Identify Fraud from Enron Email and Financial Data

Chun Zhu

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 the public record, including tens of thousands of emails and detailed financial data for top executives.

In this project, I will play detective, and use machine learning skills to build an algorithm to identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset.

Short Questions

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?

The goal of the project is to build a predictive, analytic model to identify persons who may have committed fraud based on the related financial and email dataset. The dataset already includes the label 'poi' to distinguish culpable persons: 1 for person of interest (POI), 0 for non-POI. In general, the dataset contains 146 data points with 14 financial features, 6 email features and 1 labeled feature (POI). The missing values are checked for each feature, just as the table 1 shows in the next page. It is clear that the labeled feature 'poi' has no missing values, while feature 'loan_advances' has only 4 valid values.

For outlier investigation, I plotted the scatter plot of 'salary' and 'bonus'. In Fig 1, it is clear that there is an outlier which is not labeled as POI. This outlier is '`TOTAL`'. By looking at all the names, there is one name which seems not an individual – '`THE TRAVEL AGENCY IN THE PARK'.` One more outlier is '`LOCKHART EUGENE E`' with all feature values missing. So I deleted these 3 outlies and 143 data points remained.

Feature	**missing number**
loan_advances	142
director_fees	129
restricted_stock_deferred	128
deferral_payments	107
deferred_income	97
long_term_incentive	80
bonus	64
from_this_person_to_poi	60
from_poi_to_this_person	60
from_messages	60
shared_receipt_with_poi	60
to_messages	60
other	53
expenses	51
salary	51
exercised_stock_options	44
restricted_stock	36
email_address	35
total_payments	21
total_stock_value	20
poi	0

Table 1. The number of missing values for features in enron dataset

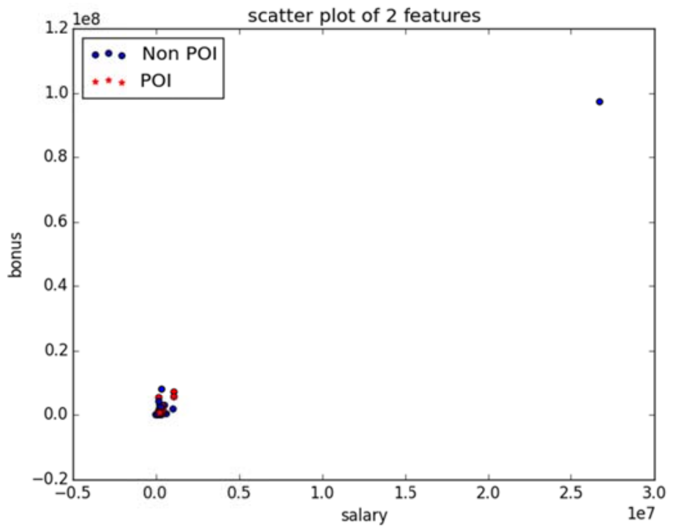

Fig 1. Scatter plot of salary and bonus

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. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importance of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.

In the feature selection process, I used SelectKBest module to automatically select 10 best features based on the feature scores. Table 2 shows the scores for related features and the fraction of valid entries for both POI and non-POI subsets.

Feature	**Score**	**fraction of valid data points for POI**	**fraction of valid data points for non-POI**
exercised_stock_options	24.81	0.67	0.71
total_stock_value	24.18	1	0.86
bonus	20.79	0.89	0.52
salary	18.28	0.94	0.62
deferred_income	11.45	0.61	0.3
long_term_incentive	9.92	0.67	0.42
restricted_stock	9.21	0.94	0.74
total_payments	8.77	1	0.84
shared_receipt_with_poi	8.58	0.78	0.58
loan_advances	7.18	0.06	0.02
expenses	6.09	1	0.61
from_poi_to_this_person	5.24	0.78	0.58
other	4.18	1	0.58
from_this_person_to_poi	2.38	0.78	0.58
director_fees	2.12	0	0.13
to_messages	1.64	0.78	0.58
deferral_payments	0.22	0.28	0.26
from_messages	0.16	0.78	0.58
restricted_stock_deferred	0.06	0	0.14

Table 2. Feature Scores by SelectKBest. The highlighted features in blue are the ones selected. The features marked in red should not be included in the final analysis.

