Enron fraud 2001, “Person of Interest (POI)” identifier

Adrián Vera Ros. Nov 18

Summary

The objective of this project is to use machine learning to build a “Person of Interest” identifier for the Enron fraud case.

Founded in 1985, Enron became one of the largest companies in the United States during the 90’s. Little was known that this growth was founded on illegal accounting practices and corporate fraud. By the end of 2001 it filed for bankruptcy and became one of the biggest cases of fraud worldwide.

During the Federal Investigation that followed, confidential information like emails and personal financial data became public. With this information, our objective is to be able to identify if someone was involved in the fraud through machine learning classification algorithms.

Dataset

· Financial Features: 14 integer features
- 10 payments features
- 4 stock features

· Email features: 6
- 5 integer features
- 1 string feature (email address)

· POI feature (Boolean): A person of interest (POI) is defined as an individual who was indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Total number of features: 21
Data points: 146

Among the 146 DATA POINTS, 18 are labeled as POI. This uneven distribution (12,3% vs 87.7%) is the first major difficulty found in this dataset. This information asymmetry might become a challenge in order to learn patterns in our data.

Looking at the insiderpay.pdf file (added) it we can draw many conclusions about the dataset information.
1) It can be seen as Total payments and Total Stock Value are two features that are direct sum of all other features in their categories. Since the Enron scandal was fundamentally based on the company results and stock prices, we think that total stock value is going to be one of the top features for the final identifier.

2) There are two rows that don’t fit in the dataset: “TOTAL” and “THE TRAVEL AGENCY IN THE PARK”. http://www.businesstravelnews.com/More-News/Enron-s-Agency-Changes-Name-Reaffirms-Corp-Commitment. These two data points are seen as outliers and will be excluded from the dataset.

3) There is a lot of NaN data. Since most of our data is numeric and we only have one string feature (email address), we are going to convert any NaN point to 0 and ignore the string feature for future calculations. But first we are going to check which features are most common for the POI and the completeness of the information we have.

Feature	Count	POI	NaN	% Missing
poi	143	18	0	0%
total_stock_value	125	18	18	12%
total_payments	123	18	20	13%
restricted_stock	109	17	34	23%
exercised_stock_options	101	12	42	29%
salary	94	17	49	34%
expenses	94	18	49	34%
other	91	18	52	36%
to_messages	86	14	57	39%
from_messages	86	14	57	39%
shared_receipt_with_poi	86	14	57	39%
from_poi_to_this_person	86	14	57	39%
from_this_person_to_poi	86	14	57	39%
bonus	81	16	62	43%
long_term_incentive	65	12	78	54%
deferred_income	48	11	95	66%
deferral_payments	38	5	105	73%
loan_advances	3	1	140	97%

Excluding the totals, the stock related features are the most common for the POI group. It also can be seen as deferred income/payments and loan advances are the most uncommon features and with lower POI presence.

4) Finally, it has been found how for one of the data points, all features have NaN values. This person is “LOCKHART EUGENE E.” and was removed from the dataset as well.

Feature selection

Before proceeding into the algorithm analysis, we have to optimize the input information in order to get better results. For that, we will exclude features with no transcendence for our analysis, create new ones that might have and select the best features available.

1) Apart from email address, we are going to **exclude** loan_advances and deferral_payments. Not only for the general lack of data about it but also because the low number of POIs that have data in those features.
2) We are going to **create** new features:
    a. From_poi_ratio: ratio of emails received from poi vs total emails received.
    b. To_poi_ratio: ratio of emails sent to poi vs total emails sent.
    c. Stock_ratio: ratio of stock payments vs total payments

We want to check the proportion of emails that a person has with direct relation to poi and, since the Enron scandal was closely related with inside information and stock movements, we want to see if there’s a higher coefficient in stock earnings vs total earnings.

3) **Select best features**

We run SeleckKbest with k='all' to obtain the score for each feature.

Feature	*Score*
exercised stock options	24,81508
total stock value	24,18290
bonus	20,79225
salary	18,28968
to poi ratio	16,40971
deferred income	11,45848
long term incentive	9,92219
restricted stock	9,21281
total payments	8,77278
shared receipt with poi	8,58942
expenses	6,09417
from poi to this person	5,24345
other	4,18748
stock ratio	3,42822
from poi ratio	3,12809
from this person to poi	2,38261
director fees	2,12633
to messages	1,64634
from messages	0,16970
restricted stock deferred	0,06550

As suspected, stock related variables have a higher relation with the POI datapoints. Of the three variables we created, only "to poi ratio" has high transcendence. For the next steps in our project we will use the best 15 features.

