Project 5: Identify Fraud from Enron Emails

Xin Pang

1. Introduction and data exploration

The goal of this project: given a dataset describing certain aspects of the employees (e.g. payment, stock, and email interaction data), by applying feature engineering techniques, we can build a model to predict who the POIs are among all the employees. Machine learning methods are important in this project due to its ability to learn the necessary features to distinguish POIs from non-POIs in an automated way.

1.1 Dataset description

The dataset used in this project contains 146 employees' data and 21 columns. 18 out of 146 employees are POIs and the rest are non-POIs. From the result of previous exercises we know that the dataset contains the following:

- Label (1)
  - poi
- Payment features (9+1)
  - salary, bonus, long_term_incentive, deferred_income, deferral_payments, loan_advances, other, expenses, director_fees, total_payments
- Stock features (3+1)
  - exercised_stock_options, restricted_stock, restricted_stock_deferred, total_stock_value
- Email/communication features (6)
  - to_messages, from_messages, from_this_person_to_poi, from_poi_to_this_person, shared_receipt_with_poi, email_address

20 out of 21 columns contain missing values with only poi column not containing any 'nan' values. 0 instead of 'nan' values are filled in to deal with missing values. Based on the insider pay document, this is reasonable since in the document '-' represents that the person does not have any value attached with the specified attribute, e.g. salary / bonus / loan advances etc.. In the script, this is done by specifying remove_NaN to true when using function 'featureFormat'.

Besides, one employee's data are all 0. It does not make sense that this person does not have neither payment nor stock, we can infer this person's information is missing. Thus his data are excluded from the dataset. Afterwards 145 employees' data remain.

1.2 Dataset exploration

Below tables show the statistics of all features listed in the dataset. By applying describe function to the dataset, we got the mean, standard deviation, quartiles, min and max. It is easily seen that all features related to employee payments seem unusual when looking at the max value. For example the total payments – the 75% value is 1.98E+06, whereas the max value is 3.10E08, which is 157 times the 75% value. Also, this 75% value is smaller than the mean value. This indicates the mean value is actually influenced by one or few extremely large values. Therefore, the max of total payments can be considered as an outlier.

Similarly, we can find features such as salary, deferral payments, exercised stock options etc. also contain outliers. Here is a reference from Forbes[1] regarding the Enron payment: "*Between 1996 and 2000, the average chief executive salary and bonus increased by 24% to $1.72 million, according to a Forbes study of proxy reports. Total CEO*

[1] Pay Madness At Enron - https://www.forbes.com/2002/03/22/0322enronpay.html

*Compensation, including stock options and restricted stock grants, grew 166% to an average of $7.43 million. In the same period, corporate profits grew by 16%, and per capita income grew by 18%. Enron was at the cutting edge of this trend. The stated goal of its board of directors was to pay executives in the 75th percentile of its peer group. In fact, it paid them vastly more and on a scale completely out of whack with the company's financial results–even if its reported financial results are accepted as accurate.*" In the article, the author analyzed Enron's payment problem. The quote above mentioned that Enron paid executives much more than 75th percentile of the peer group. And the underlined features are the ones that can be confirmed by our dataset.

Table 1-1. Descriptive analysis of all features in Enron data

	salary	to_messages	deferral_payments	total_payments	exercised_stock_options
count	1.45E+02	1.45E+02	1.45E+02	1.45E+02	1.45E+02
mean	3.68E+05	1.23E+03	4.42E+05	4.38E+06	4.21E+06
std	2.21E+06	2.23E+03	2.75E+06	2.70E+07	2.62E+07
min	0.00E+00	0.00E+00	-1.03E+05	0.00E+00	0.00E+00
25%	0.00E+00	0.00E+00	0.00E+00	1.03E+05	0.00E+00
50%	2.11E+05	3.12E+02	0.00E+00	9.67E+05	6.09E+05
75%	2.71E+05	1.61E+03	1.03E+04	1.98E+06	1.73E+06
max	2.67E+07	1.51E+04	3.21E+07	3.10E+08	3.12E+08

Table 1-2. Descriptive analysis of all features in Enron data

	bonus	restricted_stock	shared_receipt_ with_poi	restricted_stock_ deferred	total_stock_value
count	1.45E+02	1.45E+02	145	1.45E+02	1.45E+02
mean	1.34E+06	1.76E+06	697.765517	2.07E+04	5.89E+06
std	8.12E+06	1.09E+07	1075.128126	1.44E+06	3.64E+07
min	0.00E+00	-2.60E+06	0	-7.58E+06	-4.41E+04
25%	0.00E+00	3.25E+04	0	0.00E+00	2.52E+05
50%	3.00E+05	3.61E+05	114	0.00E+00	9.76E+05
75%	8.00E+05	8.53E+05	900	0.00E+00	2.33E+06
max	9.73E+07	1.30E+08	5521	1.55E+07	4.35E+08

Table 1-3. Descriptive analysis of all features in Enron data

	expenses	loan_advances	from_messages	other	from_this_person_to_poi
count	1.45E+02	1.45E+02	145	1.45E+02	145
mean	7.12E+04	1.16E+06	361.075862	5.89E+05	24.455172
std	4.34E+05	9.68E+06	1445.944684	3.69E+06	79.527073
min	0.00E+00	0.00E+00	0	0.00E+00	0
25%	0.00E+00	0.00E+00	0	0.00E+00	0
50%	2.15E+04	0.00E+00	17	9.72E+02	0
75%	5.39E+04	0.00E+00	52	1.51E+05	14
max	5.24E+06	8.39E+07	14368	4.27E+07	609

