Enron person of interest identification

Krasin Georgiev

1. Introduction

The objective of this project is to build a person of interest identifier (POI) based on financial and email data from ENRON corporation. For the purpose general data exploration and machine learning techniques will be used. Acceptable performance is precision > 0.3 and recall > 0.3 on unseen data. Most of the information in this report is based on Udacity Introduction to Machine Learning course materials [1].

2. Background

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives. A list of persons involved in the fraud case (individuals who were indicted, reached a settlement, or plea deal with the government, or testified in exchange for prosecution immunity - POI) is available.

3. General Approach

After some data exploration and cleaning, preliminary classifier was selected and the whole infrastructure was tested using combinations of intuitive features. Random forests classifier was our first choice. It is simple to use and very powerful. It is nonlinear and does not require normalization. And finally there is a feature importances metric which is very useful for feature selection.

Then, data were split to test and train set for local testing. Results from study of this dataset were compared with the one achieved by official evaluation code based on shuffling and cross-validation.

Feature selection and different classifiers testing and tuning follow.

4. Data Description and Exploration

4.1. Data and features

Input information consists of financial information tables and full text emails. Financial features are imported directly. Email features are generated from the emails. Finally we have labels for POI

- Financial features, numeric:

['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']

- Email features
    - numeric:

['to_messages', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi']

    - Email features, text:

['email_address']

- POI label, factor:

[‘poi’]

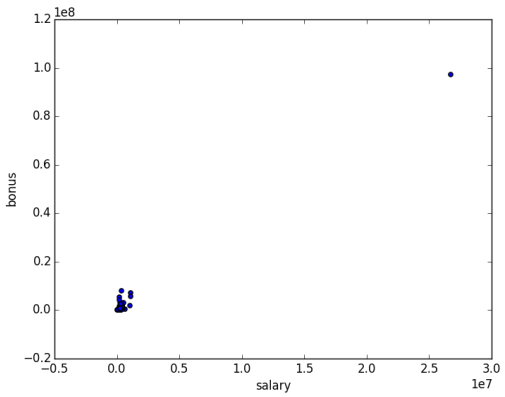
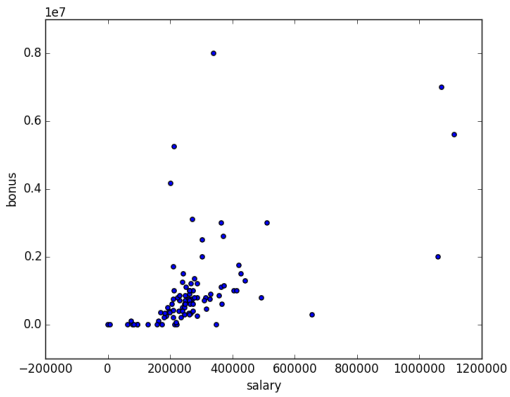
Therefore there are 19 initial numeric features and one label.

4.2. Basic statistics

Data were initially explored using summary statistics, scatter plots and a boxplot. The main ones are:

- total number of data points: total data points: 145
- number of POI: 18 (12%)

An obvious outlier is discovered. It is visible on the scatter plot of salary and bonus features - all data points are concentrated in a region far away from the outlier point.

Initial data lot	Plot after outlier removal
	

4.3. Outliers removal

From the raw data tables we can see that our data contain the 'Total' sum of all records. It is removed with the following line of code:

```
data_dict.pop( 'TOTAL', 0 )
```

5. Feature Creation and Selection

5.1. Assessment of available features

Classification using “salary” as the only one feature was used initially to achieve bug free code. Then classification with all starting features was performed. Random forest classifier was selected, as explained in section “General Approach”.

Features	Precision	Recall	F1	Dataset
“salary”	**0.240**	**0.200**	**0.218**	**full, shuffeled, cv**
“salary”	0.11	0.14	0.12	50% split
all	**0.31**	**0.14**	**0.19**	**full, shuffeled, cv**
all	0.33	0.20		50% split, random

				state 40
all	0.25	0.10	0.14	50% split, random state 42

Using all available features does not lead to significant improvement of the model and the performance is below the required level. Therefore smart selection of features and adding new features is needed.

Another conclusion is that having simple data split introduces significant random element in the analysis. At the same time if we do cross-validation we use information from all available data.

