Udacity Data Analyst Nanodegree
Project 5: Intro. to Machine Learning: Enron POI Identifier

By Stephen Fox
February 2017

I. Overview

From 1998 – 2000, I worked as a chemical engineer in Houston in a building across the street from the Enron building. I occasionally wondered what exactly people at Enron did, as their business model was unclear to me. By October 2001, the answer to my question cleared up a bit: some senior people at Enron had been up to no good. By December 2001, Enron had ceased operations and filed for bankruptcy[1]. In the ensuing investigations that followed, several senior management figures were found guilty of financial crimes and sentenced to prison, although there remained a sentiment among the public that many guilty parties had escaped justice.

The goal of this project is to use machine learning to identify 'persons-of-interest' (POI), or those people potentially involved in the criminal actions committed at Enron, from a list of Enron senior management. The data set provided for this project consists of a variety of financial features (salary, bonus, etc.) and email features (number of to messages, shared receipt with POIs etc.) along with a POI label for each of 146 data points (people). Machine learning is useful for identifying complex interactions and patterns amongst features and allowing one to classify unlabeled data points based on those patterns. For example, there might be a combination of salary, bonus and emailing habits that are shared amongst all POIs. This type of model could be a very helpful tool for investigators in identifying possible other POIs for further investigation.

II. Data Exploration, Outlier Identification & Cleaning

The coding portion of this project was completed in the accompanying jupyter ipython notebook (poi_id.ipynb). The first step of the analysis was to conduct exploratory data analysis (EDA), which included identifying and dealing with outliers. The following key metrics were deduced from the data set:

- Gross data set size: 146 'people'
- Number of features per entry: 20
- Number of POIs: 18 (12.3% of the data set)

The full list of features is provided in the Appendix. Many of the data points had missing ('NaN) values for many of the features. For example, the 'bonus' feature only contained data for 82 of the 146 data points (56%). For the EDA and outlier detection portion of the project, 'NaN' values were converted to zero, to allow for various sanity checks. For the machine learning portion of this project, 'NaN' values were ignored, since there is the

[1] https://en.wikipedia.org/wiki/Enron

potential for the machine learning algorithms to pick up on ‘NaN’ labels as a possible identifier of POIs, which would be undesirable.

The data set is small (only 146 data points), incomplete and imbalanced (the non-POI class is 88% of the data). This will pose a challenge for machine learning, which relies on having plenty of data for building and validating models. To compensate for this small data set, validation will need to utilize advanced shuffling techniques (e.g. StratifiedShuffleSplit) to get the most out of the data. The model will also need to be evaluated by scoring metrics beyond accuracy (e.g. precision and recall), since with a highly imbalanced data set, a model that always guesses a data point is non-POI will do quite well on accuracy (88%).

The next step in the analysis was to identify and remove outliers. Two non-human outliers were identified in the data, namely the ‘TOTAL’ line (which summed up all the financial data for the 144 actual people in the data set) and the ‘TRAVEL AGENCY IN THE PARK’ line, which was a non-human entry with minor financial impact. Since neither of these data points is a human, both were removed, leaving 144 legitimate people for further analysis.

After the outliers were removed, a sanity check was conducted on the financial data, to ensure the ‘total_payments’ and ‘total_stock_value’ features were equal to the sum of the other relevant financial features for each data point. This analysis revealed that for two individuals (Sanjay Bhatnagar and Robert Belfer), the data had been erroneously entered. These points were manually corrected, based on data in the Enron Financials PDF provided as part of the project materials. After the outliers were removed and the erroneous data points were cleaned, the cleaned data set was ready for the next step in the project, namely feature engineering.

III. Features Engineering

Synthesizing new features from existing features can be a powerful way to incorporate human intuition into a model and boost the model’s predictive ability. I created four new features that I believe might be of value in predicting POIs. The first two synthetic features deal with financial parameters:

- TP_TSV_ratio = total_payments / total_stock_value
  These two existing features encompass all the financial data. Taking the ratio provides a measurement of how much of an individual's compensation was paid in non-stock vs. stock payments, which could impact their behavior as it relates to short-term vs. long-term decision making.

