Kathleen Cravotta December 2016

Description from Project Submission Page:	*Feedback from Round 1 Submission:*
Document the work you've done by answering (in about a paragraph each) the questions found **here**. … 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!	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.

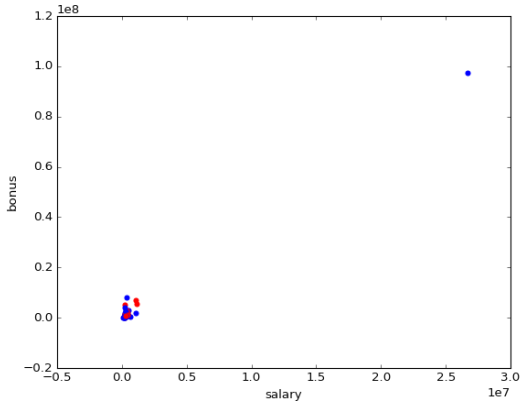
Enron Submission Free-Response Questions

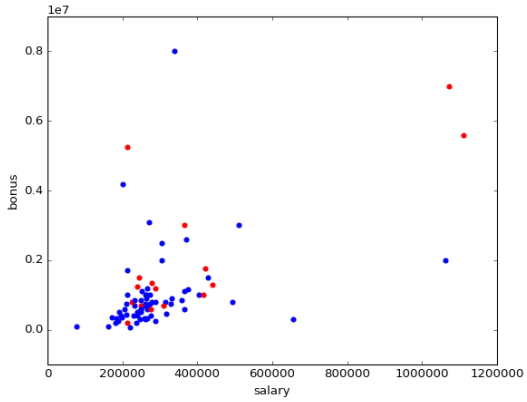
1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [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 th at 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 th e 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 features 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 based on of exercised stock options, because that activity would be bound by the individual's held stock value. As per below, we can see that the new features had a small impact on the performance of the Decision Tree.

Results WITHOUT crafted features (all features – multiple algorithms):

```
++++++++++++++++++++++++++++++++++++
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 0.344848484848 , Recall 0.758333333333 , F1 0.391245421245
```

Results WITH crafted features (all features – multiple algorithms):

```
++++++++++++++++++++++++++++++++++++
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 0.425 , Recall 0.408333333333 , F1 0.34
```

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.

Preliminary Exploration: 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 Decision Trees and correlations in K Best. They actually had very different results. For example, the decision trees determined that 'other' was important, while K Best features did not see a big correlation

DT Importance:	AB Importance:	K Best Features:
to_messages : 0.0	to_messages : 0.0197593425823	to_messages - 1.07304130128
email_from_poi_ratio : 0.0	email_from_poi_ratio : 0.0277077693591	email_from_poi_ratio - 1.10880861696
restricted_stock_deferred : 0.0	restricted_stock_deferred : 0.0	restricted_stock_deferred - 0.0914782577338
deferral_payments : 0.0	deferral_payments : 0.0247863342737	deferral_payments - 0.0845550236097
bonus : 0.0	bonus : 0.0402258590179	bonus - 12.8824406178
total_stock_value : 0.0746007468271	total_stock_value : 0.0700948931873	total_stock_value - 28.3846130519
expenses : 0.208956631796	expenses : 0.0957178645192	expenses - 4.60786666878
from_this_person_to_poi : 0.0	from_this_person_to_poi : 0.0391803527522	from_this_person_to_poi - 0.0038014279063
deferred_income : 0.0987801021886	deferred_income : 0.0225017147503	deferred_income - 10.5894517626
shared_receipt_with_poi : 0.0	shared_receipt_with_poi : 0.0224070400598	shared_receipt_with_poi - 5.24581390225
long_term_incentive : 0.0	long_term_incentive : 0.0242565783808	long_term_incentive - 6.74513952871
salary : 0.0	salary : 0.113657959528	salary - 13.0205111495
total_payments : 0.0	total_payments : 0.0519006594209	total_payments - 7.45704181186
loan_advances : 0.0	loan_advances : 0.0	loan_advances - 7.03793279819
restricted_stock : 0.0	restricted_stock : 0.0335747421505	restricted_stock - 9.66060392085
email_to_poi_ratio : 0.0	email_to_poi_ratio : 0.0553214705143	email_to_poi_ratio - 1.34557028619
from_poi_to_this_person : 0.0	from_poi_to_this_person : 0.0386487187755	from_poi_to_this_person - 0.861094988258
exercised_stock_options : 0.0	exercised_stock_options : 0.245867435538	exercised_stock_options - 30.1478672127
from_messages : 0.0372618896376	from_messages : 0.0	from_messages - 0.512098795839
other : 0.580400629551	other : 0.0558384513263	other - 3.47044692251
exer_stock_ratio : 0.0	exer_stock_ratio : 0.0185528138636	exer_stock_ratio - 0.584189962463
director_fees : 0.0	director_fees : 0.0	director_fees - 1.43893345283

After the initial exploration, I used a K Best algorithm to pick the best features. A list of 7 features worked best with the Gaussian Naïve Bayes model, and a list of 14 parameters worked best for the Decision Tree.

