Enron POI Identifier Report

Yu, Bing
yu.bing@ge.com

1. Project Introduction

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.

The project goal is to use financial and email data which has been made public as a result of the Enron scandal to identify the persons of interest (POI) in the fraud case, which means individuals who were indicted, reached a settlement, or plea deal with the government, or testified in exchange for prosecution immunity

Using machine learning would help to develop a prediction function to identify POIs based on given features & data points. This report documents the machine learning techniques I used in building a POI identifier.

2. The Enron Data Overview

For the dataset, it has combined the Enron email and financial data in to a dictionary. The financial data comes from the enron61702insiderpay.pdf source, and the email data comes from Enron email corpus. In the combined dictionary, each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name and the value is another dictionary, which contains the names of all the features and their values for that person.

2.1 Data Exploring

In the dataset, there are 21 features, which fall into three major types of features, namely financial features, email features and POI labels.

- financial features (all units are in US dollars):
  ['salary','deferral_payments','total_payments',
  'loan_advances','bonus','restricted_stock_deferred','deferred_income','total_stock_value','expenses','exercised_stock_options', 'other', 'long_term_incentive',

'restricted_stock', 'director_fees']

- email features (units are generally number of emails messages; notable exception is ‘email_address’, which is a text string):
  ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', 'shared_receipt_with_poi']

- POI label(boolean, represented as integer):
  [‘poi’]

Other characteristics for this dataset:

- Total number of data points: 146
- Total number of features: 21
- Total number of POI: 18
- Total number of non-POI: 128
- Number of people have a known email: 111
- Number of people have a quantified salary: 95
- Number of people have a quantified exercised_stock_options: 102

2.2 Outlier Investigation

The first step to investigate in this dataset is to identify the outliers, especially in the financial data. Select the features of ‘salary ’ and ‘bonus’ as a starting point. From the scatter plot of Figure 2-1, it is easy to identify one outlier with extreme large value while comparing with other points. By going through the source data, the key-value is ‘TOTAL’, which is the summary line in the financial datasheet. This is not a valid data point for this analysis, and this outlier was removed by hand before proceeding.

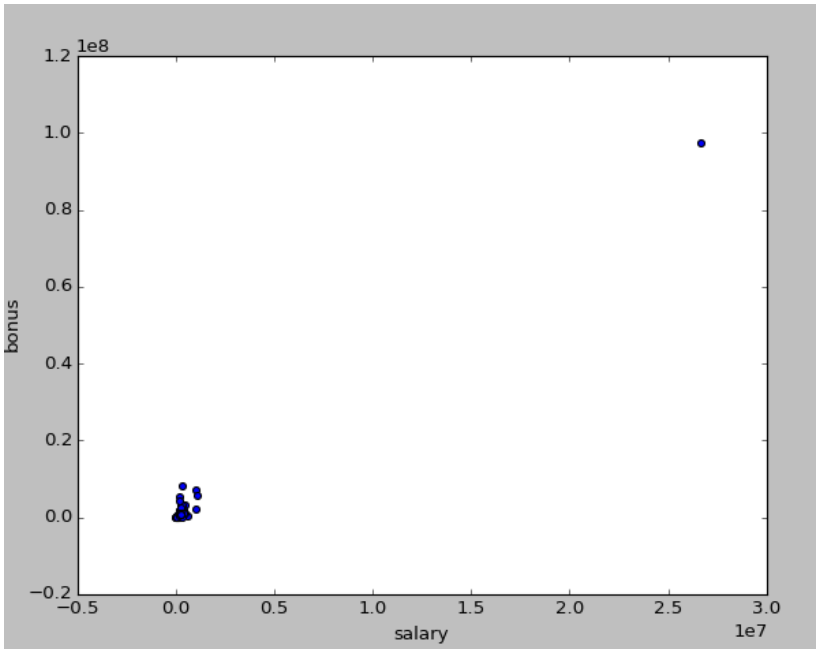

Figure 2-1 Scatter plot for bonus and salary for outlier analysis

After removing the outlier from the dataset, the updated scatter plot of Figure 2-2 still shows three more financial outliers. Those are the executive leaders for Enron, who have relatively higher salary & bonus. Those are valid data points, and it is entangled with the corporate fraud. Thus I kept those points for future analysis.

