Identify Fraud from Enron Email

Jean Paul Barddal

October 25, 2016

1 Introduction

This report details the Enron Email Fraud Identification project that aims at building employees that committed fraud, namely the Persons of Interest (POIs). It discusses several basic tasks of machine learning, including (i) extracting basic information about the dataset, (ii) the creation of new attributes, (iii) outlier identification and removal, (iii) data scaling, (iv) classification tuning, and (v) evaluation metrics for classifier comparison. This report is based on a work that used scikit-learn (`http://scikit-learn.org/stable/`).

This report is divided as follows. Section 2 introduces the data set and some of its characteristics. It also introduces the feature selection methods used and the results obtained. Later, it shows how outlier detection was performed, and how data was scaled prior to learning. Section 3 shows the rationale behind the classifier tuning process adopted and the results obtained. It also includes a thorough discussion of the results obtained by classifiers used in (i) the original dataset, (ii) the original dataset with the new features and (iii) the dataset with the selected attributes.

2 About the data set

2.1 Original data set

The developed script gathers some basic information on the data set, such as:

- Number of instances = 146
- Number of POIs = 18
- Number of features = 21

Table 1: Features selected for usage in each classifier.

Learner	Features selected
Gaussian NB	{exercised_stock_options}
SGD	{salary, total_payments, loan_advances, bonus, restricted_stock_deferred, deferred_income, total_stock_value, expenses, exercised_stock_options, director_fees, to_messages, from_poi_to_this_person, from_messages, shared_receipt_with_poi}
kNN	{total_payments, exercised_stock_options}
Decision Tree	{salary, total_payments, to_messages}

- Ratio between POIs and non POIs = 12%, 88%

Therefore, this is a low-dimensionality data set, yet, is unbalanced. If one assumes a ZeroR learning scheme as a baseline, it would be necessary to achieve accuracies above 88% to provide meaningful results. Roughly speaking, this occurs since classifiers, when applied over unbalanced data sets, will be biased towards the majority class, and if it forgets the minority class, and vote "not a POI" for each instance, it would obtained an 88% accuracy.

2.2 Creating new features

Two new features were created: "total_asset" and "fraction_of_messages_with_poi". The first is simple the sum of all the assets of an instance, i.e. "salary", "bonus", "total_stock_value" and "exercised_stock_options", while the second is the ratio between messages to and from a POI and all the messages.

2.3 Feature selection

The proposed method for feature selection was to maximize the F1-Weighted[1] metric of the selected subset of features given each classifier. In practice, this is mix between a step-wise and a wrapper-like approach, where attributes are selected sequentially (using SelectKBest) and evaluated using different classifiers. Using SelectKBest, features are selected regardind their direct correlation to the class. The evaluation using different classifiers is due that different classifiers work differently with different attributes. Table 1 presents the selected features for each classifier.

Table 2 presents the feature scores obtained using the SelectKBest method.

[1]F1-Weighted was used in replacement of F1 since the dataset is unbalanced.

Table 2: Features scores.

Feature	Score
to_messages	18.28968404
deferral_payments	0.23309098
expenses	8.77277773
deferred_income	7.18405566
long_term_incentive	20.79225205
restricted_stock_deferred	0.24705326
shared_receipt_with_poi	11.45847658
loan_advances	24.17997245
from_messages	6.09417331
other	24.81507973
director_fees	4.18747751
bonus	9.92218601
total_stock_value	8.82867901
from_poi_to_this_person	2.1263278
fraction_of_messages_with_poi	1.64634113
from_this_person_to_poi	5.24344971
restricted_stock	0.16970095
salary	2.38261211
total_payments	8.58942073
exercised_stock_options	15.36927686
total_asset	5.39937029

2.4 Outlier detection and data cleansing

If one looks at the PDF provided, which was used to create the data set, he will see that two instances, namely "TOTAL" and "THE TRAVEL AGENCY IN THE PARK" have a very distinct behavior. The first instance seems to be the sum of all others and is not a representative example for training, while the second has values in only two of its attributes.

On top of that, all of the employee named "LOCKHART EUGENE E" was also removed, since more than 95% of its attributes were NaNs[2].

Next, the data set was cleaned by turning all NaN data into 0 and all negative values into their absolute value, since the data in the provided PDF file has no values below zero.

2.5 Data scaling

Since not all attributes are bounded in the same interval, e.g. most of the e-mail related attributes have a range which is much smaller than the asset ones that might scale way higher, data scaling is needed to make sure that all features have the same weight. The procedure adopted for data scaling was simple, which is also the rule of thumb for many books and papers on machine learning. The MinMaxScaler from scikit-learn was used to transform features into the $[0; 1]$ interval. The transformation of an attribute X was done as follows:

$$\sigma_X = \frac{(X - \min X)}{\max X - \min X}$$

$$X_{scaled} = \sigma_X \times (\max X - \min X) + \min X$$

Scaling is also important since many learning algorithms, e.g. SGD, Gaussian NB, assume that the data distribution follows a Gaussian distribution. Even though the same does not hold for kNN, scaling is also important to avoid that too great distances jeopardize distance computations, since a single feature may "dominate" the overall distance computation. Also, decision tree is able to work with unscaled data, but data scaling was also applied to ease the comparison of the results against the other classifiers.

3 Results

This section presents the results obtained in the Enron dataset. First, it is detailed how classifier's were tuned, and these tuned versions are then

[2]NaN = Not a Number

Table 3: Algorithms' parameters used for tuning.