The red rows in Table 2 mark the features not suitable for the learning algorithms based on the fraction of valid data points. For example, `total_stock_value`, `total_payments`, `expenses` and `other` are valid for all POIs, but they are only valid for less than 100% non-POIs. This could lead to the classifier simply using the existence (or non-existence) of those features on a testing point to predict if it was a POI, instead of using the values of those features. As a result, those features were removed from the dataset. However, `total_stock_value` and `total_payments` have very high score. To compensate the loss of information containing in these 2 features, I engineered a new feature `financial_aggregate` - the sum of `exercised_stock_options`, `total_payments`, and `total_stock_value` - to capture how much wealth an individual has. I also noticed that the features with 10 highest scores are almost all financial features. And I wish to add some more email features. In my opinion, there should be stronger email connections between POIs than between POIs and non-POIs. So I engineered 4 new features:

- ‘`financial_aggregate`’: The sum of `exercised_stock_options`, `total_payments`, and `total_stock_value` to capture how much wealth an individual has.
- ‘`poi_ratio`’: The ratio of the total number of emails from and to POIs to the total number of emails sent and received for a person.
- ‘`fraction_from_poi`’: The fraction of all emails to a person that were sent from POIs
- ‘`fraction_to_poi`’: The fraction of all emails that were sent from a person which were addressed to POIs

The scatter plot between `fraction_from_poi` and `fraction_to_poi` in Fig 2 suggests that there is a pattern for POIs.

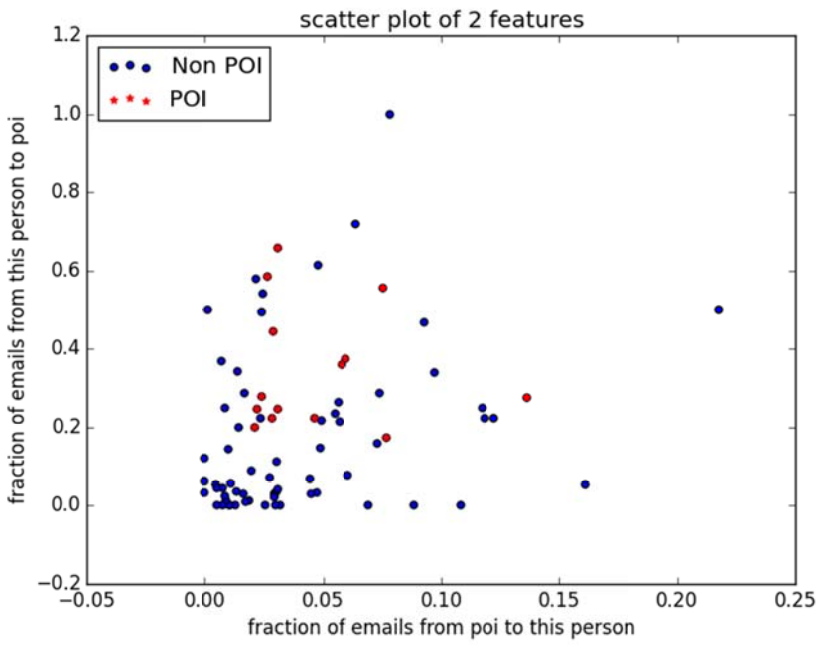

Fig 2. Scatter plot of `fraction_from_poi` and `fraction_to_poi`

These 4 new features were included in the final analysis, as they increased the precision and recall for most of the machine learning algorithms tested. The table on the next page shows the relevant scores of the features in the final analysis by SelectKBest.

Feature	**Score**
exercised_stock_options	23.96
bonus	19.98
financial_aggregate	18.42
salary	17.43
fraction_to_poi	15.71
deferred_income	11.04
long_term_incentive	9.472
restricted_stock	8.812
shared_receipt_with_poi	8.119
loan_advances	7.008
poi_ratio	5.044
fraction_from_poi	2.882

Table 3. Feature Scores by SelectKBest.

Since different features have very different data range in the dataset, I scaled all the features using a min-max scaler prior to training the classifier. Feature scaling ensures that all the features are weighted evenly for the classifiers.

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? 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?

I tried Logistic Regression, K-means clustering and Adaboost. And I ended up using Adaboost, because Adaboost results in both precision and recall better than 0.3.