5.2. Adding new features

New features were calculated based on email contacts relationships:

- email available, 'maildata'
- ratio of emails to POI and all send emails, 'to_ratio'
- ratio of emails from POI and all received emails, 'from_ratio'
- sum of above ratios, 'comm_ratio'
- communication with POI, topoi>60 and frpoi>60, ‘comm’
- communication with POI, topoi>0 and frpoi>30, ‘comm2’
- total emails to/from POI, 'comm_sum'
- max of emails to/from POI, 'comm_max'
- min of emails to/from POI, 'comm_min'

Also, transforming all features by taking logarithm was tested. No significant changes were recorded.

Features	Precision	Recall	F1	Dataset
all	**0.34**	**0.16**	**0.22**	**full, shuffeled, cv**
all	0.40	0.20	0.27	50% split, random state 40
all	0.22	0.20	0.21	50% split, random state 42

5.3. Feature selection

Several algorithms for feature assessment and selection were used:

- feature importances metric of Tree based classifier (Random Forest)
- p-values for coefficients in linear regression model
- Univariate feature selection [2] - the effect of every single feature testing using SelectKBest with f_classif

After fitting the model with all available features we have:

Features importance. Sorted list of top features.

0.256344 | bonus
0.153917 | total_stock_value
0.144239 | salary
0.133040 | exercised_stock_options
0.083285 | long_term_incentive
0.058375 | from_ratio
0.053524 | from_this_person_to_poi
0.037439 | shared_receipt_with_poi
0.030704 | comm_ratio
0.014675 | restricted_stock
0.013750 | from_poi_to_this_person
0.010549 | deferral_payments
0.010161 | expenses

P-values for the coefficients of a linear regression, ordered

0.001304 | comm2
0.010991 | comm_max
0.012261 | comm_min
0.029769 | comm_sum
0.095676 | maildata
0.125158 | from_this_person_to_poi
0.167295 | shared_receipt_with_poi
0.169917 | from_poi_to_this_person
0.195791 | to_messages
0.308069 | expenses
0.398635 | comm_ratio
0.429045 | from_ratio
0.486762 | from_messages

SelectKBest:

['salary', 'total_payments', 'loan_advances', 'bonus', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock']

A custom list of features from above lists is created. It is important to add ‘poi’ as first feature, otherwise the code will break.

Based on the results 'comm_max', 'comm_min', 'comm_sum' are removed.

Features	Precision	Recall	F1	Dataset
[‘poi’, 'bonus', 'total_stock_value', 'salary', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock',	0.356	0.164	0.22	full, shuffeled, cv

'comm2', 'comm_max', 'comm_min', 'comm_sum']				
[‘poi’, 'bonus', 'total_stock_value', 'salary', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'comm2']	**0.356**	**0.201**	**0.256**	**full, shuffeled, cv**

6. Pick and tune an algorithm

6.1. Try a variety of classifiers

Random forests, Logistic Regression, Gaussian Naive Bayes and SVM were tested. Feature scaling was performed in order to be able to get good results for the logistic regression and SVM. Pipelines were used in order to do multi-stage operations, e.g. feature scaling first, applying classifier than [3].

Classifier	Precision	Recall	F1
RandomForestClassifier	0.348	0.192	0.248
GaussianNB	**0.425**	**0.306**	**0.355**
LogisticRegression, with feature scaling	0.467	0.165	0.244
SVC, no feature scaling	error	error	error
SVC(kernel="rbf", C=100), with feature scaling	0.138	0.063	0.087

We get best performance with Gaussian Naive Bayes algorithm, one of the fastest and simplest algorithms.

6.2. Tune the classifier

Because there are no parameters to tune for Gaussian Naive Bayes classifier we will try to improve the performance of the Random Forests. The idea is to test the model with different settings and select the one with best performance, i.e highest score. We select F1 as optimization metric, because it better reflect precision and recall than accuracy for skewed data.

