Identify POIs from Enron Fraud Emails

Machine Learning Models Application

Lin Zhu

Feb 07, 2016

I. Definitions

Project Overview

Enron Corporation was an American energy company based in Houston, Texas, who was named by Fortune as "America's Most Innovative Company" for six consecutive years. After its accounting scandal was revealed, Enron's stock price plummeted from over $90 to just pennies. A few days later, Enron filed for bankruptcy.

Problem Statement

This project is based on the Enron's public email and financial data from Federal investigation. The goal of this project is to build a person of interest ("poi") identifier, to correctly recognize employees that were directly involved in the fraud ("poi") and employees that were innocent ("non-poi")

II. Analysis

Data Exploration

Initial Exploration

This dataset contains information for 146 employees, with 21 features for each of them. Among them, 18 people were tagged with "poi" and the rest 128 are "non-poi". Regarding those features, "email_address", "from_messages", "to_messages", "from_poi_to_this_person" and "from_this_person_to_poi" are email relevant attributes, the rest are financial related features.

Outlier Identification and Correction

With initial observation, the "TOTAL" is obviously not an employee. I moved it. "THE TRAVEL AGENCY IN THE PARK" looks like an outside agent, which was labeled with "non-poi". However, considering it might contribute to the "non-poi" classification, I kept it.

For features "deferred_income" and "restricted_stock_deferred", all values should be negative. The rest financial features should have positive sign. An investigation was conducted. Affected people and features were shown below.

- "BELFER ROBERT"
    - "restricted_stock_deferred": should be negative
    - 'deferral_payments': should be positive

    - 'total_stock_value': should be positive
- "BHATNAGAR SANJAY"
    - "restricted_stock_deferred": should be negative
    - 'restricted_stock': should be positive

All those signs were corrected.

Missing Value

For all features, except "poi", all the other features have more or less missing values. A pot showed the missing value allocation between "poi" and "non-poi".

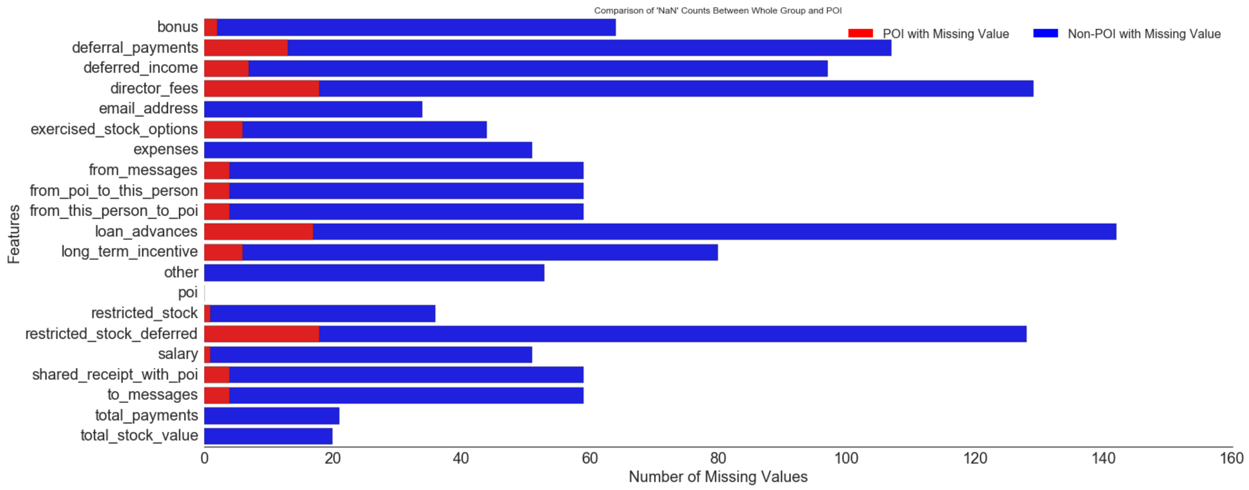


Figure 1. Missing value counts comparison between "poi" and "non-poi"

The chart (Figure 1) above showed missing value counts for "poi" and "non-poi". We could find that some features, e.g. "loan_advances", "deffered_incom", have significant missing values, while others, e.g. "total_payments", "total_stock_value" have comparatively small amount missing values. Table 1 below showed features with more than 60 percentage valid values.

Table 1. Features with more than 60% valid values

	feature	nonpoi_null	nonpoi_r	poi_null	poi_r	total_null	total_r
13	poi	0	0.0000	0	0.0000	0	0.0000
20	total_stock_value	20	0.1575	0	0.0000	20	0.1379
19	total_payments	21	0.1654	0	0.0000	21	0.1448
4	email_address	34	0.2677	0	0.0000	34	0.2345
14	restricted_stock	35	0.2756	1	0.0556	36	0.2483
5	exercised_stock_options	38	0.2992	6	0.3333	44	0.3034
6	expenses	51	0.4016	0	0.0000	51	0.3517
16	salary	50	0.3937	1	0.0556	51	0.3517
12	other	53	0.4173	0	0.0000	53	0.3655

In this table, “nonpoi_null”, “poi_null” and “total_null” indicate missing value counts in “non-poi”, “poi” and whole group, separately. While “nonpoi_r”, “poi_r” and “total_r” represent the ratio of missing value out of all input value of this feature, for the group “non-poi”, “poi” and whole group, separately.