Table 1-4. Descriptive analysis of all features in Enron data

	director_fees	deferred_income	long_term_incentive	from_poi_to_this_person
count	1.45E+02	1.45E+02	1.45E+02	145
mean	1.96E+04	-3.85E+05	6.69E+05	38.489655
std	1.19E+05	2.39E+06	4.06E+06	74.088359
min	0.00E+00	-2.80E+07	0.00E+00	0

25%	0.00E+00	-3.83E+04	0.00E+00	0
50%	0.00E+00	0.00E+00	0.00E+00	4
75%	0.00E+00	0.00E+00	3.75E+05	41
max	1.40E+06	0.00E+00	4.85E+07	528

To get a better idea, below shows two visualization examples of the statistics using boxplot. For complete boxplots please refer to Appendix 7.2 – boxplots of Enron data features. The boxplot considers values that is 1.5 times the mean value or more to be outliers. From the plots we can see based on this criteria, all of the features contain outliers. And in each of the plots, there is one or two outliers that are particularly much larger than other values.

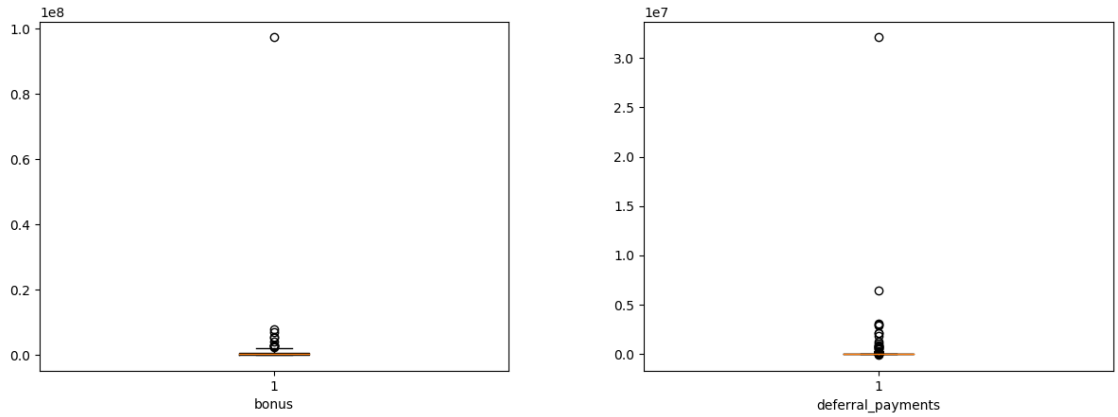


Figure 1. Example box plots of two features in Enron data

Although in the above analysis many outliners are found, we still need to be cautious about how to process them. Whether to remove or not depends on the scenario. In Enron's case, POIs tried to maximize their own benefits by transferring company assets to their own. Thus it makes sense to observe many outliers in salary, stock options, etc.

The plot below visualizes total stock value vs. total payments. Points in red are the POIs and blue are the non-POIs. On the left hand side we can see there is only one particular point far away from others at the right top corner who is a non-POI. This is possibly an outlier. The plot 2.b is a zoom-in plot of 2.a, focusing on the majority in the left bottom corner. The data points in 2.b is quite spread, compared with 2.a. And also, considering the outlier is a non-POI, whereas in the dataset we have abundant data points of non-POI, I decided to remove this outlier.

There is also another data point far away from others in 2.b, however, it might be considered as an outlier, but I decided not to remove it, due to 1) while the payments are much larger than others, the total stock value seems to be not very different from others i.e. about 1.5 times the second largest value in 2.b 2) it is a POI, if removing the complete record we will have even less data points to learn from, given that we already have only ~12% of POIs.

Therefore, only the data point in the top right corner in 2.a is removed.