K best features	Gaussian NB	Best K Neighbors	Best Decision Tree	Ada Boost Decision Tree

1	0.40 | 0.33	0.15 | 0.08	0.15 | 0.44	0.35 | 0.23
2	0.40 | 0.28	0.10 | 0.03	0.07 | 0.15	0.25 | 0.13
3	0.38 | 0.32	0.15 | 0.08	0.17 | 0.64	0.35 | 0.23
4	0.42 | 0.42	0.35 | 0.16	0.16 | 0.42	0.20 | 0.17
5	0.38 | 0.42	0.38 | 0.26	0.17 | 0.23	0.25 | 0.30
6	0.37 | 0.43	0.43 | 0.26	0.28 | 0.36	0.13 | 0.17
7	0.41 | 0.48	0.30 | 0.23	0.25 | 0.33	0.23 | 0.23
8	0.35 | 0.42	0.30 | 0.23	0.13 | 0.44	0.23 | 0.23
9	0.35 | 0.38	0.10 | 0.15	0.13 | 0.44	0.23 | 0.23
10	0.35 | 0.38	0.10 | 0.15	0.12 | 0.34	0.23 | 0.18
11	0.35 | 0.38	0.10 | 0.15	0.29 | 0.82	0.20 | 0.13
12	0.35 | 0.38	0.10 | 0.15	0.32 | 0.66	0.30 | 0.18
13	0.35 | 0.42	0.10 | 0.15	0.32 | 0.66	0.30 | 0.18
14	0.35 | 0.42	0.10 | 0.15	0.34 | 0.76	0.28 | 0.18
15	0.35 | 0.42	0.10 | 0.15	0.32 | 0.66	0.28 | 0.18
16	0.35 | 0.42	0.10 | 0.15	0.32 | 0.66	0.28 | 0.18
17	0.35 | 0.42	0.10 | 0.15	0.34 | 0.66	0.28 | 0.18
18	0.35 | 0.42	0.10 | 0.15	0.33 | 0.66	0.28 | 0.22
19	0.28 | 0.42	0.10 | 0.15	0.24 | 0.33	0.18 | 0.15
20	0.25 | 0.57	0.10 | 0.15	0.30 | 0.43	0.28 | 0.18
21	0.20 | 0.47	0.30 | 0.25	0.25 | 0.33	0.28 | 0.18
all	0.20 | 0.47	0.30 | 0.25	0.42 | 0.41	0.28 | 0.18

Best features for Gaussian NB: ['poi', 'bonus', 'total_stock_value', 'deferred_income', 'salary', 'total_payments', 'restricted_stock', 'exercised_stock_options']

Best features for DT: ['poi', 'bonus', 'total_stock_value', 'expenses', 'deferred_income', 'shared_receipt_with_poi', 'long_term_incentive', 'salary', 'total_payments', 'loan_advances', 'restricted_stock', 'email_to_poi_ratio', 'exercised_stock_options', 'other', 'director_fees']

When using a decision tree, I still got better results by simply looking at the initial decision tree's importance ranking. While the Ada Boosting did not help, I was able to use its feature importance rankings to pin down what works well in a decision tree. The values that the Ada Boost decision trees deemed important (at over .07) were: 'poi', 'other', 'expenses', 'total_stock_value', 'exercised_stock_options'.

```
Importance:
other : 0.206906967731
expenses : 0.793093032269
total_stock_value : 0.0
exercised_stock_options : 0.0

{'min_samples_split': 2, 'criterion': 'gini', 'max_depth': 2, 'class_weight':
{False: 1, True: 12}}
```

I used these features when searching for a Decision Tree with Grid Search. Since only 2 of these features were deemed important by the Decision Tree, I was simply able to reduce the features list down to: 'poi', 'other', 'expenses'.

```
DecisionTreeClassifier(class_weight={False: 1, True: 12}, criterion='gini',
    max_depth=2, max_features=None, max_leaf_nodes=None,
    min_impurity_split=1e-07, min_samples_leaf=1,
    min_samples_split=2, min_weight_fraction_leaf=0.0,
    presort=False, random_state=None, splitter='best')
Accuracy: 0.76753         Precision: 0.34749        Recall: 0.84700
F1: 0.49280      F2: 0.65786
Total predictions: 15000  True positives: 1694     False positives: 3181
False negatives:  306     True negatives: 9819
```

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"]

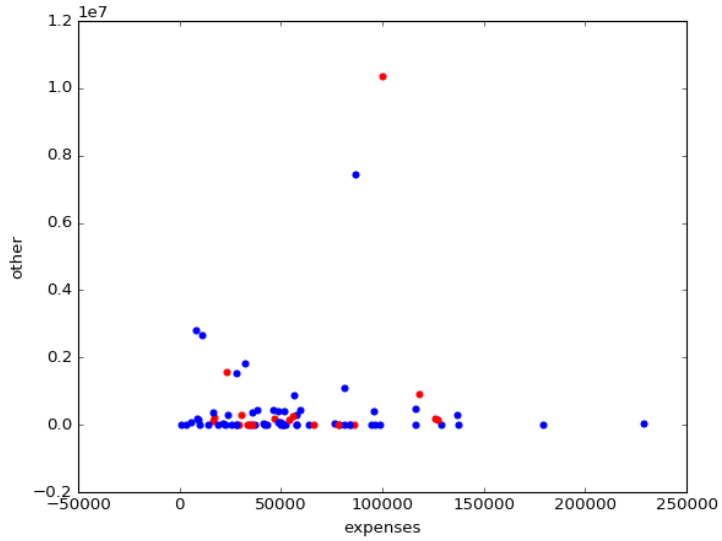
I tried Naïve bayes, AdaBoost Decision Tree, Grid Searched K-neighbors, and Grid Searched Decision Tree. I also tried a variety of SVM configurations using Grid Searches, but in some cases the SVM failed and in others it got hung, so I omitted it. To compare performance between the other algorithms, 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.

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 = 12

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 'expenses' were used by the decision tree to get good results. 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 faction 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.