- deferred_total_ratio = (deferred_income + restricted_stock_deferred) / (total_payments + total_stock_value)
  This feature will measure how much of an individual's compensation was electively deferred, relative to total compensation. If an individual chose to defer a large portion of their compensation, they might be more interested in the long-term performance of the company.

The second two synthetic features deal with email habits:

- to_poi_ratio = from_this_person_to_poi / from_messages
  This feature measures the proportion of a person's sent email messages that went to POIs. It is possible that POIs communicated more frequently with each other.

- From_poi_ratio = from_poi_to_this_person / to_messages
  This feature measures the proportion of a person's received email messages that came from POIs. It is possible that POIs communicated more frequently with each other.

The effectiveness of these synthetic features will be explored during the algorithm selection process.

IV. Algorithm Selection

Seven different classification algorithms were tested. In all cases, the default settings were employed for the first pass analysis discussed in this section. Each model was tested against the data set under two conditions, namely excluding the four new features and including the four new features. In each case, the precision, recall, F1-score and processing time were recorded. The results are presented in Table 1.

Table 1: First Pass Algorithm Selection Results

Algorithm	New Features?	Precision	Recall	F1 Score	Run time (ms)
GaussianNB (GNB)	No	0.50	0.40	0.44	3
DecisionTreeClassifier (DTC)	No	0.00	0.00	0.00	3
SupportVectorClassifier (SVC)	No	0.00	0.00	0.00	3
KNeighborsClassifier (KNN)	No	0.00	0.00	0.00	4
RandomForestClassifier (RFC)	No	0.50	0.20	0.29	44
GradientBoostingClassifier (GBC)	No	0.25	0.20	0.22	67
AdaBoostClassifier (ABC)	No	0.17	0.20	0.18	130
GaussianNB (GNB)	Yes	0.14	0.80	0.24	3
DecisionTreeClassifier (DTC)	Yes	0.33	0.40	0.36	3
SupportVectorClassifier (SVC)	Yes	0.00	0.00	0.00	3
KNeighborsClassifier (KNN)	Yes	0.00	0.00	0.00	4
RandomForestClassifier (RFC)	Yes	1.00	0.20	0.33	44
GradientBoostingClassifier (GBC)	Yes	0.25	0.20	0.22	74
AdaBoostClassifier (ABC)	Yes	0.20	0.20	0.20	135

The following key observations were made when comparing the effect of excluding or including the new features:

- Including the new features **improved** the first pass results for the following algorithms:
    - DTC
    - RFC
    - ABC
- Including the new features had **no impact** on the first pass results for the following algorithms:
    - SVC
    - KNN
    - GBC
- Including the new features **worsened** the first pass results for the following algorithms:
    - GNB

So for the most part, the new features either improved or had no impact on the first pass results for the algorithms of interest. In addition, the one algorithm that was worsened by the presence of the synthetic features (GNB) has very few tuning parameters and therefore is not nearly as interesting to me for the validation, tuning and optimization portion of the project. Based on these results, I will proceed with a features_list that includes the four new features.

The following key observations were made from the first pass algorithm selection results, under the scenario where the new features were included:

- DTC was the only algorithm to achieve the desired 0.3 score on precision and recall using this default model. It also had a very short run time compared to other promising candidates and hence will definitely proceed to the next stage.

- SVC and KNN failed to correctly identify any POIs, with precision and recall values of 0.00. This does not mean that they might not work well if optimized later, but since I plan on moving forward with a single algorithm, they will not be selected, given other algorithms showed promise.

- The run times on RFC, GBC and ABC were orders of magnitude longer than the more simple algorithms. This is not a problem at this stage, but could become an issue as model complexity and grid search expands during the next stage of the project.

After running the DTC classifier along with the selected features and data set (clf, features_list, my_data set), the following results were achieved via tester.py:

- Precision = 0.29874
- Recall = 0.29650

Therefore, a standard DTC model is close to achieving the project requirements. Based on these results, I will proceed with the optimization of the DTC model, since the advantage in run time over RFC, GBC and ABC is a compelling advantage for efficiently completing the optimization process. If I am unable to achieve satisfactory results with DTC, then I will consider one of the other promising models for further development.