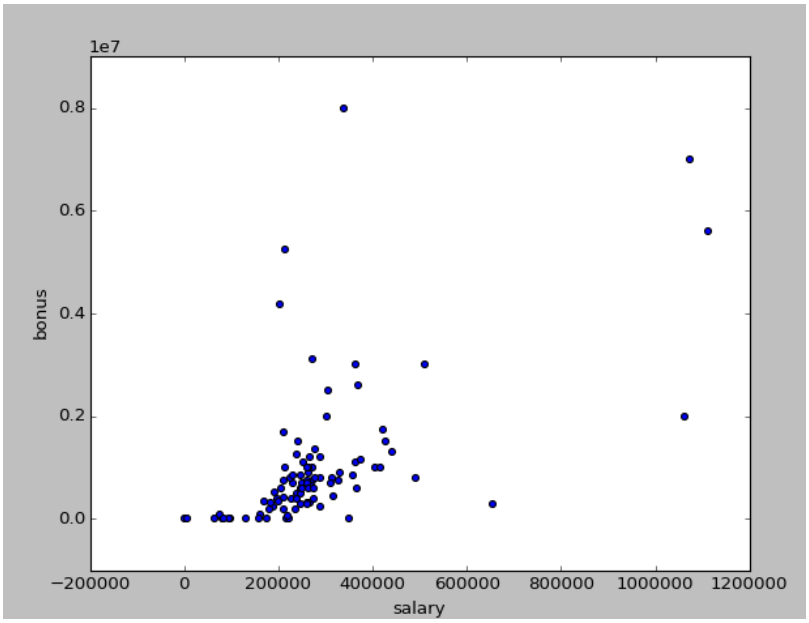

Figure 2-2 Scatter plot for bonus and salary after removing outliers

3. Feature Selection and Processing

The data set has 21 features in total, I need to analyze the features and select the proper features for building the identifier.

3.1 New features

In the email features, there are features of 'from_poi_to_this_person' and 'from_this_person_to_poi', which indicate the number of emails from and send to the person of interest. The hypothesis is that there might be stronger email connections between POIs than between POIs and non-POIs. And the scatterplot of these two features of Figure 3-1 suggests that there might be some truth to that hypothesis (red points indicate poi, while blue points indicate the non-POIs).It is using the absolute value of the number of emails, since the number of total messages is different, it makes more sense to use the fraction.

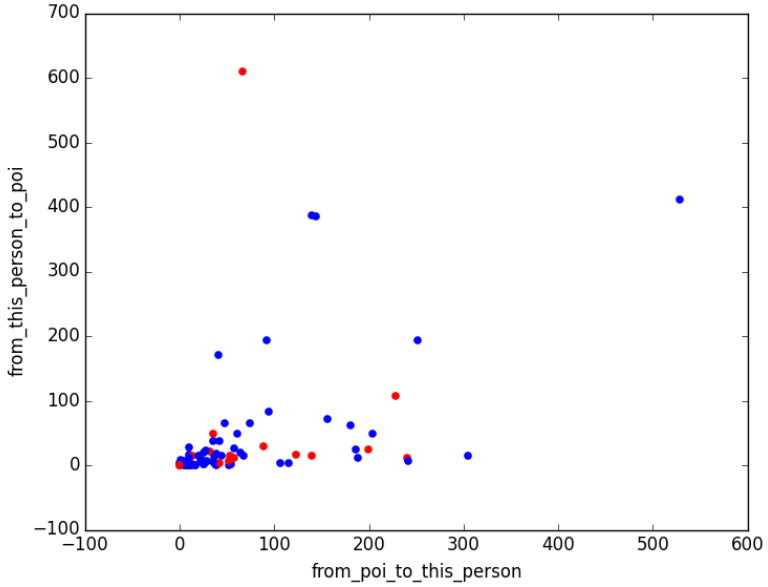

Figure 3-1 Scatter plot for messages from poi to the person and from this person to poi

I created two new features:

- fraction_from_poi – the fraction of all emails to a person that were sent from a person of interest
- fraction_to_poi – the fraction of all emails that a person sent that were addressed to persons of interest

From the scatterplot of these two features in Figure 3-2, it can be found that the points of POI formed a cluster, which will be helpful to be used for identify POIs

