Identifying Fraud from Enron Email

Machine Learning Final Project

by Dirk Kalmbach

(September 2015)

***Executive summary**: Enron was on of the largest corporate frauds in US history. This report describes the implementation of an machine learning alghorithm to identify persons involved in the fraud by analyzing their financial information and email conversation behavior. It is the final project in the Udacity Machine Learning course. All code to perform the alghorithm can be found in the corresponding python file `poi_id.py`.*

1 Summary

About Enron

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives.

Goal of this project

The model made for this project is seeing if certain financial characteristics and e-mail communication behavior can be used to identify a POI from a non-POI.[1] Machine Learning is an adequate method for this as it "*explores the study and construction of algorithms that can learn from and make predictions on data.*" (wikipedia)

Dataset

This dataset consists of personal **financial information** like salary, bonuses, loan advances, total payments as well as **email information** like how many emails from and to POIs and how many emails in total were sent to / received from for top executives.[2]

After identifying visually one outlier which probably resulted from an incorrect data transfer and eliminating it manually, the dataset contains 145 persons. 18 among them are POIs.

The dataset has a lot of missing values for Non-POIs (see Appendix for more detail). There are also many more Non-POIs than POIs in the dataset. However, the fact that missing values reflects only Non-POIs evens this somehow out as a lot of features for Non-POIs are implemented with zero value. The small dataset has some consequences on the model building approach. First of all, more advanced crossvalidation methods are appropriate as a simple split in training and testing set cannot ensure that both sets contain POIs and Non-POIs for example. I therefore validate the algorithm using a k-fold-crossvalidation, where the validation will be repeated and averaged k times. Secondly, metrics like recall and precision are also more approbriate than accuracy.[3]

2 Features selection

The dataset contains 7 features representing email information and 13 features representing financial information. One feature (`other`) is not assignable. Another feature indicates whether the person is an POI or not. A table with all features and missing values can be found in the Appendix.

[1] We define a POI as a person who was either indicted, settled without admitting or testified in exchange for immunity for prosecution during the Enron trial. Find out more about Enron participants at *USATODAY.com (2015)*.

[2] We extracted these information in class from „The Enron Email Corpus“ *(Cohen 2015)* which can be also investigated online at http://www.enron-mail.com/email/.

[3] To provide a simple explanation of both concepts, let us imagine a training set which only contains one or two POIs will learn to identifies every person as a Non-POI. As a result, the test set now contains allmost all POIs. And yet the number of Non-POIs is still higher than the number of POIs in the test set. This results in a high accuracy - the model overfits.

I started with some obvious features like salary and bonus, added successively other features and tried different machine learning algorithm with different tuning parameters and documented the performance in a spreadsheet. Although this strategy gave me some feeling about which features seems to be important and which not, I did not achieve a recall and precision score higher than 0.3.[4]

I then switched to the automatic feature detection tool SelectKBest (k=7)[5] which suggested to keep the following features according to the f1-score[6]:

- bonus
- total_stock_value
- salary
- exercised_stock_options
- deferred_income, and
- long_term_incentive, as well as
- fraction_to_POI,

The latter feature was constructed by hand and measures how many emails were sent to POIs as a percentage of all sent emails.

I also constructed a second feature which measures the fraction of received emails from POIs among all received emails. However, as this feature was not selected by SelectKBest I did not use it for the final model.

The intuition behind constructing these features was that POIs may have stronger email-connections among each other (they need more to communicate to each other, ask question, answer question, clarify things, meet for dinner, etc.). Higher values of these features indicate stronger email-connection between the person and POIs. Both new features were added to the list of all features, and then SelectKBest was run on all features. The decision wether to use the new features or keep the old features was based on the F1-score.[7]

It was not necessary to scale features as Naive Bayes does not require this.

3 Alghorithm

I used the Gaussian Naive Bayes algorithm. I also implemented a Decision Tree, Random Forest and Adaboost but Naive Bayes performed best. The

[4] One of the requirements to pass the project.

[5] k=7 with f_classif algorithm was used as the suggested features maximized precision and recall among k=1 to 10:

k	feaures	recall	precision	f1
1	exercised_stock_options	0.32	0.46	0.38
2	.. + total_stock_value	0.27	0.47	0.34
3	.. + bonus	0.35	0.49	0.41
4	.. + salary	0.32	0.5	0.39
5	.. + fraction_to_poi	0.3	0.46	0.36
6	.. + deferred_income	0.35	0.48	0.4
7	*.. + restricted_stock*	*0.38*	*0.46*	*0.42*
8	.. + total_payments	0.32	0.40	0.36
9	.. + shared_receipt_with_poi	0.31	0.39	0.35
10	.. + long_term_incentive	0.31	0.38	0.34

[6] The F1-score is the harmonic mean of recall and precision.

