Udacity Machine Learning Project Q&A

Justin "Roy" Garrard | Data Analyst Nanodegree | 08/05/17

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 objective of this project was to create and tune a classifier capable of identifying Persons Of Interest (POI's) from the provided Enron data set. Machine learning was particularly well-suited to the task given its ability to weigh categories of information against one another. For instance, two of the features used were a person's salary and the total number of messages they've sent and received. By considering the ranges those features tended to fall under for identified POI's, the algorithm was able to point out statistically-likely POI's.

The Enron data set took some work to make serviceable. To begin with, it only contained information on 146 persons, a worryingly small number for machine learning purposes that only became smaller as the data set was cleaned. Most features were applicable to less than 100 persons, while some were applicable to less than 50. Only 18 POI's were present.

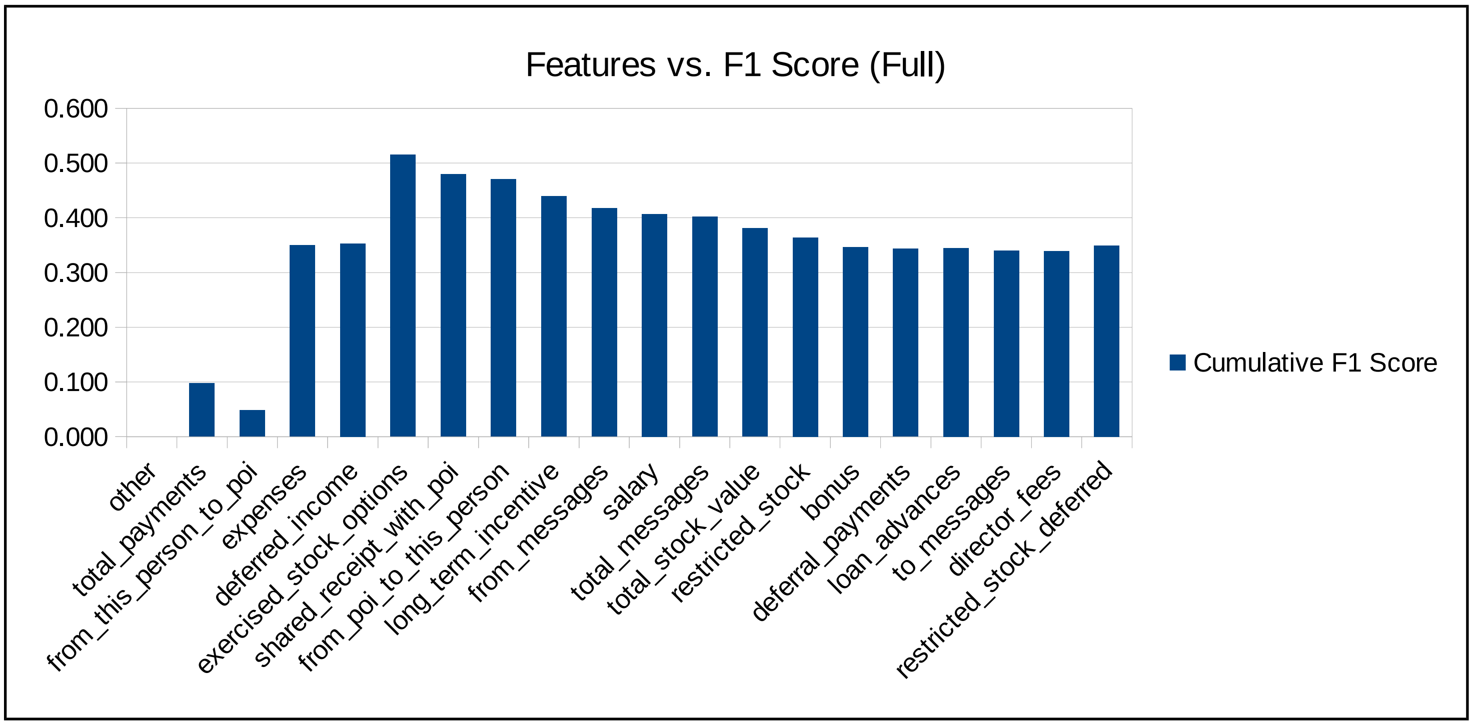
Outlier removal further reduced the data set to 141 points (18 POI's). Outliers removed include the 'TOTAL' row, two individuals with unusually small salaries, a person with negative stock value, and an employee with a positive 'restricted_stock_deferred' value.[1] The latter two are likely data entry errors as the pdf report differs from the programmatic amounts.

