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 random forest classifier with entropy as splitting criterion and 50 number of estimates. While the precision of this classification is 0.5653, the value of recall is 0.384. This means that, if my algorithm detects someone as person of interest, this person is with likelihood of 56.53% person of interest. On the other hand POI's will be with likelihood of 38.4% 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]

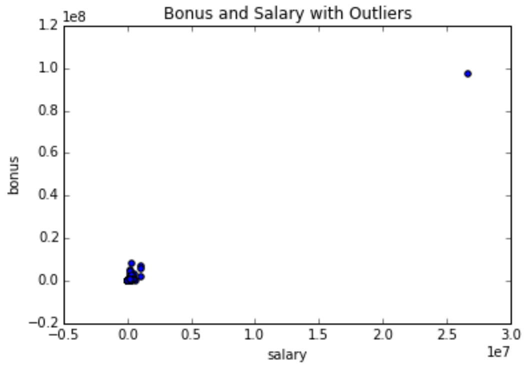
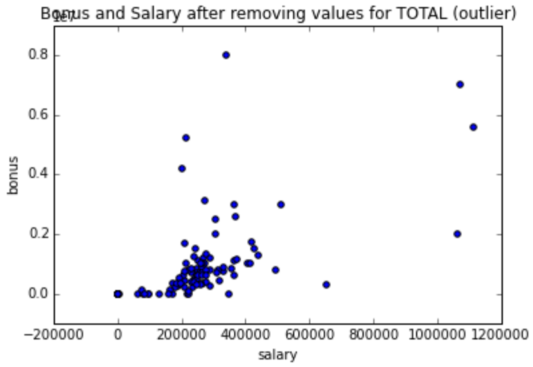
Finally, I examine the missing values by each observation. By doing this, you can see that more than 85% information is missing by following observations: Eugene Lockhart (95.24% missing), The Travel Agency in the park, David Whaley, Bruce Wrobel and Wendy Gramm with 85.71% missing information for each 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.

3. Feature Selection

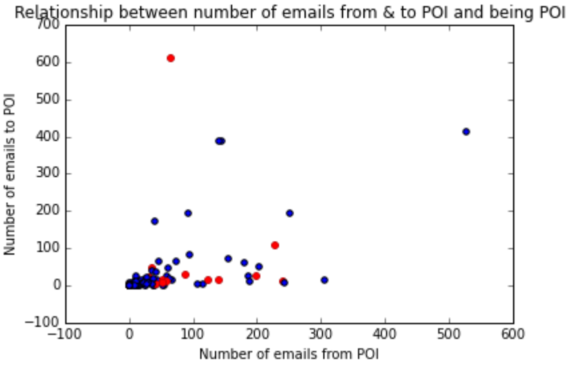
In the previous section you saw that there are 19 features in total + email address and poi variable, identifying whether the person is person of interest or not. In this section we

[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 .

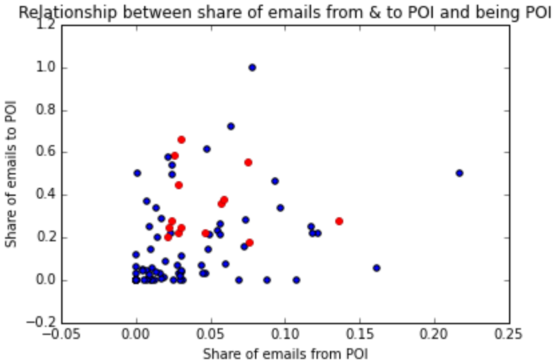
examine which of these features should be used in the final algorithm. In other words, I examine, which of these features have the largest impact on the final identification. Moreover, I will introduce two new features to the dataset.

