Identify Fraud from Enron Email

Katherine Tansey

13 December 2015

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?

The aim of the project is to identify potential person of interest within Enron. Enron was an energy/commodities company that declared bankruptcy in 2001. It turned out that there was structured accounting fraud on going within the company with various employees partaking in the fraud, which attempted to portrait the financial state of the company to be better than it was. Machine learning is a useful tool to identify patterns in data, develop an algorithm, and use that algorithm to predict the outcome in a completely new dataset. Using the data from emails as well as financial information from employees and machine learning, the project tries to identify those involved in the known fraud from those that were innocent bystanders.

There were two outliers in the dataset. From the input table, there was a row that totaled all the values from the other rows. This was not information for an employee, and was an outlier on all values (being the total for each value). This was removed from the dataset. The second was called "THE TRAVEL AGENCY IN THE PARK" which is information not for an employee but for a travel agency, and therefore unlikely to ever be a POI for this reason this was also removed. After removal of the outliers, there are 144 datapoints in the dataset, 18 of which are persons of interest (POIs) with 126 non-POIs.

Number of missing data points per feature:

Feature	Number of missing values
salary	50
to_messages	58
deferral_payments	106
total_payments	21
loan_advances	141
bonus	63
long_term_incentive	79
restricted_stock_deferred	127
total_stock_value	19
expenses	50
from_poi_to_this_person	58
exercised_stock_options	43
from_messages	58
other	53
from_this_person_to_poi	58
deferred_income	96
shared_receipt_with_poi	58
restricted_stock	35
director_fees	128

Some of these contain missing data for most datapoints.

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.

I select features using the SelectKBest function using the f_regression method. This selects the features with the highest F-value between label/feature. Features selected were: deferral payments, total payments, loan advances, restricted stock deferred, deferred income, total stock value, expenses, exercised stock options, other, long term incentive, to messages, from messages, from this person to poi, shared receipt with poi, to this person from poi ratio.

The feature "to this person from poi ratio" is a feature that I created, and this quantifys the proportion of emails for each person that were to a POI. The idea being that there might be more interaction (through emails) between POIs than those who are not POIs. There selection as important features reflects that this may be the case. When this feature was not included in the analysis, the precision and recall were lower than when it was included (precision=0.19262 and recall=0.31300), which highlights the importance of this feature in identifying POIs.

Scaling was done when testing the performance of the LinearSVC algorithm, as this classifier is sensitive to the scale of features and does not work best when features are on differing scales. I used the MinMaxScaler and the default of standardizing features to fall within a range of 0 to 1. This set all the features to the same scale, which is suitable for optimal performance from LinearSVC. However, as I found the best results using a Decision Tree classifier, features were not scaled as this is not necessary for tree based algorithm as they are not affected by different scales.

Feature importances for selected features:

Feature	Importance
total_stock_value	13.3108
loan_advances	12.827
exercised_stock_options	12.6273
to_messages	10.6238
poi	10.2925
deferred_income	6.607
deferral_payments	5.7785
total_payments	5.3444
long_term_incentive	4.7566
other	2.8089
from_this_person_to_poi	2.4314
expenses	1.8215
to_this_person_from_poi_ratio	1.0853
shared_receipt_with_poi	0.4664
salary	0.2936
from_messages	0.293
from_poi_to_this_person	0.2416
restricted_stock_deferred	0.1124

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I ended up getting the optimal performance with Decision Tree Classifier algorithm. I tried several other algorithms including AdaBoost, LogisticRegression, and LinearSVC. The performance appeared to be the best with Decision Tree Classifier compared to the other methods, as this was the one that had the best

performance in both precision and recall that I was able to obtain. The other classifiers did not perform as well.

Classifer	parameters	precision	recall
LogisticRegression	n_features=18, C=10, tol=1e-16	0.13499	0.20100
AdaBoostClassifier	n_features=13, n_estimators=40	0.38616	0.42400
LinearSVC	n_features=3, C=1000, tol=1e-128	0.16378	0.24600
DecisionTreeClassifier	n_features=18, criterion=entropy, min_samples_split=3	0.43343	0.62500

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).

Tuning the parameters of an algorithm means choosing altering (optimising) the parameters of the choosen algorithm to obtain the best performance. If this is not done well, the algorithm can under perform, and give less favourable outcome. I tuned the algorithm using gridsearch of sklearn, which tests mutliple different values for each parameter than selects the optimal values for each parameter that maximize the performance of the algorith. I choose two different parameters to be altered for Decision Tree Classifier: min_samples_split (varying from 2 to 6), and criterion (gini or entropy).

This results in the optimal performance with min_samples_split set to 4 and criterion set to entropy. This results in a precision of 0.43871 and a recall of 0.63700.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

The classic mistake is not spliting your training and testing prior to classification, which can give over performance of results by contaminating your algorithm by training on your testing dataset.I split the data into training and testing prior to running the algorithm. I randomly selected 10% of the data to set aside as testing data, and this was not used (or seen) by the algorithm during testing. Furthermore, I use kfold cross validation, with the number of sets set to 10. The performance for the algorithm differed slightly for each of the folds, but they were averaged together to give an overall estimate of the performance of the algorithm. K-fold cross validation the training dataset was split into 10 different sets, with 9 sets being used to create the algorithm and this was then testes on the 10th sets (the one not used to train). Then this final algorithm is used on the testing dataset, which has never been seen by the algoritm.

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.

I assessed the performance of my classifier using the tester.py functions and looking at precission and recall. Precision is the ratio of true positive to all those identified as positives (true positive plus false positives). True positives are people who are identified by the algorithm as a POI and are actually POI, whereas false postiives are people identified by the algorithm as a POI but are not actually POI. Precision therefore is the number of POIs correctly identified by the algorithm divided by the total number of people the algorithm believes are POIs. Recall is the ratio of true positives to all true events (true positive plus false negatives). False negatives are people who are identified by the algorithm as not being POI, but are actually POIs, so recall is quantifying how many POIs are identified by the algorithm given the true number of POI in the dataset. These metrics give you an idea of how well your algorithm is doing at classifying individuals into their correct category and also inform you about how many errors the alogrithm is making. Recall was difficult to keep high, and was much harder to get above 0.3 than precision.