Tuning the parameters of an algorithm refers to changing the algorithm's input parameters (usually across a range of values) and measuring the performance of each combination of them in order to determine the optimal set of input parameters. If parameter tuning is not done well, the performance of the algorithm could potentially be improved, because the "best" parameters for the problem were not used, or at least a better combination existed.

I tuned the following parameters for all these 3 algorithms by GridSearchCV module and used a stratified shuffle split on a thousand fold:

- Logistic regression: C , tol and class_weight
- Adaboost: n_estimators, learning_rate and algorithm
- K-means clustering: n_init and tol (n_clusters is always 2)

The best estimator for each classifier gives results in the following table:

Classifier	Precision	Recall	F1 score
Logistic Regression	0.29	0.65	0.40
K-means Clustering, K=2	0.16	0.53	0.24
Adaboost	0.36	0.31	0.34

Table 4. Precision and Recall for different classifiers with best tuned parameters.

From the table we can tell the best classifier is Adaboost, which gives both precision and recall better than 0.3.

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

Validation refers to the process of training and testing a machine learning algorithm in order to assess its performance, and to prevent overfitting. Over-fitting means the model is trained and performs very well on the training dataset, but markedly worse on the cross-validation and test datasets.

Aside from overfitting, another classic mistake is to not shuffle and split the testing and training sets adequately. If there are patterns of inserting data points into the dataset, the algorithm could wind up in training data from one class, but testing data from another, leading to very poor performance.

For this effort, I used both the tester function provided by Udacity and the GridSearchCV function with StraitifiedShuffleSplit of 1000 folds to validate the analysis as well as to identify the optimal combinations of parameters.

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.

I used 3 metrics to evaluate the performance of the algorithm: precision, recall and F1 score. Precision captures the ratio of true positives to the records that are actually POIs. For this case, a high precision value means POIs identified by the algorithm tended to be correct, while a low value means there were more false alarms, where non-POIs were flagged as POIs. Recall captures the ratio of true positives to the records flagged as POIs, which describes sensitivity. In this case, a high recall value means that the algorithm is able to detect more POIs from the dataset, while a low value mean there are more POIs not detected by the algorithm. F1 score is a weighted average of precision and recall, where an F1 score reaches its best value at 1 and worst at 0.

In this project, both precision and recall have to be at least 0.3. Adaboost achieved this goal.

```
Pipeline(clf=AdaBoostClassifier(algorithm=SAMME.R,
        base_estimator=DecisionTreeClassifier(compute_importances=None, criterion=gini, max_depth=8,
          max_features=None, min_density=None, min_samples_leaf=1,
          min_samples_split=2, random_state=None, splitter=best),
        base_est...one, base_estimator__splitter=best,
        learning_rate=2.5, n_estimators=40, random_state=None),
    clf__algorithm=SAMME.R,
    clf__base_estimator=DecisionTreeClassifier(compute_importances=None, criterion=gini, max_depth=8,
          max_features=None, min_density=None, min_samples_leaf=1,
          min_samples_split=2, random_state=None, splitter=best),
    clf__base_estimator__compute_importances=None,
    clf__base_estimator__criterion=gini, clf__base_estimator__max_depth=8,
    clf__base_estimator__max_features=None,
    clf__base_estimator__min_density=None,
    clf__base_estimator__min_samples_leaf=1,
    clf__base_estimator__min_samples_split=2,
    clf__base_estimator__random_state=None,
    clf__base_estimator__splitter=best, clf__learning_rate=2.5,
    clf__n_estimators=40, clf__random_state=None,
    scale=MinMaxScaler(copy=True, feature_range=(0, 1)), scale__copy=True,
    scale__feature_range=(0, 1))
        Accuracy: 0.82300  Precision: 0.36201  Recall: 0.31350   F1: 0.33601   F2: 0.32213
        Total predictions: 14000   True positives:  627 False positives: 1105   False negatives: 1373   True negatives: 10895
```

Comments

Enron dataset is a very unbalanced dataset with only 18 POIs labeled out of 146 records. Most of the machine learning algorithms perform much better in balanced dataset.

References

- Udacity course, “Introduction to Machine Learning”
- Machine Learning (Stanford/Coursera)
- Scikit-learn Documentation