Identify Fraud from Enron Email

--- Hang Zhu

Q: 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?

Project Background

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, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

In this project, the goal is to build a classifier to predict if a person is Person of Interest in the fraud case. Machine Learning could be help to accomplish the goal since it can learn the pattern from training data given and with the pattern it would predict Person of Interest. Features are required for Machine Learning. In this project, features involve Enron email information and financial data. I used all 23 features at the beginning as exploration.

Data Exploration:

Total data points: 145

Poi: 18

Non-poi: 127

Number of features(final): 19

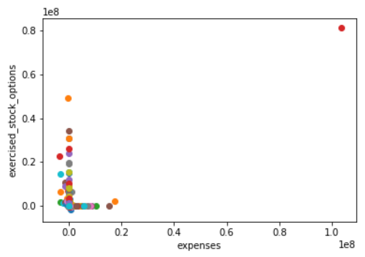
Features with most missing values:

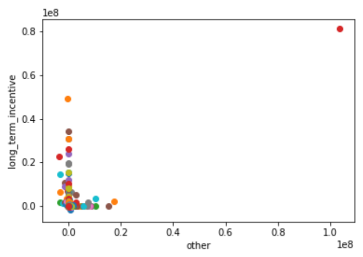
loan_advances(142), director_fees(129),

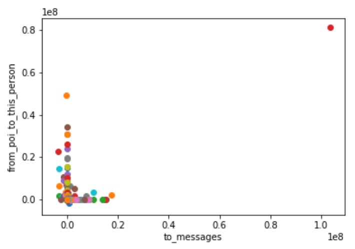
restricted_stock_deferred(128), deferral_payments(108), deferred_income(97)

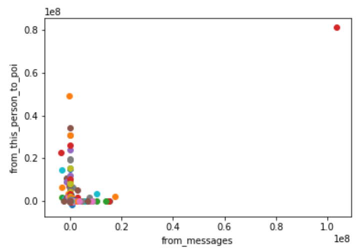
Check outliers.

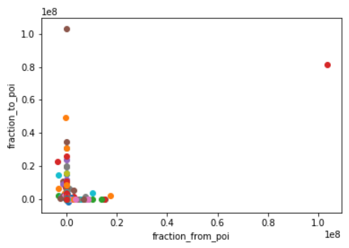
I plotted some pairs of features in a scatter plot and found out that there is always a point which is far away from other points in each plot. Thus, this point should be regarded as outlier. After checking given data description and data, I found out it is ‘Total’. This point was removed. After removing this point, other points are pretty normal even if there are a couple of points with high value.

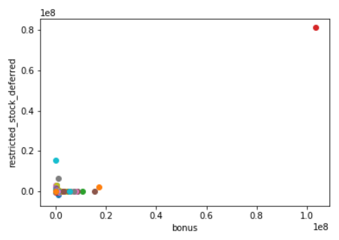












Q: What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it.

Feature Engineering:
There are 5 features which include at least 60% NaN values, so I removed them. Another feature 'email' is string feature, so I also remove it.
The features I chose at the beginning are:
'salary', 'total_payments', 'bonus', 'restricted_stock_deferred', 'total_stock_value', 'expenses' ,'restricted_stock' 'exercised_stock_options', 'other', 'long_term_incentive', 'to_messages', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'

Feature scaling:
I didn`t scale the features because scaling would mask the information from the features.

Features Selection:
I used feature importance from Random Forest to do feature selection. Feature importance is below.
('salary', 0.038058805289984968), ('total_payments', 0.044365007355075516), ('bonus', 0.18002893828487671), ('restricted_stock_deferred', 0.0), ('total_stock_value', 0.10343626193186568), ('expenses', 0.064938881040025487), ('exercised_stock_options', 0.10274444217320386), ('other', 0.078676661137683221), ('restricted_stock', 0.047221461510536099), ('long_term_incentive', 0.048459264078019024), ('to_messages', 0.024369884835001125), ('from_poi_to_this_person', 0.05935958485475068), ('from_messages', 0.029066407946626859), ('from_this_person_to_poi', 0.083779509350048603), ('shared_receipt_with_poi', 0.095494890212302203)

Feature selected:
Only 'restricted_stock_deferred' has importance 0, so I will remove this feature and keep others.

