Enron “Person of Interest” Machine Learning Identifier

Short Answers

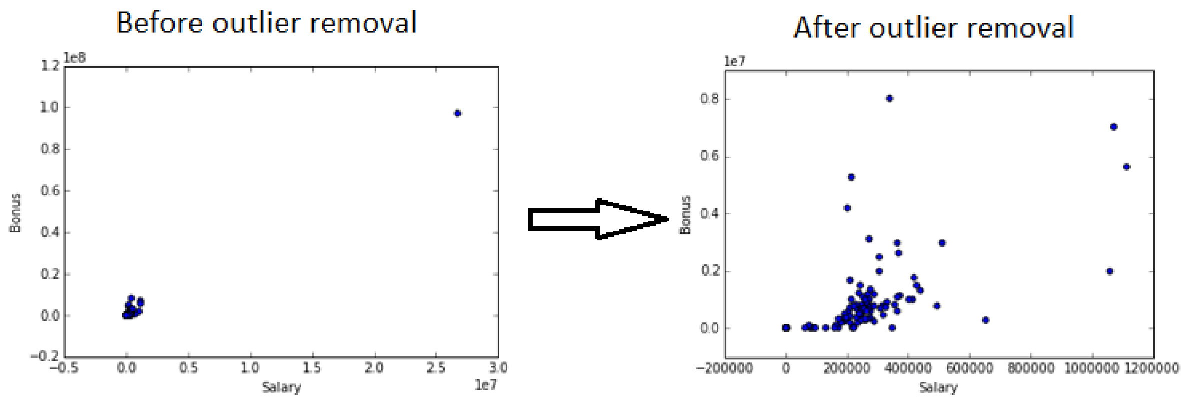
Daniel Wolf
May 1st, 2015

1) **Summarize the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: “data exploration”, “outlier investigation”]**

Enron, formerly one of the largest companies in the US, was dismantled in the early 2000s due to widespread corporate fraud. As part of the legal investigation, a large amount of email and financial data become publicly available. The goal for this project is to create a prediction model for whether an Enron employee is a “Person of Interest” (POI). This can be accomplished by applying machine learning algorithms to the public email and financial data. As the Udacity course states, machine learning brings together computer science and statistics to harness predictive power.

The features for this analysis include 14 financial features (e.g. salary, bonus), 6 email features (e.g. # of to and from messages), and also thousands of text emails that were sent between Enron employees. There are 146 total “observations” for the financial and email features, almost all of which correspond to individual Enron employees. The final feature that is provided is the POI label that shows to whether the person has likely committed fraud. Out of the 146 observations, 18 have the POI label. This is a relatively small sample size for a prediction algorithm, which could present a challenge in this analysis.

One of the first steps was to look for outliers. A quick visualization on Bonus vs Salary reveals a significant outlier in the top right corner of the left graph below. A review of the financial information found that this outlier corresponds to the “TOTAL” line item in the financial information, so it was removed by deleting it immediately after loading the dataset.



Removing outliers above a certain threshold percentage is not valid for this dataset, because some of the most suspect POIs would be removed from analysis. However, a line by line review of the financial data revealed an observation for “THE TRAVEL AGENCY IN THE PARK”. Since this does not correspond to an individual, it was also removed, leaving 144 individuals for the analysis.

The final step I took in exploring the data was to look for data points that did not correspond to a number, which are represented by “NaN” values. The chart below shows that some of the variables (red highlights) have a low percentage of populated values. I should consider excluding high percentage NaN features since they may not provide significant value or information to the dataset as a whole.

Feature	% NaN out of 144
exercised_stock_options	30%
total_stock_value	13%
bonus	44%
salary	35%
sent_to_poi_percent	40%
deferred_income	67%
long_term_incentive	55%
restricted_stock	24%
total_payments	15%
shared_receipt_with_poi	40%
loan_advances	98%
expenses	35%
from_poi_to_this_person	40%
other	37%
rec_from_poi_percent	40%
from_this_person_to_poi	40%
director_fees	89%
to_messages	40%
payments_to_stocks	26%
deferral_payments	74%
from_messages	40%
restricted_stock_deferred	88%

2) **What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that doesn’t come ready-made in the dataset--explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]**

While preparing for feature selection, I first decided to create three new features that I thought would be relevant to predictive ability. Two of these new features were calculated by finding the fraction of emails a person received (or sent) that involved a POI out of the total emails a person received (or sent). If a high percentage of an employee’s emails involved a POI, it may be more likely that they are also a POI. These two new features are called sent_to_poi_percent and rec_from_poi_percent. The third feature I created determined the ratio of total payments to total stock value. I felt this could contribute to the predictive model if there are any POI patterns for payments vs stocks. This feature was called payments_to_stocks.