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 this 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 often. 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. Visualization can serve as a good indicator whether a feature might play a role for the classification. In addition, different feature selection techniques (see next subsection) can help me to see, whether the feature is going to be important for the classifier. That subsection shows that while the share of emails sent from POI's doesn't seem to play a role, the fraction of emails sent to POI's might play an important role. Finally, to quantify the impact of this new feature (fraction_to_poi) on the performance of the algorithm (precision, recall and f1 score), I run the final algorithm with and without this feature in the first part of the appendix. The results show that all important evaluation metrics decreases. F1 score decreases from 0.4573 to 0.3597. Moreover the probability of predicting someone as POI provided that person being POI (recall) decreases by 10 percentage points. Finally, the probability of person being POI provided our algorithm is predicting person as POI (precision) decreases by seven percentage points (from 56.53% to 49.47%).[8]

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.[9]

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.[10] 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

[8] Note, however, that there are algorithms where the fraction to poi improves the performance of the algorithm (see the feature selection subsection).

[9] 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. Note, that in the final algorithm I am not using any feature connected with persons of interest.

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

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 ANOVA for the scoring of each feature. Here, I am using similar approach as. 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 (k=10)**
Features (rank SelectKBest)	**Importances (1. round)**	**Importances (2. round)**	**Importances (3. round)**	**Scores**
Fraction to poi (5)	0.1423376345595663	0.14671317191062891	0.15584851382443335	12.391880807187118
Exercised Stock Options (1)	0.11827449756571268	0.12478631970997719	0.12871589791127519	17.858942364993133
Expenses (11)	0.10796868751822149	0.11253956395711491	0.12717608208312786	4.6905259687094922
Bonus (3)	0.10631601933585366	0.10743907700541296	0.1255204138977839	15.62292503120505
Other (12)	0.094749880679550577	0.090363614599730691	0.10626078110437322	3.5752113814622719
Shared receipt with poi (9)	0.071379710607573241	0.075093709615261031	0.094658435775630018	6.4371033154690034
Total Stock Value (2)	0.063838321178229782	0.06058200069549808	0.072816758282141111	17.396693243224092
Deferred income (6)	0.053369773475269847	0.056693763843709964	0.059950756563358867	9.0116849852164851
Total payments (10)	0.037191239664672082	0.038557670860322538	0.048590290315070457	6.3595307130316554
Restricted stock (8)	0.035024650914665209	0.036635637407744455	0.044518043395777321	7.202841180154576
Long term incentives (7)	0.033347096683081244	Excluded after 2. round		8.0210515619772966
Salary (4)	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 8 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, total payments and restricted stock.[11] The analysis above gives me a good hint what features might be useful for the final algorithm.

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

However, it is still not that clear how many features should I use and what are the right features.

In order to investigate these questions, first I look at the relationship between number of features and different evaluation metrics (precision, recall and f1 score). More precisely, I choose K best features (either according to decision tree or SKBest), run different algorithms and look at the performance. I used default versions of Gaussian Naïve Bayes, Decision Tree, KNN, Random Forest and Adaboost (no tuning).[12] The results can be seen in the figures on the next page.[13] The figures indicate that the score first increases with the number of features and then decreases.[14] The peak varies by both classification and feature selection method. The peak lies by 3,4 or 6 features when I use decision tree to select the most important features. In case of SelectKbest the peak lies either by 1 or 3 features. This means that the following features might play a central role for the final classification: fraction to poi, exercised stock options, expenses, bonus, other, shared receipt with poi and total stock value.

However, it is still not clear what features exactly should be used. The investigation above partly examines this question by taking into account always subset of best features. These subsets are however determined by the ranking of importances of these features. To tackle this problem in more general way, I examine the impact of all possible feature combinations (out of seven features chosen above) on the evaluation score.[15] The optimal set of features is then that one, which maximizes the F1 score. The results can be found in the table below:

Default Algorithm	**Gaussian NB**	**Decision Tree**	**KNN**	**Random Forest**	**Adaboost**
Features_combi	(exercised stock options)	(bonus, shared receipt with poi, fraction to poi)	(exercised stock options, bonus, fraction to poi)	(exercised stock options, bonus)	(exercised stock options, total stock value, expenses, other, fraction to poi)
Precision	0.5974	0.4236	0.6906	0.5432	0.4691
Recall	0.3618	0.402	0.35	0.3543	0.3932
F1_score	0.4506	0.4125	0.4646	0.4289	0.4278

The table shows that depending on the algorithm. Different set of features maximizes F1 score. I have decided to use exercised stock options and bonus in my final classification as I believe that the performance can be improved through an appropriate tuning.[16]

[12] Default Algorithm is defined as the algorithm used as default by scikit-learn. The algorithm tuning is conducted in the next section.

