Data Analyst Nanodegree

Project 5

Identifying fraud from Enron Email

Michal Mašika

“I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate.”

1. Introduction

In 2000, Enron belonged to one of the world's major electricity, natural gas, communications and pulp and paper companies, with claimed revenues about $ 111 billion. In that time it was named "America's Most Innovative Large Company" for six consecutive years by Fortune.[1] Less than two years later, the company fell into the bankruptcy due to one of the largest corporate fraud in the US history.

How it is possible, that so big company disappeared almost overnight? How did it manage to fool the regulators and the whole investment community? What are the measures to avoid such situation again? These and many others questions were examined during the resulting federal investigation. Moreover, during the investigation, a significant amount of data entered the public record, including tens of thousands of emails and detailed financial data for top executives.[2]

In this paper, I combine this data with a list of persons of interest in the fraud case. The persons of interest (POI's) are people, who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity. The goal of this paper is to examine, whether there are any patterns in email and financial data, to identify people who were involved in the fraud case. In order to do this I use different supervised learning methods. The final algorithm is decision tree classifier with entropy as splitting criterion. While the precision of this classification is 0.4048, the value of recall is 0.5016. This means that, if my algorithm detects someone as person of interest, this person is with likelihood of 40.48% person of interest. On the other hand POI's will be with likelihood of 50.16% truly detected.

Although the dataset used for this analysis has specific characteristics, I believe that this work has broader implication as the skills demonstrated in this paper can be used in any other real world problems (e.g. spam detection, face detection …).

The paper is structured as follows. In the next section, I present the data and identify outliers. Section 3 deals then with the optimal feature selection. In section 4, I pick and tune the algorithm which helps us to answer the question. Section 5 describes validation strategy as well as different evaluation metrics. The final section concludes.

---

[1] https://en.wikipedia.org/wiki/Enron

[2] The Enron Email Dataset constitutes one of the biggest Email corpuses in the public record. You can find more about this dataset in https://www.cs.cmu.edu/~./enron/

2. Understanding Dataset

The main goal of this paper is to build identifier of persons of interest (hereafter POI's). POI's are people who were involved in the massive corporate fraud by Enron. These are people who were i) indicted, ii) settled without admitting guilt or iii) testified in exchange for immunity. In other words, I look for patterns in the data which helps me to identify POI's. Machine learning seems to be here very useful as there are many ML algorithms dealing with pattern recognition in order to use for predictions. The main ML tasks are typically classified into three broad categories: i) supervised learning (e.g. support vector machines), ii) unsupervised learning (e.g. clustering) and iii) reinforcement learning.[3] As my data contains information about, who is and who is not POI, I will use here different supervised learning algorithms. In the following, I describe my data.

Data – General Overview

My data is constructed as dictionary with people names as a key and dictionary with different features as a value. It contains information about 146 people (mainly the Enron executives), for each of which we have 21 different features. The features come from three different data sources:

i) Financial Data

There are totally 14 financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees', 'bonus'] (all units are in US dollars)

ii) Email Data

This data comes from the Enron Email corpus. The main features (in total six) generated from the email data are as follows:

['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'][4]

iii) Manual List with the persons of interest

The feature in the dataset is 'poi' – boolean indictaing whether the person is POI or not. There are 18 POIs (i.e. 12.33%) in my data. This might constitute a challenge as the information for the training data might be too little. Actually, there is 35 POI's in the list. This means than 17 POI's are missing in my data. Unfortunately, for many of them I

[3] See https://en.wikipedia.org/wiki/Machine_learning
[4] All features except 'email_address' are numeric. This is a string.

don't have neither financial nor email information and therefore including these into dataset is useless.[5]

Data – Missing Values

By examining data in more detail I can see that there is a lot of missing values in the data. For example, there is no bonus information by 43.84 % people in my data. Moreover, I have email information only for 86 people. This means, that for 60 people (i.e. 41.1 %) the email data is missing. The following features are features with the highest amount of missing data: i) loan advances (97.26% missing), ii) director fees (88.36% missing) and iii) restricted stock deferred (87.67% missing). Therefore, these features might not be that useful for the classification.

