Answers to Machine Learning Questions

by Richard Haughton

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"]*

… Summarize for us the goal of this project..:

The goal of the project was to try and identify the Enron fraudsters (or at least Persons of Interest - POI) using machine learning techniques. From a learning perspective the goal was to develop skills in selection and tuning of machine learning algorithms and the feature selection/preparation that goes with it.

... how machine learning is useful in trying to accomplish it:

Machine learning is a useful tool in this process because it allows us to extract/combine/manipulate features from within a large(ish) data corpus to categorise each employee as 'potential fraudster/not'. A process that would require a huge amount of manual effort and subject matter expertise otherwise.

... give some background on the dataset and how it can be used to answer the project question:

The dataset contains emails (body and metadata) and financial details for many of Enron high ranking employees (labelled with a POI indicator), allowing us to form supervised learning models to predict those POI's.

```
Total Number of data points: 146
Number of POI: 18 (whereas there were 35 in total)
Number of Non POI: 128
Number of Features: 21
Features with missing values: (20 features):

- salary: 51
- to_messages: 60
- deferral_payments: 107
- total_payments: 21
- long_term_incentive: 80
- loan_advances: 142
- bonus: 64
- restricted_stock: 36
- restricted_stock_deferred: 128
- total_stock_value: 20
- shared_receipt_with_poi: 60
- from_poi_to_this_person: 60
- exercised_stock_options: 44
- from_messages: 60
- other: 53
- from_this_person_to_poi: 60
- deferred_income: 97
- expenses: 51
- email_address: 35
- director_fees: 129
```

Predicting POI using ML techniques is challenging with this dataset because their are relatively few data points with an in balance of positive/negative cases (few POI's). Features are also sparsely populated which can lead to misleading conclusions.

... Were there any outliers in the data when you got it, and how did you handle those?

There was one blatent outlier which we identified in class which turned out not to be a 'real person' but a 'total' accross people (most financial features were extreme compared to others). I removed this data point prior to analysis.

Given the obvious 'non person' nature of the 'TOTAL' data point I decided to check the names of all other data points, I also sorted alphabetically to check whether there were any duplicates (there weren't). All seemed valid people except 'THE TRAVEL AGENCY IN THE PARK'

I continued outlier analysis by visualising each feature individually against salary on a scatter plot, conditionally coloured by 'poi'. Having done this I took a closer look at various stand out data points for specific features (see/run enron_outliers.py).

There were a number (see examples below) but non were on balance 'odd enough' across features to remove (though arguably I could have 'massaged' the outstanding feature value I suppose).

examples:

```
'BHATNAGAR SANJAY' - huge 'restricted_stock_deferred' (suspicious that
'restricted_stock_deferred' was equal to 'total_payments' - typo?)
'KAMINKI WINCENTY J - large 'from_messages'
...
```

I also noticed that KEN LAY (a poi) was a huge outlier for several financial features, I temporarily removed him to see whether other outliers would emerge, they did but again I didn't consider them conspicuous enough to remove.

One technique is to systematically remove a percentage of outliers for each feature but given the small number of data points and that many of the outliers are in fact poi's I decided this would be counter productive.

So in the end I removed just two outliers:

```
TOTAL - Removed
THE TRAVEL AGENCY IN THE PARK - Removed
```

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"]

...What features did you end up using in your POI identifier, and what selection process did you use to pick them?

Feature selection and algorithm selection went hand in hand and was an iterative process roughly as follows:

```
1. Coded versions of several algorithms
- Naive Bayes
- SVM - struggled to get this to work with the stratified tester
- Decision Tree
- Ada Boost
- Random Forest
- K Nearest Neighbours

2. Baseline performance with a single feature
- salary (as given in base code)
- bonus (tried bonus too as it looked like a better indicator)

3. Re-run with all the (numeric) given features

4. Inspect feature importances for each of the decision tree based algorithms

5. Manually choose set of features based upon the above

6. Re-run each algorithm with the subset of features

7. Repeat 5 and 6
```

(NOTE: I felt this manual process was practicable given the relatively small number of features)

Chosen features and feature importances:

```
bonus   0.142857142857
exercised_stock_options 0.0816326530612
expenses        0.163265306122
from_this_person_to_poi 0.142857142857
other   0.285714285714
total_stock_value       0.122448979592
mentions (user defined)        0.0612244897959
```

...Did you have to do any scaling? Why or why not?

I performed min max rescaling in particular for the k nearest neighbour algorithm because it is sensitive to magnitude of distances and clearly with some features measured in hundreds and others in millions this would skew the results. Sadly after feature scaling performance of the algorithm dropped off a cliff!

No feature scaling is necessary for decision tree based algorithms as each feature is essentially considered independently.