This table showed that the weight of missing value out of total value varies among “poi” and “non-poi” groups, giving a specific feature. For “expense”, all “poi” have valid inputs, however, over 40% “non-poi” contain missing value.

Create New Features

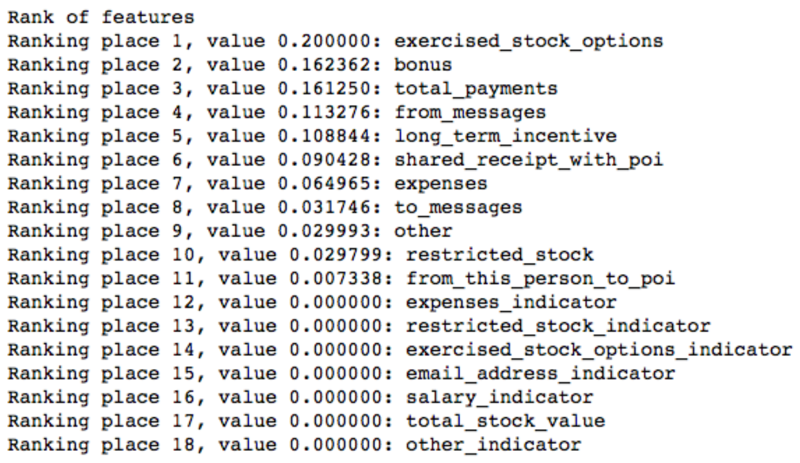
The table 1 above indicated for some features, the “poi” always has valid values. This sparked me an idea that, for some features, “poi” are more likely to have valid input. Take “email_address” for example, all “poi” have an email address inputs, but it is not necessarily applicable to those “non-poi”. Regarding this, I built a couple of feature indicator as new features, assigned “1” to the indictor if a person has valid input under this feature, and assigned “0” if not. New features were created using top 8 features with most valid inputs.

- 'salary_indicator'
- 'email_address_indicator'
- 'exercised_stock_options_indicator'
- 'expenses_indicator'
- 'other_indicator'
- 'restricted_stock_indicator'
- 'total_payments_indicator'
- 'total_stock_value_indicator'

Those new features were added onto the original dataset. Feature selection was conducted based on the whole collection of original and new features.

Feature Selection

Decision Tree Classifier was utilized for the feature selection, because it is simple to understand and easy to use, and it is not picky on the format or scale of this database. This selection was conducted on basis of all old and new features



```
Rank of features
Ranking place 1, value 0.200000: exercised_stock_options
Ranking place 2, value 0.162362: bonus
Ranking place 3, value 0.161250: total_payments
Ranking place 4, value 0.113276: from_messages
Ranking place 5, value 0.108844: long_term_incentive
Ranking place 6, value 0.090428: shared_receipt_with_poi
Ranking place 7, value 0.064965: expenses
Ranking place 8, value 0.031746: to_messages
Ranking place 9, value 0.029993: other
Ranking place 10, value 0.029799: restricted_stock
Ranking place 11, value 0.007338: from_this_person_to_poi
Ranking place 12, value 0.000000: expenses_indicator
Ranking place 13, value 0.000000: restricted_stock_indicator
Ranking place 14, value 0.000000: exercised_stock_options_indicator
Ranking place 15, value 0.000000: email_address_indicator
Ranking place 16, value 0.000000: salary_indicator
Ranking place 17, value 0.000000: total_stock_value
Ranking place 18, value 0.000000: other_indicator
```

Table 2. Feature importance ranking (partial features)

Table 2 above showed that 11 features contribute directly to the final classification.

Unfortunately, all those new indicators were contributing little regarding their feature importance. The final feature list I selected were features with high than 0.05 importance score. They are:

"poi", "exercised_stock_options", "bonus", "total_payments", "from_messages",\
'long_term_incentive', "shared_receipt_with_poi", 'expenses'

III. Methodology

Machine Learning Algorithm

Four machine learning estimators were adopted and compared, including Gaussian Naïve Bayes Classifier, Support Vector Classifier, Decision Tree Classifier and K-nearest Neighbors Classifier.

Those four methods will be applied through the following steps:

- Set feature list: ["poi", "exercised_stock_options", "bonus", "total_payments", "from_messages", 'long_term_incentive', "shared_receipt_with_poi", 'expenses']
- Split features and labels dataset
- Split training and test dataset from features and labels, respectively: 0.2 test size
- The following transformation steps are the same for each estimator, so pipeline was applied to each of them.
  - Standardization: standard scalar
  - Select best features: select 2 best indicator
  - Apply estimator
  - Train and predict
  - Compare estimator performance

Evaluation Metrics

The final dataset is unbalanced with 18 “poi” (class 1) and 127 “non-poi” (class 0), thus the estimator performance should be evaluated regarding both classes. “f1-score”, “precision” and “recall” were selected as evaluation metrics.