Moreover, if I look at the missing values by POI label, I can see huge differences. For example, while bonus data is missing by 11.11% POI's, by non-POI's this number is 48.44%. Similar difference can be observed by salary numbers (POI: 5.56% vs. no-POI: 39.06%) or email data (POI: 22.22% vs. no-POI: 43.75%). In general, the share of missing values is larger by no-POI's. This might constitute a problem for the classification as the lack of information can be picked up by an algorithm as a clue that they're POI's![6]

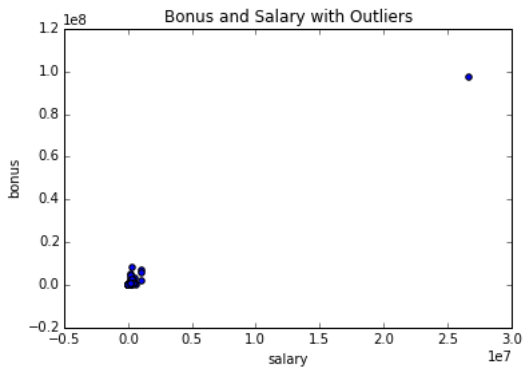
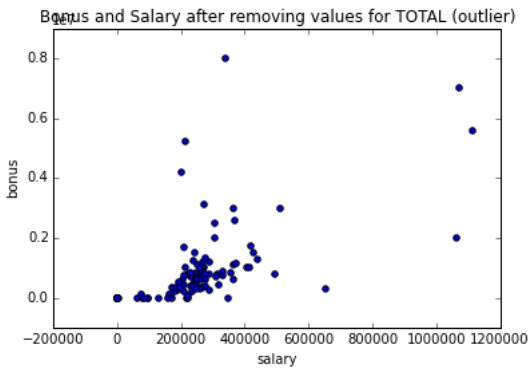
Finally, I examine the missing values by each observation. By doing this, you can see that more than 85% information is missing by following observations: Eugene Lockhart (95.24% missing), The Travel Agency in the park, David Whaley, Bruce Wrobel and Wendy Gramm with 85.71% missing information for each of this observations. The Travel Agency doesn't apparently constitute any person. Therefore, I decided to remove it from the dataset.

Data - Outliers

The classification might be negatively influenced by outliers in the data. Therefore, it is important to check whether there are any outliers in the data. In order to do this, I visualize bonus and salary. The figure below suggests that there is one strong outlier with bonus around $ 100 Million and salary over $ 25 Million. It is the observation for "TOTAL" (i.e. aggregated values over all observations). Thus, I decided to exclude this observation as well. The question remains whether there are any other outliers after removing "TOTAL". This is examined in the second figure.

[5] John Forney constitutes the only exception as I have his email data. I decided to not include him into dataset as it will generate additional missing values on the financial data. The problems connected with missing values are described in the main text.

[6] There are different techniques dealing with this problem (e.g. imputation). However, these go beyond the scope of this paper.

Figure 1 (with 'Total')	Figure 2 (without 'TOTAL')
	

The second figure suggests that there are 4 more observations which might be considered as outliers. These data points belong to John Lavorato, Kenneth Lay, Jeffrey Skilling and Mark Frevert. Two of these persons are actually persons of interest. It seems that these outliers are not there due to "wrong data". Therefore, I leave these observations in the dataset. Moreover, this figure suggests that the "outliers" might be useful instrument to identify POI's.

Thus, in the second step I examine outliers by all features in a programmatic way. First I look at the distribution of each feature (in particular 25th and 75 percentile). Then, I determine the lower bound (25th percentile – 1.5*interquartile range) and upper bound (75th percentile + 1.5*interquartile range) which I use in the next step for outliers detection.[7] In the last step, I count how many times each observations contains an outlier. Following people contains the highest amount of outliers: POI's: Timothy Belden, Kenneth Lay and Jeffrey Skilling, no-POI's: Mark Frevert, John Lavorato, Lawrence Whalley. This confirms that the outliers might play an important role for identifying people, who were involved in corporate fraud.

In summary, I excluded two observations (TOTAL, THE TRAVEL AGENCY IN THE PARK), so that I have 144 people in my dataset in total. 18 of these observations (i.e. 12.5%) are considered to be POI's. In the next section we examine the feature selection.