1 James Bannatine (former CEO), for having a salary of $477
Rodney Gray (former CEO), for having a salary of $6,615
Robert Belfer (former Director), for being the only employee with a negative stock total
Sanjay Bhatnagar (former CEO), for being the only employee with a positive restricted_stock_deferred value

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", "intelligently select features", "properly scale features"]

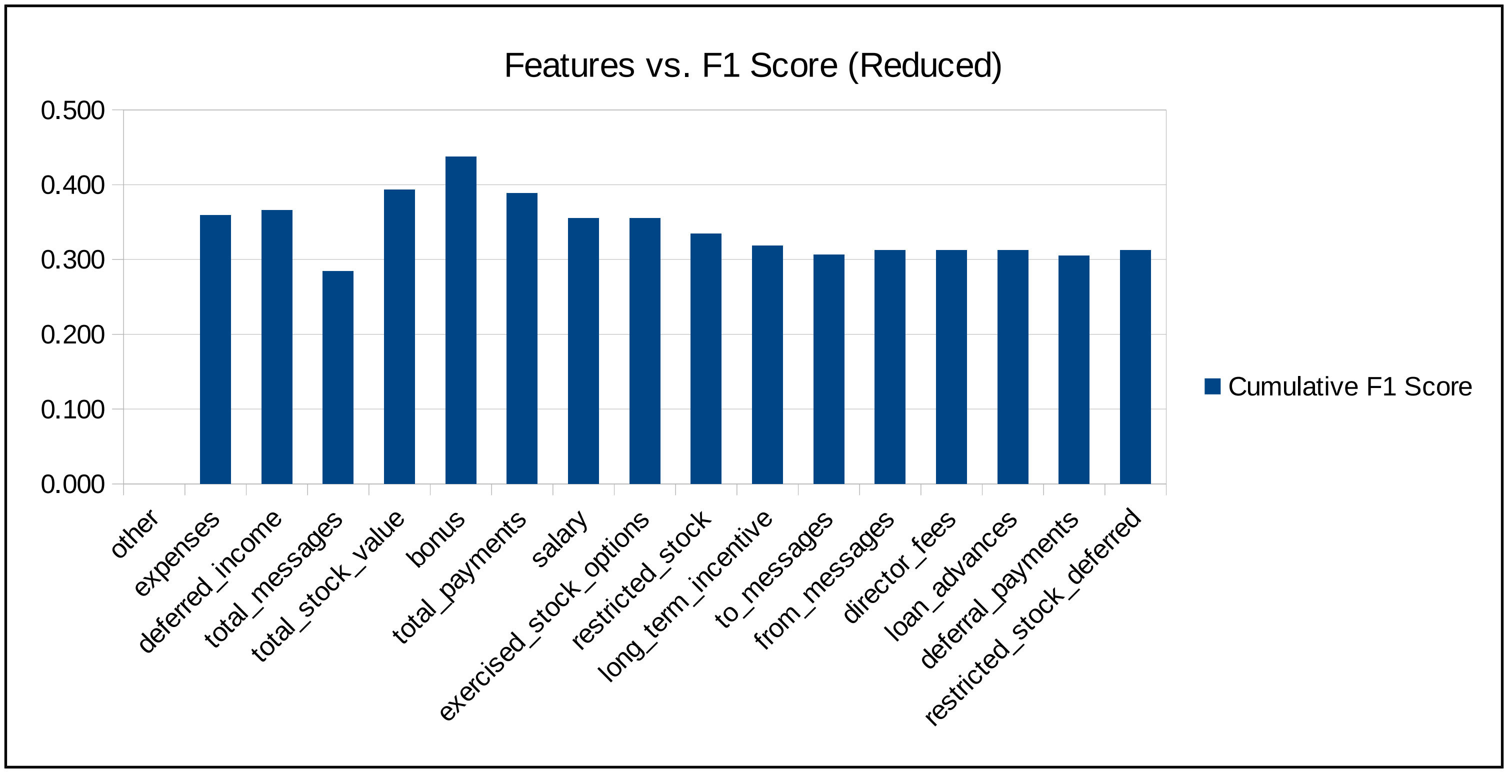
The final selection of features were as follows: salary, bonus, total_stock_value, expenses, and total_messages. No scaling was applied to the final result (the AdaBoost classifier did not appear to benefit and the RandomForest classifier was not expected to benefit).