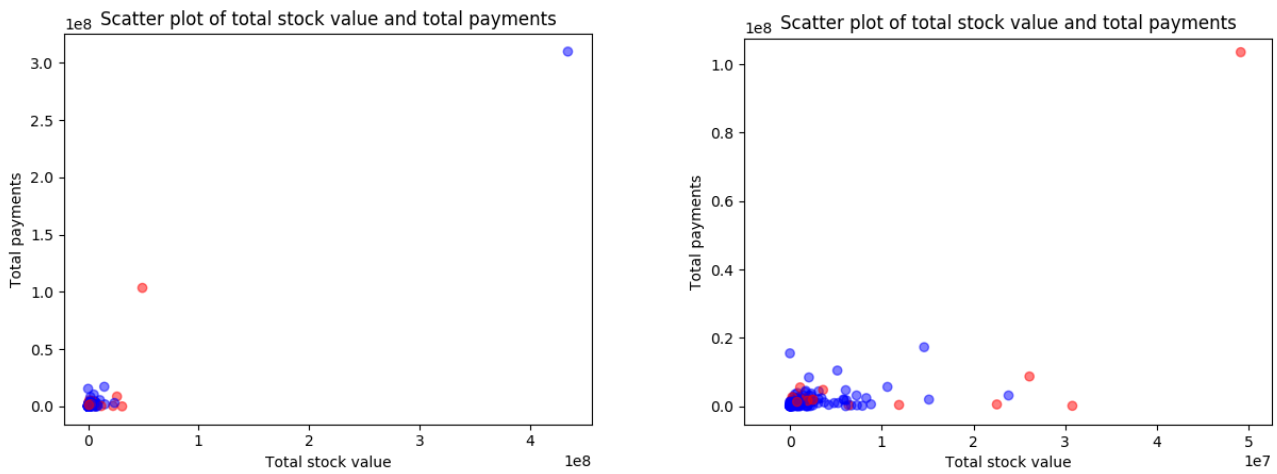


Figure 2.a Scatter plot of total payments and stock values

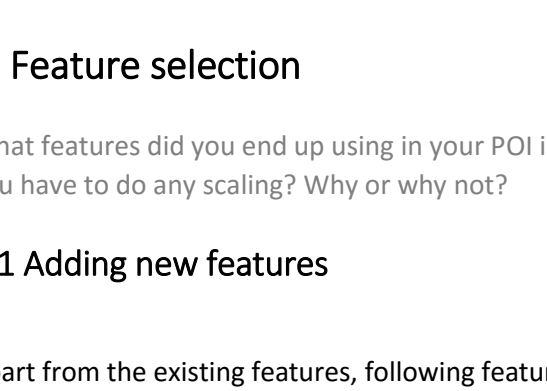
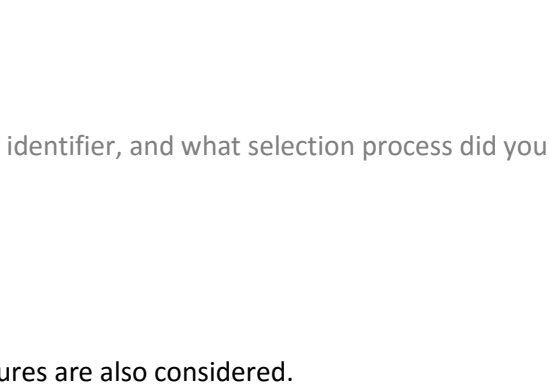


Figure 2.b Scatter plot zoom-in



2. Feature selection

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?

2.1 Adding new features

Apart from the existing features, following features are also considered.

1. from_this_person_to_poi_fraction: calculates the portion of POI related emails among all the emails a person sent. My hypothesis is that if the percentage of POI communication overall the total communication is high, it means this person is closely connected with POIs. Then this person is also likely a POI.

2. from_poi_to_this_person_fraction: calculates the portion of POI related emails among all the emails a person received. The reason is same as above.

3. bonus_over_payment_ratio: calculates the portion of bonus over total payments. If a person is a POI, then it is likely that he has bonus as major income rather than salary or others.

4. exercised_stock_ratio: calculates the portion of exercised stock over the total stock value. The hypothesis is a POI is likely to exercise the stock rather than leaving it still in the company's hold.

2.2 Feature transformation and selection

In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importance 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"]

2.2.1 Remove features with low variance

Before starting looking at which features are important for predicting the classifier, I would like to first eliminate features with nearly 0 variance. Because features with 0 variance indicate they are almost the same for each of the data record, i.e. nearly the same for POIs and non POIs. Thus they usually contain little information in predicting the classifier. Removing them could reduce the dimension and help simplify our input dataset a bit.

For this I used VarianceThreshold function. Since this function is based on the absolute values as thresholds, it does not make sense to compare for instance salary with from-messages. Thus scaling is applied here to avoid dominance of certain features. After min-max scaling and computing the variance, we have the following table. It seems there's not any feature containing nearly 0 variance.

Table 2. Feature variance after min-max scaling

feature	scaled_variance
exercised_stock_ratio	0.13436104
shared_receipt_with_poi	0.03765992
from_this_person_to_poi_fraction	0.03417767
from_poi_to_this_person_fraction	0.02776289
to_messages	0.02156128
from_poi_to_this_person	0.01955359
from_this_person_to_poi	0.01693518

loan_advances	0.01321813
bonus_over_payment_ratio	0.0123857
from_messages	0.01005785
total_payments	0.00755323
other	0.00744692
deferral_payments	0.00725292
director_fees	0.0072456
deferred_income	0.00721674
exercised_stock_options	0.00699143
total_stock_value	0.00695624
long_term_incentive	0.006952
bonus	0.00691246
expenses	0.00683061
salary	0.00680789
restricted_stock	0.00672274
restricted_stock_deferred	0.00390675

2.2.2 Select K best

Univariate feature selection method is used for feature selection. The idea of univariate feature selection is to analyze the relation between independent variable (in our case it is one of the features) and dependent variable (i.e. the labels). As a starting point, I chose the “select K best” method. In this method, there are a few score functions: chi-2, f_classif and mutual_info_classif. Considering all the features are continuous variables, and we are having a classification problem, mutual_info_classif is used with number of selected features K=15. Chi-2 is not used because in order for it to apply, the features should be categorical. Below shows the feature score value in descending order. From this plot we can see the engineered feature ‘from_this_person_to_poi_fraction’, ‘from_poi_to_this_person_fraction’, ‘exercised_stock_ratio’ are selected features. Whereas the original features such as ‘to_messages’, ‘director_fees’ and ‘loan_advances’ are not selected as features. Overall, coming up with new features is indeed very helpful for us to understand the dataset.

Even though we chose K=15 at the beginning, it is obvious from figure 3 that for features starting from ‘total_payments’, the importance becomes almost 0. Therefore, the decision is to exclude all features starting from ‘total_payments’. Afterwards, we have 13 features left.