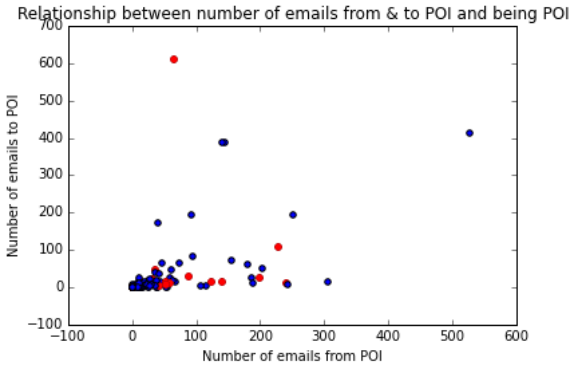
[7] Everything what is below lower bound or above upper bound is considered as (mild) outlier. Outlier definition was taken from http://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm .

3. Feature Selection

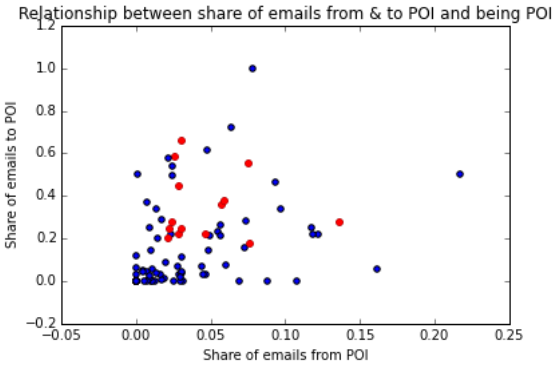
In the previous section you saw that there are 19 features in total + email address and poi variable, identifying whether the person is person of interest or not. In this section we examine which of these features should be used in the final algorithm. In other words, I examine, which of these features have the largest impact on the final identification. Moreover, I will introduce two new features to the dataset.

Creating new feature

In the first step, I examine already created features: It might be that the POI's send emails to other POI's at a rate much higher than the normal population. The following figure shows exactly the relationship. It seems that there is no clear pattern between number of emails to/from POI and being POI (red points).



Although the absolute numbers of emails to/from POI doesn't seem to have huge impact on being POI, it might be that the shares of emails to/from POI will be good indicator. It is reasonable to assume that POI's send emails to other POI's much ofter. To this end, I create two new variables: i) fraction_from_poi is share of emails from POI on all emails the person received and ii) fraction_to_poi is the sare of emails to POI on all emails the person sent.



The figure above indicates that people sending relatively small numbers of emails to POI's (the share of emails to POI is small) are very likely not POI's.

However, using this new variable in the algorithm might be problematic. By creating those variables, I am also using the test set labels. This might negatively influence the performance of the algorithm as I am mixing the train and test data. Moreover, such model cannot be used for detecting fraud in other companies, without knowing the POI's. On the other hand, I might argue, that the identification can be used for the cases, where I already now some POI's and I am interested in prediction, whether a new observation (new person with both financial and email data) is POI or not. Let's assume that this is the question here.[8]

Feature Selection

As seen above, there are many features in this dataset. But, what features should I select or in other words what features shouldn't be used at all? Answering this question is important because of the following reasons: i) Features add to the data just noise but no information, ii) too many features might cause overfitting, iii) some of the features are strongly related to each other and therefore don't bring that much additional information or iv) many features slow down training or testing process.

In order to select right features I use two approaches: decision tree classifier and the SelectKBest approach. In general, in both approaches it is important to split the data into training and testing dataset, in order to avoid bias in my accuracy.[9] The decision tree approach actually fits the model and takes these features that are the best for the fit. Here, in order to perform feature selection I am splitting the data set into training and testing data, fit the model and save the importances of each feature. This procedure is repeated 1000 times. Finally, I am taking the average of each importance and rank the feature according to their importance. This procedure is repeated three times, where in each round I exclude the "worst-performing" features. The advantage of this approach is that each time I have new training data set (cross-validation) and therefore new decision tree (might overcome some issues with overfitting – therefore 3 rounds). The results of decision tree model can be found in the table below.