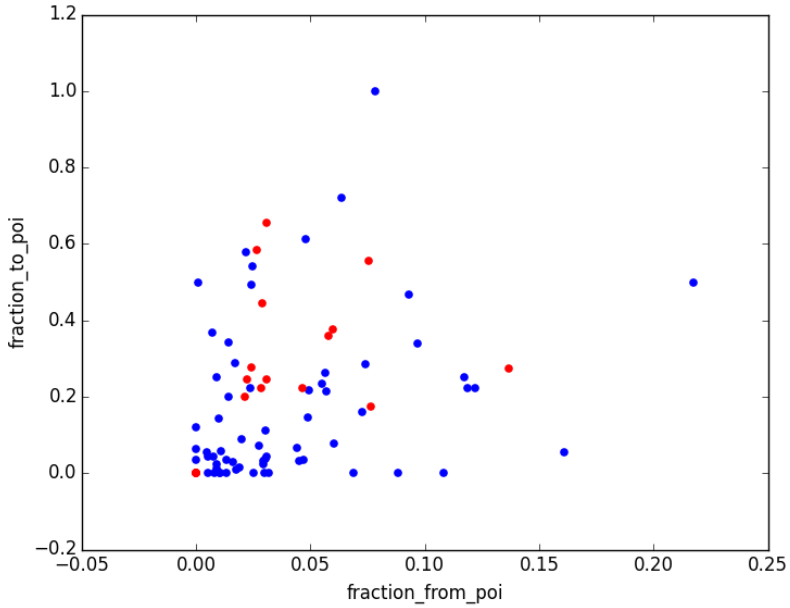

Figure 3-2 Scatter plot for the fractions of messages from poi to the person and from this person to poi

3.2 Feature Selection

In order to select the proper features, I firstly ranked all of the features through univariate feature selection. And then use Naïve Bayes classifier to test the right number of features to be used.

Univariate feature selection works by selecting the best features based on univariate statistical tests. Use Univariate feature selection - SelectPercentile to get the score for each feature, and use f_classif, which is ANOVA F-value between labe/feature for classification tasks.

Features sorted by score:
[('exercised_stock_options', 25.097541528735491),
('total_stock_value', 24.467654047526398),
('bonus', 21.060001707536571),
('salary', 18.575703268041785),
('fraction_to_poi', 16.641707070468989),
('deferred_income', 11.595547659730601),
('long_term_incentive', 10.072454529369441),
('restricted_stock', 9.3467007910514877),
('total_payments', 8.8667215371077717),
('shared_receipt_with_poi', 8.7464855321290802),
('loan_advances', 7.2427303965360181),
('expenses', 6.2342011405067401),
('from_poi_to_this_person', 5.3449415231473374),
('other', 4.204970858301416),
('fraction_from_poi', 3.2107619169667441),
('from_this_person_to_poi', 2.4265081272428781),
('director_fees', 2.1076559432760908),
('to_messages', 1.6988243485808501),
('deferral_payments', 0.2170589303395084),
('from_messages', 0.16416449823428736),
('restricted_stock_deferred', 0.06498431172371151)]

Based on the feature importance, use Naïve Bayes to test the right number of features for the final classifier. The detailed test results have been summarized in the Table 3-1 below, using the validation & evaluation metrics from the tester function, for the detailed explanation, please check the section of Validation Analysis and Performance Evaluation in later part.

From the evaluation summary, it could be seen that when using Top 7 features, it had the largest F1 score, which did give better performance in both precision and call, the final features_list is shown as below:

features_list = ['poi', 'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'fraction_to_poi', 'deferred_income', 'long_term_incentive']
The new created feature 'fraction_to_poi' is used in the final version of feature list as it did help to improve the classifier precisions & recall.

