Data Analyst Nanodegree
Project: Identifying Fraud from Enron Email
Project Report

Submission #1
Author: Konstantin Kunz
Date: 2015-10-25

Question (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”]

Answer (1)

The aim of this project is to create a person of interest (poi) identifier within the scope of the Enron fraud case of the early years of this millennium. A person of interest is a person that could be linked to being part of the fraud going on in the company. We use machine learning to create the identifier because of the diverse and complex nature of the data. The complexity makes it impossible to create an analytical model of the system, hence we use machine learning methods. The data is already labeled as belonging to a poi or not. Machine learning can use these labels in order to create an identifier via supervised learning. It might be that some people are actually persons of interest but were never prosecuted.

The data we have at our hands are the Enron email corpus and the financial data of the listed people.
There are totally 146 data rows, with 21 features each. The features can be split up into email-related features and financial features.
The 146 data rows contain 18 persons of interest - 12% of the data rows are POIs.
In this data set we do not have a lot of data, and even less data of the POIs. This is critical as we have to be careful and not discard a lot of data during the outlier selection. It might even be that the outliers have special predictive importance.
If we look at the totally available data points there are totally 1708 non-zero (non-NaN) data points. 269 of these non-zero data points fall into POI data rows - this represents about 16% of the non-zero data points. We have slightly more data from POIs than from non-POIs. This is a good sign - at least the data amount for POIs is higher than for the other persons.
On the other hand a NaN value could also actually be valuable information because it encoded the value 0 in financial features in this dataset.
The top 3 features with the most non-zero data points are total_stock_value, total_payments, email_address. The top 3 features with most zero data entries are loan_advances, restricted_stock_deferred, director_fees. An initial guess is that features that have more data points have more predictive power and vis-versa. This remains to be confirmed during the feature selection.
When looking at just the POIs there are certain financial features that a large part of the POIs have non-zero values (excluding the total data points):

expenses (100%), other (100%), restricted_stock (94%), salary (94%), bonus (88%)
Comparing this to the non-POI group:
restricted_stock (73%), exercised_stock_options (70%), salary (60%), expenses (60%), other (58%)
Since the POIs have a different distribution of the features with non-zero data this could give us a discriminative advantage when building the identifier.

The dataset had a variety of outliers.

(1) One of the more obvious ones was the addition of the 'TOTAL' data row in the dataset. This is accumulation of all financial data and thus does not represent an individual. It can be taken out of the data entirely.
(2) Another less easy to find outlier was the 'The Travel Agency in the Park'. This data row represents a travel agency owned by a relative of a former Enron chairman. The data for this data row was only the 'other' payments. There were no emails or any other data. Since it is not a person, and the available data is so limited, I deem this data invaluable for our analysis and will take it out of the data set entirely.
(3) A third outlier I noticed was the negative value for ‘restricted_stock’ of 'BHATNAGAR SANJAY'. All values of this feature should be positive, which made me curious. The original data set had no dash for 'other' payments, and what happened is that all values got shifted over. To unbundle this the values had to be shifted back according to the order they are in the financial data sheet.
(4) In the same way the financial feature values of 'BELFER ROBERT’ got shifted. With respect to the financial data sheet they are shifted one to the left.
(5) Another outlier that should be treated is the data of 'LOCKHART EUGENE E’. He had only zero data entries, which makes him a zero information data point. This data will also be taken out of the data set.

So far for the data points that had to be treated. There are also a not negligible amount of data points that seem like outliers but are actually valid data points. Here I will only list some extreme examples as there are plenty of them.

(1) ‘loan_advances’ of 81.5 MUSD for ‘LAY KENNETH L’
(2) 8 MUSD ‘bonus’ of 'LAVORATO JOHN J’
(3) 10.3 MUSD ‘other’ payments for ‘LAY KENNETH L’
(4) ‘total_stock’ values above 20 MUSD for ‘LAY KENNETH L’, ‘RICE KENNETH D’, ‘SKILLING JEFFREY K’, ‘HIRKO JOSEPH’, ‘PAI LOU L’

Question (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”]

Answer (2)

While looking at the different features I thought of some features that could bring added value to the classifier problem.