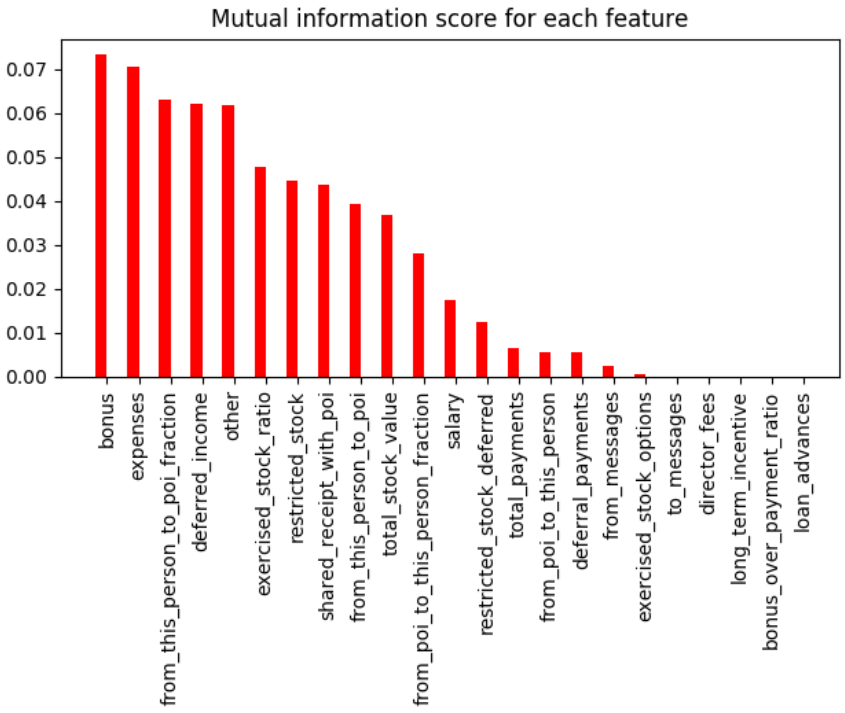


Figure 3. Feature importance ranked by mutual information score

3. Model building

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"]

In this exercise, I used Support Vector Machine and Naïve Bayes to solve the POI and non-POI classification problem. The advantage of using SVM are the follows: first it can deal with non-linear scenarios by applying the kernel trick; second is the regularization parameter, this can avoid overfitting of the model; also SVM is defined by a convex optimization problem with no local minima. Naïve Bayes is chosen because it can work well in small dataset and is relatively simple.

3.1 Support Vector Machine

3.1.1 Standardize features

Before applying training data to SVM model, the dataset needs to be standardized with mean = 0 and unit variance. This is necessary for SVM because due to calculating the margin (distance) between data points. If the dataset is not standardized, then it can occur that some columns with much larger variance will dominate the margin maximization process. For instance in our case, the total payments column is ranging from 0 to 3.1e8, whereas the other column from poi to this person is ranging from 0 to 528. It might be the case that from poi to this person will have little impact on determining the POI attribute whereas the total payments will decide POI to a large extent. Therefore, the dataset is standardized.

3.1.2 Dataset imbalance

Afterwards I tried to apply SVM to the standardized dataset directly. When looking at the F1 score, I encountered the following error message: "UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no predicted samples". It is very strange at the first sight. But later I figured out what has happened. Since F1 score is

calculated as (2 * precision * recall) / (precision + recall), if one of precision and recall is 0, then F1 is calculated with divide-by-zero error. That's why the error message appeared.

By looking at precision and recall separately, I found out this problem with F1 is due to both precision and recall. In the dataset there is no observation predicted as positive (POI). Thus TP and FP are 0. It results in precision and recall both to be 0. Hence when computing F1 we got a divide-by-zero error. Remember in the original dataset we only have 18 out of 145 people labelled as POI, the POI vs. Non-POI percentage is 12.4% vs. 87.6%. Thus we can conclude the actual problem is coming from the high imbalance of the dataset and SVM's not being able to learn the characteristics of POI data.

3.1.3 Cross validation performance

After identifying the problem, as a starting point I used SVM with class weight 0: 0.124 and 1:0.876, which is exactly the opposite of 0 and 1 percentages. By applying a stratified shuffle split with 5 splits to SVC, with class weight specified and all other parameters default, we have the following performance results.

Table 3. SVM model performance with 5 split cross validation

	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Average
F1 score	0.380952	0.470588	0.5	0.470588	0.333333	0.431092
Precision	0.25	0.333333	0.333333	0.333333	0.230769	0.296154
Recall	0.8	0.8	1	0.8	0.6	0.8

3.2 Naïve Bayes Classifier

NB is considered mainly due to its simplicity and good performance for small datasets. There are certain assumptions before building the model: 1) each feature is independent of each other given the class label 2) for numerical features the values follow normal distribution. To satisfy the first assumption, Pearson correlation is calculated and one in a pair of highly correlated features is removed. To satisfy the second, standardization is applied.

3.2.1 Correlation

Pearson correlation is calculated per two feature pair. Table below shows the correlation per two features. Positive correlations are colored blue and negative correlations are colored yellow. High correlation values are shown with dark colors for positive and negative ones respectively. Here we define correlation larger than 0.9 to be highly correlated. Then I started looking at each of the feature and identify how many features are correlated with it. The left most column listed all feature names as well as number of highly correlated features in brackets.

