were a number of outliers which could skew the results (be used as a primary predictor) but due to the validity of the data, these points could not be removed. Although performance was tested with and without feature scaling as a reassurance to the process, the final model did not utilize feature scaling.

Feature Selection

Feature selection was performed by SelectKBest and by observing the the best scores. No features used were manually picked. However although I did feature selection but I never selected these features because later I used pipeline to pass SelectKBest and it choose the features. The following are the top 10 features I got.

- 'salary',
- 'total_payments',
- 'loan_advances',
- 'bonus',
- 'deferred_income',
- 'total_stock_value',
- 'exercised_stock_options',
- 'long_term_incentive',
- 'restricted_stock',
- 'shared_receipt_with_poi'

Feature Engineering

Three features were engineered for testing of the model.

- poi_ratio_messages - a fraction of the sum of emails that were sent to/from a POI to total mails.
- restricted_stock_ratio - a fraction of the restricted stocks to total stocks value
- money_ratio - a fraction of expenses to salary

With the project goal of identifying POIs, I believed adding two to three additional features which calculated the percentage/ relationship of a POI with other employees at the company via their 'to' and 'from' email interaction would have shed insightful and useful information, allowing the algorithm to use these values as predictors.

Like-wise, I used the other 2 calculated feature to help as a predictor for the algorithm.

I used SelectKBest and their scores to determine the features and their scores, the number 10 for k was decided upon after a number of performance and tuning which determined this to deliver the best mix of performance (timing), precision and recall(used in later part of the project)

Below is a table of features and their scores:

```
Features and their Scores
features: salary score: 18.575703
features: total_payments score: 8.866722
features: loan_advances score: 7.242730
features: bonus score: 21.060002
features: deferred_income score: 11.595548
features: total_stock_value score: 24.467654
features: exercised_stock_options score: 25.097542
features: long_term_incentive score: 10.072455
features: restricted_stock score: 9.346701
features: shared_receipt_with_poi score: 8.746486
```

As you can see, the three fields that I created, poi_ratio_messages, money_ratio and restricted_stock_ratio were not listed in the 10 best features, meaning they were hardly used by any of my models.

Features Rescaling / Normalization

Further more, for many of my classifiers, I did employ a normalization technique, using the sklearn MinMaxScaler. This re-scaled the financial features to be values between 0 and 1. For my chosen algorithm, this was very important, as many of the features were on vastly different scales. For example, salary and from_messages, are two features that represent two different types of numeric data, one being currency and one being a count of messages, where salary is on a much broader scale than from_messages. This requires normalization to ensure that the algorithm is not inappropriately swayed by the fact that one feature is on a different scale.

However it should be noted that for my final Naïve Bayes classifier I did not use MinMaxScaler. I don't know but by scaling the features I got low scores as opposed to when I did not scale them. This was very weird to me as I thought scaling would increase performance.

ML Model

3.What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

The POI identifier model uses the GaussianNB as it provided the best validation results. The other main algorithms used were Decision Tree, AdaBoost and Random Forest algorithm all of which performed adequately in one aspect or another. For example, Random Forest provided the best accuracy score but had low Precision and Recall scores.

Below are the performance of the algorithms with some fine tuning.

Algorithm	**Accuracy**	**Precision**	**Recall**
Naïve Bayes	0.8636	0.4	0.4
Decision Tree	0.84	0.4	0.3
Random Forest	0.901	0.2	1.0
ADA Boost	0.84	0.20	0.25

Generally, I found that performance ranged widely between these, as you can see. I spent many hours tinkering with the pipeline parameters and researching the algorithms, running each at least 5-10 times. I ultimately choose the Naive Bayes classifier, as I found that it produced the best results. This classifier does not have the variety of parameters to select from (as opposed to others), therefore I focused on getting SelectKBest and PCA setup in my pipeline appropriately. I ultimately found with the aid of GridSearchCV that 5 features and 2 PCA components achieved the best results.

Tuning

4.What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you *picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: "tune the algorithm"]*

Tuning the parameters of algorithm is simply the process of changing, testing and updating the parameters in order to get the right mix or settings where once completed, the parameters are optimized to produce the best results. Most ML algorithms have parameters and in some cases there are defaulted values, so it's not always necessary to "tune" an algorithm but in a lot of cases it is. If you do tune your algorithm but not very well, you could end up with a model that seems correct but is actually providing false data. For example, I had an issue where my accuracy score was very high during tuning (train test sets) but in during validation the precision and recall were less than 0.1. I eventually realized that I was using "accuracy score" as the benchmark for my tuning but my validation tests were using the F1-score, Precision and Recall(for Random Forest classifier).

In the case of my final model, I used the GridSearchCV function to determine the optimized parameters. Given a set of parameters, this function evaluates (fit, transforms) all of the possible combinations, then returns a classifier, that provides the best score.

As mentioned above, with my chosen classifier, I did not have many Naive Bayes parameters to tune, but I did have other techniques in my pipeline that required thought and tuning, such as SelectKBest and PCA. GridSearchCV did much of the work, running all the variations of these parameters and selecting the one that resulted in the best performance. The final classifier and parameters used are given below:

```
parameters = {'SKB__k': range(4,7),

              'SKB__score_func': [f_classif],

              'PCA__n_components': [2],

              'PCA__whiten': [True]}
```

```
Pipeline(memory=None, steps=[('SKB', SelectKBest(k=5, score_func=<function f_classif at x000000001013A668>)), ('PCA', PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,  svd_solver='randomized', tol=0.0, whiten=True)), ('NaiveBayes', GaussianNB(priors=None))])

Accuracy: 0.87393       Precision: 0.54905      Recall: 0.30500 F1: 0.39216     F2: 0.33476

Total predictions: 15000        True positives:  610    False positives:  501

False negatives: 1390   True negatives: 12499
```

Validation

5.What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

Validation is the process of checking your model's prediction against data that wasn't used to train your algorithm/model. A classic mistake of overfitting occurs when you train the algorithm on all available data instead of splitting it into training and testing data. Overfitting causing the model to merely memorize classification and not 'learn' to generalize and apply this information to new datasets.

The final model used Stratified Shuffle Split cross validation iterator to randomly create multiple train test sets of data. This was an ideal approach given the small dataset and even smaller number of POIs within the dataset.

Model Results

6.Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]

The final model uses Precision, Recall and F1 scores to evaluate how good the model is in predicting POIs. The raw data can be see below. Each observation is a test, and each test made 15,000 predictions.

- Accuracy: 0.87393 Precision: 0.54905 Recall: 0.30500 F1: 0.39216 F2: 0.33476
- Total predictions: 15000 True positives: 610 False positives: 501 False negatives: 1390 True negatives: 12499
- *Precision is the measurement of how many selected items were identified as relevant*
- *Recall is the measurement of how many relevant were selected*

The model's precision is approx. 55% i.e from the people classified as POIs by the model, 55% of them are actual POIs. However, the model's recall is approx. 30% i.e from the number of actual POIs in the total dataset, the model correctly identifies 30% them. It can be concluded that although the model "spreads a wide net" it will capture over 30% of actual POIs. I hope to tune this and get a good recall score in the meantime.

References

- https://discussions.udacity.com/t/GridSearchCV-and-testingtraining-data/36107
- http://stackoverflow.com/questions/22903267/what-is-tuning-in-machine-learning

- https://discussions.udacity.com/t/project-fear-strugging-with-machine-learning-project/198529/48
- https://discussions.udacity.com/t/starting-the-final-project/179090/13
- http://scikit-learn.org/stable/