Udacity Introduction to Machine Learning Project **Greg Hein - 6/19/2016**

Identify Fraud from Enron Email

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?

The dataset used in this project contains information gathered from the Enron Email Dataset (http://www.cs.cmu.edu/~enron/). According to Wikipedia: "Enron Corporation was an American energy, commodities, and services company based in Houston, Texas" (USA). In 2001 it was revealed that the company's "financial condition was sustained by an institutionalized, systematic, and creatively planned accounting fraud". During the subsequent investigations by the U.S. Securities and Exchange Commission and the Federal Energy Regulatory Commission, email data from employees of Enron (mostly senior management) were made public. After the investigation, certain employees were found guilty of varying degrees of involvement in the fraud, and those persons are labeled as "persons of interest" or "poi" in our dataset.

The goal of this project is to use machine learning, specifically the scikit-learn library for the Python programming language, to create a classifier that will attempt to recognize Enron persons of interest given the other feature data in the dataset. There are 146 data points (Enron employees) in the original dataset. Of those 146, 18 are persons of interest and 128 are not poi's. There are 21 features for each person in the original dataset. The features given for each employee in the dataset are given in the following table, along with the percent of missing ('NaN') values for each feature.

Feature	% Missing Values	Feature	% Missing Values
Salary	34.93	Loan Advances	97.26
To Messages	41.10	From Messages	41.10
Deferral Payments	73.29	Other	36.30
Total Payments	14.38	From This Person To POI	41.10
Exercised Stock Options	30.14	POI	00.00
Bonus	43.84	Director Fees	88.36
Restricted Stock	24.66	Deferred Income	66.44
Shared Receipt With POI	41.10	Long Term Incentive	54.79
Restricted Stock Deferred	87.67	Email Address	23.97
Total Stock Value	13.70	From POI To This Person	41.10
Expenses	34.93		

After inspecting the dataset, two of the data points were recognized not to be actual Enron employees, and obviously could not be a person of interest (or a non-poi). One of the data points was “Total”, and the other was “The Travel Agency In The Park”. These outliers were removed from the dataset.

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.

The features chosen for the POI identifier are as follows: 'poi', 'salary', 'expenses', 'shared_receipt_with_poi', 'deferred_income', 'total_stock_value', 'exercised_stock_options', and 'sum_fractions_to_and_from'. After running SelectKBest for multiple values of K with 4 different Machine Learning algorithms, certain features repeatedly came up as good choices to help identify persons of interest, and those were hand-picked and used in the final identifier. There was no scaling done in the final identifier, as the algorithm chosen (AdaBoost with default base estimator of Decision Tree) did not benefit from scaling.

Three features were engineered that did not come ready-made in the dataset. The fraction of each person's emails that were to a POI and from a POI ('fraction_to_poi' & 'fraction_from_poi') were calculated and added to the dataset. These were chosen with the theory that persons of interest likely had a higher percentage of their emails sent both to and from other persons of interest. Also, those numbers were summed to give a combination of email fractions to and from persons of interest ('sum_fraction_to_and_from') and that feature was also added to the dataset. Assuming the two fractions were helpful in identifying persons of interest, a combination of them might be an even more powerful predictor. In testing various identifiers, the sum of fractions of emails to and from POI appeared to be the best of the three at helping identify POI's and, as listed above, was used in the final POI classifier.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

Four different algorithms were tested as POI identifiers. Those four were: Gaussian Naive Bayes, Decision Tree, Random Forest, and AdaBoost (with default base estimator of Decision Tree). After trying both Principle Component Analysis and SelectKBest in combination and separately, the features used in the final analysis were the target feature of 'poi', 6 of the other original features, and one of the features that was engineered as listed above. The same features were then used to compare each algorithm. Multiple tuning parameters were tested using Grid Search, and the best performance found for each algorithm is given in the table at the top of the next page.

Algorithm	Accuracy	Precision	Recall
AdaBoost	0.87	0.53	0.42
Gaussian Naive Bayes	0.87	0.49	0.39
Decision Tree	0.83	0.37	0.38
Random Forest	0.87	0.51	0.32

The best performing algorithm with respect to Precision and Recall was the AdaBoost classifier, and that was chosen as the final POI classifier. From the scikit-learn documentation, AdaBoost is “is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases”.

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?

Parameter tuning is choosing the settings of the parameters that affect the behavior of a machine learning algorithm, with the goal of optimizing the algorithm's performance on the data set. If done well it can improve the performance of the algorithm. If not done well it can lead to over-fitting. In scikit-learn, all algorithms that have parameters that can be tuned will have a default setting for those parameters. The process followed with the four algorithms tested was to first run the algorithm with the default settings. After that Grid Search was used to test different parameter settings. Grid search is an approach to parameter tuning that will systematically go through each combination of algorithm parameters specified in a grid. This helped to inform the final choices of parameter settings. The final algorithm chosen for the POI classifier, AdaBoost, was used with all default settings except one. The parameter “n_estimators”, which is the maximum number of estimators at which boosting is terminated, was changed from the default setting of 50. The performance for different numbers of estimators is summarized in the table below.

Number of estimators	Accuracy	Precision	Recall
30	0.86	0.47	0.38
50	0.87	0.51	0.41
70	0.87	0.52	0.42
90	0.87	0.53	0.42
100	0.87	0.53	0.41
120	0.87	0.52	0.40

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is the process by which a statistical model (in our case the POI classifier) is evaluated to see how well it would generalize to an independent data set. If done incorrectly, it will often lead to over-fitting. This would typically be the result if using the same data to train the model and test the model. To avoid this over-fitting, in supervised machine learning it is common practice to train the model on a portion of the data and then test the model on the remaining data. This was done in the final POI classifier by utilizing scikit-learn's "train_test_split" function training on 30% of the data and testing on the remaing data. Additionally, the evaluation program provided (tester.py) uses scikit-learn's StratifiedShuffleSplit method which takes validation a step further and uses multiple stratified random splits of the data into training and testing sets. The program tester.py was utilzed as a final check on the classifier.

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.

The two primary evaluation metrics used in this project are *precision* and *recall*. The requirements for completion of the project was at least a measurement of 0.3 for both of the evaluation metrics precision and recall. As shown previously all of the algorithms tested surpassed this benchmark, with AdaBoost giving the highest values of precision and recall.

Precision is a measure of how many of the data points that are classified as positive are positive. The measure of precision given by the POI identifier chosen is 0.52. This would mean that out of every 100 persons that are classified as a person of interest by the classifier, 52 of them would actually be persons of interest.

Recall is a measure of how many of the data points that are positive are classified as positive. The measure of recall given by the POI identifier chosen is 0.42. This would mean that for every 100 persons that are persons of interest, 42 would be classified as a person of interest.