[13] Note, that to compute the evaluation metrics I use crossvalidation described in the next section. The definition of precision, recall and f1 score can be found as well in the next section.

[14] In particular, I am taking f1 score as the main score. Next section offers an explanation why f1 score might be the best choice.

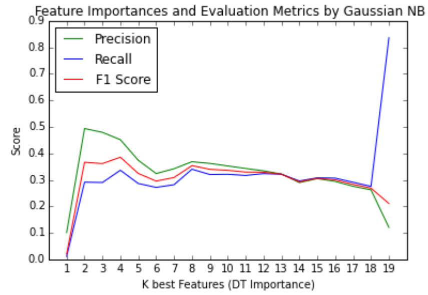
[15] Assume that the list of chosen features is [a,b,c]. The list of all combinations is then [[a],[b],[c],[a,b],[a,c],[b,c],[a,b,c]]

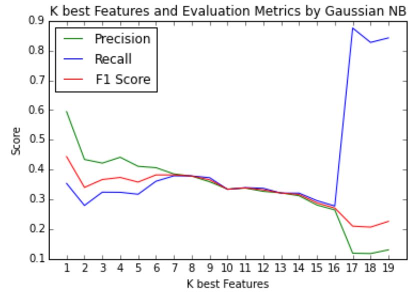
[16] This choice might seem to be not optimal as the performance of Gaussian NB as well as KNN is higher. Note, however, that by Naïve Bayes the tuning is somehow restricted. Moreover, I have also tried to scale the features in order to use them in KNN approach. The performance of the KNN approach has decreased after feature scaling.

Feature Importances (DT)

