ENRON POI IDENTIFIER REPORT

Introduction

As one of the largest corporate fraud cases in American history, the downfall of Enron is both fascinating and uniquely well-documented (thanks to the ensuing federal investigation). In this project, I use email and financial data for 146 executives at Enron to identify persons of interest in the fraud case. A person of interest (POI) is someone who was indicted for fraud, settled with the government, or testified in exchange for immunity. This report documents the machine learning techniques used in building a POI identifier.

The Enron Data

The first step of this project is examining the data for any outliers or obvious mistakes. There was one significant outlier that arose from a spreadsheet summary line that got transcribed into the dataset; this outlier was removed by hand before proceeding. Two more financial outliers (Ken Lay and Jeff Skilling) were left in because they are clearly real Enron characters, and the fact that they made so much money off the company is not noise or a mistake--it's tightly entangled with the corporate fraud.

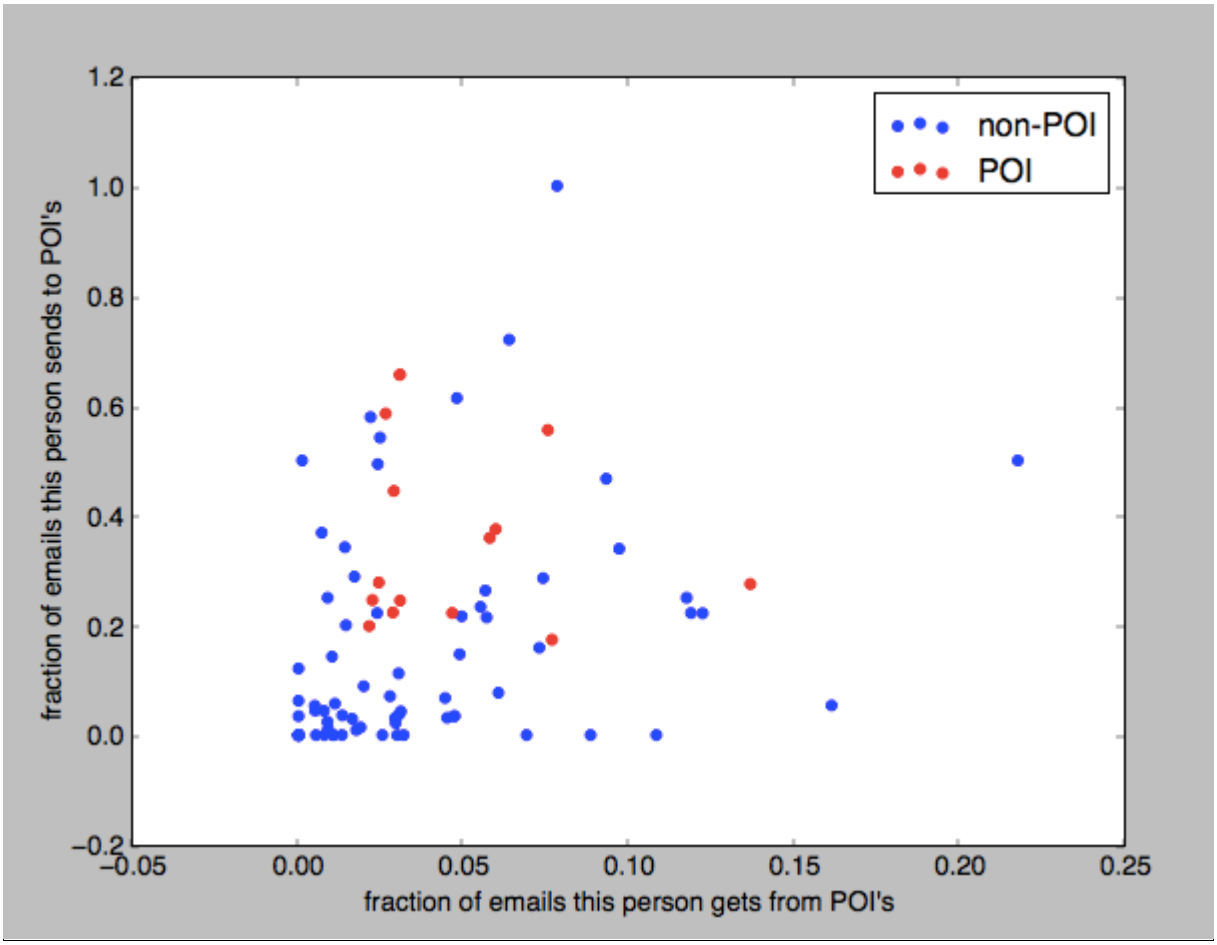
Total Person entries in the data set – 146
Total features for each person – 21
Count of Persons of Interest – 18
Count of people with salary entries - 95
Total salary paid for all the people combined - 53408458
Total salary paid only for the persons of interest - 6518563
Count of the people with bonus entries - 82
Total amount paid in terms of bonus for all the people combined -194687238
Total bonus paid only for the persons of interest - 33199999
Count of people who have email address - 111
Count of people who have value for total payments - 21
Count of people who have value for total stock value – 20
Total amount in terms of total stock value held by everyone - 853518639
Total amount in terms of total stock value held by only POI – 164982077
Percentage of people in the dataset having 'NaN' for their total payments – 14.4
Percentage of POIs in the dataset having 'NaN' for their total payments - 0

