Project report: Identifying Fraud from Enron Email

1) Project Goal and Dataset

The goal of this project is to find or identify people of interest (POI) from dataset by using machine learning algorithms.

Machine learning algorithms will investigate the features, including financial features and email features, of POI and NOPOI and determine which ones are POIs.

The dataset of this project contains 146 entries. 18 of them are marked as "POI". The others are "NOPOI". Each entry is a dictionary, which records the financial and email information of one. There are 21 features including 'POI' in each record. There are a lot of "NaN" in the dataset. The following table shows the numbers of "NaN" in each feature. (I did not include feature 'email_address' since it's not used in the algorithm).

feature	NaN numbers
bonus	64
deferral_payments	107
deferred_income	97
director_fees	129
exercised_stock_options	44
expenses	51
from_messages	60
from_poi_to_this_person	60
from_this_person_to_poi	60
loan_advances	142
long_term_incentive	80
other	53

restricted_stock	36
restricted_stock_deferred	128
salary	51
shared_receipt_with_poi	60
to_messages	60
total_payments	21
total_stock_value	20

For a typical entry, the record structure is like this:

{‘ ALLEN PHILLIP K’:{‘bonus’:4175000, ‘deferral_payments’:2869717,
‘deferred_income’:-3081055, ‘director_fees’: NaN, ‘email_address’:
‘phillip.allen@enron.com’, ‘exercised_stock_options’:1729541, ‘expenses’:
13868, ‘from_messages’:2195, ‘from_poi_to_this_person’:47,
‘from_this_person_to_poi’:65, ‘loan_advances’:NaN,
‘long_term_incentive’: 304805, ‘other’: 152, ‘poi’: FALSE,
‘restricted_stock’:126027, ‘restricted_stock_deferred’: -126027, ‘salary’:
201955, ‘shared_receipt_with_poi’: 1407, ‘to_messages’: 2902,
‘total_payments’: 4484442, ‘total_stock_value’:1729541}}

‘ALLEN PHILLIP K’ is the name of this person and the other keys are the features of this person. By investigating these features, machine learning algorithm can dig POIs from the dataset.

Except for persons in the dataset, there are also invalid entries. There is one entry called ‘TOTAL’, which is the sum of all the persons in this dataset. If machine learning algorithms need to work, this entry should be removed.

At first I thought the following people are outliers:

a, "LAVORATO JOHN J" whose bonus is much higher than other NOPOI;

b, “BELFER ROBERT", who has negative total_stock_value;

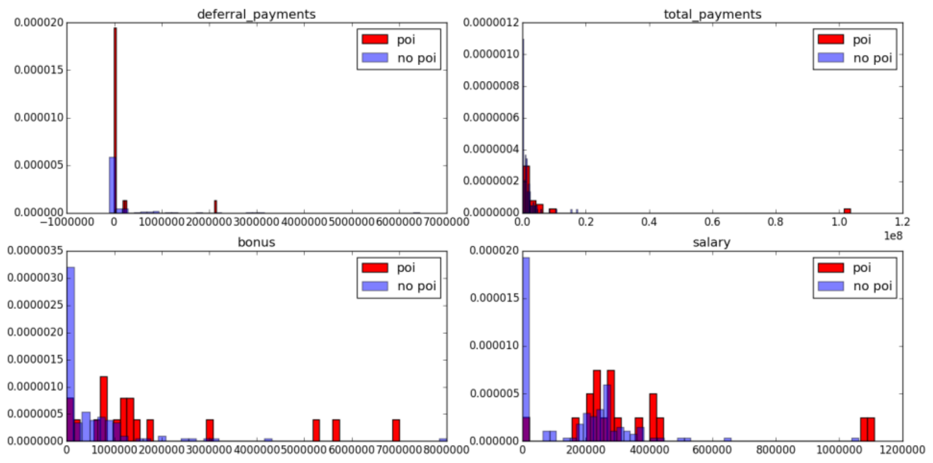
c, "FREVERT MARK A", whose salary is much higher than any other NOPOI

