

<i>Description from Project Submission Page:</i>	<i>Feedback from Round 1 Submission:</i>
<p>Document the work you've done by answering (in about a paragraph each) the questions found <a href="#">here</a>. ...</p> <p>Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response!</p>	<p>In the written response, ideally every step/choice of parameters and algorithm selection, tuning or feature selection should be followed by a table with the adequate figures reporting the impact of that specific step/choice on the chosen performance metrics and a consequent rationale for the choices made in the light of those results. .... It is very important, in every scientific work, to explain and justify every step/choice as thoroughly as possible, no matter how obvious to you the various steps might look.</p>

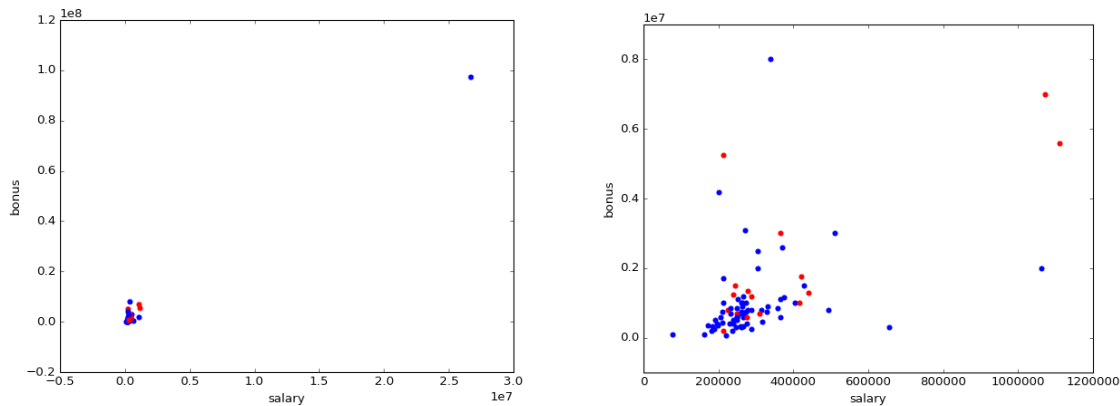
### Enron Submission Free-Response Questions

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 the project is to find patterns in the data that may be used to signal whether or not a person was involved in fraud at Enron. The data used in my code has financial data, such as bonus and salary, as well as summary email data, such as the number of emails from that person. This data was compiled by Katie at Udacity or her colleagues using publicly available financial and email information of Enron employees or contractors.

```
Number of records: 145
Number of poi: 18
Number of non-poi: 127
Number of Features: 21
```

I explored the data using 2-D visuals and max check to see if there were any outliers. The financial outlier 'TOTAL' popped up, which de-ja-vu we discovered during the lessons. I removed it from the working dictionary before formatting the features for the machine learning algorithms. Here are the before and after plots of salary and bonus, where the first contains the outlier.



I continued to explore 2D plots without prominent discovery. I then took note of missing values in the data, signified by 'NaN'. I saw that there was very little data for loan\_advances, restricted\_stock\_deferred, and director\_fees.

```

Number of missing data per feature
59 - to_messages
107 - deferral_payments
64 - bonus
20 - total_stock_value
51 - expenses
59 - from_this_person_to_poi
0 - poi
97 - deferred_income
34 - email_address
59 - from_poi_to_this_person
51 - salary
21 - total_payments
80 - long_term_incentive
142 - loan_advances
36 - restricted_stock
128 - restricted_stock_deferred
59 - shared_receipt_with_poi
44 - exercised_stock_options
59 - from_messages
53 - other
129 - director_fees

```

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, if 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 and reasons for your choice of parameter values. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

I created 3 parameters based on the existing data. Since the number of emails per person could be on different scales, I used a ratio in place of the number of emails to and from a person of interest so that these would be in balance with the person's email activity. I also used ratio in place of exercised stock options, because that activity would be bound by the individual's held stock value. Ultimately the ratio for exercised stock options didn't seem to be used by the classifiers.

The SVM classifiers did not work, so I tried a scaler on the features for them but it didn't help. I did also use scaled values for testing K-neighbors algorithm.

In my initial exploration, I used all the features (except email address) to get a feel for how to configure the algorithms, how was the performance, and what was feasible. I also checked the importance of features in the AdaBoost and GridSearch Decision Tree.