Feature Processing

Once the data was cleaned of outliers, the next step was selecting features to use. This was an iterative process, where a handful of features were selected for a first pass based on visualizations of the data that suggested that they might have some discriminatory power. These features were then used in the next steps of the process, selecting and evaluating a classification algorithm. Once a decision tree was selected as the algorithm for the POI identifier (further details on that process are given in the

next section), the features were revisited for optimization. No feature scaling was deployed, as it's not necessary when using a decision tree.

Two new features were created and tested for this project. These were



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

The hypothesis behind these features was that there might be stronger email connections between POIs than between POIs and non-POIs, and a scatterplot of these two features suggests that there might be some truth to that hypothesis.

In order to find the most effective features for classification, feature selection was deployed to rank the features.

- The top 8 features were selected for use in the classifier.
- These were features selected using the SelectKBest feature selection process.

ENRON POI IDENTIFIER REPORT

- The selected features were "fraction_to_poi", 'salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options' and 'restricted_stock'
- It does not mean there are only 8 best features.
- Tried different sets of features for each classifier
- Trying both smaller and larger numbers of features negatively impacted the precision and recall.
- As can be seen from the list of features used in the classifier, out of the two new fractional email features only "fraction_to_poi" was used in the final version because it helped the precision and recall.

Algorithm Selection and Tuning:

Below three algorithms were tried for classifying the POI. These are some of the basic classification algorithms being used and decided to try these three algorithms.

- Naïve Bayes
- Decision tree
- Support Vector Machine
- Random Forest

Steps following for selecting the algorithm and tuning the parameters:

Reason for trying for various classifiers:

- There is no single best algorithm to solve the problems.
- Different algorithms perform well for different data sets.
- It is very important to try different algorithms to find which one performs well
- Dataset at hand and the configuration of the algorithms plays a very important role on how well they can perform.

Reason for parameter tuning:

- Machine learning models are parameterized so that their behavior can be tuned for a given problem.
- Models can have many parameters and finding the best combination of parameters can be treated as a search problem.
- It is very important to find the best combination of the algorithm parameters which helps us to achieve the best results.
- Playing around the algorithms by fine tuning is very important to find the best combination.
- We can try choosing various combinations of values for parameters manually which would be time consuming and tiring.
- There are other automated ways like in sklearn grid search and random search methods to find the best possible parameter combinations.

ENRON POI IDENTIFIER REPORT

Naïve Bayes:

- Created a classifier using the Naïve Bayes algorithm.
- Used 8 best features selected using the SelectKBest selection method.
- It does not mean only 8 best ones are present.
- As the precision and the recall was not good enough tried various combinations of these and other features.
- Finally selected these features for Naïve Bayes classifier - 'salary', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'restricted_stock'

Using stratified StratifiedShuffleSplit Cross Validation was able to achieve the below results:

- Manually tried numerous combinations of values for folds and random_state in StratifiedShuffleSplit method.
- For Folds tried values between 1 to 20
- For random_state tried values between 12 to 50

Accuracy: 0.85529 Precision: 0.49167 Recall: 0.38350
F1: 0.43090 F2: 0.40115
Total predictions: 14000
True positives: 767 False positives: 793
False negatives: 1233 True negatives: 11207

In fact tried other methods of cross validation like KFold and train_test_split. Achieved the same results as stratified.