On the other hand the SelectKBest approach doesn't fit any model. It basically looks how the features are correlated with the outcome variable. More specifically, the SelectKBest uses

---

[8] A good discussion about this topic can be found under https://discussions.udacity.com/t/mistake-in-the-way-email-poi-features-are-engineered-in-the-course/4841/2. I discuss this topic a little bit in the last section of this paper.

[9] Guyon (2003) Guyon, I. (2003) "An Introduction to Variable and Feature Selection", The Journal of Machine Learning Research, Vol. 3, pp. 1157-1182

ANOVA for the scoring of each feature. Here, I am using similar approach as above (with K=10). I.e. I run this approach 1000 times and store the scores. Finally, I am taking the average of the scores for each feature. Results of both procedures can be found in the following table:

Approach	**Decision Tree**			**SelectKBest**
Features	**Importances (1. round)**	**Importances (2. round)**	**Importances (3. round)**	**Scores**
Fraction to poi	0.1423376345595663	0.14671317191062891	0.15584851382443335	12.391880807187118
Exercised Stock Options	0.11827449756571268	0.12478631970997719	0.12871589791127519	17.858942364993133
Expenses	0.10796868751822149	0.11253956395711491	0.12717608208312786	4.6905259687094922
Bonus	0.10631601933585366	0.10743907700541296	0.1255204138977839	15.62292503120505
Other	0.094749880679550577	0.090363614599730691	0.10626078110437322	3.5752113814622719
Shared receipt with poi	0.071379710607573241	0.075093709615261031	0.094658435775630018	6.4371033154690034
Total Stock Value	0.063838321178229782	0.06058200069549808	0.072816758282141111	17.396693243224092
Deferred income	0.053369773475269847	0.056693763843709964	0.059950756563358867	9.0116849852164851
Total payments	0.037191239664672082	0.038557670860322538	0.048590290315070457	6.3595307130316554
Restricted stock	0.035024650914665209	0.036635637407744455	0.044518043395777321	7.202841180154576
Long term incentives	0.033347096683081244	Excluded after 2. round		8.0210515619772966
Salary	0.026444273044019837	0.030462647454747832	Excluded after 3. round	13.187140586255813
Other Features	Excluded after 1. or 2. round			

By examining the table above, we can see that following 7 features belong to the best 10 features by both approaches: fraction to poi, exercised stock options, bonus, shared receipt with poi, total stock value, deferred income and total payments.[10] I run different classification with different subsets of these features in order to decide what features I should take for the final analysis. The features chosen for final classification are described in the next section.

Feature Scaling

The selected features differ in their magnitudes. In particular, the email features have by far smaller magnitude than the finance features. Thus, the feature scaling might seem to play an important role. In this case, however, I have decided not to conduct feature scaling as in my final analysis I am using decision tree and random forest which are unaffected by feature scaling.

[10] Note that by running the code slightly different results might be observed as I haven't set random state by decision tree.

4. Tuning, validation and evaluation of algorithm

The section above suggests seven features which should be taken into account by building a classifier. In this section I build, tune, validate and evaluate different algorithms. Although all of the features might be somehow relevant for the algorithm, I decided to choose only some of them. This was done by iterative process, where I tried out several subsets of these features. The main reason for this is to avoid overfitting. For the final algorithm I have chosen following features: fraction to poi, total stock value, shared receipt with poi and bonus.

Moreover, in order to pick the "ideal" algorithm I make use of K-Fold cross validation technique with K=3. This improves the evaluation procedure as the results are not affected by only one training/test set. I started with following default algorithms: Naïve Bayes, Decision Tree, K Nearest Neighbors, Random Forest and Adaboost. Results of these algorithms can be found in the following table:

Score\Algorithm	**Fold (size of test set, #pois)**	**Naïve Bayes (Gaus.)**	**Decision Tree**	**KNN**	**Random Forest**	**Adaboost**
ACCURACY	1 (44,7)	0.841	0.75	0.864	0.795	0.818
	2 (44,6)	0.795	0.818	0.886	0.818	0.818
	3 (43,5)	0.860	0.837	0.884	0.884	0.837
	Average	0.832	0.802	0.878	0.832	0.825
PRECISION	1 (44,7)	0	0.167	1	0	0
	2 (44,6)	0.333	0.333	0.667	0.25	0.25
	3 (43,5)	0.429	0.333	0	0.5	0.25
	Average	0.254	0.278	0.556	0.25	0.167
RECALL	1 (44,7)	0	0.143	0.143	0	0
	2 (44,6)	0.5	0.333	0.333	0.167	0.167
	3 (43,5)	0.6	0.4	0	0.4	0.2
	Average	0.367	0.292	0.159	0.189	0.122
F1 SCORE	1 (44,7)	0	0.154	0.25	0	0
	2 (44,6)	0.4	0.333	0.444	0.2	0.2
	3 (43,5)	0.5	0.364	0	0.444	0.222
	Average	0.3	0.284	0.231	0.215	0.141