# of features	Accuracy	Precision	Recall	F1	F2
Top 1	0.83418	0.60329	0.25700	0.36045	0.29033
Top 2	0.84069	0.46889	0.26750	0.34066	0.29264
Top 3	0.84300	0.48581	0.35100	0.40755	0.37163
Top 4	0.84677	0.50312	0.32300	0.39342	0.34791
Top 5	0.84879	0.45558	0.30000	0.36177	0.32199
Top 6	0.85236	0.47723	0.35100	0.40449	0.37061
Top 7	**0.84907**	**0.46376**	**0.36100**	**0.40629**	**0.37818**
Top 8	0.84650	0.45370	0.36500	0.40454	0.37985
Top 9	0.84427	0.39434	0.31350	0.34930	0.32690
Top 10	0.83973	0.37743	0.31100	0.34101	0.32235
Top 11	0.77867	0.24654	0.32100	0.27889	0.30272

Table 3-1 Algorithm performance results for applying Naïve Bayes to different number of features

3.3 Feature Scaling

For further analysis of using different algorithm, feature scaling might be needed. For Naïve Bayes and Decision Tree algorithm, feature scaling is not required. When trying SVM algorithm, using MinMaxScaler from preprocessing package to rescale the features, so that all of the features would be in range of [0, 1].
X_std = (X - X.min) / (X.max - X.min)
X_scaled = X_std * (max - min) + min

The feature scaling is deployed in the function of **test_classifier_withscale**, which can be found in the poi.py

4. Algorithm Selection and Tuning

From previous section, I have decided to use the seven features of 'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'fraction_to_poi', 'deferred_income', 'long_term_incentive' for the classifier, and I tried three different algorithm: Naïve Bayes(NB), Decision Tree(DT) and Support Vector Machine(SVM), by using shuffle split for cross validation to evaluate the algorithm performance (details could be found in section 5 & 6), the results is summarized in table 4-1.

Algorithm	Parameter	Accuracy	Precision	Recall	F1	F2
NB	-	0.84907	0.46376	0.36100	0.40629	0.37818
DT	min_samples_split=2	0.80136	0.29129	0.27250	0.28158	0.27606
DT	min_samples_split=3	0.80643	0.29312	0.25150	0.27072	0.25885
DT	min_samples_split=4	0.80121	0.27461	0.23850	0.25528	0.24494
DT	min_samples_split=5	0.80436	0.28175	0.23850	0.25833	0.24605
SVM	kernel='rbf', C=1, gamma=0.0	0.85714	-	-	-	-

Table 4-1 Algorithm performance results based on seven selected features

For Decision Tree algorithm, the parameter min_samples_split is tuned by hand. Min_samples_split is the minimum number of samples required to split an internal node. Tuning this parameter could help to avoid the issue of over fitting which creates over-complex trees that do not generalize the data well. From Table 4-1, DT performed the best when setting min_samples_split to 2.

When trying SVM, the predictions showed high biased: all test data was labeled 0 as non-POI. The detailed results are shown as below. Although the accuracy is high, it had zero precision and recall. When trying different value for parameter of C & gamma, the results remain the same. When the number of features is larger, SVM gives poor performance. The result is impacted by feature noise.

Total predictions: 14000 True positives: 0 False positives: 0
False negatives: 2000 True negatives: 12000

Since Naïve Bayes have the best precision and recall score, Naïve Bayes was selected as the final classifier.

In order to understand more of the different algorithm, I tried these three algorithms over one feature - exercised_stock_options , which has the most impact score on the results from section 3. The evaluation results has been summarized in table 4-2

Algorithm	Parameter	Accuracy	Precision	Recall	F1	F2
NB	-	0.83418	0.60329	0.25700	0.36045	0.29033
DT	min_samples_split=2	0.84600	0.57727	0.57150	0.57437	0.57265
DT	min_samples_split=3	0.84900	0.58706	0.57150	0.57917	0.57455
DT	min_samples_split=4	0.85209	0.59749	0.57150	0.58421	0.57652
DT	**min_samples_split=5**	**0.85273**	**0.59948**	**0.57250**	**0.58568**	**0.57770**
SVM	kernel='rbf', C=1, gamma=0.0	0.87327	0.88846	0.34650	0.49856	0.39465
SVM	kernel='rbf', C=10, gamma=0.0	0.87327	0.88846	0.34650	0.49856	0.39465
SVM	kernel='rbf', C=100, gamma=0.0	0.87327	0.88846	0.34650	0.49856	0.39465

Table 4-2 Algorithm performance results based on exercised_stock_options

Both Decision Tree & SVM showed better performance with one feature. And Decision Tree had the best performance when min_samples_split is 5. For SVM, I tried different value for parameter C, C controls tradeoff between smooth decision boundary and classifying training points correctly. A large C means you will get more training points correct. But in this case the results remain the same.