Decision Tree:

- Created the decision tree classifier.
- Used the same set of features used for Naïve Bayes algorithm with train_test_split method of cross validation.
  Got the below results:
  Accuracy: 0.78729 Precision: 0.26625 Recall: 0.27850
  F1: 0.27224 F2: 0.27596
  Total predictions: 14000
  True positives: 557 False positives: 1535
  False negatives: 1443 True negatives: 10465

- To improve the results tried to find the best features for Decision tree using feature_importances_ attribute of the classifier.
- "fraction_from_poi", "fraction_to_poi", 'bonus', 'total_stock_value', 'expenses', 'exercised_stock_options' and 'long_term_incentive' came out as the most important features.

ENRON POI IDENTIFIER REPORT

- Manually tried various combinations of the parameters for decision tree classifier and was finding it difficult to find which value was the giving the good scores.
- Using the GridSearchCV selected the best possible values for the parameters like splitter, criterion, min_samples_split and min_samples_leaf
- Tired setting the random_state parameter for Decision Tree.
- Parameters selected using the GridSearchCV did not help much to improve the results so had to do some manual changes
- Got the below results:
  Accuracy: 0.82986 Precision: 0.40817 Recall: 0.42450
  F1: 0.41618 F2: 0.42113
  Total predictions: 14000
  True positives: 849 False positives: 1231
  False negatives: 1151 True negatives: 10769

  Using other validation methods like KFold and StratifiedShuffleSplit was able to achieve the same results as above.

Support Vector Machine:

- Support Vector Machine was performing well in terms of accuracy
- But in terms other evaluation metrics like recall and precision it was performing very badly.
- Feature scaling was performed for SVM algorithm.
- Normalization method of feature scaling was used.
- SVM relies on the magnitude of the values hence normalization method of scaling was used.
- Manually tried varying the parameter values for kernel, C and gamma.
- There was good improvement with respect to recall and precision.
- Tried two different methods of validation but could not test using tester.py
  Train_test_split method results:

  F1 - 0.352941176471
  Precision - 0.375
  Recall - 0.333333333333
  Accuracy - 0.792452830189

  StratifiedShuffleSplit method results:

  F1 - 0.4
  Precision - 0.333333333333
  Recall - 0.5
  Accuracy - 0.785714285714

ENRON POI IDENTIFIER REPORT

Random Forest:

- Random Forest classifier was created using the same features used for Naïve Bayes and got the below results:
  Accuracy: 0.84786 Precision: 0.42389 Recall: 0.18100
  F1: 0.25368 F2: 0.20443
  Total predictions: 14000
  True positives: 362 False positives: 492
  False negatives: 1638 True negatives: 11508

- To improve the results tried to find the best features for Decision tree using feature_importances_ attribute of the classifier.
- "fraction_from_poi", "fraction_to_poi", 'salary', 'deferral_payments', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock' were found to be important features based on the results from feature_importances_.
- First tried using the train_test_split validation method.

  Achieved the below results:
  F1 - 0.5
  Precision - 0.666666666667
  Recall - 0.4
  Accuracy - 0.916666666667

  But when tested with the tester.py the result was as below:

  Accuracy: 0.86473 Precision: 0.47943 Recall: 0.16900
  F1: 0.24991 F2: 0.19414
  Total predictions: 15000
  True positives: 338 False positives: 367
  False negatives: 1662 True negatives: 12633

- Tried removing some of the features having lesser importance as per feature_importances_ attribute but did not help much.
- Tried various combinations of parameters for the algorithm.
- Tried using GridSearchCV also but had to vary the values manually to find the best possible combination.
- With KFold validation method was able to achieve below results:
  Accuracy: 0.85673 Precision: 0.41367 Recall: 0.17850
  F1: 0.24939 F2: 0.20140
  Total predictions: 15000
  True positives: 357 False positives: 506
  False negatives: 1643 True negatives: 12494

- With StratifiedShuffleSplit method of validation was able to achieve the below results:

Accuracy: 0.86429 Precision: 0.56964 Recall: 0.20450
F1: 0.30096 F2: 0.23457
Total predictions: 14000
True positives: 409 False positives: 309
False negatives: 1591 True negatives: 11691

- Random forest precision score was getting better and better but the recall was not improving. It might be due to lack of enough data.

