Project Overview

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, 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 put my new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal. To assist me in my detective work, the data combined with a hand-generated list of persons of interest 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.

Summary

The goal of this project is to develop an algorithm that could have identified people of interest related to its eventual financial disaster. This project will use machine learning with supervised learning since we know the outcome of guilty parties. The data used in this project is comprised of many financials and emails from dozens of former Enron employees. Financial data includes categories such as salary, bonus, exercised stock options, etc. The emails data is all the available email traffic to or from a former employee.

There are 146 people in the data set with 24 features each. There are 18 POI's and 128 Non-POI's. The data set is considerably small in quantity. Ideally, we would want a data set in the thousands. The higher the quantity of data points, the less sensitive to accuracy, precision, and recall numbers variance when train-test splitting. Randomly splitting the data into a training set and testing set can be problematic if most of the POI’s are moved to the train dataset. There’s a distinct possibility since the total number of POI’s is a little over 10% of the whole dataset. This is why tuning and validation are critically important.

Two outliers were identified and removed: *TOTAL* and *THE TRAVEL AGENCY IN THE PARK (TAP)*. Both were removed mainly because they were not people. TOTAL is obviously not relevant for determining specific people engaged in financial misconduct. TAP was removed as we are not focused on potential companies of financial misconduct—only people. TAP was 50% owned by Sharon Lay, Ken Lay’s sister. TAP received commissions for managing travel books for Enron.[1] While the family connection may be suspicious, it is outside the scope of the project.

Features

All available features from the data set were used except the actual email addresses. Two new features were created but not used: total_comp and fraction_to_poi. The feature total_comp sums salary, bonus, total_stock_value, and exercised_stock_options. Fraction_to_poi generates the fraction of emails to and from the POI. The idea was to compress features into a bigger metric; however, neither feature seemed to help with accuracy, precision, or recall.

The features list was pass through SelectKBest to identify the top 6 features. There was a total of 18 identified features. Every number of top features from 4 to 12 was tested. A feature quantity of 6

[1] Texas agency takes a huge hit from Enron's fall. March 5, 2002. http://www.travelweekly.com/Travel-News/Travel-Agent-Issues/Texas-agency-takes-a-huge-hit-from-Enron-s-fall

produced the highest accuracy, precision, and recall. Here are the top 6 features and scores using SelectKBest:

```
#1 exercised_stock_options with score 24.8150797332
#2 total_stock_value with score 24.1828986786
#3 bonus with score 20.7922520472
#4 salary with score 18.2896840434
#5 deferred_income with score 11.4584765793
#6 long_term_incentive with score 9.92218601319
```

The feature loan_advances have the highest number of *NaN* values. The high number of NaN's thins our data quantity even further. This ultimately makes it even harder to design a functional machine learning algorithm. Below is a list of Total NaN's by feature in the data set:

```
* salary with 50 NaN's
* to_messages with 58 NaN's
* deferral_payments with 106 NaN's
* total_payments with 21 NaN's
* exercised_stock_options with 43 NaN's
* bonus with 63 NaN's
* director_fees with 128 NaN's
* restricted_stock_deferred with 127 NaN's
* total_stock_value with 19 NaN's
* expenses with 50 NaN's
* from_poi_to_this_person with 58 NaN's
* loan_advances with 141 NaN's
* from_messages with 58 NaN's
* from_this_person_to_poi with 58 NaN's
* deferred_income with 96 NaN's
* shared_receipt_with_poi with 58 NaN's
* restricted_stock with 35 NaN's
* long_term_incentive with 79 NaN's
```

Algorithm

Two algorithms were tested with the dataset: Guassian Naïve Bayes and AdaBoost. Scaling was not utilized with GNB as it was not required. Naïve Bayes resulted the following:

```
Accuracy: 0.84714
Precision: 0.45679
Recall: 0.37000
F1: 0.40884
F2: 0.38462
Total predictions: 14000
True positives:  740
False positives:  880
False negatives: 1260
True negatives: 11120
```

AdaBoost was used with MinMaxScaler for scaling although scaling did not appear to make a difference in results. These are the results:

```
Accuracy: 0.84029
Precision: 0.38499
Recall: 0.19750
F1: 0.26107
F2: 0.21881
Total predictions: 14000
True positives:  395
False positives:  631
False negatives: 1605
True negatives: 11369
```

The results where almost identical; however, GNB had a higher recall.

Parameter Tuning

Parameter tuning can be a very important process when algorithms require it. In this project, Guassian Naïve Bayes was used which does not require parameter tuning. AdaBoost does utilize parameter tuning which was executed in this project. The algorithm and n_estimators parameters were investigated for AdaBoost. The algorithm parameter for ADB allows you to alternate between discrete and real boosting. The n_estimators parameter defines the maximum number of estimators at which boosting is terminated.[2] The change from real and discrete algorithm selection did not appear to adjust accuracy, precision, and recall. Iterating from 2 to 12 by 2 for n_estimators only adjusted the recall slightly. Changes in accuracy and precision were negligible. A few high end values were tested. A value of 100 appeared to break the algorithm—or at least take longer to calculate than I had patience.

Validation

Validation is an important process with machine learning. The focus of proper validation is to reduce the to prevent overfitting as well as ensure that results can be repeatable by other users. The focus of this project was to get the accuracy, precision, and recall as close to a value of 1 as possible. The more prominent overfitting is, the higher the likelihood of misidentification of new potential data points. New data points do not apply to this project; however, it would apply to systems with streaming data from sensors as an example.

Validation is also important to ensure that the algorithm tuning is repeatable. Anyone should be able to use `poi_id.py` to produce the same results documented in this report. The `test.py` file was used for validation of the project which uses Stratified Shuffle Split (SSS) from SKLearn. SSS takes random test-train splits to generate metrics (e.g. recall and precision) over a series of user defined iterations. This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.[3] The `test_classifier` function iterates the results over a user-defined fold count. The default iteration value is 1,000.

Metrics

Precision and recall are used to gauge the efficacy of the investigated algorithms. Precision is the ratio of true positives to the sum of true positives and false positives. Recall is the ratio of true positives to the sum of true positives and false negatives.[4] The closer to 1 for both metrics, the lower the error rate. An example related to this project would be if an individual is falsely identified as a person of interested when in fact they are not (i.e. precision). Another example would be if an individual is not identified as a person of interest when in fact they are (i.e. recall). These examples are the reason we want to make sure all persons of interest are identified and no one is falsely identified.

[2] sklearn.ensemble.AdaBoostClassifier. 2014. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[3] sklearn.cross_validation.StratifiedKFold. 2014. http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedKFold.html

[4] Precision and Recall. September 16, 2016. https://en.wikipedia.org/wiki/Precision_and_recall