<b>DT Importance:</b> to_messages : 0.0 email_from_poi_ratio : 0.0 restricted_stock_deferred : 0.0 deferral_payments : 0.0 bonus : 0.0 total_stock_value : 0.0746007468271 expenses : 0.208956631796 from_this_person_to_poi : 0.0  deferred_income : 0.0987801021886 shared_receipt_with_poi : 0.0  long_term_incentive : 0.0 salary : 0.0 total_payments : 0.0 loan_advances : 0.0 restricted_stock : 0.0 email_to_poi_ratio : 0.0 from_poi_to_this_person : 0.0  exercised_stock_options : 0.0 from_messages : 0.0372618896376 other : 0.580400629551 exer_stock_ratio : 0.0 director_fees : 0.0	<b>AB Importance:</b> to_messages : 0.0197593425823 email_from_poi_ratio : 0.0277077693591 restricted_stock_deferred : 0.0 deferral_payments : 0.0247863342737 bonus : 0.0402258590179 total_stock_value : 0.0700948931873 expenses : 0.0957178645192 from_this_person_to_poi : 0.0391803527522 deferred_income : 0.0225017147503 shared_receipt_with_poi : 0.0224070400598 long_term_incentive : 0.0242565783808 salary : 0.113657959528 total_payments : 0.0519006594209 loan_advances : 0.0 restricted_stock : 0.0335747421505 email_to_poi_ratio : 0.0553214705143 from_poi_to_this_person : 0.0386487187755 exercised_stock_options : 0.245867435538 from_messages : 0.0 other : 0.0558384513263 exer_stock_ratio : 0.0185528138636 director_fees : 0.0
--	--

Then I ran a series of k-fold tests using different sets of features. First, I used all the features. Then I tried using the features I noted during my initial exploration and features based on importance above.

All of the features:

'to\_messages', 'email\_from\_poi\_ratio', 'restricted\_stock\_deferred', 'deferral\_payments', 'bonus', 'total\_stock\_value', 'expenses', 'from\_this\_person\_to\_poi', 'deferred\_income', 'shared\_receipt\_with\_poi', 'long\_term\_incentive', 'salary', 'total\_payments', 'loan\_advances', 'restricted\_stock', 'email\_to\_poi\_ratio', 'from\_poi\_to\_this\_person', 'exercised\_stock\_options', 'from\_messages', 'other', 'exer\_stock\_ratio', 'director\_fees'

Features based on exploration intuition:

'salary', 'bonus', 'total\_payments', 'exercised\_stock\_options', 'shared\_receipt\_with\_poi', 'expenses', 'email\_to\_poi\_ratio', 'email\_from\_poi\_ratio', 'exer\_stock\_ratio'

Features based on example DT importance:

'other', 'expenses', 'total\_stock\_value', 'exercised\_stock\_options', 'long\_term\_incentive',  
'from\_this\_person\_to\_poi', 'from\_messages', 'restricted\_stock'

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”]

I tried Naïve bayes, AdaBoost Decision Tree, Grid Searched K-neighbors, and Grid Searched Decision Tree. I also tried a variety of SVM configurations using Grid Searches, but in some cases the SVM failed and in others it got hung, so I omitted it. To compare performance between the other algorithms, I measured the scores for the various models with various features lists. Ultimately, I chose a Decision Tree that was discovered through the Grid Search, and had interesting tuning (described in #4). Naïve Bayes and the chosen Decision Tree both offered pretty good performance compared to others.

**Test with all of the features:**

Gaussian NB Classifier

Precision 0.204848484848 , Recall 0.466666666667 , F1 0.27380952381

Ada Boost Decision Tree Classifier

Precision 0.283333333333 , Recall 0.183333333333 , F1 0.206666666667

Grid Search - Best K Neighbors Classifier

Precision 0.3 , Recall 0.25 , F1 0.25

Grid Search Decision Tree Classifier

Precision 0.225 , Recall 0.133333333333 , F1 0.161904761905

**Test with features based on exploration intuition**

Gaussian NB Classifier

Precision 0.333333333333 , Recall 0.333333333333 , F1 0.316666666667

Ada Boost Decision Tree Classifier

Precision 0.2 , Recall 0.133333333333 , F1 0.156666666667

Grid Search - Best K Neighbors Classifier

Precision 0.3 , Recall 0.203333333333 , F1 0.206666666667

Grid Search Decision Tree Classifier

Precision 0.234487734488 , Recall 0.88 , F1 0.350665445665

**Test with features based on example DT importance**

Gaussian NB Classifier

Precision 0.3 , Recall 0.25 , F1 0.25

Ada Boost Decision Tree Classifier

Precision 0.35 , Recall 0.266666666667 , F1 0.296666666667

Grid Search - Best K Neighbors Classifier  
Precision 0.4 , Recall 0.233333333333 , F1 0.283333333333

Grid Search Decision Tree Classifier  
Precision 0.341666666667 , Recall 0.825 , F1 0.444047619048