In summary, when using one feature to build the classifier, Decision Tree with min_samples_split =5 has a better performance, this algorithm could be used when we have limited information. For this dataset, exercised_stock_options only have 102 values, it would be challenging in real world to just use one feature to identify POI. When using multiple features to build the classifier, Naïve Bayes has a better performance with seven selected features. In order to fully utilize the dataset and explore more characteristics of the POI, I used Naïve Bayes on seven features -'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'fraction_to_poi', 'deferred_income', 'long_term_incentive' as the final classifier.

5. Validation Analysis

If we are using the same data to train the prediction function and test it, the model would just repeat the labels of the samples, that would have a perfect score but would fail to predict anything useful on new dataset. This is a classical methodological mistake. To avoid this mistake, it is common to use cross validation, which will hold out part of the available data as a test set.

The problem with splitting dataset in to training data & testing data is that you want to maximize the test data to have a better validation, at the same time want to maximize the training data to have best learning results. In this case, the Enron dataset is not balanced, there are only 18 POIs out of 145 samples after removing the outlier. I tried to use SelectPercentile to select 50% of the data as training data, and the remaining as testing data. The data volume is too small for an effective training. Using K-fold method with setting K as 3 will improve the validation a little, for each training cycle, I used two thirds of the data for training and one third for testing, and got the results for three times. The K-fold validation was deployed through **test_classifier_Kfold** function.

In order to get a better validation results from a larger data points, I utilized the shuffle split cross validation from the tester_classifier function.
The ShuffleSplit iterator will generate a user defined number of independent train / test dataset splits. Samples are first shuffled and then split into a pair of train and test sets. Setting the folds as 100, I had 13000 predictions in total to validate the results. And it is possible to control the randomness for reproducibility of the results by explicitly seeding the random_state pseudo random number generator, in this case it was set as 42. The results from table 3-1, 4-1 and 4-2 are all based on shuffle

split cross validation.

6. Performance Evaluation

Accuracy is the ration of no of items in a class labeled correctly vs all items in that class. It could not show the error on side of guessing innocent or guilty. So precision and recall are used to evaluate algorithm performance.

The precision is the ratio of true positive / (true positives + false positives) .The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. If the algorithm has a good precision score, it means that whenever a POI gets flagged in my test set, I know with a lot of confidence that it's very likely to be a real POI and not a false alarm.

The recall is the ratio of true positives/(true positives + false negatives). The recall is intuitively the ability of the classifier to find all the positive samples. If the algorithm has good recall, that means that, nearly every time a POI shows up in my test set, I am able to identify him or her.

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is: F1 = 2 * (precision * recall) / (precision + recall). If the algorithm has a really great F1 score, this is the best of both worlds. Both my false positive and false negative rates are low, which means that I can identify POI's reliably and accurately. If my identifier finds a POI then the person is almost certainly a POI, and if the identifier does not flag someone, then they are almost certainly not a POI

7. Conclusions

Based on previous analysis, I choose Naïve Bayes to build the final classifier, and used seven features -'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'fraction_to_poi', 'deferred_income', 'long_term_incentive' as the final classifier. After shuffle split cross validation, the classifier had precision score of 0.46, and recall score of 0.36. It means using this identifier to flag POI's, 46% of the POIs flagged are real POIs, while 54% are false alarm. And given all a POI in the data set, 36% of the time the identifier would catch that person and flag POI, and 67% of the time it would miss him or her.

The result is acceptable for analysis, but far more from good identifier to be used in

real life. If either precision or recall were close to 1.0, then it would be doing better in identify POI. When using Decision Tree with one feature exercised_stock_options, the identifier had a precision score of 0.60 and recall of 0.57 which could perform better in identifying POIs. But it only utilized one feature, which did not reflect the full characteristics of the POI.

The challenge of analysis this dataset is imbalance. There are only 18 POIs in data set, while in real world there should be 35. And the combined financial & email information is imbalanced, not all of the persons have full information about their financial status and email messages.

One thought to improve the identifier is to have a larger data set. It would require lots of work to collect more financial information & emails. That might not be possible with limited resources. The other thought is to explore more on the email information. The current features only use the number of messages related to POI not with the content of the email messages. Applying machine learning to study the content of the email might help to build a better identifier.