Select KBest

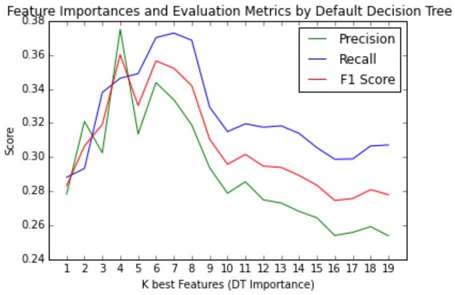
Gaussian NB

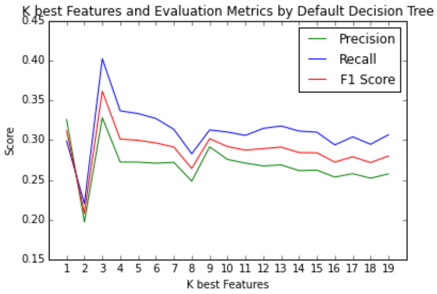




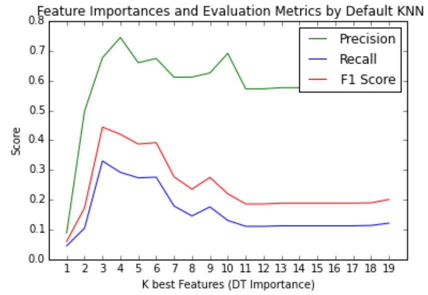
Decision

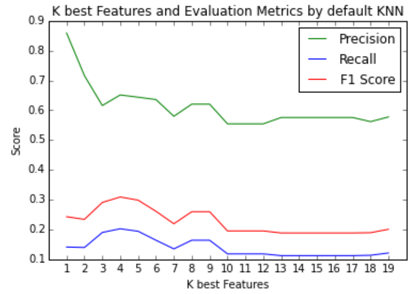
Tree





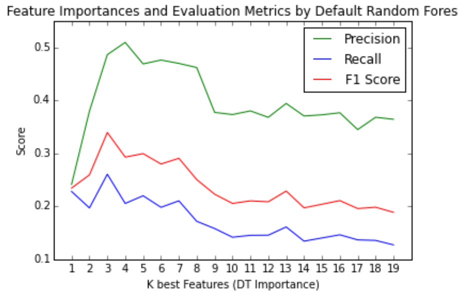
KNN

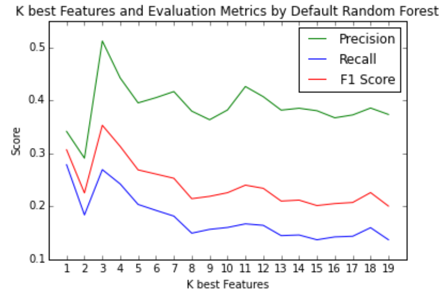




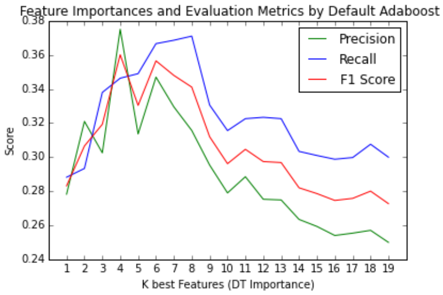
Random

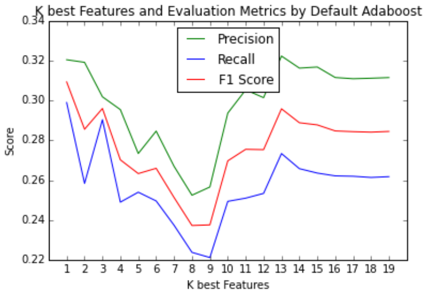
Forest





Adaboost





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.

4. Tuning, validation and evaluation of algorithm

In this section I will validate, tune and evaluate my final algorithm. As noted in the previous section I have chosen following features: exercised stock options and bonus for my final algorithm.

Validation is important technique by supervised learning as it basically allows the generalization of results (i.e. results can be used also on the data the model hasn't seen before). The basic idea behind this is to split the data into testing and training dataset and fit the model only on the training data. Without doing this, the model would have high performance, but it would fail to predict anything useful on yet-unseen data. In other words, we would overfit the model.[17] Moreover, splitting the data only once (training/testing) might lead to suboptimal results as well. One reason for this is that the training set might contain some "outliers" which negatively influence the final performance. Another reason is that by using only one training set the parameters can be tweak such that the risk of overfitting increases. I conduct some parameter tuning later in this section. A solution to this problem is cross-validation procedure.

In this project, I make use of stratified shuffle split with 1000 re-shuffling and splitting iterations. This procedure randomly split the data into testing and training 1000 times. However, it preserves the share of targeted class as in the complete data set. This is particularly important here as I have small number of observations and the classes (POI and no-POI) are highly unbalanced. In order to evaluate my classification, 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:

[17] For more discussion see http://scikit-learn.org/stable/modules/cross_validation.html.

Score\Algorithm	Naïve Bayes (Gaus.)	Decision Tree	KNN	Random Forest	Adaboost
Accuracy	0.8389	0.7923	0.8661	0.847	0.8122
Precision	0.5056	0.3687	0.7183	0.5432	0.4039
Recall	0.2997	0.3945	0.2868	0.3543	0.332
F1 Score	0.3763	0.3812	0.401	0.4289	0.3644
True Positives	1798	2367	1721	2126	1992
False Positives	1758	4053	675	1788	2940
True Negatives	29242	26947	30325	29212	28060
False Negatives	4202	3633	4279	3874	4008

The table above shows the performance of each classification after 1000 reshuffle and splitting iterations. 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. The share of POI's is only 12.5% 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 87.5%). 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.[18] 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.

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

The following example (I use the results of random forest) illustrates how all these metrics are computed. There are 37000 observations (6000 of them are POI's). As the number of true positives is 2126, the recall is 03543. The number of false negatives is therefore 3874. As the number of false positives is 1788, the precision is 0.5432. The number of true negative is therefore 29212. Below you can find the confusion matrix of this example.

