1. Goal

The goal of this project is to create a model that can predict based on a number of features if an employee of Enron is a person of interest (POI) or not (non-POI). We provide our model with characteristics (features) of persons and if these persons are POIs or not (labels). We train our model on these features and labels. Thereafter we test our model for another set of persons and see if our model predicts the correct label (POI or non-POI).

Dataset

The dataset has data for 146 Enron employees. 18 of them are marked as persons of interest in the Enron fraud case. These POIs are individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

The dataset consists of 20 features and the POI label. There are two kinds of features:

- Email features (6)
    - emailto_messages, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi
        - units are the number of email messages
    - Email_address
        - Text
- Financial features (14)
    - 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
        - units are in US dollars

This dataset is imbalanced, with only 18 of the 146 data points (=employees) being marked as POIs. Machine learning provides best predicting models when working with a balanced dataset, meaning that positive and negative data points make up equal parts.

Missing values

	total	poi	non poi	% missing poi	% missing non poi
from_messages	60	4	56	22	43
from_poi_to_this_person	60	4	56	22	43
from_this_person_to_poi	60	4	56	22	43
shared_receipt_with_poi	60	4	56	22	43
to_messages	60	4	56	22	43
email_address	35	0	35	0	27

Table 1: Missing email features

For the email features, the email address is missing for 35 non-POI employees. All other email features are missing for 60 employees, 56 of these are non-POIs. The source of these email features is unclear, but it seems data for these employees are missing. So we are dealing with an incomplete dataset.

For the financial features, there are a different number of missing values per feature as can be seen in the table below. For this part of the data I feel confident that if a feature is missing for an employee this employee did not receive that particular type of financial compensation. For example only 11% of the POIs are missing bonus data, while 48% of the non-POI employees do. Most missing values are for loan advances, director fees and the deferred features which seems to be expected as these are not standard types of financial compensation.

	total	**poi**	**non poi**	**% missing poi**	**% missing non poi**
loan_advances	142	17	125	94	96
director_fees	129	18	111	100	85
restricted_stock_deferred	128	18	110	100	85
deferral_payments	107	13	94	72	72
deferred_income	97	7	90	39	69
long_term_incentive	80	6	74	33	57
bonus	64	2	62	11	48
other	53	0	53	0	41
expenses	51	0	51	0	39
salary	51	1	50	6	38
exercised_stock_options	44	6	38	33	29
restricted_stock	36	1	35	6	27
total_payments	21	0	21	0	16
total_stock_value	20	0	20	0	15

Table 2: Missing financial features

There was one employee (Eugene E Lockhart) for who all features are missing. I decided to drop him from the dataset, resulting in 145 employees.

Outliers

When plotting the features in boxplots (see next two pages) with POI vs non-POI on the x-axis, I noticed that the largest data point was often in the non-POI group which was unexpected. Looking closer at the original pdf and cross referencing the numbers, I found that the row 'TOTAL' was included. I also noticed that there was a row named 'THE TRAVEL AGENCY IN THE PARK', which did not sound as an Enron employee either. Both rows were dropped from the dataset, resulting in a dataset of 143 employees (of which 18 POIs).

