Identify Fraud from Enron Email

Questions

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 goal of this project is to construct predictive model that could identify individual as a participant of fraud AKA Person of Interest (POI), based on financial and email data from Enron corpus. We will use different machine learning techniques to accomplish this goal.

The dataset contains 146 records with 1 label (POI), 14 financial features and 6 email features. There are 18 records which are labeled as POIs.

After further exploration I decided to remove three records:

1. ***TOTAL*** as it was revealed as an outlier, because it contained summed values for all financial data points
2. ***LOCKHART EUGENE E*** as it's meaningless to process data containing only NaNs
3. ***THE TRAVEL AGENCY IN THE PARK*** as in seems to be a data-entry error and not an individual

While the first one is clearly an outlier, the following two are clearly not correct entries, as they are not related to a “person”

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.*

Three new features are implemented: "fraction of emails to poi", "fraction of emails from poi", "fraction of emails with shared receipt with poi".

Fractions may be more accurate to use instead of the absolute amount of messages to/from/with poi. After that I looked at the feature importance scores of my extended set of features:

Feature	Score
Exercised stock options	24.815079733218194
Total stock value	24.182898678566879
Bonus	20.792252047181535
Salary	18.289684043404513
Fraction of emails to poi	16.409712548035813
Deferred income	11.458476579280369
Long term incentive	9.9221860131898225
Restricted stock	9.2128106219771002
Fraction of emails with poi	9.1012687391935341
Total payments	8.7727777300916792
Shared receipt with poi	8.589420731682381
Loan advances	7.1840556582887247
Expenses	6.0941733106389453
From poi to this person	5.2434497133749582
Other	4.1874775069953749
Fraction of emails from poi	3.1280917481567281

Feature	Score
From this person to poi	2.3826121082276739
Director fees	2.1263278020077054
To messages	1.6463411294420076
Deferral payments	0.22461127473600989
From messages	0.16970094762175533
Restricted stock deferred	0.065499652909942141

Using SelectKBest and comparing results of simple classifier (logistic regression with default tol an C) for different k, I got results by stratified cross-validation with 1000 folds:

k	Precision	Recall
1	0.25	0.47
2	0.26	0.32
3	0.31	0.40
4	0.30	0.47
5	0.34	0.67
6	0.34	0.70
7	0.33	0.68

k	Precision	Recall
8	0.33	0.68
9	0.32	0.64
10	0.27	0.57
11	0.25	0.53
12	0.17	0.93
13	0.17	0.92
14	0.17	0.92
15	0.17	0.90

6 features represent the highest score (one of which is created by me "fraction of emails to poi"). I tried decision tree as well on scaled features, although for this algorithm it wasn't necessary.

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

Finally, Logistic Regression (tuned) was the algorithm, selected. Naive Bayes, Decision Tree and SVM were also implemented. Table below shows with two metrics averaged by 1000 differently tests dataset created with StratifiedShuffleSplit for the tested algorithms.

Algorithm	Precision	Recall
Naive Bayes	0.48	0.35
Logistic Regression	0.34	0.70

Algorithm	Precision	Recall
Tuned Logistic Regression	0.30	0.94
Decision Tree	0.29	0.23
Tuned Decision Tree	0.37	0.58
SVM	0.25	0.24
Tuned SVM	0.29	0.35

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?*

Parameters have a deep impact in the performance of the algorithms. In order to tune (choosing a set of optimal parameters of an algorithm), we need to decide the target function. (e.g. minimize loss function). Tuning can improve the results or our model, but due to data limitations, an excess of tuning implies overfitting.

The required threshold of 0.3 for precision was set as a target. Additionally, Maximizing recall is the next step. With the GridSearchCV from [0.01, 0.1, 1, 10, 100, 1000] and tol from [0.0001, 0.001, 0.01, 0.1], I obtained the best parameters C=0.1 and tol=0.1.

During manual tuning near C=0.1 and tol=0.1 I was getting results that was very close to each other, with the maximum recall=0.94 and precision=0.30 (when C=0.02, tol=0.1).

C	tol	Precision	Recall
0.01	0.01	0.27	0.94
0.01	0.05	0.27	0.94
0.01	0.10	0.26	0.94

C	tol	Precision	Recall
0.02	0.10	0.30	0.94
0.02	0.20	0.29	0.94
0.025	0.10	0.31	0.93
0.025	0.20	0.30	0.93
0.025	0.30	0.28	0.93
0.03	0.10	0.31	0.92
0.03	0.20	0.30	0.92
0.03	0.30	0.29	0.93

For all algorithms I used the parameter class_weight as 'balanced'. Our classes have very different sizes and algorithms are showing better results when classes are balanced.

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

We would like to test our algorithm in a different set of data with the same characteristics. However, in our case dataset is limited, so we can't afford to held back data for validation. If we tune our algorithm on one specific test dataset, we can make a classic mistake of overfitting. We would get algorithm that shows high performance on one particular test dataset, but it's performance on other data could be much worse. In order to more objectively validate algorithm performance

without validation dataset, we could use stratified cross-validation. This technique generates asked number of different splits of data on train and test datasets in such manner, that proportion of classes is the same in train and test datasets in each split. I useds split with test_size = 10% of all dataset (default value in StrifiedShuffleSplit in sklearn).

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.*

As we can see in a table above, after tuning Logistic Regression showed average value of recall 0.94 and average value of precision 0.30. Recall can be explained as probability that our classifier will identify POI as POI, so we can say our algorithm "catches" 94% of all real POIs. Precision can be explained as probability that a POI identified as POI, is actually POI, so we can say our algorithm wrongfully "catches" 70% of non-POIs. Although classifier with such precision value can be considered as bad for different problems, in our case high recall is much more important.