V. Algorithm Validation & Optimization

An important part of developing a robust and effect algorithm is the validation step. In machine learning, the model is optimized on training data and then validated on validation / test data. The purpose of validation is to test the model on data that is independent of the data used to build the model. This step is crucial, since it validates whether the model will perform well when facing new data, which is ultimately what is needed to support a claim that one has developed a 'learner'. A classic mistake if validation is not done properly is to have an overfitted system. This is a system that performs very well on training data but fails to perform well when faced with new data.

Given how small the Enron data set is, this project made use of 'StratifiedShuffleSplit' for validation, which allows one to perform thousands of validation steps by shuffling and splitting the small data set in many random ways. This validation strategy was automatically incorporated into the grid search hyper parameter tuning process discussed next.

As a final step, the DTC model hyper parameters were tuned. Tuning is a process of manually or systematically adjusting the various parameters for a given model and determining whether the adjustment increases or decreases the score of interest. In this case, the score of interest is the F1 score, since it is a function of both precision and recall. The expectation was that by maximizing F1, the tuning process would also result in improved precision and recall scores. If tuning is not done well, one can be left with a suboptimal model that doesn't perform nearly as well as could be achieved with proper tuning, as will be seen in the results presented later in this section.

I used piping and grid search to conduct a systematic tuning process for the model. The pipe consisted of a SelectKBest module for optimizing the number of features fed into the DTC, followed by the classifier itself. For a DTC, feature scaling will have no effect, since the splits will be the same regardless, so was not used. To keep the run times manageable, I kept certain parameters constant on certain runs and let the grid search process test combinations of other parameters. In this way, I probed most if not all of the possible parameter combinations in order to arrive at the optimal combinations. Since there are slight differences between the F1 score generated by tester.py and the pipe, the results of each grid search run were also compared to the tester.py results, to check whether the tester.py scores also continued to improve with each model change and grid search optimization step. The results of this iterative grid search tuning process are presented in Table 2.

Table 2: Hyper Parameter Tuning Process Results

Model / Change (***)	Run time (s)	Grid Search F1 Score	Tester.py Precision	Tester.py Recall	Tester.py F1 Score
DTC only (*) (**)	0.003	0.36	0.299	0.297	0.298
k = 23 (fixed) (**)	0.3	0.297	0.299	0.297	0.298
k = 16	8	0.314	0.313	0.325	0.319
k = 22; 'splitter' = "random"	15	0.405	0.338	0.367	0.352
k = 17 'max_leaf_nodes' = 14	59	0.448	0.470	0.361	0.408
'max_leaf_nodes' = 15 'class_weight' = "balanced"	135	0.461	0.357	0.499	0.416
k = 23 'max_leaf_nodes' = 8 'max_depth' = 5	149	0.475	0.336	0.794	0.472
'max_depth' = 4 'min_samples_split' = 10	31	0.492	0.346	0.811	0.485

(*) pipe NOT used
(**) grid search NOT used
(***) only those changes relative to the previous model are shown

In addition to the tuned parameters shown in Table 2, other parameters ('criterion' and 'min_samples_leaf') were also tested (results not shown). However, for those parameters, the DTC defaults were found to be optimal. As seen in the table, the tuning steps resulted in increased F1 scores. These generally translated into improved precision and recall scores too, and the 0.30 thresholds for each measurement were consistently achieved. Note that the tuning process greatly increased the run times, so tuning more complex models (RFC, GBC, ABC) might have been quite time consuming.

The final model has a precision of 0.35 and a recall of 0.81, both in excess of the 0.30 minimum requirements for the project. Precision (or positive predictive value) is the fraction of retrieved instances that are relevant[2]. If precision is equal to 1, all the true positives (e.g. POIs) have been identified without any false positives (non_POIs who were identified as POIs). Recall (or sensitivity) is the fraction of relevant instances that are retrieved. If recall is equal to 1, all the true positives (e.g. POIs) have been identified without any false negatives (POIs who were identified as non-POIs).

I am pleased that the recall is so much higher than the precision in this case, because I believe this tool would be most useful for conducting an initial screening for potential POIs, with the intent that humans then evaluate each positive result in more detail, to determine whether a criminal investigation is warranted. Thus, it is more important to identify as many true positives as possible with very few false negatives (i.e. high recall is desired) instead of worrying too much about false positives (i.e. low precision is

[2] https://en.wikipedia.org/wiki/Precision_and_recall

acceptable is recall is high), given that one is merely conducting an initial screening step. False positives could then be weeded out in the next step of the process.