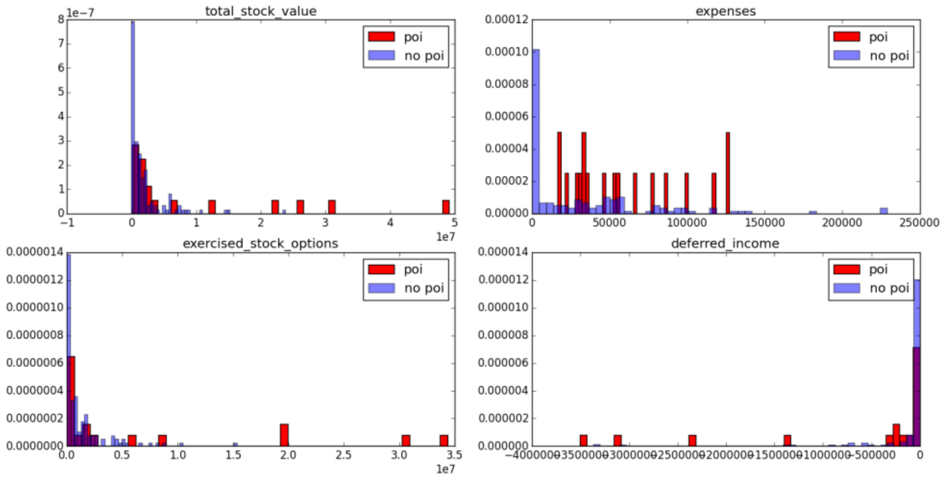
But after learning comments from grader, I think I should be more careful, since the dataset is tricky. After checking the information of "LAVORATO JOHN J" and "FREVERT MARK A", I do agree that they are real in the dataset. But “BELFER ROBERT"’s total_stock_value should be typo or a mistake. This makes the other information for this person not trustable. I’m going to consider this person to be an outlier and remove the entry.