After creating my new features, I used the univariate feature selection tool SelectKBest to rank my features by their predictive ability. The chart below shows each feature's importance, along with the percentage of NaN values. The blue lines indicate the features that I created, so one of my new features has a relatively high importance. The features that had high percentage of NaN (shown in red) had relatively low impact, which validated my decision to remove them.

Rank	Feature	Importance	% NaN out of 144
1	exercised_stock_options	24.815	30%
2	total_stock_value	24.183	13%
3	bonus	20.792	44%
4	salary	18.290	35%
5	sent_to_poi_percent	16.410	40%
6	deferred_income	11.458	67%
7	long_term_incentive	9.922	55%
8	restricted_stock	9.213	24%
9	total_payments	8.773	15%
10	shared_receipt_with_poi	8.589	40%
X	loan_advances	7.184	98%
11	expenses	6.094	35%
12	from_poi_to_this_person	5.243	40%
13	other	4.187	37%
14	rec_from_poi_percent	3.128	40%
15	from_this_person_to_poi	2.383	40%
X	director_fees	2.126	89%
16	to_messages	1.646	40%
17	payments_to_stocks	0.629	26%
18	deferral_payments	0.225	74%
19	from_messages	0.170	40%
X	restricted_stock_deferred	0.065	88%

The final step I took before applying machine learning algorithms was to scale my features using MinMaxScaler. Algorithms that depend on the distance between data points (e.g. SVM) rely on feature scaling because the financial data can be orders of magnitude larger than the number of emails. Other algorithms, such as Decision Trees, do not require scaling since the features are divided into categories. Feature scaling does not adversely affect algorithms that do not require it, so I went ahead and applied scaling upfront.

I decided to use the top 8 features in my final POI classifier (rank 1 to 8 in the chart above). These features achieve solid performance based on metric evaluation (see section 3). I also considered the Bias-Variance dilemma in my feature selection. Even though the model has acceptable performance with as few as 3 features, I am a little concerned that only 3 features would over-simplify and lead to high bias. Also, only using the top 3 features would exclude any of the email features, which I believe should be a factor in classifying POIs. Alternatively, I did not want to include more features than necessary or I would have issues with high variance and overfitting. Using 8 features is the right balance between selecting a model that passes the performance requirements and does not have too much bias or variance. It also maximizes the recall (see section 6).

3) What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]

I applied several types of algorithms to my dataset using different numbers of features and decided to use GaussianNB with the top 8 features. I applied an exhaustive approach to determine the best number of features and the best algorithm. The chart below reflects my analysis using precision and recall, with the green cells showing every scenario that had greater than 0.3 for both metrics.

Classifier:	**GaussianNB**		**DecisionTree**		**Kneighbors**		**RandomForest**		**AdaBoost**	
# of Features	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
2	0.46889	0.2675	0.22396	0.201	0.68929	0.193	0.30727	0.169	0.39952	0.2515
3	0.48581	0.351	0.3606	0.3835	0.63795	0.2555	0.59617	0.296	0.34211	0.299
4	0.50312	0.323	0.32189	0.333	0.65971	0.2685	0.50416	0.2425	0.31411	0.207
5	0.45558	0.3	0.29325	0.3365	0.64164	0.282	0.4644	0.225	0.27734	0.227
6	0.47723	0.351	0.28292	0.289	0.68675	0.2565	0.47776	0.188	0.33575	0.2545
7	0.46376	0.3615	0.28541	0.268	0.60926	0.2105	0.47301	0.1665	0.31565	0.238
8	0.4537	0.365	0.26786	0.255	0.68883	0.259	0.43296	0.155	0.31357	0.238
9	0.37697	0.311	0.27655	0.293	0.63878	0.168	0.39038	0.138	0.29742	0.225
10	0.35801	0.3095	0.33105	0.314	0.63878	0.168	0.42574	0.1505	0.35567	0.268
11	0.35949	0.3115	0.30016	0.285	0.63878	0.168	0.45633	0.1515	0.37814	0.2785
12	0.35378	0.3085	0.30451	0.287	0.63878	0.168	0.45789	0.1495	0.37977	0.2835
13	0.3562	0.296	0.33436	0.327	0.64636	0.191	0.44113	0.1405	0.39677	0.3075
14	0.34965	0.2965	0.33452	0.329	0.64636	0.191	0.45526	0.145	0.38526	0.298
15	0.31643	0.287	0.31844	0.31	0.64636	0.191	0.41571	0.127	0.38263	0.282