The final, optimized model used all 23 features from the loaded features list, which consisted of the 20 original features minus email_addressed plus the four synthetic features discussed earlier. The feature scores from the SelectKBest process are presented in Table 3, in descending order.

Table 3: SelectKBest Feature Scores for Optimized Model

Feature	SelectKBest Score
total_stock_value	22.51
exercised_stock_options	22.35
bonus	20.79
salary	18.29
to_poi_ratio	16.41
deferred_income	11.42
long_term_incentive	9.92
total_payments	9.28
restricted_stock	8.83
shared_receipt_with_poi	8.59
loan_advances	7.18
expenses	5.42
from_poi_to_this_person	5.24
other	4.20
from_poi_ratio	3.13
from_this_person_to_poi	2.38
director_fees	2.13
to_messages	1.65
restricted_stock_deferred	0.77
TP_TSV_ratio	0.61
deferred_total_ratio	0.49
deferral_payments	0.23
from_messages	0.17

It is no surprise to see that the top 5 features are dominated by financial metrics with large values, such as total stock value, exercised stock options, bonus and salary. It is satisfying to see that one of the synthetic features (‘to_poi_ratio’) made the top five list.

Interestingly, of the 23 features fed into the classifer, only six features had feature importance scores above zero (at least when 6 significant figures are considered). Those six features, and their respective importances, are given in Table 4 in descending order.

Table 4: DTC Feature Importances for Optimized Model

Feature	Importance
to_poi_ratio	0.518960
restricted_stock	0.174142
total_stock_value	0.111818
TP_TSV_ratio	0.101889
to_messages	0.061206
restricted_stock_deferred	0.031985

Of the top five features from the SelectKBest score, only two of them (‘to_poi_ratio’ and ‘total_stock_value’) remain in the top 5 list for feature importances. It is very satisfying to see that one of the synthetic features (‘to_poi_ratio’) has the highest importance and another synthetic feature (‘TP_TSV_ratio’) makes the top 5 feature importance list too. Given that only six features had importances above zero, it should in theory be possible to achieve the same results by only feeding these six features into the DTC. However, doing so would require a whole new optimization and validation process, and was not done here.

VI. Conclusion

An Enron POI classifier has been built and optimized here. It is recommended to use all 20 of the original features with the exception of the ‘email_address’ and also to feed in four synthetic features, namely the ‘TP_TSV_ratio’, ‘deferred_total_ratio’, ‘to_poi_ratio’ and ‘from_poi_ratio’. These 23 features are then fed into a decision tree classifer, where the following parameters have been changed from the default settings:

```
clf = DecisionTreeClassifier(splitter="random",
                    max_depth=4,
                    min_samples_split=10,
                    max_leaf_nodes=8,
                    class_weight="balanced",
                    random_state=42)
```

This model achieves a precision of 0.35 and a recall of 0.81, which exceeds the project requirements and also has the desirable feature of not missing too many true positive POIs (as shown by the high recall), even if the false positive rate is quite high (as shown by the low precision relative to recall).

References

1. ‘Enron61702insiderpay.pdf’ - this file was provided by Udacity and used for the purposes of data checking and cleaning financial data
2. Many coding questions were addressed on http://stackoverflow.com/
3. The Udacity forums were an invaluable resource for project specific questions
4. http://scikit-learn.org was a key resource for using the various sklearn libraries and features
5. The feature_format.py and tester.py files (instrumental for formatting the data and testing the algorithm, respectively) were provided by Udacity
6. Much learning and information was obtained from the Udacity slides for Data Analyst Nanodegree: Project 5
7. Wikipedia (https://en.wikipedia.org/wiki) provided some additional information on the Enron story and also on precision and recall

Appendix: Full Feature List – Enron Data set

financial features:
['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees']

(all units are in US dollars)

email features:
['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi']

(units are generally number of emails messages; notable exception is ‘email_address’, which is a text string)

POI label:
[‘poi’]

(boolean, represented as integer)