The table above shows the performance of each classification in each fold and the average performance. The examined evaluation metrics are accuracy, precision, recall and F1 score.

Accuracy measures the probability of predicting the correct class. This measure is not ideal here as the classes are skewed. In average the share of POI's is only 13.7% in the test data. In other words we will end up with pretty high accuracy when we just identify all the people as not persons of interest (exactly 86.3%). Therefore, we have to consider other evaluation measures.

Precision is probability of person being POI provided our algorithm is predicting person as POI. This measure should be maximized if we want to be sure that the algorithm correctly predicts POI's. In other words it should be used when we want to minimize the number of non-POI's being identified by classifier as POI's (false positives). The disadvantage of focusing only on this metric might lead to many POI's identified as non-POI's (false negatives).

Recall on the other hand is the probability of predicting someone as POI provided that person being POI. This measure should be maximized if we want to detect all the POI's. It other words it should be used if we want to minimize false negatives. The disadvantage of it is that it might lead to higher number of non-POI's being detect as POI's.

F1 Score takes the best of both previous metrics. It can be interpreted as the weighted average of precision and recall.[11] High F1 score means, if my identifier finds a POI then the person is almost certainly a POI, and if the identifier does not flag someone, then they are almost certainly not a POI. The following example (I use the third fold of random forest) illustrates how all these metrics are computed. There are 43 people in the test set (5 of them are POI's). As the recall is 0.4, the number of true positives is 2. The number of false negatives is therefore 3. As the precision is 0.5, the number of false positives is 2. The number of true negative is therefore 36. Below you can find the confusion matrix of this example.

		Actual	
		POI	NON-POI
Prediction	POI	2	2
	NON-POI	3	36

[11] F1=2*(precision*recall)/(precision+recall) (see http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)

In order to evaluate my algorithm, I take into account precision, recall and f1 score. I particularly focus on the f1 score. This metric is also used for optimizing my classifications. I have decided to tune both decision tree and random forest. By this end I use GridCV which systematically goes through multiple combinations of parameters which influence the performance of chosen classification.

Decision Tree

In this case, I tune following two parameters: criterion and min_samples_split. Min_samples_split influence the complexity of the tree and therefore it might have an impact on overfitting. In general holds: the higher this number, the less complex the tree and therefore the lower likelihood of overfitting. Here, I try following numbers: [2,6,10,20,50,100,400]. Criterion describes splitting rules, how the tree is build. Here, I vary between gini and entropy. By doing so, I can see that the "optimal" tree with respect to F1 score is with entropy criterion and min_samples_split=2.

Random forest

In addition to the parameters above I also tune here the number of estimators, which effects the variance of the estimator. Here, I try following numbers: [2,5,10,20,50,80,100,200,300]. The "optimal" random forest has following parameters: criterion='gini', min_samples_split=20 and n_estimators=50. The following table illustrates the results if I use the optimal algorithms. I present here only the average metrics (i.e. not for each fold):

Evaluation\Algorithm	**Decision Tree**	**Random Forest**
Accuracy	0.832	0.878
Precision	0.405	0.5
Recall	0.502	0.244
F1 Score	0.447	0.324

The table demonstrates that I was able to improve performance of both algorithms. Since the decision tree performs better, I decided to use this classification as the final algorithm. Moreover, the recall is better than my precision, which means that I am pretty good in identifying POIs whenever they show up. The cost of this is that I relatively often get false positives (i.e. non-POI's get flagged). More concrete, given the person is a POI, this person

will be flagged with probability of 50.2%. On the other hand, given I identify someone as a POI, this person is with likelihood of 40.5%.[12]

5. Conclusion

In this paper I have used the email Enron dataset combined with the financial data to investigate, whether there are data patterns which help me to identify people involved in the Enron corporate fraud. To this end I examined the data first with the following feature selection by means of both decision tree and select KBest technique. I end up with following features: share of emails sent to poi (fraction_to_poi), number of emails shared with POI (shared_receipt_with_poi), bonus and total stock value. Afterwards I used different supervised learning approaches. Finally, I tuned two of them and selected decision tree as the "most optimal" one. I ended up with the F1 score of 0.447.