The process by which the feature set was selected involved an automated component and a trial-and-error section. The automated component made use of the RFE (Recrusive Feature Elimination) classifier working in conjunction with the core AdaBoost classifier.



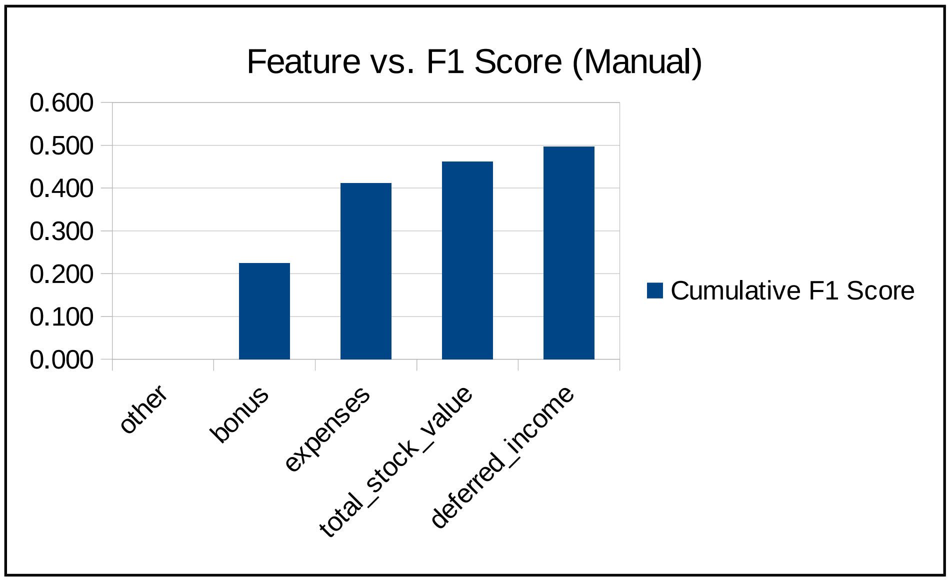
A bar chart showing the resulting F1 score as each feature is added to the AdaBoost classifier. Features are organized left-to-right by RFE importance. Due to data constraints, a classifier could not be built with the total_stock_value feature alone.

While the results were promising, the provided features include a few that presume knowledge of POI's ('from_this_person_to_poi', 'shared_receipt_with_poi', and 'from_poi_to_this_person'). It may be unwise to include these in a general-purpose classifier. With that in mind, a second feature chart was created:



As with the previous plot, the bar chart demonstrates the effect of features using an RFE. This chart excludes the features 'from_this_person_to_poi', 'shared_receipt_with_poi', and 'from_poi_to_this_person'. Predictably, F1 scores drop.

This information, combined with human intuition, resulted in the final classifier. The trial-and-error feature list had an F1 score that was moderately higher than the automated result (0.497 vs. 0.437).



A bar chart showing the resulting F1 score for the final set of features. As with the previous plots, an RFE determined the order of importance for each dimension. It's interesting to note that the ordering differs from the previous charts.

Whether this result is a quirk of the small dataset or a surprisingly effective configuration is uncertain. Regardless, it demonstrates that feature engineering remains as much an art as a science.

Feature List	**Coefficient Weights**	**Results**[2]
Automated ['poi', 'total_messages', 'bonus', 'total_stock_value', 'expenses', 'other', 'deferred_income']	[0.08 0.13 0.18 0.23 0.21 0.17]	Accuracy: 0.86736 Precision: 0.55496 Recall: 0.36100 F1: **0.43744** F2: 0.38813
Human ['poi', 'bonus', 'total_stock_value', 'expenses', 'other', 'deferred_income']	[0.13 0.15 0.27 0.34 0.11]	Accuracy: 0.88036 Precision: 0.62227 Recall: 0.41350 F1: **0.49685** F2: 0.44324

Three features relating to e-mails were developed with the hope of identifying persons of interest by who they confided in most frequently. The first was a count of the total number of messages sent and received. The second and third were ratios of email correspondence with POI's. The ratio features were later realized to be potentially dangerous, as they were based upon features with presumptive knowledge of POI's ('from_this_person_to_poi' and 'from_poi_to_this_person'). As such, they were abandoned.

