Identifying Persons of Interest from the Enron Corpus

Grace Pehl, PhD
Intro to Machine Learning Project
Udacity Data Analyst Nanodegree

1. *Summarize for us 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?]*

The goal of this project is to explore some of the tools and techniques of machine learning while trying to identify persons of interest (POIs) in the Enron scandal. This project uses information from the Enron corpus, which contains the email messages of about 150 people, mostly senior management of Enron. The corpus was made public by the US Federal Energy Regulatory Commission during its investigation into the collapse of Enron. It is widely used in machine learning(*).

Exploration of the dataset found 146 entries including 18 flagged as a POI. Each entry had 21 possible features. While all entries contained a POI flag, the remaining each of the remaining features had a missing value for at least 20 people.

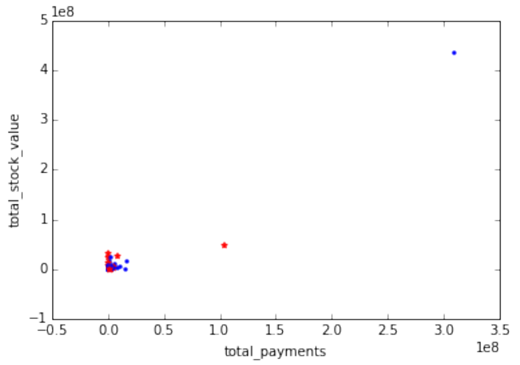
Number of People Missing a Value for Each Feature:

Feature		**Feature**		**Feature**		**Feature**		**Feature**	
Salary	51	Deferral payments	107	Loan advances	142	Total payments	21	Emails sent	60
Bonus	64	Exercised stock options	44	Director fees	129	Total stock value	20	Emails received	60
Expenses	51	Restricted stock	36	Deferred income	97	Email address	35	Emails from POI	60
Other	53	Restricted stock deferred	128	Long term incentive	80	Emails sent also to POI	60	Emails to POI	60

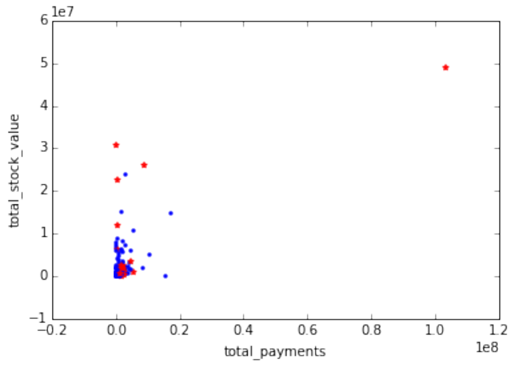
The features can be divided into financial features and email features, which came from different sources. Sixty of the people in the dataset had no values for the email features (including 4 Persons of Interest) and 3 people had no values for the finanical features (Ronnie Chan, William Powers, and Eugene E. Lockhart, none of whom are Persons of Interest).

The smallest numbers of missing entries were the financial features 'total_payments' and 'total_stock_value'. A scatterplot of these features revealed one point with values much larger than the other points. Comparing the values to the financial information given in 'enron61702insiderpay.pdf' revealed that the point belonged to 'TOTAL'. This point was removed from the dictionary of data as being an artifact of the spreadsheet. A second scatterplot revealed a different point with values much larger than all of the other points. This point belonged to Kenneth Lay, a Person of Interest. This very meaningful point was left in the dataset.

Outlier point is “TOTAL”

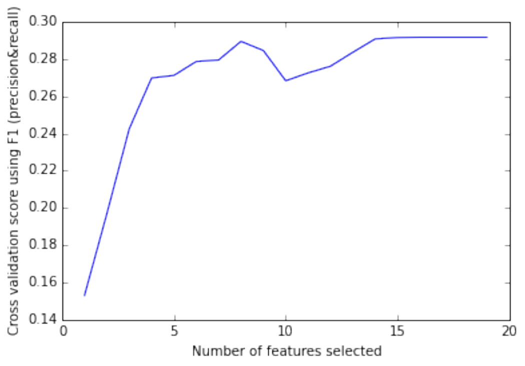


Outlier point is “LAY KENNETH K”