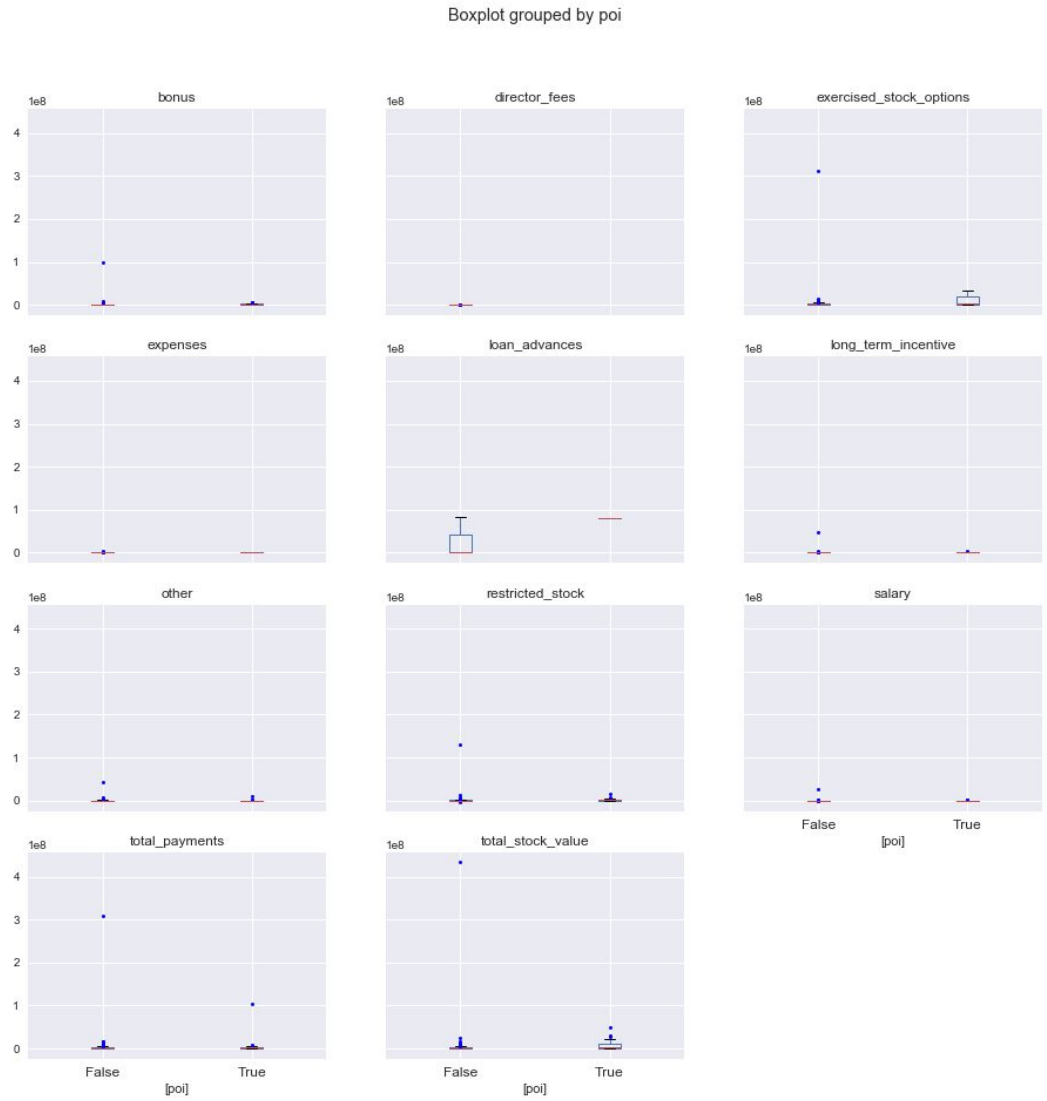

Figure 1: Boxplots financial features

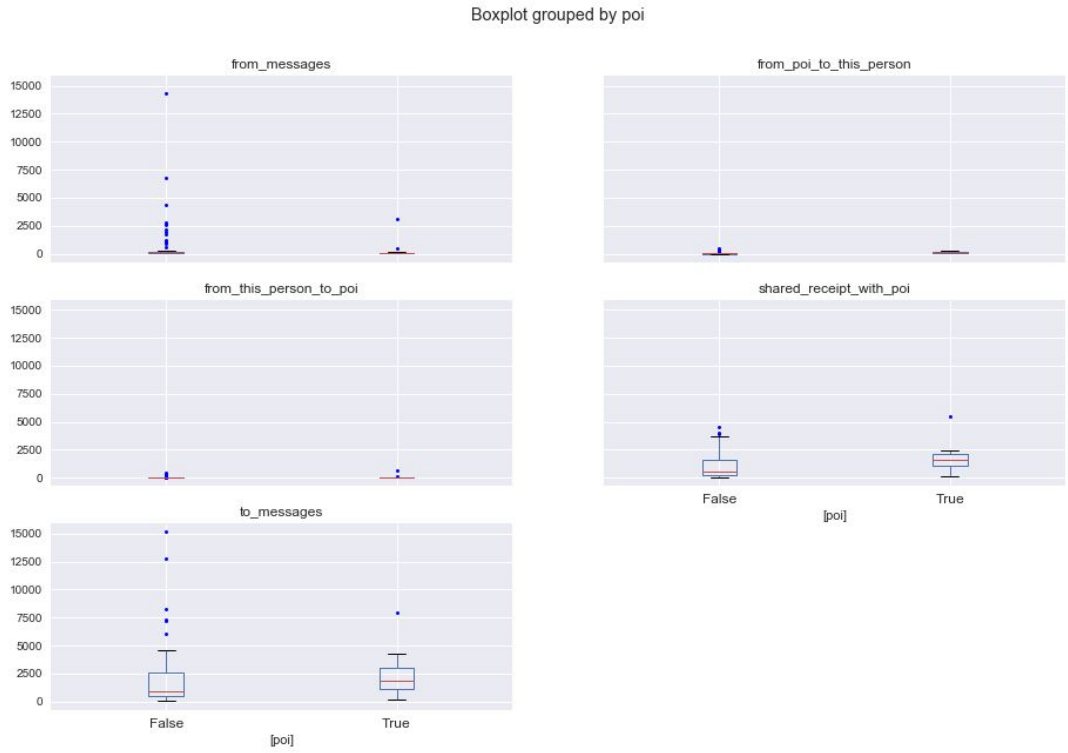

Figure 2: Boxplots email features

When trying to come up with new features I summed up the payments columns and found that the calculated total payments were different for two employees (Belfer and Bhatnagar) than the feature 'total_payments' was displaying. I dropped the rows for these employees and added the correct numbers from the pdf to the dataset.

2. Features

New feature creation

I created two new features, namely:

- Bonus to salary ratio (bonus_salary_ratio)
- Bonus to total payments ratio (bonus_tp_ratio)

I created these two new features by dividing the bonus value for each employee by their salary or total payments value. Resulting in a ratio, ranging from 0 to 1.

Bonuses are irregular types of compensation and are depending on a company's performance. It also seemed to a more common compensation type for POIs than for non-POIs based on the missing values (11% vs 48%). If a bigger part of your compensation is based on a bonus it might that you are more likely to push business in order to receive a higher bonus.

Feature selection
I looked at the results with different sets of features. I looked at all original features (n=20) and at the original ones plus the two newly created features (n=22). When looking at the histograms it seemed that 10 features had significant different distributions for POIs (bonus, exercised_stock_options, long_term_incentive, restricted_stock, salary , to_messages, shared_receipt_with_poi, total_payments, total_stock_value), so I decided to look also at these features in isolation (n=10) and also at these 10 plus the two created features (n=12).

