Identifying Fraud from Enron Email

A Udacity Nanodegree project by Jay Wijaya

19 Aug 2015

Notes:

1. "final_project/poi_id.py" contains only the final steps of my analysis, not the documentation of it.
   I DO NOT PUT MY EXPLORATORY CODE IN THIS DOCUMENT AND I DO NOT INTEND TO DO SO, PLEASE READ MY DOCUMENTATION INSTEAD. (sorry had to make this clear, as previous reviewer did not get this)

2. My work is documented in detail in document "Identifying Fraud from Enron Email". You can find code and explanations there. Locally, its ipython notebook file is located at "final_project/Identifying Fraud from Enron Email.ipynb" and html file at "final_project/identifying-fraud-from-enron-email.html".
   PLEASE FIND THE ANSWERS FOR RUBRIC (e.g. total number of data points, allocation across classes, etc) **IN THAT DOCUMENT.**

3. Another document "Bag of Words Implementation" explains how I implemented TfidfVectorizer to use email messages in POI prediction. Although the model acquired better score there, it is incompatible with tester.py. Locally, its ipython notebook file is located at "final_project/Bag of Words Implementation.ipynb" and html file at "final_project/bag-of-words-implementation.html".

4. I'm having problem with running SVC in my dataset. The prediction score always end up 0. Could you enlighten me on this, please? Please see this in my documentation, NOT at poi_id.py.

Answers

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”]*

Answer:

This project attempts to predict the likelihood of someone being a suspect of Enron fraud conspiracy by looking at given dataset. We call the suspects Person of Interest (POI). The dataset contains insider pays to all Enron executives as well as emails sent through their company accounts, and their POI status. We use machine learning to learn the insider pays and emailing habits of POIs and non-POIs and see if we can find a pattern there, then use the classifier created to predict the likeliness of someone with a particular pattern of being a POI or not.

During my analysis, I found an outlier in the dataset with name "TOTAL" which I removed. Later I found two other outliers "BELFER ROBERT" and "BHATNAGAR SANJAY". They have several wrong values, which I then corrected from document *enron61702insiderpay.pdf*.

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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]*

Answer:

In the end I ended up using the following feature in my POI identifier: '*bonus*', '*salary*', '*deferral_payments*', '*deferred_income*', '*director_fees*', '*exercised_stock_options*', '*expenses*', '*from_poi_to_this_person_ratio*', '*from_this_person_to_poi_ratio*', '*loan_advances*', '*long_term_incentive*', '*other*', '*restricted_stock*', '*restricted_stock_deferred*', '*salary*', '*shared_receipt_with_poi*'. I scaled these features with MinMaxScaler algorithm since they have much different value range (i.e. number of messages and dollar range in income / stocks).

Initially I tried using only features with high correlation to "poi" but turns out F1 score was higher with including all other parameters. During further analysis, I found that several income related fields are combined into "total_payments" and other stock related fields are combined into "total_stock_value" so I then removed these two fields from my feature list. I also created "from_this_person_to_poi_ratio" feature ("from_this_person_to_poi" / "from_messages") and "from_poi_to_this_person_ratio" feature ("from_poi_to_this_person" / "to_messages") - then found out that by including these features instead of their substances I improved my prediction model. As a side project, I also engineered features from email messages by Tfidf-vectorizing them. Please see "Bag of Words Implementation" document

for detail. I tried running PCA on my features to make them generalize better but it resulted in lower scores so I removed that step in my pipeline.

Feature importances in my DecisionTree:

1. exercised_stock_options (0.40)
2. from_this_person_to_poi_ratio (0.13)
3. shared_receipt_with_poi (0.13)
4. long_term_incentive (0.12)
5. from_poi_to_this_person_ratio (0.11)
6. other (0.07)
7. restricted_stock (0.03)
8. bonus (0.01)
9. salary (0.00)
10. restricted_stock_deferred (0.00)

3. *What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]*

Answer:

I ended up using DecisionTreeClassifier after spot checking various other classifiers.

Since our case is classification problem, I tried DecisionTreeClassifier, SVC, GaussianNB, then I tried joining the best out of these three (DecisionTreeClassifier) with AdaBoostClassifier. I then dug deeper into trying other tree algorithms: RandomTreeClassifier and GradientBooster. I found most about this from this article. I also initially tried using DummyClassifier to see how Machine Learning algorithm fare against random guessing, and the result was quite good it seems (about 3x better).

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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]*

Answer:

It means trying out different configurations in the machine learning algorithm used for the problem set. In my project, found that one of two things could happen when you do not tune your algorithm:

1. It hurts your computational performance. This happened when I used TfidfVectorizer just as it was. Since it had no limitation on how many words to score the algorithm

ran for an awfully long time, that I finally decided to stop it and did some pruning by adding

2. It hurts your prediction performance. This is quite obvious as with tuning (and correct cross validation technique) you would end up with a model that better represent your problem.

I tuned the parameters using sklearn's GridSearchCV with F1 score as its scoring metric. I then keep the best parameters and used it in the final classifier.

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"]*

Answer:

Validation in Machine Learning means simply what the word means, validating your prediction model with the dataset given to you to find its likely performance when given data it has never seen before. In other words, estimating model performance from training data. Some classic mistakes of validation are *using global information, using future data* (i.e. training phase uses data that should not have been available at the time analysis is running)*, and picking the best model*. This paper has excellent explanation on them.

In validating my model, I created a GridSearchCV to test out parameters I was interested in, ran it on a cross-validation set I created (NOT from *test_classifier* method), then use the best parameters found by GridSearchCV in *test_classifier* method. That way I avoided the third mistake I mentioned above.

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"]*

Answer:

My final prediction model performance:
Accuracy: 0.86047 Precision: 0.47906 Recall: 0.53200 F1: 0.50415 F2: 0.52050
Wall time: 1.32 s (for running *test_classifier* method)

Precision and recall: Here is the best way I found to remember them - Precision says how **exact** you were among your pick-ups. For example, as you picked, say 2 correct items out of all 7 items you took, you were **28% exact**. Recall says, how **complete** you were among your pick-ups. For example, if you took 2 correct items out of all 4 correct items in total, you were **50% complete** in identifying all the correct items.

F1-score: is the harmonic mean - a **weighted average of precision and recall**. Since this is a representative of both precision and recall I prefer to use this as my formal metric. An interesting thing was I got **3.7** when using purely bag of words implementation + DecisionTreeClassifier without other features given in dataset, and down to **2.1** when other features added into it, and that was before I optimized the model parameters; In other words it has the potential to be a better predictor. However, since it went too far from the intended direction of this project I decided not to pursue that.