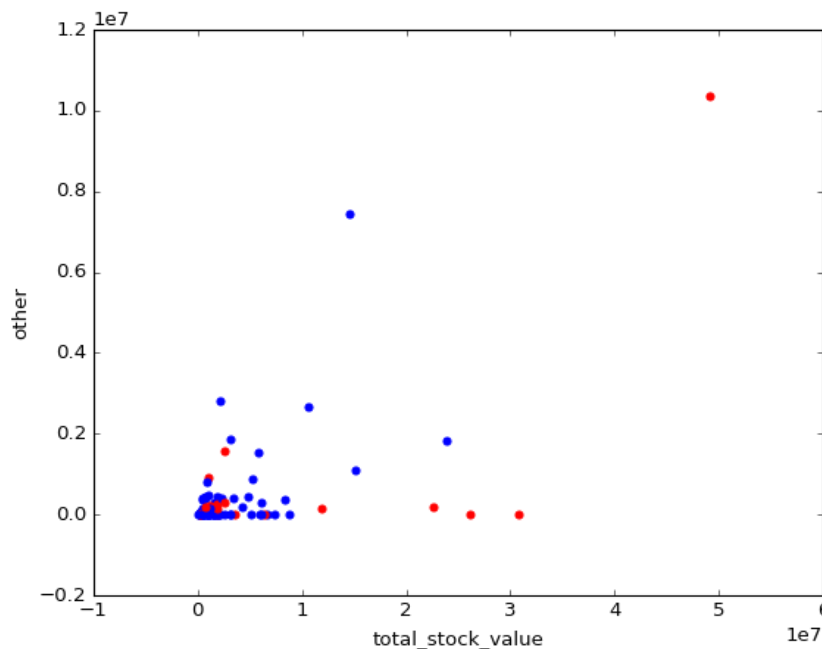
4. What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain 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"]

There are a variety of tuning parameters that apply to different algorithms that affect how the algorithm will make choices. Tuning can achieve things like preventing over-fitting or underfitting, improving execution time, and assigning priority to classes. The settings may be specific to the problem; for example, we have much fewer positive POIs than negative in the Enron Data Set. I used Grid Search to evaluate different combinations of parameters for the Decision Tree model. The parameters that I chose to vary were 'max\_depth' and 'min\_samples\_split' for how well to fit the training data, 'criterion' to see if gini or entropy will be better for classification of this problem, and 'class\_weight' since we had unbalanced classes. I specified 'f1' to be the scoring method of the Grid Search for choosing the best decision tree.

The best tuning parameters for the decision tree were:

min\_samples\_split: 2 - criterion: 'gini' - max\_depth: 2 - class\_weight: False = 1, True = 4

The class weight for positive POI's was higher which helped identify them since they were sparser. Interestingly, the max\_depth was 2, and the features 'other' and 'total\_stock\_value' were used by the decision tree to get good results. I wonder if there is an accidental 'signature' in 'other', but it would take much more investigation. The documentation indicates that 'other' is for "items such as payments for severance, consulting services, relocation costs, tax advances and allowances..", etc.



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 is testing how well the model has been trained. Scores are obtained during validation that can be used to help choose models and also to communicate the performance of the model.

Validation can be wrong when using the same data that trained the classifier, to test the classifier. When the classifier over-fits to the training data, it will appear to have high accuracy. Another thing we are looking at when validating our Enron data set is that the occurrence of positive nodes is sparse. In those cases, only looking at accuracy will make the algorithm appear to have good performance when it frequently returns negative.

To do good validations, separate data should be used to test the algorithm, that wasn't used to train the algorithm. Since the available data to work with is limited, the data set must be split in to a training set and a testing set. Furthermore, the model can be measured using the same data set over and over again, but with different split permutations. I used a fixed split of data as I explored different classifiers, and then I used 10 k-fold testing to get a closer look at the performance of classifiers with different permutations. When I looked at the performance, instead of looking at accuracy, I focused on Precision and Recall, below. These measures are especially helpful with measuring performance of predicted classification of the skewed data set.

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"]

Precision and Recall are especially important metrics for the sparse POI classifier.

Precision measures how likely the classifier is to be correct when it predicts a person of interest. In other words, it represents the fraction of predicted positives that are actually correct. The more precision the model gets, the more you can be sure that a prediction of a positive label is correct. My decision tree has a precision of .34. So, given a data point that the classifier predicts is a person of interest, the chances that this data point is actually a person of interest is 34%.

Recall measures how likely the classifier is to correctly identify a person of interest when it sees an actual POI. Whereas precision represents the fraction of predicted positives, recall is based on the fraction of actual positives that are correct. The higher the recall score for the model, the less likely the model will miss prediction of positive labels. My Grid Search DT has an average recall of .8, so it will correctly identify 80% of the persons of interest it sees. So, in a given set of data to predict on, the model is expected to fail identifying 20% of the persons of interest.