Classifier	Precision	Recall	F1
RandomForestClassifier(n_estimators=10, max_features=2, min_samples_split=4)	0.433	0.205	0.278

n_estimators = [10, 100, 200, 500], max_features = [None, 2, 4, 6]			
RandomForestClassifier(n_estimators=50, max_features=3, min_samples_split=4) (manually tested)	0.450	0.213	0.290
SVC(kernel="rbf", C=1000000, gamma = 0.0) C = [100, 100000, 1000000, 10000000], gamma = [0, 2, 10]	0.400	0.223	0.286

7. Test and evaluate the identifier

Validation is technique for assessing how the results of a statistical analysis will generalize to an independent data set. If we asses our results only based on the data used to create the model we will have too optimistic results in case of overfitting. To overcome this problem the full dataset is split into train set and test set in equal proportions (50% test set). **Feature selection is based on the train set only**. The test set is used only to calculate model performance (precision and recall in our case). This is one of the reasons our model not to have very high performance.

To evaluate performance we use cross-validation routine provided by Udacity in test.py file. Precision and recall are used consistently through all analysis to evaluate algorithm performance. The precision can be interpreted as the likelihood that a person who is identified as a POI is actually a true POI; the fact that this is 0.43 means that using this identifier to flag POI's would result in 57% of the positive flags being false alarms. Recall measures how likely it is that, given that there's a POI in the test set, this identifier would flag him or her; 31% of the time it would catch that person, and 69% of the time it wouldn't [5].

8. Summary and Conclusions

The main features of the model identified were ['poi', 'bonus', 'total_stock_value', 'salary', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'comm2']. This includes six financial features and one reflecting email communication. Only the train set is used for feature selection.

Selected classifier is Gaussian Naive Bayes with precision of 0.425 and recall 0.306. The other tested models couldn't achieve better result even after some parameter tuning. Therefore the model could not be used on its own to make predictions. It will miss nearly 70 % of the targets. And the identified one will be true in nearly 60% of all cases.

Removing our added feature further increase the performance of the model :) This could be explained with the fact that feature selection was tuned for another classifier (Random Forests). Final feature set is ['poi', **'bonus'**, **'total_stock_value'**, **'exercised_stock_options'**, **'restricted_stock'**] with model performance **precision: 0.44** and **recall: 0.32**.

Major limitation of the dataset is that there are only 18 cases of POI. "There were 35 people who were POIs in "real life", but for various reasons, half of those are not present in this dataset--by the "more data is better than a finely-tuned algorithm" maxim of machine learning, having a larger dataset is likely the single biggest way to improve the performance of this analysis" [5].

Model could be improved to some extent if additional features are created based on more advanced processing of email corpus available. Another option is to do more extensive

parameter tuning for Random Forest or SVM models, which is computationally intensive task and could be time consuming. In any case, overfitting is very likely.

References

1. Udacity, Final Project Sample Report, https://www.udacity.com/wiki/ud120/finalproject
2. Scikit-learn documentation, Feature Selection, http://scikit-learn.org/stable/modules/feature_selection.html
3. Scikit-learn documentation, Pipeline and FeatureUnion: combining estimators, http://scikit-learn.org/stable/modules/pipeline.html
4. Grid Search: Searching for estimator parameters, http://scikit-learn.org/stable/modules/grid_search.html
5. Udacity, Enron POI Report Example.
6. Alex Brazhenko, Udacity Nanodegree. Project 4. Identifying Fraud from Enron Email, https://github.com/alexbra/ud-nd-project4/blob/master/poi_id.py

Code snippets

```
################################################################################
# Classifier parameter tuning
# Grid search with pipeline

parameters = {'SVC__C':[10, 100, 10000, 1000000], 'SVC__gamma':[0, 1, 2, 4, 8,
100]}
parameters = {'svc__C':[100, 100000, 10000000], 'svc__gamma':[0, 2, 10]}

SVM = Pipeline([('scale', preprocessing.StandardScaler()), ('svc', SVC())])
clf = grid_search.GridSearchCV(SVM, parameters, scoring='f1')

test_classifier(clf, my_dataset, features_list)
print clf.best_estimator_

################################################################################
## Classifiers tested

from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=50, max_features=None)

from sklearn import linear_model
clf = linear_model.LogisticRegression()

from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()

from sklearn.svm import SVC
clf = SVC(kernel="rbf", C=10000)
clf = SVC(kernel="linear", C=10000)

from sklearn.pipeline import Pipeline, make_pipeline
from sklearn import preprocessing
clf = make_pipeline(preprocessing.StandardScaler(),
linear_model.LogisticRegression()) # big effect of normalization
```