# P5 – Evaluating the Enron DataSet, R2

Judy Jensen

## Q1: Summarize the project Dataset and Questions:

The question with the Enron dataset is if one can use machine learning to build a model that predicts if a person in the dataset is a person of interest in the Enron case (involved in the legal proceedings with a status of: turned witness, found guilty, penalty without admitting guilt). Notably, the task in this project is to identify outliers

The dataset includes 144 people and 21 features that can be categorized into financial features, email features and the poi feature. In addition, there are 17 additional POIs that do not have financial features available and are not included in the analysis. 12% of the people in the dataset are POIs. Because the dataset is small and I am trying to identify a small portion of the people, recall and precision are the relevant evaluation metrics.

Utilized exploratory data analysis – specifically stats.describe and scatterplots to identify features of interest and outliers.

Most features have very high spreads, which may lead to scaling depending on the algorithm used, or some feature engineering. There are several features with very few data points, and these will be investigate further to perhaps remove them (highlighted in yellow)

| Feature | Count | min | max | mean |
|---|---|---|---|---|
| bonus | 81 | 70000 | 8.00E+06 | 1.20E+06 |
| deferral_payments | 38 | -102500 | 6.43E+06 | 8.42E+05 |
| deferred_income | 48 | -3504386 | -833 | -5.81E+05 |
| director_fees | 16 | 3285 | 137864 | 89822.875 |
| exercised_stock_options | 101 | 3285 | 3.43E+07 | 2.96E+06 |
| expenses | 94 | 148 | 2.29E+05 | 5.42E+04 |
| from_messages | 86 | 12 | 1.44E+04 | 6.09E+02 |
| from_poi_to_this_person | 74 | 1 | 528 | 75.41891892 |
| from_this_person_to_poi | 66 | 1 | 609 | 53.72727273 |
| loan_advances | 3 | 400000 | 8.15E+07 | 2.80E+07 |
| long_term_incentive | 65 | 69223 | 5.15E+06 | 7.46E+05 |
| restricted_stock | 109 | -2604490 | 1.48E+07 | 1.15E+06 |
| restricted_stock_deferred | 17 | -1787380 | 15456290 | 621892.8235 |
| salary | 94 | 477 | 1.11E+06 | 2.84E+05 |
| shared_receipt_with_poi | 86 | 2 | 5.52E+03 | 1.18E+03 |
| to_messages | 86 | 57 | 1.51E+04 | 2.07E+03 |
| total_payments | 124 | 148 | 1.04E+08 | 2.62E+06 |
| total_stock_value | 125 | -44093 | 4.91E+07 | 3.35E+06 |

There are a few notable outliers. First, is the “Total” key which was erroneously introduced when the financial data was merged with the Enron emails. Also, TRAVEL AGENCY IN THE PARK is intuitively not a person.  These two keys are removed from the dataset.  The other outliers contain important information on the POIs, and so are maintained.

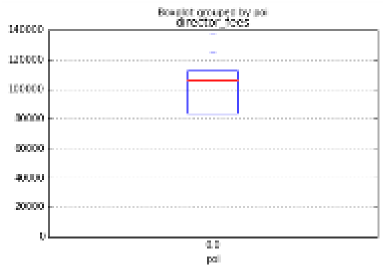
## Q2:  Feature Selection

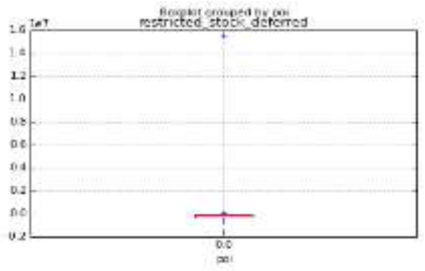
The goal is to use the fewest number of features to optimize the model.  I used three methods to build the feature list.  I used visualizations to first eliminate features, and then identify features to include or to engineer.  I then engineered 5 features that seemed like they would improve the model, and finally, I used the feature_importance_ attribute in the selected algorithm to highlight the most relevant features.  Features were manually tested along the way.

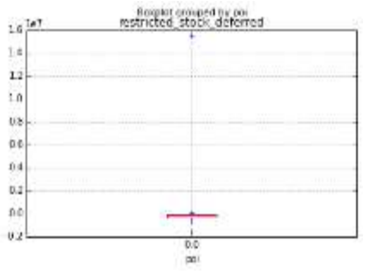
### Step 1:  Eliminate poor quality features

The three features identified in step 1 are eliminated.  They do not contain any POIs, and intuitively this is because the sample set is so small (and the number of POIs is also small), and not because there is any reason to believe they are a strong indicator of a person being a POI or not.