[7]

old features		new features
Naive Bayes (criterion=‘gini‘) with features: • from_messages • to_messages • from_poi_to_this_person, • from_this_person_to_poi	**vs**	Naive Bayes (criterion=‘gini‘) with features: • fraction_to_POI • fraction_from_POI
F1: 0.00493		*F1: 0.01338*

following table shows the recall and precision scores of these algorithms:

Algorithm	Parameters	Recall	Precision
Naive Bayes	*criterion='gini'*	*0.36*	*0.46*
Decision Tree	criterion="gini", min_samples_split=2	0.26	0.28
	criterion="gini", min_samples_split=3	0.25	0.29
	criterion="gini", min_samples_split=4	0.24	0.28

Random Forest	min_samples_split=11,
	n_estimators=6
		0.22	0.51

Adaboost	algorithm=‘samme‘	0.32	0.24

Table 1: Recall and Precision for different algorithm

4 Tuning

Parameter tuning (i.e.: finding the best combination of parameters of an algorithm) usually is an important step in machine learning as this often results in higher validation scores. On the other side, many machine learning algorithm have a lot of parameters which - especially when different algorithm are compared - often leads to many possible combinations, or like Domingos (2012) writes: "*Machine learning algorithms have lots of knobs, and success often comes from twiddling them a lot, so this is a real concern.*"

As Naive Bayes does not have any tunable parameters, this step was omitted.

Nevertheless, I also run a decision tree algorithm (among other algorithm) to see wether it beats the Naive Bayes or not. I tried different values for the min_sample parameter[8] and compared the results to the Naive Bayes scores.[9] But none of these beat the Naive Bayes algorithm so I kept the latter.

5 Validation

Validation means proofing or examining wether an algorithm works properly (i.e. predicting the expected outcome). In machine learning this can be done by splitting the dataset into a training and testing set. The former is used to train the algorithm, whereas the latter is used to test the algorithm by comparing the predicted results with the data in the test set.

The test set is used to select features and tune the algorithm. Depending on the model's performance on the test set, the feature set and/or tuning gets readjusted.

Validation is one of the most important parts in machine learning. Without validation it is possible that an algorithm memorizes all features, i.e. showing high performance scores only at the training set.

I began with deploying a 2-fold and a 3-fold cross validation to validate the results. In the beginning, the metrics showed acceptable results. Precision and recall were 0.6 and 0.4 for the „7-feature-solution“ with naive bayes.[10]

One drawback of k-fold cross validation is that for small sample sizes the results often does not vary so much. As a consequence, I switched over to use a *shuffle split cross validation*.[11]

Testing the algorithm with the latter resulted in a precision score of 0.46 and a recall score of 0.367 for the Naive Bayes.

[8] How many persons in a node are required to split a node.

[9] Other parameters are the splitting criteria (gini vs entropy), the max_depth to limit the size of the tree and therefore the processing time, min_samples_leaf, min_weight_fraction_leaf and max_leaf_nodes.

[10] Precision and recall were 0.4 0.5 for k=2, respectively.

[11] Stratified shuffle split cross validation with 1000 re-shuffling & splitting iterations.

6 Evaluation

As Table 1 shows precision was 0.46 and recall was 0.36. Accuracy was 0.85 and the f1-score was 0.41.

Accuracy is the fraction of correctly identified POIs and Non-POIs from all persons, i.e.: the probability that a randomly selected person was classified correctly either as a POI or a Non-POI (based on its financial data and email communication behavior).

Precision is the fraction of correctly identified POIs among all (real) POIs, i.e. the probability that a randomly selected person among all as POIs classified person actually is a POI.

Recall is the fraction of correctly identified POIs among all (correct or wrong) identified POIs, i.e. the probability that a randomly selected person among all real POIs actualy is a POI.

Accuracy, precision, recall, and f1-score all range from 0 to 1, with 1 being optimal.

7 Reflection

The alghorithm I implemented used only a few features. Better results could be achieved by digging deeper into the email information, e.g. using text learning to analyze the content as well as analyzing the time structure of the email communication. However, I did not implement any natural language processing alghorithms as the goal of this project was to achieve a precision and recall score above 0.3.

Some minor critics about the quality of the original datasets are that the emails does not include attachments, and that some messages have been deleted "as part of a redaction effort due to requests from affected employees" (Cohen 2015).

8 References

Cohen, William. W. „Enron Email Dataset.“ May 8, 2015. Accessed September 13, 2015. http://www.cs.cmu.edu/~./enron/.

Domingos, Pedro. A few useful things to know about machine learning. Commun. ACM. 55 (10):78–87, 2012.

USATODAY.com, „A look at those involved in the Enron scandal.“ December 28, 2005, Accessed September 13, 2015. http://usatoday30.usatoday.com/money/industries/energy/2005-12-28-enron-participants_x.htm

Wikipedia contributors, "Machine learning," Wikipedia, The Free Encyclopedia, Accessed October 14, 2015. https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=685498675.

9 Appendix

List of all features

		missing values		
	name	POI	Non-POI	*total*
1	salary	50	0	*50*
2	bonus	62	0	*62*
3	total stock value	20	0	*20*
4	exercised_stock_options	38	0	*38*
5	deferred income	90	0	*90*
6	exercised_stock_options	38	0	*38*
7	long_term_incentive	74	0	*74*
8	director_fees	111	0	*111*
9	total_payments	21	0	*21*
10	loan_advances	125	0	*125*
11	expenses	51	0	*51*
12	deferral_payments	94	0	*94*
13	restricted_stock_deferred	110	0	*110*
14	from_messages	56	0	*56*

15	to_messages	56	0	*56*
16	from_this_person_to_poi	56	0	*56*
17	from_poi_to_this_person	56	0	*56*
18	shared_receipt_with_poi	56	0	*56*
19	shared_receipt_with_poi	56	0	*56*
20	email_adress	35	0	*35*
21	other	53	0	*53*

Table 2: Variables and missing values

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.