Features selected in best model
Next to selecting some features by hand (see above) I also used SelectKBest to select the best features. I tuned *k* in SelectKBest with GridSearchCV. The range of values was 2 to 10.

For the best model (Decision Tree with 12 initial features), SelectKbest selected the following 9 features with the highest scores:

Feature Ranking: (importance - score)
salary (0.388 - 18.29)
exercised_stock_options (0.307 - 22.349)
bonus (0.305 - 20.792)
bonus_salary_ratio (0.0 - 10.784)
bonus_tp_ratio (0.0 - 20.716)
restricted_stock (0.0 - 8.825)
long_term_incentive (0.0 - 9.922)
total_stock_value (0.0 - 22.511)
total_payments (0.0 - 9.284)

There are three features that have importance scores. If these features were dropped the model's fit would decrease. Salary is the most important feature, while the exercised stock options and bonus features are of similar importance.

Scaling
I scaled the features when using the k-nearest neighbours (KNN)model. Scaling is necessary with this model, as it looks at the distance between data points. When features have very different ranges the ones with a larger range will influence the model disproportionately. By scaling the features, we are able to compare the features in the Enron dataset which have very different ranges. I used StandardScaler and added it to the pipeline when testing the KNN model.

3. Classifiers
I tried out three different classifiers, namely:
1. GaussianNB (GNB)
2. Decision Tree (DT)
3. KNearest Neighbor (KNN)

All three classifiers were used together with SelectKBest in a pipeline (I used this article + many forum posts to get this working). I tuned the *k* with a range from 2 to 10.