Adding Features
I created 5 features: 'fraction_from_poi', 'fraction_to_poi' , 'fraction_stock_incentive', 'ratio_salary_bonus', 'ratio_salary_restricted_stock'

'fraction_from_poi': the fraction of 'from_poi_to_this_person' messages in all of 'to_messages'
'fraction_to_poi': the fraction of 'from_this_person_to_poi' messages in all of 'from_messages'
The two new features transform the number of messages into proportion of messages associated with poi, which gives better indication.

'fraction_stock_incentive': the division of 'exercised_stock_options' and 'long_term_incentive'.
This feature gives the ratio between excercised stock and long term incentive. Both of them measure the treasure except salary and bonus and both of them would be relatively large, so I would like to see the ratio

'ratio_salary_bonus': the ratio between bonus and salary
'ratio_salary_restricted_stock': the ratio between restricted_stock and salary
The two features give ratio between restricted stock or stock and salary. For POI, some of them would have pretty normal or low salary but very high bonus or restricted stock

Performance comparison

I used Logistic Regression as model and test on validation data with performance as precision/recall.

Precision: the proportion of people who are predicted as POI and are POI actually in all people predicted as POI.

Recall: the proportion of people who are predicted as POI and are POI actually in all people who are POI actually.

Features	Precision(1)	Recall(1)
Original	0.40	0.40
Selected from Random Forest	0.50	0.40
Add'fraction_from_poi', 'fraction_to_poi'	0.50	0.40
Add 'fraction_stock_incentive'	0.50	0.40
Add 'ratio_salary_bonus'	0.50	0.40
Add 'ratio_salary_restricted_stock'	0.50	0.40

After selecting features using Random Forest, the performance increases, but after adding the extra features, no increase for performance. However, the test data set is too small and the features still have importance over 0 in Random Forest, so I would still keep these new features.

Final Features used:

'salary', 'total_payments', 'bonus','restricted_stock', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'to_messages', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi', 'fraction_from_poi', 'fraction_to_poi', 'fraction_stock_incentive', 'ratio_salary_bonus', 'ratio_salary_restricted_stock'

Q: What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is vitally important. We can split data as training data and test dat. After fitting a model with training data, we should measure the performance on an test data set and we can compare the performances for different algorithms or feature set using this validation. On the other hand, we can also know if our model is overfitting by compare performance on validation data and training data. If I didn`t do validation and used performance on training data as criteria to select features or models, then we can`t handle overfitting problem since the training accuracy would much higher than final test performance and the comparison would be hard.

Q: 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? What parameters did you tune?

Algorithms and parameter tuning

I used multiple algorithms and conducted parameter tuning so that I can get a set of parameters which could make the performance best for this algorithm on the split test data set. I used F1 score as criteria for parameter tuning and precision/recall as final performance measure.

Logistic Regression
Parameters:
'C': [5, 10, 20, 40 , 80, 160, 320, 5000, 1000, 5000],
'penalty': ['l1', 'l2']
Best parameter: {'penalty': 'l1', 'C': 40}
Performance on tester.py: Precision: 0.60 Recall: 0.60

Decision Tree
Parameters:
'max_depth': [5, 8, 10, 12, 14, 15],
'min_samples_split': [2,4, 6, 8],
'min_samples_leaf': [1, 2]
Best parameter: {'min_samples_split': 6, 'max_depth': 5, 'min_samples_leaf': 1}
Performance on tester.py: Precision: 0.25 Recall: 0.20

Naïve Bays
Precision: 0.50 Recall: 0.40

Random Forest
'n_estimators': [100],
'max_depth': [5, 10],
'min_samples_split': [2,4, 6],
'min_samples_leaf': [1, 2, 4],
'n_jobs': [-1]
Best parameters:
{'min_samples_split': 2, 'n_estimators': 100, 'n_jobs': -1, 'max_depth': 10, 'min_samples_leaf': 1}
Precision: 1.00 Recall: 0.20

For comparison, I use precision/recall as performance measure

Algorithm	Precision(1)	Recall(1)
Logistic Regression	0.60	0.60
Decision Tree	0.25	0.20
Naïve Bayes	0.50	0.40
Random Forest	1.00	0.20

After I used the four algorithms and tuned the parameters to get the best performance, I would choose Logistic Regression as my final model. 0.6 Precision means that 60% of people are actually POI among all people who are predicted as POI. 0.6 Recall means that 60% of people are predicted as POI among all people who are actually POI.