Enron POI Identifier Report

Brian Hurn
September 2015

Background

I remember hearing about the fall of the energy trading firm Enron in 2001 and watching the news reports showing dejected employees, cardboard boxes in hand, walking out of their Houston headquarters. Four years later, when I saw the documentary *Enron: The Smartest Guys in the Room*, in addition to the sympathy I felt for their employees I had a growing sense of anger toward the criminal mismanagement of the company.

The intent of this project is to examine a dataset that includes financial compensation and summary email information for a subset of Enron employees using machine learning algorithms to develop a person of interest (POI) identifier. POIs are labeled in the dataset and are those who came under additional scrutiny or were subject to investigations or criminal charges in the aftermath of the collapse. One can imagine the FBI or Justice Department employing similar analysis techniques as part of their investigation to ensure that it included the majority of individuals within the company with potential involvement in criminal activity.

Machine learning algorithms are well suited for classification problems (POI or not?) like this. Given a data set of sufficient size with known labels, the algorithms can be trained through a process called supervised learning to develop a classification model. This model can then take data from either a test set (a subset of the original data not used for training) or from a new data set with matching features and make predictions for the appropriate label (or class) for each data point.

Data Exploration

The final project dataset included 146 entries, one for each person, and each entry included 23 features. Because the dataset was so small it was easy to use the variable explorer in Spyder to see that two entries, TOTAL and THE TRAVEL AGENCY IN THE PARK, were bogus and should be purged prior to analysis. I also ran summary statistics (using scipy.stats.describe) for the 21 (19 original and 2 derived) numeric variables, and many showed abnormally large means driven by the outliers in the TOTAL entry that had been carried into the dataset by mistake. Once the TOTAL entry was eliminated, the means and maximums of the variables across the remaining 144 subjects reverted to much more reasonable values.

Other than the poi feature, which was added manually by the dataset developer, none of the other features is present for all 144 entries. The counts for each feature are shown in the column titled “n” in the table in the next section. While the top three features (those selected for the final classifier as described below) show 43, 19, and 63 missing values respectively, they were present for more than half of all entries and provided sufficient predictive power to meet the goals of the project. In addition, these three features had 12, 18, and 16 values respectively for the 18 individuals identified positively as POIs.

Feature Selection
In addition to the 19 numeric features in the dataset, two new features were generated. Both features represent the proportion of emails sent to and received from POIs; t_s_ratio is from_this_person_to_poi divided by from_messages, and f_t_ratio is from_poi_to_this_person divided by to_messages. Normalizing these quantities to the proportion of messages sent/received allows them to represent the relative level of interaction with POIs, rather than the absolute number of messages exchanged (which could be driven by reporting structures or propensity to email).

Feature scaling, which typically involves converting a variable to a value between 0 and 1 based on its position between the minimum and maximum values, was not required in this project. Scaling is often necessary when creating compound variables through addition or subtraction of two or more variables that have widely different magnitudes, ranges, or units, but that was not the case here.

Univariate feature selection was used to develop a prioritized list of variables for use with the selected algorithms. SelectKBest by default supplies a one-way ANOVA score between each selected feature and the labels. The score for each feature is shown in the fourth column below. The values shown in the four rightmost columns in the table indicate the single-variable Gaussian Naive Bayes performance for the top 9 features as scored by the test_classifier function supplied in the course. See the Validation section below for more details.

Ranked List of Features

Rank	Feature	n	Score	Accuracy	Precision	Recall	F1 Score
1	exercised_stock_options	101	24.82	0.904	0.461	0.321	0.378
2	total_stock_value	125	24.18	0.861	0.615	0.264	0.369
3	bonus	81	20.79	0.789	0.580	0.178	0.272
4	salary	94	18.29	0.780	0.464	0.110	0.177
5	t_s_ratio	86	16.41	0.848	0.031	0.002	0.004
6	deferred_income	48	11.46	0.818	0.595	0.281	0.382
7	long_term_incentive	65	9.92	0.841	0.185	0.033	0.056
8	restricted_stock	109	9.21	0.810	0.372	0.066	0.111
9	total_payments	123	8.77	0.822	0.058	0.010	0.017

10	shared_receipt_with_poi	86	8.59				
11	loan_advances	3	7.18				
12	expenses	94	6.09				
13	from_poi_to_this_person	86	5.24				
14	other	91	4.19				
15	f_t_ratio	86	3.13				
16	from_this_person_to_poi	86	2.38				
17	director_fees	16	2.13				
18	to_messages	86	1.65				
19	deferral_payments	38	0.22				
20	from_messages	86	0.17				
21	restricted_stock_deferred	17	0.07				

As shown in the Evaluation Metrics section, the top three features (exercised_stock_options, total_stock_value, and bonus) delivered the optimal performance for both the Naive Bayes and Decision Tree classifiers examined. While total_stock_value is a sum of stock-based compensation (including exercised_stock_options), it provides a boost in recall that improves overall performance versus a two-feature model using exercised_stock_options and bonus alone. From a qualitative perspective, it appears that those individuals who became persons of interest were those who also had the most to gain through incentive payments. The derived feature t_s_ratio was not included in the final model, but it did provide benefits to the models employing the top 5 through 8 features, all of which were relatively strong performers. The feature f_t_ratio did not score well and was not utilized.

Algorithm Selection
The tables in the Evaluation Metrics section show the performance of the Gaussian Naive Bayes and Decision Tree classifiers that were analyzed. An attempt was made to use a Support Vector Machine classifier with both 1 and 3 features, but both failed to deliver any results after one hour of training time.