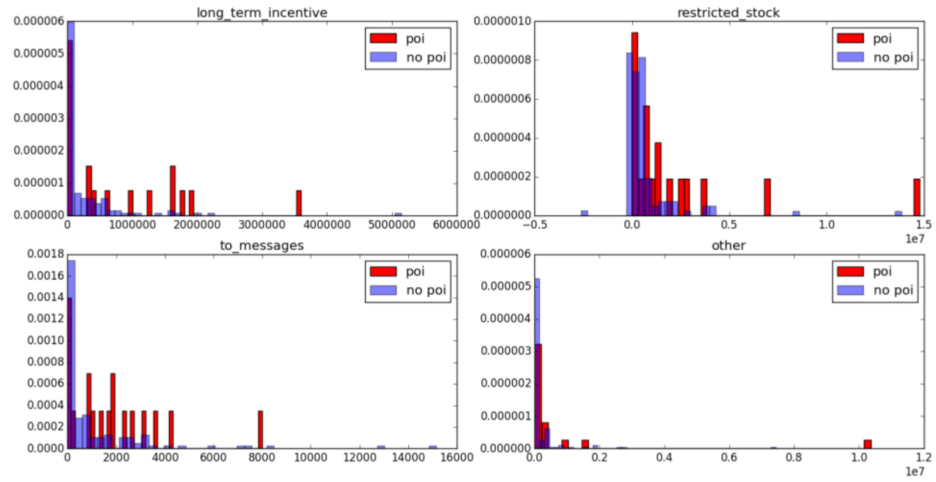
2) Feature Selection

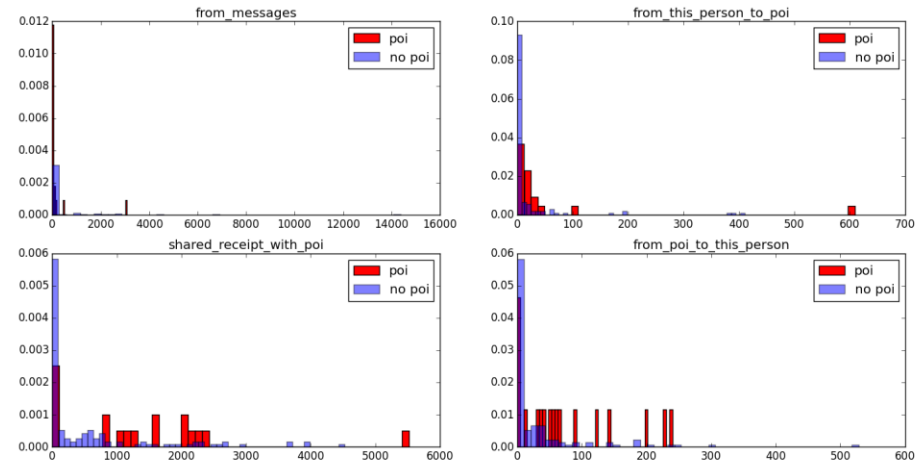
I used two method to select features. One is learned from previous projects. I plot histograms of each feature, with people divided into POI and NOPOI. By visualizing the relation between POI and each feature, I can have an idea if one feature is important in the machine learning process.

The following are the plots:




total_stock_value
poi
no poi
1e−7
1e7
expenses
poi
no poi
exercised_stock_options
poi
no poi
1e7
deferred_income
poi
no poi


long_term_incentive
poi
no poi
restricted_stock
poi
no poi
1e7
to_messages
poi
no poi
other
poi
no poi
1e7


from_messages
poi
no poi
from_this_person_to_poi
poi
no poi
shared_receipt_with_poi
poi
no poi
from_poi_to_this_person
poi
no poi

The 6 most important features I think show patterns are: 'bonus', 'salary', 'expenses', 'to_messages', 'shared_receipt_with_poi', and 'exercised_stock_options'.

This method is kind of subjective. To be more reasonable, I also tried another method which is learned from this project. I used SelectPercentile(Of course we can also use SelectKBest) to help me find the most important features.

feature	score
'bonus'	12.15709286117505
'salary'	9.9303714270637933
'shared_receipt_with_poi'	7.894448934887321
'deferred_income'	5.7767545331486465
'exercised_stock_options'	4.2076459245218478
'total_stock_value'	4.0796810045078313
'expenses'	3.7579758271831878
'from_poi_to_this_person'	3.0239991777153379
'total_payments',	2.6021910283844236
'from_this_person_to_poi'	2.5885235915555658
'long_term_incentive'	1.947174355759907
'restricted_stock'	1.7919403654424106
'director_fees'	1.4847627104859711
'to_messages'	1.2325183574802649
'restricted_stock_deferred'	0.98364731777469339
'loan_advances'	0.18799289520426291

'other'	0.17075999147185841
'from_messages'	0.075380745648050154
'deferral_payments'	0.0013171731874316078

If I use Decision tree to find the importance of the features, I would get:

feature	score
exercised_stock_options',	0.19999999999999976
'bonus'	0.16236158094081607
'other'	0.1388529689505914
'restricted_stock'	0.12048043256659144
'total_payments'	0.11287477954144617
'from_this_person_to_poi'	0.087189204011633867
'expenses'	0.064965420713307881
'shared_receipt_with_poi'	0.05772005772005772
'from_messages'	0.055555555555555552
'salary'	0.0
'deferral_payments'	0.0
'loan_advances'	0.0
'restricted_stock_deferred’	0.0
'deferred_income'	0.0
'total_stock_value'	0.0
'long_term_incentive'	0.0
'director_fees'	0.0
'to_messages'	0.0

'from_poi_to_this_person'   0.0

The importance of features calculated by the above two methods are not exactly the same. Next I'm going to test how many features I should use.