However, they are still too many features to be run, and not all of them are equally important; so after deciding which algorithms we are going to use, we will proceed to select the most optimal parameters in order to get the best result. For that, we will rescale the data through MinMaxScaler in order to reduce the magnitude of features with big difference in data figures and have GridSearchCV, in conjunction with SelectKBest, help us decide how many features are the best for each algorithm.

Algorithm

We will follow the suggested algorithm order by sklearn in it's documentation: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

· Linear SVC
· Gaussian Naïve Bayes
· KNeighbors Classifier
· SVC (other kernels)
· Ensemble Classifiers
  - Random Forest
  - Adaboost

Due to the rbf kernel of SVM giving problems (divide by 0 result), we will only use VSC Linear.

	ACCURACY	PRECISION	RECALL	F1
SVC LINEAR	0.50020	0.14421	0.55700	0.34933
GAUSSIAN NB	0.83253	0.34852	0.29450	0.20483
KNN	0.87820	0.64636	0.19100	0.31924
RANDOM FOREST	0.86180	0.43987	0.13350	0.22910
ADABOOST	**0.85000**	**0.41427**	**0.30200**	**0.34933**

Being Adaboost an adaptative algorithm we can expect it to get great results out of the box, and it did. Before optimizing it, it's already the best-rounded algorithm given the evaluation metrics chosen for this project. So now we are going to tune it up a little to see if we can get better scores.

Algorithm parameters

Tuning up the parameters of an algorithm is to look for the optimal combination of parameters in order to get the best result. For that reason, we are going to use "Pipeline" to create a process that first rescales the dataset with "MinMaxScaler" and then automatically search the best features by using a "GridSearchCV" to cross-validate the different options of "SelectKBest" and "n_estimators".

ADABOOST	K	N_ESTIMATORS	ACCURACY:	PRECISION:	RECALL:	F1:
	13	1	0.84887	0.16708	0.03350	0.05581
	13	5	0.85833	0.44663	0.26150	0.32986
	13	**10**	**0.85700**	**0.44795**	**0.31200**	**0.36782**
	13	25	0.84373	0.39031	0.30600	0.34305
	13	50	0.84233	0.38471	0.30450	0.33994

The K=13 and n_estimators = 10 provides the best possible combination.

Validation

In order to validate our dataset, we have to separate the data we train from the full dataset. This way, we can find patterns in our data and not have an algorithm uniquely tuned to our dataset. The basis behind this is the bias vs variance trade-off. We want an algorithm able to properly classify our data but at the same time for it to be able to learn and to adapt to different datasets.

This has become unexpectedly a major issue during the elaboration of this project. We not only have a low number of datapoints, but the distribution of POI-nonPOI is also highly biased. For this same reason, using the provided 'train_test_split' returned bad evaluation metrics. For most of the classifiers it was unable to provide precision or recall figures.

Because of this, we proceeded to split our data with "StratifiedSuffleStrip". This function shuffles the data and then splits it k times, but maintaining the proportion of the binary features. This way, it doesn't provide extreme precision or recall figures.

Evaluation metrics

ALGORITHM	K:	N_ESTIMATORS:	ACCURACY:	PRECISION:	RECALL:	F1:
ADABOOST	**13**	**10**	**0.85700**	**0.44795**	**0.31200**	**0.36782**

Where:

Accuracy: percentage of correct classifications as POI or non-POI

Precision: Probability of a POI classified person being a POI
True positives / (true positives + false positives)

Recall: Probability of a POI being correctly classified.
True positive / (true positives + false negatives)

F1: 2*((precision*recall)/(precision+recall))

In a case with the magnitude of Enron, it's not only crucial to have high accuracy (identify POIs correctly), but also not to wrongly classify other people. That's why, for this project, we have looked after an algorithm with a minimum precision and recall scores.

Also, in a dataset with this much divergence, it would be easy to have high accuracy just by creating a dummy algorithm that always reply "non-POI". Other evaluation metrics as precision and recall are more resistant to this kind of datasets.