Identify Fraud from Enron Email

Nicole Mister

Enron, an energy commodities and services company, declared bankruptcy in 2001. The company then went under investigation for fraud. A corpus of emails was used in the investigation to determine persons of interest (POI) in the case. The goal of this project is to use this corpus of emails as well as employee financial data to create a machine learning algorithm to predict if an Enron employee is a person of interest.

Our dataset contains 146 records with 21 features. Of the 146 records, 18 are POI. After graphing a scatterplot of salaries and bonuses, it was clear that there were some outliers. After a quick visual inspection of enron61702insiderpay.pdf, I was able to determine "TOTAL" and "THE TRAVEL AGENCY IN THE PARK" were outliers that should be removed. I used the dicitionary pop function to remove those records. While another scatterplot indicated that the data still included additional outliers, I did not remove any additional data. The outliers included several POIs. We would therefore potentially lose pertinent data by removing these points. Additionally, by reviewing the PDF, I was able to determine several features that did not contain much data: loan_advances, director_fees, restricted_stock_deferred, deferral_payment. Additionally, the dataset contained email addresses, which in addition to not converting nicely into strings, is unlikely to contain data relevant to locating a POI. I removed these features and used 16 of the provided features for first set of algorithms.

Because I believe the proportion of emails to and from a POI may be more relevant than just the sum of emails, I created two new features, from_ratio and to_ratio. The to_ratio is the number of emails from this person to a POI divided by the total number of emails sent by this person. The from_ratio is the number of emails from this person to a POI divided by the total number of emails from this person. Adding the from_ratio and to_ratio to the classifers did improve the results of the Decision Tree, but did not impact the other classifiers.

Classifier	Before New Features	After New Features
Decision Tree	Precision: 0.22032 Recall: 0.21250	Precision: 0.31802 Recall: 0.30800
Naïve Bayes	Precision: 0.33274 Recall: 0.28250	Precision: 0.33274 Recall: 0.28250
KNeighbors	Precision: 0.64636 Recall: 0.19100	Precision: 0.64636 Recall: 0.19100

To select features, I used SelectKBest to select the features that had a p-value less than 0.05. The features selected and p-values are as follows:

0 salary 3.47827376837e-05
1 total_payments 0.00358932617252
2 bonus 1.1012987324e-05
3 deferred_income 0.000922036708467
4 total_stock_value 2.40431527604e-06

Identify Fraud from Enron Email

Nicole Mister

5 expenses 0.0147581999654
6 exercised_stock_options 1.81820487779e-06
7 other 0.0425817470123
8 long_term_incentive 0.00199418124535
9 restricted_stock 0.00286280295791

I did not use scaling for these features.

Two algorithms met the 0.30 or better precision and recall score requirement of this project: Decision Tree and Naïve Bayes. The Naïve Bayes algorithm score is slightly better with 0.379 precision and 0.338 recall. I also tried KNeighbors. KNeighbors had higher precision (0.646), but lower recall (0.191).

Tuning the parameters means to adjust settings of the algorithm which may impact how well the algorithm performs on a dataset. The Decision Tree algorithm's precision and recall dropped (0.217 and 0.073 respectively) when the minimum samples split was set to 100 and when it was set to 10 (0.235 and 0.200). Setting the criterion to entropy slightly also decreased both precision (0.26) and recall (0.231). Setting the max features to auto improved both precision (0.309) and (0.309). Still, the best classifier was Naïve Bayes with a precision of 0.379 and recall of 0.305.

Validation tests an algorithm's performance against a set of data not used in the training. If validation is not performed correctly, for instance, testing on the same data used to train, you can overfit your data so that the effectiveness of the algorithm seems better than it actually is. I validated my analysis through the StratifiedShuffleSplit function randomly assigns records to either the test or training group. I used a test group that was 30% of the dataset. The classifier is trained on the training group and is tested against the test group.

The evaluation metrics I used to evaluate the algorithms in this analysis were precision and recall. Precision is the percent of employees labeled as POIs that were actually POIs. The final classifier had a precision of 0.379 which means that of all the individuals the classifier identified as a POI, 37.9 % of them actually were POIs. Recall is the percent of POIs correctly identified. The final classifier had a recall of 0.305 which means that of all the POIs, 30.5% of them were correctly identified by the classifier.