From the table we can see that there are 8 features highly correlated with each other, i.e. salary, total payments, bonus, expenses, other, deferred income, restricted stock, total stock value. 6 out of 8 comes from payment features and 2 from stock features. It is not difficult to understand these high correlations, since employees in a comparative position will get about the same level of payment and stocks. And usually the higher position it is, the more payment and stocks a person will get. To deal with feature correlation, I removed all correlated features and only included total payments.

	salary	total_ payments	from_this_ person_to_ poi_fraction	from_poi_to_ this_person_ fraction	bonus	restricted _stock	shared_ receipt_ with_poi	exercised _stock_ ratio	restricted _stock_ deferred	total_stoc k_value	expenses	other	from_this _person_ to_poi	deferred_ income	from_poi _to_this _person
salary (7)	1.000														
total_payments (7)	0.956	1.000													
from_this_person _to_poi_fraction	-0.022	0.015	1.000												
from_poi_to_this_ person_fraction (5)	-0.029	-0.031	0.490	1.000											
bonus (7)	0.993	0.962	-0.009	-0.022	1.000										
restricted_stock (7)	0.988	0.965	-0.027	-0.050	0.983	1.000									
shared_receipt_ with_poi	-0.009	0.031	0.291	0.251	0.047	-0.020	1.000								
exercised_stock _ratio	0.054	0.064	0.071	0.022	0.062	0.049	0.061	1.000							
restricted_stock _deferred	-0.445	-0.378	-0.002	-0.030	-0.441	-0.456	0.005	-0.130	1.000						
total_stock_value (7)	0.989	0.968	-0.027	-0.048	0.986	0.993	-0.027	0.087	-0.440	1.000					
expenses (7)	0.994	0.945	-0.042	-0.042	0.987	0.981	-0.024	0.039	-0.444	0.983	1.000				
other (7)	0.963	0.983	0.000	-0.035	0.959	0.972	0.003	0.063	-0.421	0.971	0.952	1.000			
from_this_person _to_poi	-0.007	-0.004	0.086	0.088	0.042	-0.014	0.525	0.028	-0.009	-0.022	-0.012	-0.040	1.000		
deferred_income (7)	-0.971	-0.923	0.009	0.021	-0.969	-0.957	-0.007	-0.089	0.436	-0.964	-0.963	-0.942	0.024	1.000	
from_poi_to_this _person	-0.007	0.029	0.261	0.585	0.054	-0.020	0.734	0.065	-0.023	-0.019	-0.028	0.007	0.498	-0.007	1.000

Figure 4. Pearson correlation of all selected features

3.2.2 Cross validation performance

After removing highly correlated features and applying standardization, I used same settings for cross validating the NB model performance. Following table shows the performance of NB Gaussian.

Table 4. NB Gaussian model performance with 5 split cross validation

	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Average
F1 score	0.190476	0.243902	0.190476	0.227273	0.227273	0.21588
Precision	0.108108	0.138889	0.108108	0.128205	0.128205	0.122303
Recall	0.8	1	0.8	1	1	0.92

3.3 Model comparison and observations

By comparing the two performance tables above, it is quite straightforward that the F1 score and precision are much higher than those of NB, whereas the recall is slightly lower. In general, this means the SVM model we used is better at recognizing the POIs. However, it is still very difficult to draw any conclusions of SVM and NB as to which one is the best to choose, since in the model building process there are many invariants, for instance different features used, weight of labels existing or not, which kernel function is used or which type of Naïve Bayes is used. Thus one cannot say that SVM will definitely outperform Naïve Bayes. While one model has to be selected for further improvements, I will choose SVM and exclude NB here. This is mainly because of the independence and normal distribution assumptions. It is a bit “naïve” for these assumptions to hold because in reality there is hardly any dataset which can fulfill these criteria very strictly. Consequently, the performance of NB will be influenced by not complying with these assumptions. Therefore SVM is chosen for further parameter tuning.

4. Parameter tuning

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? What parameters did you tune? (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"]

If we simplify and view our machine learning process as a manufacture line which given certain material will produce some product, then our objective is to design the line in such a manner that it fulfills our objective to the most (i.e. given input produce output as accurate as possible). Manufacture lines require to configure number of machines and their power, number of workers, volume and speed of the raw materials etc. to optimize cost / throughput / profits. Similarly, in machine learning we also need to optimize the model to get best performance (depending on how best performance is defined). To this end, parameter tuning is important for us to improve from "having a model" to "having a model that performs the best".

In SVM's case, there are a few parameters: 1) penalty parameter C, the extent of "soft margin" such that some data points can be on the wrong side of the decision boundary 2) type of kernel function 3) gamma, in case of e.g. RBF, polynomial and sigmoid kernel, deciding the influence level of a support vector.

Grid search can be applied here to find the optimal parameters. Given a set of parameters, grid search is building models and testing results for each possible combination of parameters and generate the model performance respectively. To start with, I explored parameter settings with C = [1, 100], gamma = [0.001, 0.0001], weight = [0:0.2, 0:0.5, 0:0.8]. And to avoid overfitting, I used 5-fold cross validation together with grid search. Below table shows the performance of each combination in terms of precision and recall. We can see that like mentioned before, class weight is important in achieving good performance, due to highly imbalanced classes. And when class weight for non-POI is 0.2, the precision is much better than other cases. When keeping class weight and C unchanged, it seems larger gamma values will result in better precision. Also we see when C is 100, the precision and recall is much better than when C is 1.