I used the testing.py scores to determine performance. The decision tree classifier gave the best results. For all three classifiers the best f1 scores were found when either the selected features (n=10) or selected + created features (n=12) were used. For DT and KNN a grid search was performed.

	n	k	p	r	f1
GNB	12	5	**0.39084**	0.33150	0.34669
DT	12	9	0.35517	**0.55850**	**0.43421**
KNN	12	5	0.38605	0.32100	0.35053

GaussianNB

The GaussianNB classifier does not have any options for parameter tuning. I however looked at it together with KBest in a pipeline.

	n	k	p	r	f1
All features	20	6	0.37402	0.30950	0.33871
All features + created features	22	8	0.36974	0.30300	0.33306
Selected features	10	5	0.38582	0.32100	**0.35044**
Selected + created features	12	5	**0.39084**	**0.33150**	0.34669

The results show that number of initial selected features changes the results. The best results were achieved when the selected features with and without the two created features were used. The difference between both were not large.

Decision Tree

	n	k	p	r	f1
All features	20	9	0.20708	0.57600	0.30464
All features + created features	22	9	0.27603	0.55400	0.36847
Selected features	10	9	0.19755	**0.63000**	0.30079
Selected + created features	12	9	**0.35517**	0.55850	**0.43421**

Only when using n=12, the precision score was higher than the given threshold. The recall score was high for all 4 selections. Due to the higher precision score the f1 score for the n=12 selection was significantly higher.

K-nearest neighbours

	n	k	p	r	f1
All features	20	3	0.60139	0.25950	0.36256
All features + created features	22	4	0.34752	**0.32250**	0.33454
Selected features	10	3	**0.61933**	0.25950	**0.36575**
Selected + created features	12	5	0.38605	0.32100	0.35053

The models without the newly created features (n=20 and n=10) performed better than the ones were these features were included.

4. Parameter tuning

Parameter tuning is needed to fine tune your model, to select the best values of the parameters to get the best predicting model. I used GridSearchCV to find the best values of the parameters. I heavily relied on this article. GridSearchCV tries every possible combination of values for each parameter and returns the best scoring model (based on f1 score in this case) and the best performing values.

For the decision tree model, I tuned the parameters in the table below (additional to *k* in SelectKBest). The combination of values that resulted in the best performing model can be found in the last column.

	parameter	values	default value	chosen value
SelectKBest	k	range(2,10)	-	9
Decision Tree	splitter	best, random	best	random
	criterion	entropy, gini	gini	entropy
	min_samples_split	4, 6	2	4
	class_weight	balanced, None	None	balanced
	min_samples_leaf	12, 15	1	15

As can be seen most values took another value than the default value thereby increasing the performance of the model and thereby showing the importance of parameter tuning. I tried many different combination of input values for min_samples_split and min_samples_leaf. Taken into account the default values, I found that the model performed worse when the default values were included. I therefore decided to take the default values for these parameters out.

5. Validation

In order to validate your model we have to test it. The way we do this is by splitting up our dataset in a training and a test set. The trade off is by training the model on as many data points as possible to maximize the learning result. At the same time we want to keep a big enough testing set so we can validate our model. A classic mistake is to train and test your model on the whole dataset without splitting them up. When this is not done, it is hard to tell how good your model will predict when used on other data.

I used Stratified Shuffle Split to split the data in randomized train and test sets. With 100 splits and a test size of 30%. The splits are created in such a way that it takes into account the ratio between the classes (POIs vs. non-POIs), so it is representative. It calculates the average results of all splits.

6. Evaluation

The precision score shows what percentage of the employees who were labelled as POI by the model are really POIs. In the case of the decision tree classifier the precision score is 0.35517. A high precision score shows that we are not incorrectly labelling employees as POIs. As the score is not particularly high, it implicates that the model labels employees as POIs while they are actually not POIs.

The recall score shows how many actual POIs were also labelled by the model as POIs. In the case of the decision tree classifier the recall score is 0.55850. The highest of the three classifiers when testing with 12 features. This means that 55,85% of the POIs were labelled correctly as POIs by the chosen model.

I believe in the case of detecting fraud the recall score is more important as we do not want to miss out on any POIs. The F1-score is also the highest for this model (0.43421), the f1-score is the weighted average of the precision and recall score.

Sources

- Articles linked in text
- Scikit learn website
- Many posts on the Udacity forum
- Many posts on Stackoverflow