1. "director_fees".  no POIs were paid directors fees and the population is small (16).
2. "restricted_stock_deferred".  no POIs had restricted stock deferred and the population is small (17)
3. "loan_advances". No POIs and population is very small (3).



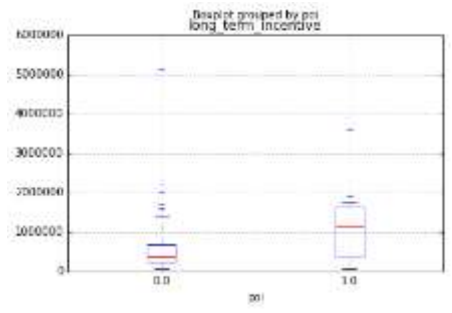


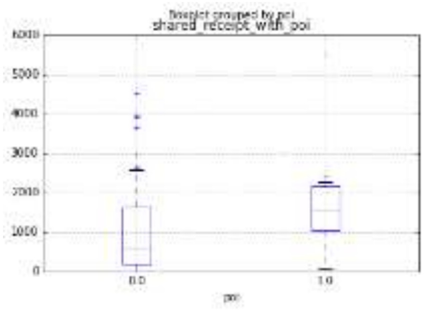


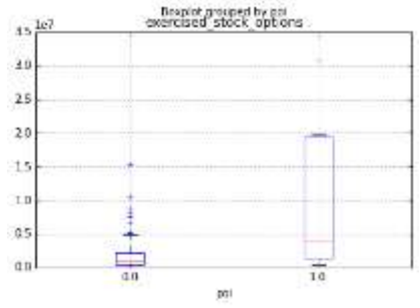
Bonus; deferral_payments; deferred_income; ~~director_fees~~; exercised_stock_options; expenses; from_messages; from_poi_to_this_person; from_this_person_to_poi; ~~loan_advances~~; long_term_incentive; restricted_stock; ~~restricted_stock_deferred~~; salary; shared_receipt_with_poi; to_messages; total_payments;total_stock_value

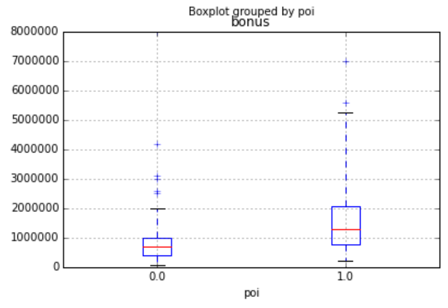
### Step 2:  Identify high impact features with good differentiation between POIs and nonPOIs

The next step was identify features with a noticeable difference between POIs and non POIs.







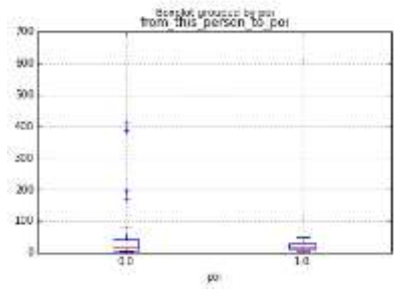


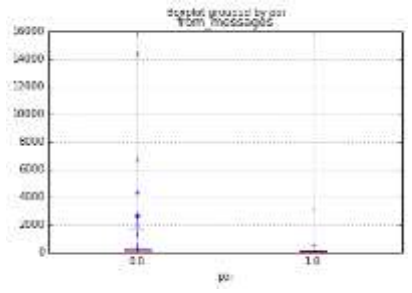
Bonus; deferral_payments; deferred_income; exercised_stock_options; expenses, from_messages; from_poi_to_this_person; from_this_person_to_poi; long_term_incentive; restricted_stock; salary; shared_receipt_with_poi; to_messages; total_payments;total_stock_value

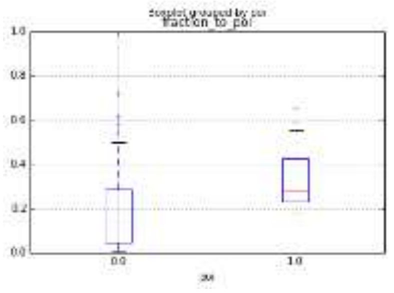
## Step 3: Engineer variables and visualize

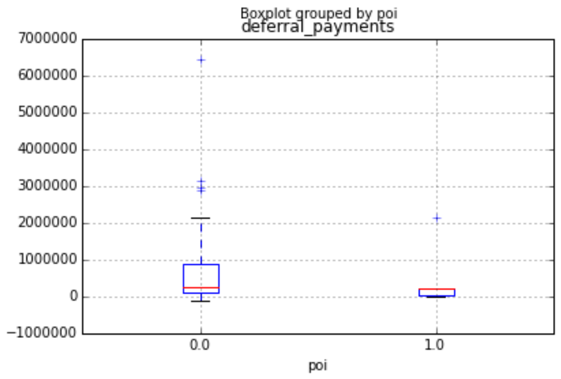
Finally, I utilized intuition to create features that might provide better results if expressed as an engineered variable. I looked at three different categories – mail (POI) / all mail, financial features / total payments and financial features / total salary. It was very easy to create and test these feature using the comuteFraction function and then visualize the results.