Table 5. Parameter tuning using grid search CV (1)

C	Gamma	Class weight	Average precision	Average recall
1	0.001	0.1	0.11410488	1
1	0.0001	0.1	0.11410488	1
1	0.001	0.2	0	0
1	0.0001	0.2	0	0
1	0.001	0.5	0	0
1	0.0001	0.5	0	0
1	0.001	0.8	0	0
1	0.0001	0.8	0	0
100	0.001	0.1	0.330639273	0.95
100	0.0001	0.1	0.162938693	1
100	0.001	0.2	0.439401249	0.758333334
100	0.0001	0.2	0.63547619	0.316666665
100	0.001	0.5	0.681666667	0.377777776
100	0.0001	0.5	0.65	0.116666666
100	0.001	0.8	0.8	0.280555554
100	0.0001	0.8	0	0

Based on the initial observations, I fixed the value of C to be 100 and weight to be 0:0.2. Then I applied the same grid search to variant gamma values, i.e. 0.1, 0.01, 0.001 and 0.0001. The result is as follows. From this result we see the optimal parameter is gamma = 0.01 where the precision results in highest at the cost of not much recall. Hence we

can conclude the optimal model among all the experimented combinations is: C = 100, gamma = 0.01 and weight = [0:0.2, 1:0.8].

Table 6. Parameter tuning using grid search CV (2)

Gamma	0.1	0.01	0.001	0.0001
Average precision	0.791667	0.62528	0.439401	0.635476
Average recall	0.65	0.9	0.758333	0.316666665

Overall, in the parameter settings, we found when C is larger, the result is slightly better. Large C means the model is actually choosing more samples as support vectors. Gamma value defines how far influence a support vector could reach. In our case, we selected a high gamma value, meaning the influence of a support vector is close to itself. We also noticed the model performs slightly better when Gamma is 0.01 compared with when Gamma is 0.1, in terms of precision and recall altogether.

In combination of the characteristics of the dataset, one can think of it as the model needs quite a few support vectors from the sample to differentiate POIs from non-POIs. This is actually understandable, because we do not have many POI samples, compared with the abundant non-POI samples. While the ultimate goal is to identify POIs, it is not easy to capture their characteristics given so small sample size. Thus many support vectors are needed to actually capture the difference between POIs and non-POIs.

5. Validation

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"]

In the model building process, it is very easy to run into overfitting. Overfitting happens when the model tries to learn each and every detail of the training set whereas loses the ability to generalize. Especially when the dataset contains noise, the model can sometimes learn the noise instead of the actual hidden knowledge. In SVM, it means too fine level of classification boundary. The idea is illustrated in the figure below. The classification boundary on the right side is seen as overfitting, because it tries to correctly classify every data point by creating decision boundaries belonging to the 'cat' class whereas the majority of data points around that area are all 'dog' class. The classification boundary on the left hand side is much more generalized.

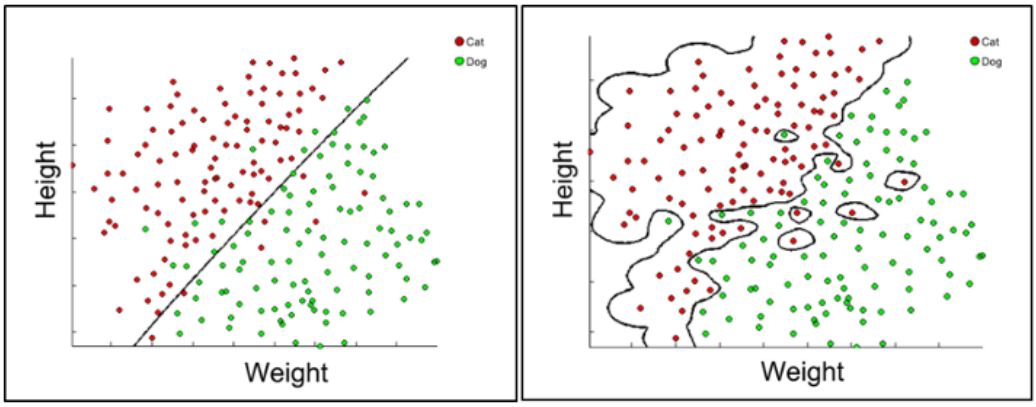


Figure 5. An example of overfitting

One way to avoid overfitting is to use cross validation as we did in previous section. By partitioning the data in to 5 folds, we train the data with 4 fold and test the model with 1 fold, which is 'unseen' to the model. The performance on the test set can actually represent the ability of the model to correctly classify on new or unseen data.

6. Model evaluation

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 algorithms performance. [Relevant rubric item: “usage of evaluation metrics”]

In the previous section, we used f1 score, precision and recall to evaluate the model. F1 score is the harmonious mean of precision and recall. Precision is TP / (TP+FP), in our case, this indicates among all the people classified as POIs, the number of real POIs. Recall is TP / (TP+FN), which indicates among all the people who are actually POIs, the number of people correctly identified.

In the result we see whenever running the algorithm, we get precision much lower whereas recall is still fine. This means the model is often confusing non-POIs as POIs, but for those who are actually POIs the can find most of them. Partially the reason is we used a weightage to let the model consider POIs to be ‘more important’, and that helped change the classification boundary towards POIs side.