The GaussianNB and DecisionTree classifiers clearly performed better than the other models, at least with the default parameters. This was interesting to me because GaussianNB is one of the simplest machine learning algorithms, and it does not even have parameters to tune. I also attempted an SVC classifier, but it took too long to train so I abandoned that route.

The reason that I chose the GaussianNB classifier with 8 features is that GaussianNB consistently performed the best and 8 features gave the highest recall. I felt it was important to maximize the recall in this exercise (see section 6).

4) What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? (Some algorithms don't have parameters that you need to tune--if this is the case for the one you picked, identify and briefly explain how you would have done it if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Most machine learning algorithms have parameters that can be tuned to improve the performance of the model. Tuning requires a knowledge of the dataset and an understanding of how the algorithms operate, and it helps improve predictive ability by slightly adjusting how the algorithm finds the optimal model. For example, one of the parameters for the DecisionTree algorithm is min_samples_split. This parameter determines whether there are enough samples to continue to split the decision tree further. If min_samples_split is too low, the classifier has the potential to overfit on the training data.

My final classifier (GaussianNB) does not have any parameters. However, I applied parameter tuning to my DecisionTree classifier to see if I could improve its performance. I used GridSearchCV to iterate through several combinations of parameters and find the optimal settings. Based on that analysis, the best combination of parameters was a min_samples_split of 5 and criterion equal to entropy. While this did improve the accuracy from 0.82367 to 0.85127, the precision/recall performance still did not exceed the performance of the GaussianNB classifier.

5) **What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]**

Validation in machine learning tests the accuracy of the predictive model. This is a crucial aspect of creating a good predictive model because I need to understand how well the model performs when applied to new data. A classic mistake when applying validation is using the entire dataset to train the algorithm. This can cause overfitting and there is no way to test how the model performs when applied to new data. So, the dataset should always be split into training and testing data.

In my analysis, I used the tester.py script that was provided with the project. This guaranteed that my algorithm will pass the required metrics and streamlines my efforts so that I can focus on feature and algorithm selection. The testing script generates training and testing data using StratifiedShuffleSplit with 1,000 folds, which is effective for a small sample size. The script also calculates several important metrics that are used to evaluate the performance of the model, including F1 score, precision, recall, and the underlying calculations.

6) **Give at least 2 evaluation metrics, and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]**

The main metrics that I considered when evaluating my model were precision and recall. Precision is the number of correctly-identified POIs (true positives) divided by the number of predicted POIs (true positives plus false positives). Recall is the number of correctly-identified POIs (true positives) divided by the number of actual POIs (true positives plus false negatives). Part of the requirements for completing this project was to achieve precision and recall above 0.3. My chart in section 3 clearly shows that precision and recall stayed above 0.3 consistently for my chosen algorithm and features.

One key point when applying my metric evaluation is that maximizing the recall is more relevant to this project, because getting a higher recall minimizes the false negatives. A false negative is when my model incorrectly does not label an individual as a POI, so low recall makes it more likely that POIs "escape" scrutiny. It is not as important to maximize precision, because false positives can still be found innocent after further inspection. In other words, maximizing recall will cast a wider net for potential POI.

Reflection

Completing this project taught me how to apply machine learning to a real-world example. I found success through ranking features and testing them with different supervised learning algorithms. Even though my model reached the required thresholds, there is still room for improvement. One area that I did not address was the huge library of emails. In a more advanced model, I could have parsed out the text in all these emails and tested for word patterns amongst POIs. This likely could increase both precision and recall. Also, it would be interesting to have a larger dataset of Enron employees, because 144 is only a small sample of all Enron employees. I look forward to applying my new knowledge to other areas to see what insights and predictive ability I can uncover.