New feature	Formula	Reasoning
total_poi_interact	from_poi_to_this_person + shared_receipt_with_poi + from_this_person_to_poi'	Sum of mail interaction of a person with POIs. Could be a measure of connectedness
from_poi_ratio	from_poi_to_this_person/ to_messages	What percentage of mails this person receives is sent by POIs. It could be that people write a different amount of mails due to communication habits. That is why ratios could give more insight than absolute numbers.
to_poi_ratio	from_this_person_to_poi/ from_messages	What percentage of mails written by this person is sent to a POI. Due to communication habits absolute numbers might be misleading.
from_non_poi_to_this_person	to_messages - from_poi_to_this_person	Differentiation between POI mails and non-POI mails.
bonus_salary_ratio	bonus/salary	This feature measures if the bonus is excessively large when compared to the salary.
total_incentives	bonus + long_term_incentive	This feature aggregates the entire cash incentives offered to a person, short term (bonus) and long term. It could help understand if the incentives where used in a mix to create fraud.
total_compensation	total_payments + total_stock_value	This feature is a hard measure of how much money a person made. It can be interesting to see if a person made moderate gains on both cash and stock but in total comes out with a top gain.
payments_stock_ratio	total_payments/ total_stock_value	This feature encodes the ratio of cash payments and stock compensation. It could give us a hint if there is an unusually high stock payment in relation to the cash and vis-versa.

The engineered features had different effects on the final classifier. The table below quantifies these effects. The ratio of the f1 score of the final classifier before and after adding the engineered feature to the dataset gives a good intuition if the feature adds

information to the classification problem. Due to computation time reasons the classifier was not tuned fully for each feature added. The only tuned parameter was the number of features used in the classifier. This doesn't narrow the power of the qualitative analysis of the effect. The two features with the large positive effect on the resulting score were total_compensation and to_poi_ratio. This coincides with the high score in the SelectKBest selection stage.

f1-score	w/ eng. feature	Ratio of f1-score with/without feature
total_poi_interact	0.25	0.99
from_poi_ratio	0.24	0.96
to_poi_ratio	0.28	1.11
from_non_poi_to_this_person	0.26	1.03
bonus_salary_ratio	0.23	0.94
total_incentives	0.25	1.01
total_compensation	0.30	1.22
payments_stock_ratio	0.23	0.94

For the final classifier feature selection I used a sequence of first selecting the k best features by using the SelectKBest function, these features were then fed to the second stage of feature selection, a principle component analysis (PCA), in order to reduce the dimensionality and generalise the features further.
The parameters of these selection stage functions were tuned using GridSearchCV to maximize the f1-score of the classifier. SelectKBest used the ANOVA f-score to compute the feature scores ranking the features. The feature scores can be seen in the table below.
The PCA parameter n_components was tuned as to select the number of principle components that explain p percent of the variance in the feature space.
Three classifiers were compared: DecisionTree, SVM & KNearestNeighbors. The detailed tuning steps and decision process are explained below in Answer (3).
The numbers of features selected by the algorithms after tuning is shown in the below table. The final classifier used 14 features, which is still a reasonable amount compared to the total number of features (27). The classifier is neither overfitting nor over-generalising.

My final classifier did not require any scaling (DecisionTree). The other classifiers that were tested in the classifier selection phase required scaling (SVM and kNearestNeighbors) since they compute the classifier based on a distance type metric. I used the sklearn MinMaxScaler to scale all features between 0 and 1 for these cases.

Algorithm	# features used	% of variance explained by PCs
DecisionTree	14	50
SVM	9	80
k-NN	3	70

Rank	Feature	Score
1	total_stock_value	22.5105490902
2	exercised_stock_options	22.3489754073
3	total_incentives	21.9717374928
4	bonus	20.7922520472
5	salary	18.2896840434
6	total_compensation	16.9893364218
7	to_poi_ratio	16.409712548
8	deferred_income	11.4248914854
9	bonus_salary_ratio	10.7835847082
10	long_term_incentive	9.92218601319
11	total_payments	9.28387361843
12	restricted_stock	8.82544221992
13	total_poi_interact	8.61664819811
14	shared_receipt_with_poi	8.58942073168
15	loan_advances	7.18405565829
16	expenses	5.41890018941
17	from_poi_to_this_person	5.24344971337
18	other	4.20243630027
19	from_poi_ratio	3.12809174816
20	from_this_person_to_poi	2.38261210823
21	director_fees	2.13148399246
22	to_messages	1.64634112944
23	from_non_poi_to_this_person	1.51861100786
24	restricted_stock_deferred	0.768146344787
25	payments_stock_ratio	0.605428626917
26	deferral_payments	0.228859619021
27	from_messages	0.169700947622

Question (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”]

Answer (3)

I ended up using a decision tree classifier in combination with a PCA and SelectKBest. First the k best features were selected, then PCA reduced them further by feeding the the principle components to the decision tree.