As the results show, the Gaussian Naive Bayes classifier provided stronger performance than the Decision Tree. Surprisingly, the Naive Bayes classifier met the target criteria (precision and recall > 0.3) with a single feature, exercised_stock_options. Based on the Accuracy and F1 score performance, the final selected classifier employed the top 3 features identified above.

Parameter Tuning
In machine learning, parameter tuning is the process of modifying one or more algorithm input parameters in a systematic way to achieve optimal results. For algorithms with variable parameters, such as Decision Tree, tuning these parameters can provide two major benefits. The first benefit is to avoid overfitting, An overfit model tracks the training data too precisely, and as a result it performs poorly when it encounters additional data points. The second benefit is to limit training, testing, and production processing times by reducing the complexity and size of the model.

The scikit-learn package includes a function called GridSearchCV that can score multiple runs of the selected algorithm using all of the combinations of the specified input parameters. In this project, the min_samples_split parameter for the Decision Tree classifier was adjusted manually and evaluated using the test_classifier function supplied in the course. The results, shown below, reveal that values of 2 and 3 provide the strongest performance, and 2 was used subsequently in evaluating the Decision Tree classifier across a range of feature counts.

Decision Tree Parameter Tuning Results (# of features = 3)

min_samples_split	**Accuracy**	**Precision**	**Recall**	**F1 Score**
2	0.801	0.361	0.381	0.371
3	0.810	0.379	0.364	0.371
4	0.808	0.368	0.346	0.356
5	0.805	0.351	0.312	0.330
6	0.811	0.367	0.318	0.340

Validation
Validation is vital to selecting the right algorithm, parameters, and input features that will deliver optimal performance in machine learning. By dividing a dataset into training and testing subsets, one can take the model developed against the training set and validate its performance against the testing set. If the training and test sets are not representative of the larger set or the future input data (especially if they do not have an appropriate mix of the target classifications), the model could be ineffective or be evaluated erroneously.

The project data set has a relatively small number of data points (144) and positive POI labels (18). Because of these low values, the test_classifier function supplied in the course was used to validate each variant of the selected classifiers and number of rank-ordered features. The function uses Stratified ShuffleSplit cross validation, which returns stratified random folds (train/test splits of the samples) through multiple iterations, providing numbers of predictions and tests much larger than could be achieved through a single fit/validate cycle. Most importantly, this technique ensures a consistent mix of POI/non-POI data points in the training and test sets to avoid situations where a mix skewed too far in one direction or the other could lead to an inaccurate model or a lack of any positive classifications in any particular iteration. KFold cross validation could have been utilized to generate multiple iterations, but its shuffle function lacks the ability to ensure an appropriate mix of classes in the training and test sets.

Evaluation Metrics

Three key metrics were used to evaluate the performance of the selected POI identifier. The first, accuracy, measures the number of correct predictions (positive and negative) as a fraction of the total number of predictions. In this project, the ability of the models to correctly identify POIs and non-POIs was quite good, with the selected model (Naive Bayes with the top 3 features) doing so 84.3% of the time across 13,000 predictions made by the cross validation algorithm. Note, however, that simply predicting “no” for all entries would yield an accuracy of 87.5% because only 12.5% of the data points are in fact POIs. This situation highlights the fact that additional evaluation metrics are vital to the selection process.

Precision, also known as positive predictive value, measures the number of true positives as a proportion of positive identifications. In other words, if the algorithm identifies a certain number of POIs, what percentage of them are actually POIs? Maximizing precision can help to reduce what in this case would essentially be false accusations.

Recall (or sensitivity) evaluates the number of positive classifications as a proportion of the number of positive elements in the data set. Stated differently, what share of all the POIs in the dataset did the algorithm detect? Increasing recall allows one to reduce the number of POIs that would be missed and not be identified for further investigation.

The F1 score provides a balanced and valuable view of the combination of precision and recall. As the harmonic mean (the inverse of the average of the inverses) of the two values, it is biased toward the lower value. This means that weak performance on one metric can not be masked by strong performance in the other. The F1 score was used as the primary evaluation metric in this project.

As shown in the two tables below, the Gaussian Naive Bayes classifier using the top three selected features (exercised_stock_options, total_stock_value, and bonus) delivered both strong accuracy as well as the highest F1 score, and both precision and recall are greater than 0.3. Nearly 49% of the time across multiple iterations, when it identifies a POI, that identification is correct. On average, the algorithm also identified just over 35% of the POIs present in the dataset.

Gaussian Naive Bayes Results

Number of Features	Accuracy	Precision	Recall	F1 Score
1	0.904	0.461	0.321	0.378
2	0.841	0.469	0.268	0.341
3	**0.843**	**0.489**	**0.351**	**0.408**
4	0.847	0.503	0.323	0.393
5	0.849	0.456	0.300	0.361
6	0.852	0.477	0.351	0.404
7	0.849	0.464	0.361	0.406
8	0.846	0.454	0.365	0.405
9	0.840	0.377	0.311	0.341

Decision Tree Results (min_samples_split=2)

Number of Features	Accuracy	Precision	Recall	F1 Score
1	0.863	0.274	0.310	0.291
2	0.768	0.220	0.200	0.209
3	0.801	0.361	0.381	0.371
4	0.789	0.321	0.334	0.327
5	0.788	0.292	0.341	0.314
6	0.793	0.277	0.280	0.279
7	0.799	0.285	0.270	0.277
8	0.795	0.271	0.255	0.263
9	0.805	0.276	0.286	0.281