The effectiveness of the remaining 'total_messages' feature is debatable. It was weighed as highly important in the Reduced Features bar chart, but only moderately influential in the Full Features chart. Additionally, it seemed to negatively impact the F1 score in both scenarios. It further reduced accuracy when added to the final classifier.

Feature List	**Coefficient Weights**	**Results**
['poi', 'bonus', 'total_stock_value', 'expenses', 'other', 'deferred_income']	[0.13 0.15 0.27 0.34 0.11]	Accuracy: 0.88036 Precision: 0.62227 Recall: 0.41350 F1: **0.49685** F2: 0.44324
['poi', 'bonus', 'total_stock_value', 'expenses', 'other', 'deferred_income', '**total_messages**']	[0.1 0.16 0.24 0.36 0.11 0.03]	Accuracy: 0.86736 Precision: 0.55496 Recall: 0.36100 F1: **0.43744** F2: 0.38813

A table showing the change in results for the final classifier with the addition of the 'total_messages' feature.

From this, one can gather that the effectiveness of a feature is dependent on the other features in its set. A future investigation might attempt to run RFE's on several feature sets, possibly shedding more light on the influence that features have upon one another.

2 AdaBoostClassifier(learning_rate=0.3, n_estimators=100, random_state=22)

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”]

The primary algorithm used was AdaBoost. RandomForest was also attempted, but found to be less effective in all respects (accuracy, precision, recall, F1, F2) for the selected features. RandomForest has more parameters to tune and may be more effective than AdaBoost with proper calibration.

---

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? What parameters did you tune? (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 items: “discuss parameter tuning”, “tune the algorithm”]

Most algorithms possess a few input parameters that can changed from the defaults. These changes may improve or reduce the overall effectiveness of the algorithm. In many cases, leaving the parameters on default is akin to leaving accuracy on the table.

In the context of the AdaBoost classifier, the two main “dials” that can be turned are n_estimator and learning_rate. Tweaking the values from default can adjust the balance between precision and recall to one more favorable for any given scenario. Various permutations were tested using an automated script. Results were assessed by the value of the classifier’s F1 score (an aggregate of precision and recall values).

learning_rate	[10, 30, 50, 70, 90]
n_estimator	[10, 30, 50, 100]

---

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 items: “discuss validation”, “validation strategy”]

Validation is the process of assessing algorithms. This is generally accomplished through a train/test division of data, where a majority of data is used to test an algorithm and a minority to assess it. This project makes use of the “train_test_split” tool to create a randomized 80/20 divide in the data. It also uses the “StratifiedShuffleSplit” tool when the tester script is invoked, which works to balance the number of POI’s and non-POI’s for testing.

Good classifier algorithms are accurate, precise, and have good recall values. The classic mistake when validating is to only assess accuracy (number of correct / total number of items), which does not take into consideration false positives. The provided tester script was used for validation, which does factor in false positives. F1 scores (an aggregate of precision and recall) were the main determinate for whether or not an algorithm was successful.

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 algorithm's performance can be summarized with the output of the tester script for its optimal parameters:

Accuracy: 0.88036 Precision: 0.62227 Recall: 0.41350 F1: 0.49685 F2: 0.44324

The overall accuracy is 88%, representing the number of correctly identified items out of the entire pool. The rate of correct positives to all positives (precision) was 62%. This implies that a little more than half the persons identified as POI are genuinely POIs. The recall rate was lower at 50%, representing the number of POIs that were identified out of all POIs.

Put another way, the classifier missed 38% of the true POIs and wrongly identified 50% of those it found. Higher levels of accuracy, precision, and recall might still be coaxed out of the classifier through additional feature engineering and parameter tuning, but what would be most beneficial is additional data.