... explain what feature you tried to make, and the rationale behind it

The three features 'from_poi_to_this_person', 'to_poi_from_this_person', 'shared_receipt_with_poi'

could be deemed to collectively amount to an implicit feature describing the strength of working relationship between a person and POI's. So I attempted to combine the three (rescaling along the way). Sadly the new feature had a negative effect upon performance contributing very little (feature importance) to the output.

I then created a feature called 'mentions' which is effectively a count of the number of times a poi is mentioned in each persons 'sent email bodies'. By mentions I mean both first and surname of a poi in the same email body or the email address of a poi in the email body. I removed one flaw that gave the algorithm too much strength which was self references.

The rationale is much like the rationale for building 'from/to poi' features, which is that 'mentions' are a potential indicator of a strength of relationship and therefore improve the likelihood that they are embroiled in the same fraudulent behaviour. Using the email body also brings in the potential that poi's that we don't have email/financial data for may be included.

With this new feature a modest improvement was seen for both precision and recall.

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"]

As discussed I tried the following:

```
- Naive Bayes
- SVM - struggled to get this to work with the stratified tester
- Decision Tree
- Ada Boost
- Random Forest
- K Nearest Neighbours
```

I managed to get decent 'precision' scores (> 0.3) from most algorithms quite quickly but struggled to get over 0.3 for 'recall'. My Support Vector Machine implementation was prohibitively slow to work with the provided 'tester' (I did get it working locally with a 30% subset).

Ada Boost and K Nearest Neighbours stood out as the best performing across 'precision' and 'recall'. After feature rescaling K Nearest Neighbours performance dropped dramatically so I abandoned that approach as 'unsafe without scaling'.

Having identified in Ada Boost an algorithm that exceeded the 0.3 precision and recall target pre-tuning I selected this algorithm and then tried to improve its scores.

For completeness 'results.ods/results.xls' contains some figures detailing performance of each algorithm with various configurations (e.g. sets of features, scaling vs not scaled ..)

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"]

Some algorithms have parameters that you can vary, tuning is the process of varying these parameters to achieve the best possible predictions. If you don't tune the algorithm there is every chance that you won't achieve the best performance, for example some algorithms have a learning_rate parameter that (kind of) dictates how much you look to improve accuracy every step of the algorithm - set this too low and your algorithm may never finish, set too high and you may never converge upon the minima (lowest error).

As I chose Ada Boost (with a Decision Tree base), the parameters I had to vary where the 'number of estimators (n)' and the 'learning rate (lr)'. My approach was a little unscientific but non the less yielded improvements over the defaults:

```
1. start with default settings (n = 50, lr = 1)
2. baseline performance (accuracy, precision, recall)
3. vary 1 parameter (lr) up and down, retesting to find an optimal (albeit not
necessarily global) value
4. keep this value and vary the other parameter up and down, retesting to find a
combined optimal value
5. repeat the process but varying the other parameter (n) first (do we end up in
the same place? - yes)
```

Note that in this instance 'optimal' value was a bit subjective, it was kind of the best F1 score but as Recall was consistently lower than Precision it was really the settings that yielded highest Recall.

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 the process of testing your algorithm/model against an independent subset of data (i.e. not used to train the algorithm) with known output variable (in this case POI). A classic mistake is one of 'over-fitting', where the algorithm performs well against the training set but does not generalise well to new data. This can happen when you have a small number of data points with a large number of features.

I used the provided 'tester' code which uses the 'stratified shuffle split' technique. In essence it generates multiple different sets of train and test data from the given data, performs the fitting/predicting on each and reports an average for several metrics (e.g. accuracy, precision, recall). This is a form of 'cross-validaion' that is useful particularly when you have relatively few data points as a way of reducing the chance of over-fitting.

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"]

```
Accuracy: 0.87586
Precision: 0.60364
Recall: 0.38150
```

For completeness:

```
F1: 0.46752
F2: 0.41181
```

```
Total predictions: 14000
True positives:  763
False positives:  501
False negatives: 1237
True negatives: 11499
```

Accuracy:

A measure (proportion) of how often the algorithm makes the correct prediction. When you have skewed data like ours, that is many more -ve than +ve outcomes (or vice versa) accuracy is not a good performance metric because you could have a high accuracy simply by selecting -ve for all. This would not help us identify POI's

Precision:

Given the algorithm predicts a given outcome, what are the chances that the prediction is correct. High precision means a low false +ve (values range from 0 - 1)

In our Enron case a high precision would mean that when we predict 'POI' it usually is a 'POI'

Recall:

For a given outcome, what are the chances that our prediction is correct. High Recall means a low false -ve (values range from 0 - 1)

In our Enron case a high recall means that we seldom predict 'Non POI' when it is a 'POI'

I used Precision and Recall as my evaluation metrics. In this instance, where we are just trying to flag up potential POI for further investigation it seems more important 'not to miss any POI' than be sure that they are 'POI' so Recall feels the more important (though proved difficult to achieve).