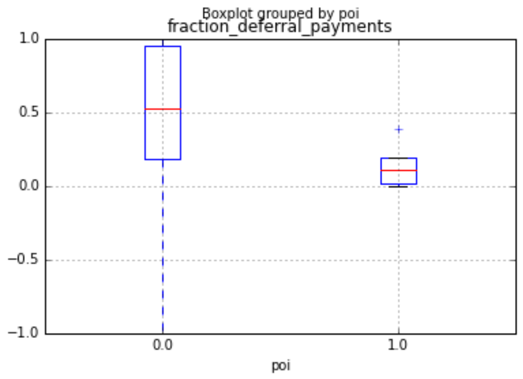
```
def computeFraction( numerator, denominator ):
"""
given a number messages to/from POI (numerator) and number of all messages to/from a person (denominator), return the fraction of messages to/from that person that are from/to a POI
"""
    fraction = 0.
    if numerator != 'NaN' and denominator != 'NaN':
        fraction = numerator / float(denominator)
    return fraction
```



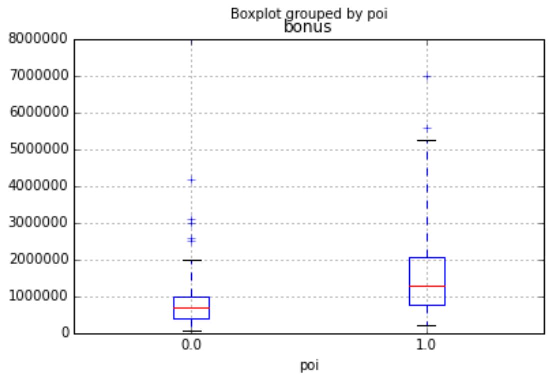


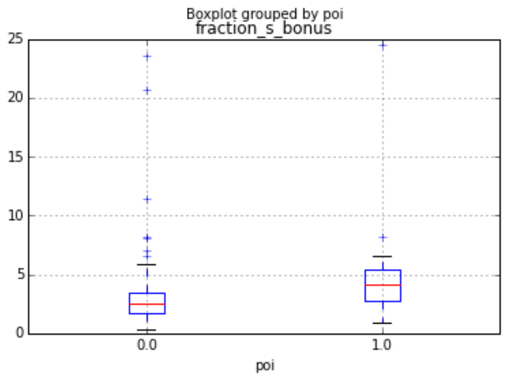






"fraction_to_poi" and "fraction_deferral_payments" both appear to be a much better features than the original features – less overlap of boxes , clearly different medians, and and less extreme outliers (whiskers set at 0,90 for all charts). Removed "from_this_person_to_poi" and "from_messages" from feature list.





However the ratio of the bonus v bonus to salary is more difficult to judge from the visualization and so both features were tested in the classifier. (This is also true of deferred_income)

The final step was to eliminate features that were better represented as an engineered feature (i.e. from_this_person_to_poi and to_messages are not in the final features list, but the ratio, 'fraction_to_poi', is.

Bonus; ~~deferral_payments~~; deferred_income; exercised_stock_options; expenses, ~~from_messages~~; ~~from_poi_to_this_person;~~ ~~from_this_person_to_poi;~~ long_term_incentive; restricted_stock; salary; shared_receipt_with_poi; ~~to_messages~~; total_payments;total_stock_value; fraction_from_poi; fraction_to_poi; fraction_deferral_payments; fraction_s_bonus; fraction_s_deferred_income

(key: green – expected high importance feature, blue – engineered feature, ~~abc~~ – replaced by engineered feature