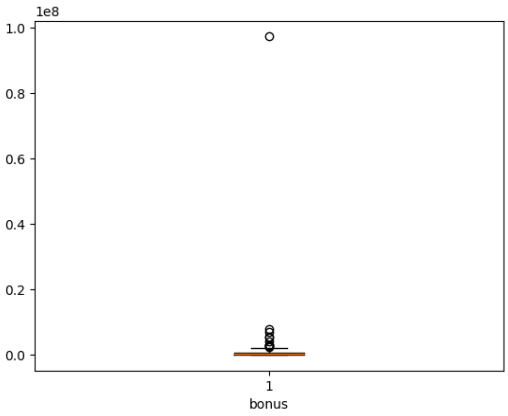
7. Appendix

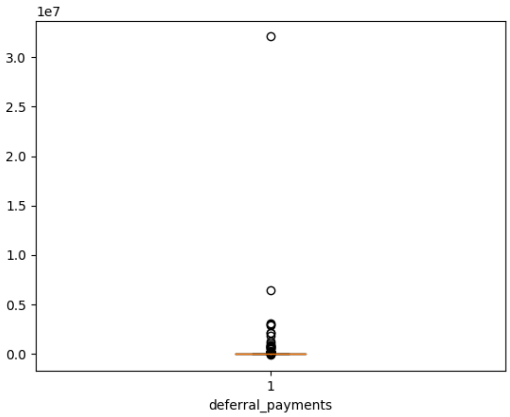
7.1 Feature importance scores

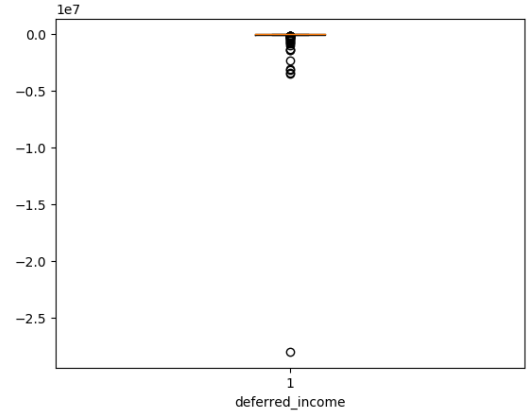
Feature scores using SelectKBest with scoring function 'mutual_info_classif'.

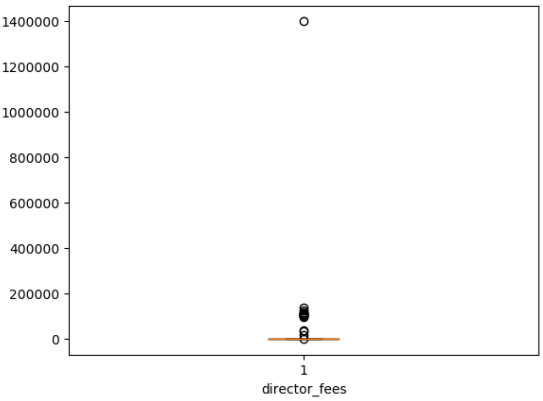
feature	score
bonus	0.073218
expenses	0.070432
from_this_person_to_poi_fraction	0.062907
deferred_income	0.062096
other	0.061883
exercised_stock_ratio	0.047688
restricted_stock	0.044442
shared_receipt_with_poi	0.043509
from_this_person_to_poi	0.039393
total_stock_value	0.036662
from_poi_to_this_person_fraction	0.028152
salary	0.017513
restricted_stock_deferred	0.012496
total_payments	0.006572
from_poi_to_this_person	0.005476
deferral_payments	0.005387
from_messages	0.002242
exercised_stock_options	0.000533
to_messages	0
director_fees	0
long_term_incentive	0
bonus_over_payment_ratio	0
loan_advances	0