There are different possibilities for improvement which can be tackled in the future work. First, the data overview section showed that there are a lot of missing values in the data. These missing values might negatively influence performance of my classification as they distribution of missing values is different for POI's and non-POI's. One possibility how to deal with this would be imputation. For example, one could use clustering methods (e.g. KMeans) to group the people in the first place. Afterwards, one can use feature means of these cluster for replacing the missing values.

Second, I used only email data on aggregated level (e.g. fraction_to_poi). One could dig a little bit deeper in the emails and use some text learning techniques (e.g. nlp) to see whether there are differences between POI's and non-POI's in the text structure.

Third, in the feature selection section I discussed that the usage of POI information in the features (e.g. fraction_to_poi) might be problematic because of two reasons: i) in order to create these features I use both test and train information. To deal with this issue, one could only use only POI's in the training data to create these features. ii) usage of POI-features makes the algorithm useless for the cases, where we want to identify POI's in other company without any information about who is a person of interest. One example, how this issue might be tackled, is shown in the first part of appendix.

[12] These numbers are a little bit different after running the tester.py: Accuracy: 0.84014, Precision: 0.43425, Recall: 0.393, F1: 0.4126

Finally, besides feature selection techniques, one could also use PCA to reduce the dimensionality. One example, how this might look like is shown in appendix.

6. References

Background – Enron Case:
https://en.wikipedia.org/wiki/Enron

https://www.cs.cmu.edu/~./enron/

Machine Learning – Background and Documentation of algorithms:
https://en.wikipedia.org/wiki/Machine_learning

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

http://scikit-learn.org/stable/modules/pipeline.html

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier

Outliers:
http://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

Features Creation & Selection:
https://discussions.udacity.com/t/mistake-in-the-way-email-poi-features-are-engineered-in-the-course/4841/2

Guyon, I. (2003) "An Introduction to Variable and Feature Selection", The Journal of Machine Learning Research, Vol. 3, pp. 1157-1182

Evaluation of Algorithms:
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Validation:
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.KFold.html#sklearn.cross_validation.KFold

Appendix

Usage of features without POI information

As discussed in the main text, the usage of the features containing the POI information might be problematic. Therefore, in this section, I train the data to identify POI's without this information. After trying out several algorithms I ended up with Gaussian Naïve Bayes. The features used in this algorithm are the following ones: total stock value, bonus and exercised stock options.

The results can be seen in the table below. Compared to the algorithm used in the main text I am now better in terms of precision, but much worse in recall. This means, that if I identify someone as POI, it is more likely now that he/she is really a POI. However, the number of false negatives is now larger (i.e. POI's that are flagged as non-POI's.[13]

Algorithm\Evaluation	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Naïve Bayes	0.853	0.467	0.359	0.377

PCA Approach

Another possibility how to reduce the number of features used for the classification is the PCA. The PCA might be here very useful as basically there are two large groups of features: financial and email data. After using of GridSearch CV and different classifiers I ended up with the PCA with number of components equal to ten and Naïve Bayes as classifier. The results can be taken from the table below. The results are worse than by algorithm used in the main text.[14]

Algorithm\Evaluation	**Accuracy**	**Precision**	**Recall**	**F1 Score**
PCA (n=10), Naïve Bayes	0.783	0.23	0.357	0.274

[13] After running tester.py – the results are a little bit different: Accuracy: 0.843, Precision: 0.486, Recall: 0.351, F1: 0.40755. Still a little bit worse than the results in the main text. But the difference is not that large

[14] After running tester.py - the results are a little bit different: Accuracy: 0.84947, Precision: 0.41248, Recall: 0.304, F1: 0.35003. This is worse than the algorithm used in this paper.