Now, I need to decide how many features I should use to do the training and testing. I used Naïve Bayes algorithm to help decide how many features I should use. The following score shows the relation between feature number and performance.

number of features	precision	recall
1	0.57980	0.17800
2	0.37692	0.1715
3	0.33723	0.173
4	0.46915	0.2965
5	0.46007	0.3485
6	0.45167	0.3575
7	0.47291	0.371
8	0.44202	0.3545
9	0.42183	0.313
10	0.36697	0.3055
11	0.33776	0.305
12	0.32978	0.3095

(If number of features is larger than 11, the precision value would become very poor.)

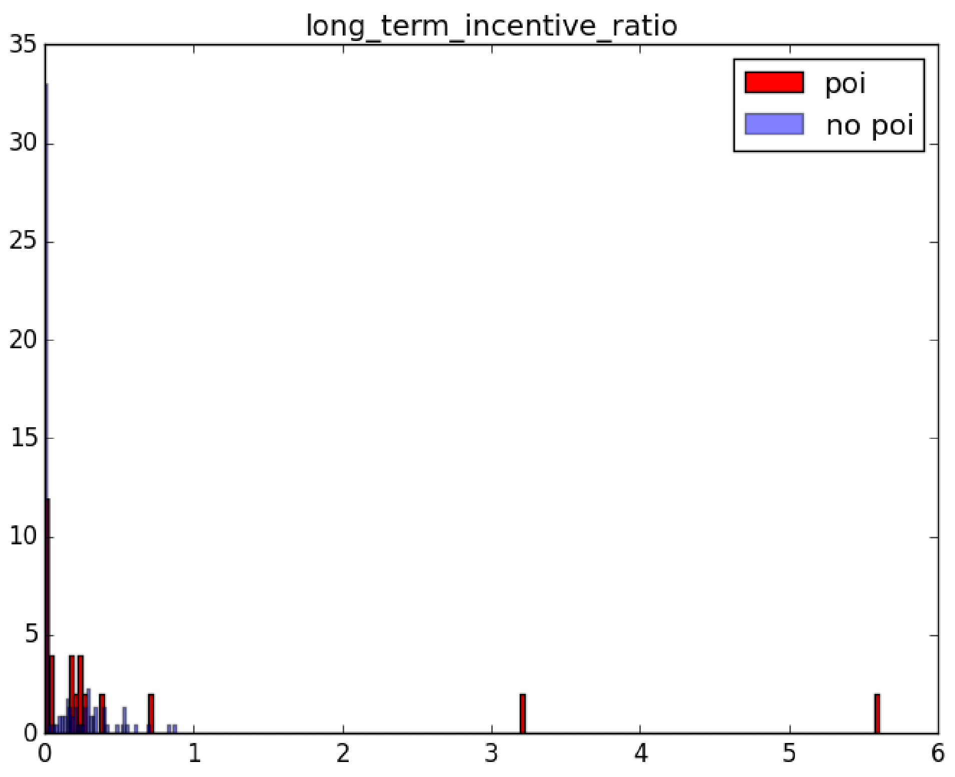
We can see if I choose 7 features, I could get the best performance. So the features I select for Naïve Bayes are ['bonus', 'salary', 'shared_receipt_with_poi', 'deferred_income', 'exercised_stock_options', 'total_stock_value', 'expenses'].

For Decision Tree, I got the following result:

number of features	precision	recall
1	0.31955	0.34
2	0.3694	0.408
3	0.29145	0.2575
4	0.29626	0.293
5	0.2584	0.2385
6	0.23686	0.2365
7	0.26322	0.2415
8	0.24646	0.235
9	0.25682	0.2305
10	0.25946	0.2435
11	0.24947	0.236

So the features I select for Decision Tree are ['exercised_stock_options', 'bonus']

In the lecture, instructor was using the ratio of 'from_poi_to_this_person' / 'to_messages' and ratio of 'from_this_perston_to_poi' / 'from_messages' to show how to create new features. I want to try something new. The feature I created is called 'long_term_centive_ratio'. Its defination is 'long_term_centive' divided by 'total_payments'.