Decided to go with **Naïve Bayes** based on the precision, recall and accuracy scores.

Analysis Validation and Performance

As per the model created and evaluated as part of the project, average precision was 0.49 and the average recall was 0.38

Validation:

- It is very important to test the model to find how well it is performing when it is presented with the data which was not used to train it.
- A good model is one which performs well even on testing set.
- A trained model is not exposed to the test dataset during training and any predictions made on that dataset are designed to be indicative of the performance of the model in general.
- By evaluating the model on the testing set one can always go back to the board and fine tune the model to perform well on the testing sets.
- The beauty of Machine Learning algorithms is unleashed when it performs well for the unknown datasets (test set).
- There are various ways in which the dataset can be divided into training and testing sets.
- Simple method is random split of the dataset
- In sklearn cross_validation has various ways of helping to create the training and testing sets.
- In this project I have utilized the below ones:
  train_test_split
  KFold
  StratifiedShuffleSplit
- Finally used the StratifiedShuffleSplit as it was the ones used for testing the models of this project.

Performance:

- The performance measure is the way you want to evaluate a solution to the problem.
- It is the measurement you will make of the predictions made by a trained model on the test dataset.
- Accuracy is simple performance metric.

ENRON POI IDENTIFIER REPORT

- Accuracy always does not gives us the right measurement of the performance of the model.
- So we use precision and recall also in order to evaluate the model.
- Precision for a class is the number of true positives divided by the total number of elements labeled as belonging to the positive class.
- Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class

Discussion and Conclusions

The precision can be interpreted as the likelihood that a person who is identified as a POI is actually a true POI; the fact that this is 0.49 means that using this identifier to flag POI's would result in 51% of the positive flags being false alarms. Recall measures how likely it is that, given that there's a POI in the test set, this identifier would flag him or her; 38% of the time it would catch that person, and 62% of the time it wouldn't.

While these numbers are reasonably good, certainly better than guessing or a very unoptimized classification process, it would still be challenging to rely exclusively on this POI identifier alone to flag POIs. If either precision or recall were close to 1.0 then it would be easier to interpret as either

(1) A very conservative algorithm that sometimes misses POI's but is rarely wrong when it does find one,

(2) A trigger-happy algorithm that rarely misses POI's but is prone to false alarms.

Biggest problem with the dataset:

- There were only 18 examples of POIs in the dataset.
- There were 35 people who were POIs in "real life", but for various reasons, half of those are not present in this dataset.
- Data set is not complete

Paths for Improvement:

- "More data is better than a finely-tuned algorithm" maxim of machine learning, having a larger dataset is likely the single biggest way to improve the performance of this analysis.
- Dig in more into the contents of the emails.
- Extract more data from the contents of the emails.
- Get into the details of the meetings and the telephonic calls between the people in the dataset.
- Also try to get more data about the meetings of the people in the dataset with the people in powerful positions in the government and its agencies during the period of the fraud.

ENRON POI IDENTIFIER REPORT

References:

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html

http://stackoverflow.com/questions/5843817/programmatically-install-nltk-corpora-models-i-e-without-the-gui-downloader

http://machinelearningmastery.com/blog/

https://www.coursera.org/learn/machine-learning

https://weka.waikato.ac.nz/

http://stackoverflow.com/questions/21391429/classification-tree-in-sklearn-giving-inconsistent-answers

http://machine-learning.tumblr.com/post/1209400132/mathematical-definitions-for-precision-recall-for

http://machinelearningmastery.com/how-to-evaluate-machine-learning-algorithms/

“I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.”