In comparison I also used a SVM and a KNearestNeighbors algorithm. During the comparison phase all three algorithms were tuned using GridSearchCV. The performance was compared by comparing the f1 score of the classifier on StratifiedShuffleSplits with a 1000 folds. Also the recall and precision for the tuned parameters are shown.

The f1 score of the different algorithms are as below. We can see that the decision tree gives the best results. I will go further in analysing this classifier in more detail.

Algorithm	**f1**	**recall**	**precision**
Decision Tree	0.39	0.39	0.40
SVM	0.20	0.21	0.18
KNearestNeighbors	0.28	0.26	0.31

Algorithm	**DecisionTree**	**SVM**	**KNearestNeighbors**
k	14	9	3
n_components	0.5	0.8	0.7
class_weight	NONE	NONE	-
min_samples_split	8	-	-
C	-	500	-
n_neighbors	-	-	3
whiten	TRUE	TRUE	TRUE

Question (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"]

Answer (4)

Tuning the parameters of an algorithm means selecting the combination of parameters that gives the best performance on the data at hand. The reason why it is necessary is because the data can have large differences that need to be taken into account in the machine learning algorithm (e.g. number of features, number of samples, number of positive labels vs negative labels and more). The performance can best measured by a variety of scores and should be defined beforehand. In our case one could argue that recall is a good performance measure since we have very little positiv samples. The f1 score is also a suitable measure since it measures a tradeoff between precision and recall.

In my case I used the GridSearchCV function to tune the three functions used in the classifier pipeline: SelectKBest, PCA and DecisionTree. The f1 score was used as performance measure. The final parameters were selected as below.

Parameter	Function	Tuned Value
n_components	SelectKBest	14
n_components	PCA	0.5
whiten	PCA	TRUE
class_weight	DecisionTree	NONE
min_sample_split	DecisionTree	8

Question (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"]

Answer (5)

Validation is the process of evaluating the performance of a classifier algorithm and thus validating the classifier. A classic mistake that can be done is to use the training data as test data. In my analysis I used the StratifiedShuffleSplit algorithm in combination with the GridSearchCV algorithm. The reason behind using this specific algorithm is that it returns randomised folds of the data, eliminating the risk of underlying patterns in the data. Also it return stratified folds, making sure that we always have enough positive labels in the training/test sets since it preserves this ratio.

Question (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"]

Answer (6)

The score that I evaluated the final parameters on was the f1 score. The f1 score can be interpreted as an average of the precision and recall.

The precision and recall are the basis of this metric.
The recall of a classifier is the amount of true positives divided by the sum of the true positives and of false negatives. In other words the probability of the algorithm to correctly predict a certain label given that the data point is actually that label. In our specific case, the probability of detecting a poi given that the data point actually belongs to a poi.
The precision is a metric that allows one to evaluate the correctness of a prediction. Given that the algorithm predicted a certain label (poi), what is the probability that this is actually this label. It is calculated as the ratio of true positives and the sum of true positives and false positives.

	f1	recall	precision
DecisionTree Classifier	0.40	0.37	0.43

Resources

Python documentation
https://docs.python.org/2/library/index.html

Sklearn documentation
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html
http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.scatter
http://scikit-learn.org/stable/modules/ensemble.html
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
http://scikit-learn.org/stable/modules/generated/
sklearn.metrics.recall_score.html#sklearn.metrics.recall_score
http://scikit-learn.org/stable/modules/generated/
sklearn.metrics.precision_score.html#sklearn.metrics.precision_score
http://scikit-learn.org/stable/modules/generated/
sklearn.cross_validation.StratifiedShuffleSplit.html#sklearn.cross_validation.StratifiedShuffleSplit
http://scikit-learn.org/stable/modules/generated/
sklearn.cross_validation.KFold.html#sklearn.cross_validation.KFold
http://scikit-learn.org/stable/modules/feature_selection.html

Wikipedia
https://en.wikipedia.org/wiki/Decision_tree_learning
https://en.wikipedia.org/wiki/Support_vector_machine
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
https://en.wikipedia.org/wiki/Machine_learning
https://en.wikipedia.org/wiki/Supervised_learning
https://en.wikipedia.org/wiki/Feature_selection

Udacity
https://plus.google.com/u/0/events/c330agh10tgbph1it9n26kd1ib0?authkey=CMrJtqDHq4TzsgE
https://www.udacity.com/course/viewer#!/c-ud120-nd
https://discussions.udacity.com/t/outliers-features-and-their-effect-on-other-features/34672/3
https://discussions.udacity.com/t/feature-selection-vs-pca/34958

Stack overflow
http://stackoverflow.com/questions/8459231/sort-tuples-based-on-second-parameter