The above is the plot of that created feature. It seems that for POIs, they tend to have higher long_term_incentive_ratio. But such a correlation is not very obvious. I need to use this feature to do predictions and then decide whether it makes prediction better.

Before adding this feature, I got the following result using the Naïve Bayes classifier provided by the code.

Precision: 0.47291 Recall: 0.371

After introducing this feature, I got the following result:

Precision: 0.51884 Recall: 0.35800

Comparing these two results, I found that the precision has increased by 0.046 larger than the decrement of recall 0.013, although precision and recall are more balanced without the new features.

I also tested Decision Tree without tuning.

Before adding new feature:

```
Precision: 0.37204 Recall: 0.41650
```

After adding new feature:

```
Precision: 0.40950 Recall: 0.43550
```

The performance is better with the new feature. So I think I would use this new feature.

So the feature list for Naïve Bayes is: ['bonus', 'salary', 'shared_receipt_with_poi', 'deferred_income', 'exercised_stock_options', 'total_stock_value', 'expenses', 'long_term_incentive_ratio'].

The feature list for Decision tree is: ['exercised_stock_options', 'bonus', 'long_term_incentive_ratio'].

For the features, there is no need to rescale them for GaussianNB() classifier or DecisionTreeClassifier() which I used in the code.

The next stage is to find the best algorithm.

3) Algorithm Selection

I tried the Naïve Bayes, SVM and Decision Tree algorithms.

The SVM gave me back error. I could not use it.

For Naïve Bayes, the result is:

Precision: 0.51884 Recall: 0.35800

For Decision Tree without tuning, the result is:

Precision: 0.40950 Recall: 0.43550

It seems Naïve Bayes gives much better precision result. Decision Tree gives much better recall value. And I haven't tuned Decision Tree yet. I cannot decide which algorithm to use. I like Decision better because the result is more balanced. The next stage is to tune Decision Tree algorithm. If I can make Decision Tree even better, I would like to use it to be the final algorithm.

4) Algorithm Tuning

The purpose of tuning is to find best sets of parameters of the algorithm. Better parameters means the algorithm can give more accurate prediction. If the parameters are not tuned well, even a good algorithm can produce bad prediction. So finding right parameter is a critical step.

The parameters I'm going to tune are:

Parameter	Values to test
n_components	[None, 1, 2]

min_samples_split	[2, 4, 6, 8]
max_depth	[None, 5, 10, 15]

I scanned the parameters. (Code in the .py file could be uncommented to run, but it will take a few minutes to finish). And I found the following parameters gives me the best result.

n_components = 2, min_samples_split = 8,max_depth = 5

And it gives the following result:

Precision: 0.54238 Recall: 0.42550

Comparing this result to the result got by Naïve Bayes, Decision Tree is much better. I would choose Decision Tree to be my final algorithm since it provides more balanced precision and recall. And the final feature list is: features_list = ['exercised_stock_options', 'bonus', 'long_term_incentive_ratio'].

5) Validation

One common mistake in train and test algorithm is to use one dataset for training and then use the same dataset for testing, as mentioned in sklearn document:

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **overfitting**.

To avoid this problem, we hold out part of the available data as a test set. Training dataset is used to train the algorithm and testing dataset is used to test the performance of the algorithm. Such a process is called “validation”. In the code, an iterator called StratifiedShuffleSplit was used to do validation. This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. I chose this cross validation because it the entries in the dataset are fully randomly split into training set and testing set. In such a case, our algorithm would be scientifically evaluated.

6) Evaluation

The result from my tuned classifier is:

```
Precision: 0.54238 Recall: 0.42550
```

```
Total predictions: 12000  True positives:  851        False positives:  718
False negatives: 1149      True negatives: 9282
```

Both precision and recall are larger than 0.3.

Precision is the percentage of true POIs if one is identified as POI. In this context, precision = 851 / (851 + 718) = 0.54238.

Recall is the percentage of POIs being identified from all POIs. In this context, recall = 851 / (851 + 1149) = 0.42550.