Classifier	Parameter	Values tested
kNN	k	$[1; 20]$
	p[5]	$\{1, 2, 3\}$
	search algorithm	brute[6]
SGD	loss	{hinge, log, squared_hinge, perceptron}
	penalty	{l1, l2, elasticnet}
Gaussian NB	–	–
Decision Tree	Criterion	{gini, entropy}
	splitter	{best, random}

evaluated against one another in three variations of the Enron data set.

3.1 Classifier tuning

When applying learning algorithms to different domains, classifiers are likely to act differently. Even though researchers try to provide reasonable default parameters, there is no "one-fits-all"[3] set of parameters that will succeed in all scenarios. Classifier tuning is a step in the machine learning evaluation that compares several different combinations of parameters with the goal of achieving higher classification rates[4]. Tuning is in practice a search problem, and thus, many different strategies can be applied to find a fair subset of parameters that maximizes your goal measure. In the performed experiments, tuning was performed using a brute force strategy, where all possible combinations of parameters were evaluated with the goal of maximizing the F1-Weighted measure in a 10-fold cross-validation scheme. F1-Weighted is a variation of the conventional F1 measure (which is an harmonic mean between the precision and recall) that accounts for instances of each class differently since the problem is unbalanced.

Table 3 presents the algorithms evaluated and the parameters tuned.

3.2 Data set variations

The tuned classifiers are finally compared against one another in four variations of the Enron data set:

- (Original) The original data set, rescaled and without outliers
- (New Atts) Original + new attributes

[3] Usually referred as the no free lunch theorem.

[4] In certain scenarios, such as stream processing, the goal could also be to find a fair trade-off between accuracy, processing time and memory usage.

Table 4: Average accuracy (%) obtained in the test script.

Accuracy (%)				
Data set	Gaussian NB	kNN	SGD	Decision Tree
Original	37.62	**87.53**	**77.35**	79.87
New Atts	37.71	86.00	62.73	**80.75**
Selected	**90.41**	84.94	70.33	76.00
Selected + New	83.29	82.16	55.07	78.29

- (Selected) New Atts + Feature selection
- (Selected + New) Selected + new attributes

These variations will allows a better understanding to the results obtained, since evaluations on how each classifier performs with and without the new attributes and/or feature selection will be possible. It is important to highlight that the last variation is used solely to verify the impact of the newly proposed attributes in the results obtained.

3.3 Overall evaluation

Validation is one of the most important parts of building and deploying applied machine learning techniques. The goal of validation is to verify whether a classifier will predict well when applied in real-world scenarios. The rationale behind most of the validation schemes is that the training set and test sets are different, however, built upon the same data generator[7]. Holdout is one of the most simple yet used validation schemes. It splits the data set into two disjoint subsets: training and test. A classifier is then built using the training set and applied to the test set. Even though holdout looks interesting, the test set could be extremely equal or different from the training set, and thus, highly over-fitted or under-fitted results could be obtained. To avoid this issue, a 10-fold cross-validation was applied, where a holdout validation was repeated 10 times and the resulting average is reported. Evaluating classifiers using cross-validation is beneficial since it assesses the variability of the results, since it is possible that during one of these folds, classifiers will obtain results that largely deviate from the mean, a fact that should be verified and analyzed. It is also worth mentioning that the same procedure was used during feature selection.

Tables 4, 5 and 6 present the results obtained by the tuned classifiers in terms of Accuracy, Precision and Recall, respectively.

[7]The same does not hold for streaming settings.

Verifying the results obtained in accuracy (Table 4), it is important to highlight that most of the results obtained are below the ZeroR baseline of 88%, with the exception of the Gaussian Naive Bayes in the Selected data set. Another interesting result that should be highlighted is the accuracy increase obtained by the same classifier amongst the data set variations, where the accuracy went from approximately 38% to 90%. On the other hand, kNN, Stochastic Gradient Descent (SGD) and the Decision Tree obtained results below the baseline, however, using the selected subset of attributes decreased the average accuracy obtained.

Besides accuracy, specific metrics for POI recognition should be evaluated. To do so, both Precision and Recall metrics were evaluated, and these metrics are computed according to Equations 1 and 2, respectively, where tp are the true positives, fp are the false positives and fn are the false negatives.

$$Precision = \frac{tp}{tp + fp} \tag{1}$$

$$Recall = \frac{tp}{tp + fn} \tag{2}$$

High precision results relates to a low false positive rate (not many false POIs detected), while an high recall relates to a low false negative rate (not many false non-POIs detected). Therefore, both precision and recall should be maximized.

According to the results obtained in Tables 5 and 6, the only classifier able to achieve precision and recall results above 0.3 is the Gaussian Naive Bayes in the selected data set. In practice, the results obtained for this combination of classifier and data set shows that it is possible to state that 50% of the instances flagged as POI by the classifier are really POIs (precision), while 32% of the POI instances are correctly classified as a POI. Again, it is highlighted the last variation of the dataset, which regards the selected features plus the newly generated attributes. Focusing on the Gaussian NB results, precision presents an increase of 0.04 while recall decreases by 0.59.

Table 5: Average precision obtained in the test script.

Precision				
Data set	Gaussian NB	kNN	SGD	Decision Tree
Original	0.16	**0.57**	0.06	0.26
New Atts	0.16	0.42	0.09	**0.29**
Selected	0.46	0.40	0.12	0.16
Selected + New	**0.50**	0.34	**0.13**	0.26

Table 6: Average recall obtained in the test script.

Recall				
Data set	Gaussian NB	kNN	SGD	Decision Tree
Original	**0.83**	**0.27**	0.04	0.28
New Atts	**0.83**	0.13	0.19	**0.31**
Selected	0.32	0.10	0.19	0.17
Selected + New	0.24	0.26	**0.29**	0.28