		Actual	
		POI	NON-POI
Prediction	POI	2126	1788
	NON-POI	3874	29212

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. Each of machine learning algorithms is parameterized and modification of those parameters can have a positive impact of algorithm performance. Such modification is called tuning, where the configuration of algorithm parameters is tweaked. The objective of algorithm tuning is to find such combination of algorithm parameters which maximizes given score (here f1 score). As mentioned above, although this strategy might be effective, it can also lead to overfitting of my model.[19]

I have decided to tune both decision tree and random forest. By this end I use GridSearchCV which systematically goes through each combination of parameters and evaluates the algorithm. It choses then that combination which maximizes the scoring function (here f1 score).

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]. 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 gini criterion and min_samples_split=2. This corresponds with the default decision tree.

[19] More information about algorithm tuning can be found under http://machinelearningmastery.com/how-to-improve-machine-learning-results/.

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]. The “optimal” random forest has following parameters: criterion=’entropy’, min_samples_split=2 and n_estimators=50. The following table illustrates the results if I use the optimal algorithms.

Evaluation\Algorithm	**Decision Tree**	**Random Forest**
Accuracy	0.7923	0.8522
Precision	0.3687	0.5653
Recall	0.3945	0.384
F1 Score	0.3812	0.4573

The table demonstrates that I was able to improve performance of random forest. I decided to use this classification as the final algorithm. Moreover, the precision is better than my recall, which means that whenever I identify someone as POI I can be relatively sure that he is POI. The cost of this is that I relatively often get false negatives (i.e. true POI’s get not flagged). More concrete, given the person is a POI, this person will be flagged with probability of 38.4%. On the other hand, given I identify someone as a POI, this person is with likelihood of 56.53%.[20]

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 first described the. After that I conducted feature selection by means of both decision tree and select KBest technique. By doing this, I end up with following features: bonus and exercised stock options. Afterwards I used different supervised learning approaches. Finally, I tuned two of them and selected random forest as the “most optimal” one. I ended up with the F1 score of 0.4573.

[20] These numbers are a little bit different after running the tester.py: Accuracy: 0.85485, Precision: 0.53592, Recall: 0.4215, F1: 0.47187

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, 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
http://stackoverflow.com/questions/16296643/convert-tuple-to-list-and-back

http://scikit-learn.org/stable/modules/cross_validation.html
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html

Tuning:
http://machinelearningmastery.com/how-to-improve-machine-learning-results/

Miscelaneous:
http://stackoverflow.com/questions/18201690/get-unique-combinations-of-elements-from-a-python-list

http://stackoverflow.com/questions/16221368/creating-dictionary-of-dictionaries-in-python-2-6

Appendix

Effect of fraction to poi on the algorithm performance

Besides the features provided I created two additional features: fraction to poi and fraction from poi. In the feature selection section it was shown that fraction from poi doesn't seem to be important. Adding fraction to poi, however, might be useful. In this section, I examine the impact of fraction to poi on the performance of the final algorithm.

		Accuracy	Precision	Recall	F1 Score
With fraction_to_poi	**Decision Tree**	0.8078	0.3022	0.352	0.324
	Random Forest	0.8676	0.4947	0.2826	0.3597
Without fraction_to_poi	**Decision Tree**	0.7923	0.3687	0.3945	0.3812
	Random Forest	0.8522	0.5653	0.384	0.4573

As can be seen in the table, adding new make the performance of the algorithm worse. For example, the recall decreases from 0.384 to 0.2826. This means that the probability of predicting someone as POI provided that person being POI decreases by 10 percentage points. The probability of person being POI provided our algorithm is predicting person as POI (precision) decreases by seven percentage points (from 56.53% to 49.47%).[21]

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 two 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.[22]

Algorithm\Evaluation	**Accuracy**	**Precision**	**Recall**	**F1 Score**
PCA (n=2), Naïve Bayes	0.8795	0.4665	0.252	0.3272

[21] Note that by adding fraction to poi to the algorithm, the optimal random forest would change as well. In this case the min_samples_split is 6 and the number of estimators is 2. The evaluation metrics are: Accuracy: 0.8643, Precision: 0.4784, Recall: 0.3454, F1 Score: 0.4012

[22] After running tester.py - the results are a little bit different: Accuracy: 0.87407, Precision: 0.55522, Recall: 0.279, F1: 0.37138. This is worse than the algorithm used in this paper.