## Step 4 – select feature list using feature_importances_

Note, feature list selection is done with "untuned" classifier

### *Test 1*

| Features Tested | feature_importances | Recall | Precision |
|---|---|---|---|
| shared_receipt_with_poi | 0.23578 | 0.3345 | 0.3345 |
| bonus | 0.220427 | | |
| fraction_s_deferred_income | 0.136189 | | |

| | | | |
|---|---|---|---|
| salary | 0.094312 | | |
| fraction_to_poi | 0.086019 | | |
| fraction_s_bonus | 0.075786 | | |
| total_stock_value | 0.065107 | | |
| exercised_stock_options | 0.058945 | | |
| restricted_stock | 0.027436 | | |
| deferred_income | 0 | | |
| long_term_incentive | 0 | | |
| total_payments | 0 | | |
| fraction_from_poi | 0 | | |
| fraction_deferral_payments | 0 | | |

## *Test 2*

Removed features with feature_importances_ = 0 with mixed results.

| Features Tested | feature_importances | Recall | Precision |
|---|---|---|---|
| fraction_s_bonus | 0.251155 | 0.3295 | 0.3373 |
| fraction_to_poi | 0.247283 | | |
| fraction_s_deferred_income | 0.180431 | | |
| bonus | 0.135139 | | |
| exercised_stock_options | 0.101669 | | |
| shared_receipt_with_poi | 0.056058 | | |
| salary | 0.028264 | | |
| restricted_stock | 0 | | |
| total_stock_value | 0 | | |

## *Test 3*

Again removed features with feature importances = 0. Feature importances changed significantly! And recall and precision both improved.

| Features Tested | feature_importances | Recall | Precision |
|---|---|---|---|
| exercised_stock_options | 0.354479 | 0.3555 | 0.3988 |
| shared_receipt_with_poi | 0.324358 | | |
| fraction_to_poi | 0.23811 | | |
| fraction_s_bonus | 0.083054 | | |
| bonus | 0 | | |
| salary | 0 | | |
| fraction_s_deferred_income | 0 | | |

## *Test 4*

Removed features with importances = 0, but recall and precision both decline!

| Features Tested | feature_importances | Recall | Precision |
|---|---|---|---|
| fraction_to_poi | 0.374664 | 0.3255 | 0.36884 |
| shared_receipt_with_poi | 0.319375 | | |
| fraction_s_bonus | 0.157827 | | |
| exercised_stock_options | 0.148133 | | |

### *Test 5*

Systematically return bonus, salary, and fraction_s_deferred_income. Move forward with Test 4 + salary as the recall and precision is the best.

| Test Description (reference feature list is test 4) | Recall | Precision |
|---|---|---|
| Test 4 + bonus: | 0.32850 | 0.36379 |
| Test 4 + salary | 0.4166 | 0.44161 |
| Test 4 + fraction_s_deferred_income | 0.4050 | 0.42542 |
| Test 4 + Return salary + fraction_s_deferred_income | 0.3985 | 0.42058 |

Final feature list prior to classifier tuning:

| Feature list | Feature importances | Source | Definition | Recall | Precision |
|---|---|---|---|---|---|
| fraction_to_poi | 0.480561 | engineered | = from_this_person_to_poi / to_messages | 0.4155 | 0.4418 |
| exercised_stock_options | 0.267928 | original | n/a | | |
| fraction_s_bonus | 0.251511 | engineered | = Bonus / salary | | |
| shared_receipt_with_poi | 0 | original | n/a | | |
| salary | 0 | original | n/a | | |

No features were scaled pending selection of the final algorithm (no scaling was required).

## Q3 – Algorithm Selection

I reviewed three classification algorithms, Naïve Bayes (NB), SVC, and Decision Tree (DT). SVC did not have sufficient data to yield results, so NB and DT were investigated further. With no tuning, DT provided better accuracy, precision and recall. NB does not have parameters to tune, so DT advanced to tuning.

| Algorithm | Parameters | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DT | default | 0.828 | 0.35816 | 0.36550 |
| NB | default | 0.8208 | 0.29977 | 0.25750 |

## Q4 – Algorithm Tuning

Tuning is performed to select the best parameters to optimize the algorithm. The DT algorithm was tuned to optimize performance (precision and recall) using two methods. I created a quick program (manual method in table) to try out the clf object with various iterations of criterion, max_depth, and min_samples_split and then selected the best set of parameters based on the results from the test_classifier in tester.

I then used GridSearchCV with estimator DecisionTreeClassifier, using the same parameters as the previous step and setting scoring = ‘f1’.  This produced a lower result than the manual method.