- “recall” (r): the ratio of true positives (tp) to all actual positives (tp + fn), indicating how much positive instances are correctly predicted
- “precision” (p): the ratio of true positves (tp) to all predicted positives (tp + fp), indicating out of those predicted positives, how much are correct
- “f-1 score” (F1): weights recall and precision equally, which is a good way to maximize precision and recall simultaneously.

$$F1 = 2\frac{pr}{p+r} \quad \text{where} \quad p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}$$

In this project, I am more concerning both “recall” and “accuracy”. At the same time, I need to take into consideration of “f1 score”, to balanced the performance from both.

All those estimators were applied with their default setting, holding all other factors constant. Results were shown in table 2 (for data, bold indicates highest value in this category). “Gaussian NB” and “KNN” hadgood performance for the precision and f1-score, however, for “poi” (class 1), but their recall were not as good. For “SVC”, there is no score due to a lack of true positive predictions. This might be resulted from unsuitable features list or not good k-best feature selection.

Results showed that the best overall performance came from Decision Tree Classifier, with averaged 0.35 f1-score, 0.36 precision and 0.34 recall. This metrics set already satisfy our requirements, but we still could conduct some refinements for better performance.

Table 2. Evaluation metrics comparison

Metrics	**Gaussian NB**	**SVC**	**Decision Tree**	**KNN**
precision	0.36	NA	0.36	**0.52**
recall	0.23	NA	**0.34**	0.15
f1-score	0.28	NA	**0.35**	0.24

Refinement

Algorithm tuning aims to get better results from algorithm. Different parameter combinations could result in various performances. The process of tuning is to find the best parameter combination given all options, with the goal to achieve maximum performance.

For the Decision tree classifier, I tuned the following metrics:

- min_sample_split: [2, 5, 10, 20, 40]
    - It represents the minimum number of samples required to split an internal nodes.
    - Smaller number will make the estimator not accurate enough to split into right number of groups, thus result in lower performance.
    - Larger number will explore the tree dimensionality and result in high computing cost.
- max_depth: [None, 2, 4, 6, 8, 10, 15, 20]
    - It represents the max depth of the tree, indicating how in depth the structure of the tree.
    - Smaller number of trees makes the model too general to reach required metrics scores.
    - Larger number in layers of trees will increase the training accuracy but might result in over fitting.

I will also tune the parameters for SelectKBest. I will use GridSearchCV for optimization. Tuned parameter includes”

- k: [1, 2, 3, 4, 5, 6, 7, 8]
    - indicates how many features will be selected as best features for fitting.

- score_func: [f_classif, chi2]
    - indicates the function used to select those k best features.

IV. Model Evaluation and Validation

Validation is the process of checking the metrics for a machine-learning algorithm given an independent dataset. A classical mistake for validation is to validate your estimator using the same dataset you train it, which is also called over fitting. This would return a set of very high metrics but it is not meaningful since the model already knows the right answer. Usually, we split the dataset into training and testing set, to train and test them separately.

In this project, I used train_test_split to generate train and testing dataset, also utilized StratifiedShuffleSplit (SSS) to generate iterations during cross validation. The SSS function generates several training and testing datasets with shuffled indices and balanced class distribution from the features and labels, allowing multiple training and testing runs within the small Enron dataset, improving the estimator's performance by supplying even class distribution.

Evaluation Metrics

The same as in the estimator comparison part, three evaluation metrics were selected for the validation and evaluation:

- precision
- recall
- f1-score

V. Conclusion

Results

The final model with best estimator selected parameters combination below:

- k=3
- score_func: f_classif
- min_samples_split=2
- max_depth=8

Table 3. Final model metrics

Metrics	**Default-Decision Tree**	**Tuned-Decision Tree**	**Improvement (%)**
precision	0.36	0.36	0%
recall	0.34	0.38	12%
f1-score	0.35	0.37	6%

The final results for tuned Decision Tree Classifier were shown above (Table 3). Both precision and recall satisfied the requirements (> 0.3). The precision score has no significant changes, still 0.36. Recall was improved by 12%, from 0.34 to 0.38, and f1-score increased by 6%, from 0.35 to 0.37.

Improvement

This project is very interesting since it explores the historical dataset from famous corporate scandal in the American history. Four estimators were evaluated and compared, cross validation was conducted and final model was selected and evaluated. Some thoughts could be applied later:

- Utilize the Tukey's Method for identifying outliers
- Build new features based on features from the final feature list, e.g. features related to emails
- Conduct a more in-depth observation for each feature: sign, outlier
- Try SVC with StratifiedShuffleSplit, to guarantee balanced class distribution for features and labels.