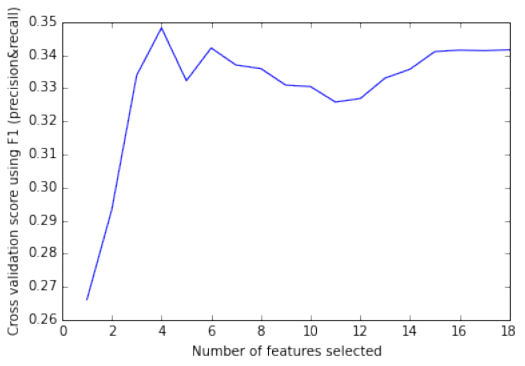


2. *What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature 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.) In your feature selection step, f you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores. [relevant rubric items: ?create new features?, ?properly scale features?, ?intelligently select feature?]*

To select features, I used recursive feature elimination with cross-validation. For each feature, a coefficient was calculated assuming that all other features were held constant. This was done by a linear support vector classifier. This classifier requires feature scaling. All features were scaled by setting the minimum value to 0 and the maxium to 1. Min-Max scaling preserves zero values in the data, which is useful in this case because missing values are coded as zero within the featureFormat function. After the coefficients are calculated, the feature whose coefficient is closest to zero is discarded and the process is repeated. At each step, the overall model is evaluated using F1 score as a metric (discussed in question 6). The optimal number of features was found to be 16 features and the features 'loan advances', 'total payments', and 'other' were eliminated. The F1 score for this model was 0.292, but individual feature coefficients are not returned from this function. Feature importances will be reported during classifier selection (question 3),



Each email-related feature had a very large range of values and it makes sense that the number of emails a person sent to POIs is going to be related to the number of emails that they ususally send. I created new features 'fraction_to_poi' and 'fraction_from_poi' as the proportion of email messages to and from POIs. I added the two engineered features to those selected above and ran the recursive feature elimination again. This time, the best results came from using only 4 features and the F1 score improved from 0.292 to 0.332. The final features were: 'expenses', 'exercised stock options', the number of emails where the person was a joint receipent with a POI, and one of the engineered features, 'fraction to POI'. This is the feature set that will be used to select a classification algorithm.



3. *What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]*

Ensemble methods do well at kaggle competitions, so I tried two of the most popular algorithms: random forest and adaboost. In random forest, a series of decision trees are built on randomized subsets of the data. At each node, a randomized subset of variables is evaluated to find the best split. This creates a "forest" of independent, randomized trees. The predictions of all the trees are averaged to assign a predicted label. I started with the default options, though I set class weights to be inversely proportional to their representation in the dataset. AdaBoost is also an ensemble of decision trees, but the trees are not independent. First, a single tree is fit using all of the data. Incorrectly assigned points are then given a higher weight (boosted) and the process is repeated. By default, adaboost uses trees of depth 1, single feature trees called decision stumps.

Model	Time	F1	Precision	Recall
Random Forest	1.166s	0.307	0.443	0.235
AdaBoost	5.472s	0.505	0.554	0.465

AdaBoost was better than random forest according to each metric, but took almost 5 times as long. I don't want to give up on random forest, so I will attempt to tune both algorithms.

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 do not 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 for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: "tune the algorithm"]*

Tuning the parameters of an algorithm refers to adjustments made when training the algorithm to improve the fit on the test data set. It is very easy to overfit the training data by including too many features, but such an algorithm will not make good predictions. Overfitting can be avoided by holding make a subset of the data to test the algorithm's predictions against.

The best parameters to tune for random forest are the number of trees built and the number of parameters evaluated at each node. With 100 trees and 4 features (all of them), the F1 score improved from 0.307 to 0.344, which is still not as good as adaboost. For adaboost, the main parameters for tuning are the number of weak estimators and their complexity. In this case the estimators are decision trees, so I tune the number of them and their maximum depth. With a maximum depth of 1 (the default) and 70 estimators, adaboost achieved an F1 score of 0.475. Since the default options were included in the grid search parameters, I don't know why the score decreased, but it may be because I am using a smaller number of iterations in the testing function.

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 refers to checking the algorithm's predictions against data that was not used for training the algorithm to prevent the algorithm from simply memorizing the training data (overfitting). However, this dataset is already very small with only 14 Persons of Interest. A single split into a training and test set would not give a very accurate estimate of error. Instead, throughout this analysis, I use the StratifiedShuffleSplit function to randomly split the data into 30% test set and 70% training while keeping the fraction of POIs in each split relatively constant.

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 final model (AdaBoost) had precision = 0.56, recall = 0.49, and F1 = 0.52. This means that given a Person of Interest, the algorithm will recognize it almost of the time and that 56% of POIs flagged by the algorithm will have actually pled guilty to a crime connected to the Enron scandal. The F1 score is a combination of the two metrics, F1 = 2 * (precision * recall) / (precision + recall).