### *Tuning – Round 1*

Features_list = poi, fraction_to_poi, exercised_stock_options, fraction_s_bonus, shared_receipt_with_poi, salary

| Tuning method | Criterion | min_samples_split | max_depth | Precision | Recall | True Positives |
|---|---|---|---|---|---|---|
| No tuning | Gini | 2 | 2 | 0.44161 | 0.4166 | |
| GridSearchCV | Entropy | 2 | 15 | 0.3950 | 0.3225 | 645 |
| Manual | Entropy | 30 | 8 | 0.4553 | 0.4439 | 866 |

Feature importances after tuning

| Feature | Feature importance |
|---|---|
| fraction_to_poi | 0.564100684 |
| exercised_stock_options | 0.314503382 |
| fraction_s_bonus | 0.121395934 |
| salary | 0 |
| shared_receipt_with_poi | 0 |

### *Tuning – Round 2*

1. Remove salary and shared_receipt_with_poi
2. Retune classifier

| Test | Result |
|---|---|
| Round 1 feature list - ‘salary’ and retune | Recall and precision decline v. Round 1 |
| Round 1 feature list - ‘shared_receipt_with_poi’ | Recall and precision decline v. Round 1 |
| Round 1 feature list – salary and shared_receipt_with_poi | Recall and precision decline v. Round 1 |

This table exemplifies the importance of feature tuning, taking a poorly performing algorithm with an unacceptable 0.210 recall to a very favorable 0.70 recall just by changing the criterion from gini to entropy!

Final classifier:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=8,
        max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
        min_samples_split=30, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
```

## Q5 – Validation Strategy

Validation is required to ensure that the predictions generation through the machine learning algorithm perform adequately on an independent set of data. The balance to be struck is using enough training data to build a robust model, but to have sufficient data in the training set to result in statistically significant results. The data was broken into training and testing sets using the split provided in the project (70% training). Because the dataset is small, K-fold cross validation was utilized for this project. This effectively allows all of the data to be used for both training and for testing by dividing up the data into buckets, running the algorithm over all of the buckets, and then averaging the results. In this manner, the 145 original samples resulted in 1200 datapoints!

In addition, the number of positives (POIs) is small (12%), and so a stratified shuffle split is utilized. This ensures that the training sets have the correct ratio of non POIs and POIs as is present in the population for optimal results.

A classic way to validate the balance between bias and variance is to compare the accuracy on the training set and the test set. Decision Tree reports a score, which is the mean accuracy on the data and labels The score of the test data is 88% and the training data is 91%, which is satisfactory.

## Q6 – Evaluation Metrics

This project is a supervised classification problem, and the appropriate evaluation metrics are the accuracy score, precision and recall which are available as methods with the DecisionTreeClassifier. In addition, the test_classifer function in tester.py returns these metrics independent of the classifier used and metrics on true positives, false positives, true negatives and false negatives.

Accuracy is not a suitable metric for this project given the small # of POIs (12%). After selecting and tuning my algorithm, the maximum accuracy was 85%, which is lower than what could be achieved by setting all of the results to non-POI for an accuracy of 88%. This is clearly not a good method for this project to identify the POIs!

Recall is the metric that most closely aligns to the goal of finding POIs in this project. For this project, recall is the *# of POIs correctly identified / True # of POIs*. First, I want to optimize the number of true positives. Next, I want to minimize the number of missed POIs (false negatives) as POIs correctly identified + missed POIs is the denominator of recall.

The other relevant metric is precision. Precision or *POIs correctly identified / all POIs identified*, is a lower priority than recall, because minimizing the number of non-POIs incorrectly categorized as POIs is less important than not missing POIs. I also tracked correct POIs (true positives)

| Classifier | Precision | Recall | True Positives |
|---|---|---|---|
| DecisionTree | 0.455 | 0.433 | 866 |

Using the scorecard returned by Test_classifier, the algorithm overperforms on true positives by identifying more than expected. It also does better than expected by not incorrectly identifying non-POIs as POIs (false positives). False negatives, or missed POIs is higher than I would prefer. This is the metric preventing an even higher recall score. Finally, this chart indicates why the recall score is higher than the precision score. The much higher number of non-POIs will drive a higher # of false positives in the precision equation denominator, while the relatively smaller number of POIs will have a smaller # of false negatives in the recall equation denominator.

| Total predictions = 13000 | Expected | Algorithm result | Difference from expected. |
|---|---|---|---|
| True Positives | 12% * 83% = 1294 | 866 | -33% |
| False Positives | 88% * 17% = 1945 | 1036 | -47% |
| False Negatives | 12% * 17% = 265 | 1134 | 328% |
| True Negatives | 88% * 83% = 9495 | 9964 | 5% |

My conclusion from the recall and precision metrics, as well as the table above is that the algorithm does not do a particularly good job of properly identifying POIs. It does not properly id POIs as often as the accuracy would predict, and it also does not mis identify non-POIs as POIs which is good. It identifies many non-POIs incorrectly as POIs which is acceptable for this project.