Enron POI Identifier Report

By Henry Wang

Introduction

The project explores algorithmic classifiers with the objective of finding persons of interest, POI, from public Enron financial and email data.

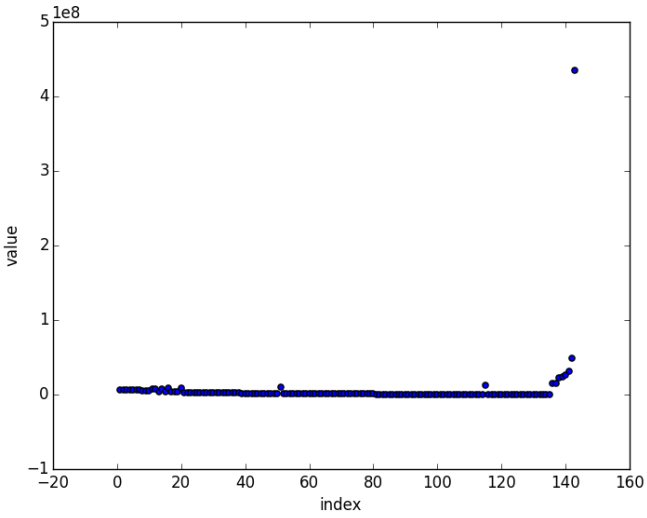
The Enron Data

The data contains 20 features on 145 persons with a subset of 18 marked with the additional POI flag. POI are those who were indicted, reached a settlement, involved in plea deals with the government, or testified in exchange for prosecution immunity. Some features did not contain data for every person.

Various supervised machine-learning algorithms are explored with this data to find the optimal classifier for the POI flag based on selected features.

Removing Outlier Data

There was one obvious outlier in the data—the total combined values for all persons. The figure below shows the data with outlier at the top of the chart for the feature "total_payments." This outlier was removed for testing.



Feature Transformation

All features are scaled from 0.0 to 1.0 for testing. This is done for some classifier, such as SVM, where the scales mattered.

Three additional features are added to describe data in terms of ratio to overall amount. The ratios help make better comparisons over absolute values.

- fraction_to_poi: ratio of emails sent to POI
- fraction_to_shared_with_poi: ratio of emails received shared with POI
- fraction_from_poi: ratio of emails from POI

Feature Selection

To select initial features, two processes were tried: PCA eigenvalues and decision tree importance.

PCA

The eigenvalues for all features of the first PCA component is analyzed (see task_2.py). Four of the five final feature selection is chosen from this list due to high eigenvalues:

- total_payments: 0.68172
- loan_advances: 0.51338
- total_stock_value: 0.40506
- exercised_stock_options: 0.29293

Decision Tree

A decision tree is analyzed with all possible combination of up to 4 features. 3,206 different combinations were tried. The sum of importances of each feature is recorded (see task_6.py). The sums of importances of the best ten performing set of features are:

- exercised_stock_options, 5.4425149427154835
- fraction_to_shared_with_poi, 2.7504467362191813
- total_payments, 1.2691611397988012
- long_term_incentive, 0.42577529949934501
- total_stock_value, 0.11210188176719012

In addition to the four features from PCA, the fraction_to_shared_with_poi feature is chosen for the final list of features. The lesser significant last item in the list is ignored to keep the feature list from being too long.

Algorithm Selection and Tuning

Multiple algorithms and parameters are tested:

- Naïve Bayes
- Decision Tree with min_sample_split of 1, 2, 5, 10, 15, 20, 25, 30
- SVM of linear/rbf kernel and C-value of 10, 100, 1000, 10000
- AdaBoost with Decision Tree of 2, 10, 20, 30 min_sample_split
- Random Forest with Decision Tree of 10, 25, 50, 100 n_estimators and 20 min_sample_split

Tuning the parameters of algorithms means adjusting possible settings to find the best settings. If not done well, the best algorithm may be overlooked. The list above includes tuned parameter values.

Score are recorded for each trial. The score is determined by the sum of precision and recall where both values are above 0.3. The top five scores were all from the decision tree classifier of min_sample_split of 25, 30, 20, 15, and 10.

Decision Algorithm Tuning

To further tune the decision tree (see task_7.py), min_sample_split values of 5 to 50 are tested with increments of 5. Each trial is averaged over 50 runs. The table below shows the result. The best decision tree of min_sample_split of 30 is saved in my_classifier.pkl.

Split	Precision	Recall	Total
5	0.46891	0.40918	0.87809
10	0.606512	0.55727	1.163782
15	0.609789	0.5568	1.166589
20	0.613056	0.55654	1.169596
25	0.612996	0.55654	1.169536
30	**0.613018**	**0.55665**	**1.169668**
35	0.612907	0.55057	1.163477
40	0.576708	0.37515	0.951858
45	0.540263	0.27694	0.817203
50	0.545138	0.27541	0.820548

Validation

Validation ensures that the classifier is not over fitting the data. Splitting the data into a training set and a testing set can do this. Also, multi-fold cross validation can be used. Because of the small sample size, all tests ran from the tester.py uses stratified shuffle split cross validation.

A classic mistake is to not shuffle the data before dividing it into a training subset and testing subset.

Conclusion

The evaluation score used is the sum of precision and recall where both values are above 0.3. Recall is the ratio of correctly detected positive provided that the actual is positive. The precision is the ratio of correctly detected positives over all detected positives. The best classifier (decision tree of min_split_split 30) has a precision of 0.613 and recall of 0.557 for a combined value of 1.170.