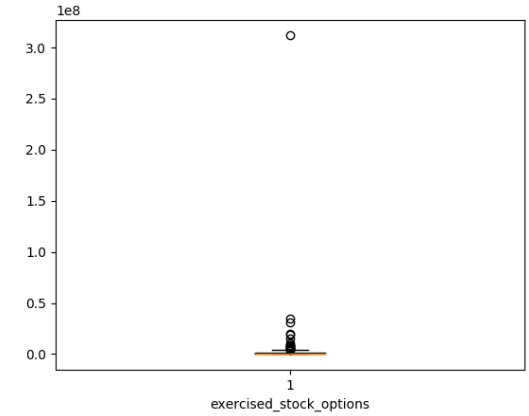
7.2 Boxplot of Enron data features

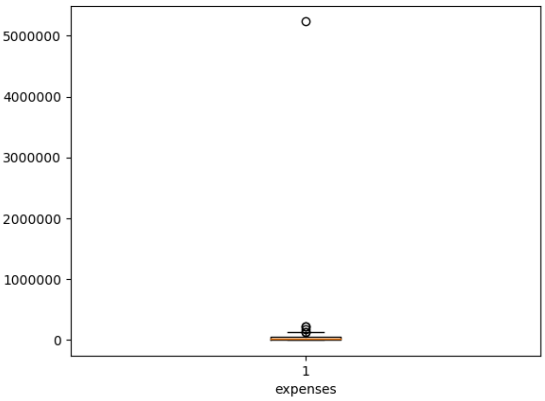


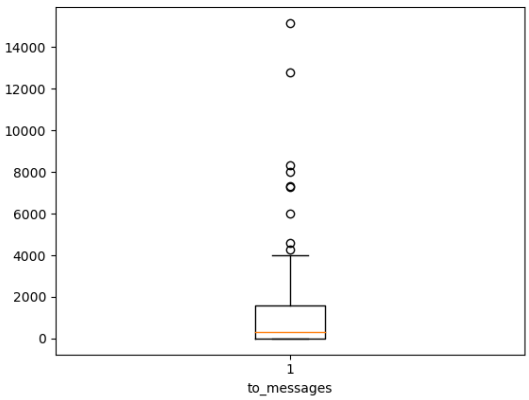


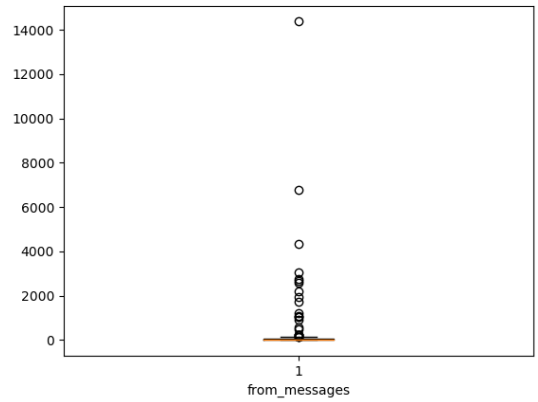

1e7
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
1
deferred_income

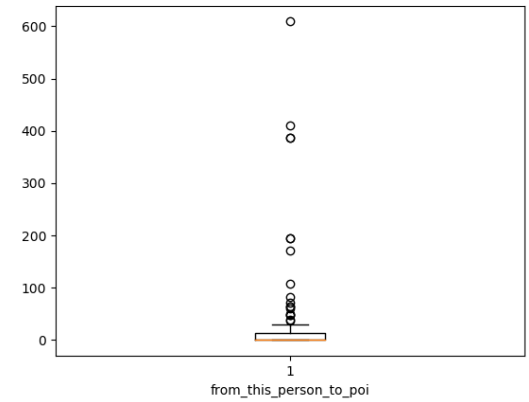

1400000
1200000
1000000
800000
600000
400000
200000
0
1
director_fees

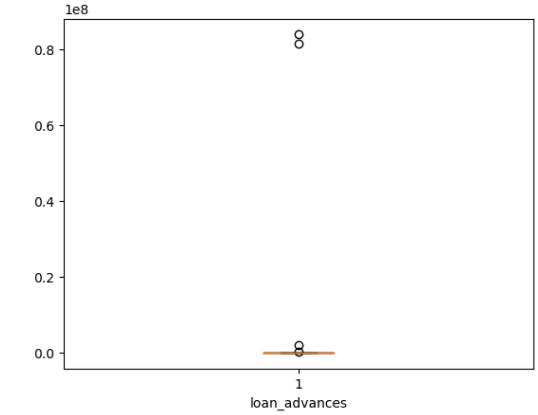

1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
exercised_stock_options

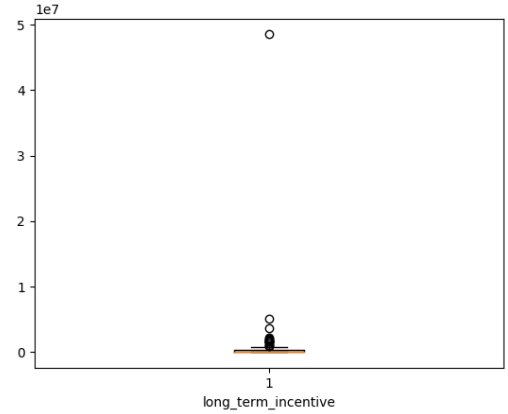

5000000
4000000
3000000
2000000
1000000
0
1
expenses

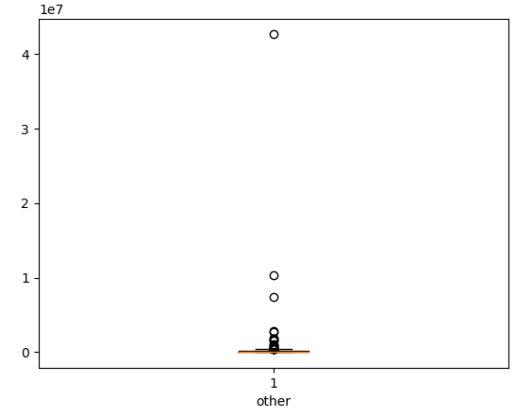

14000
12000
10000
8000
6000
4000
2000
0
1
to_messages

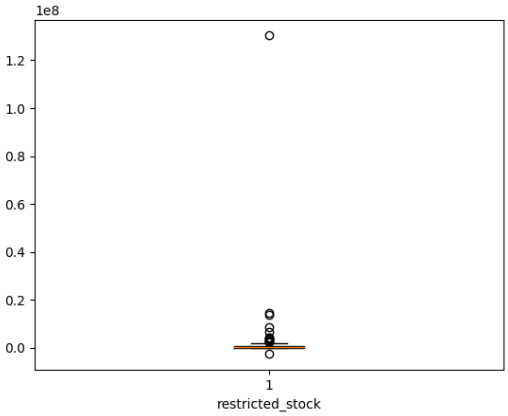

14000
12000
10000
8000
6000
4000
2000
0
1
from_messages

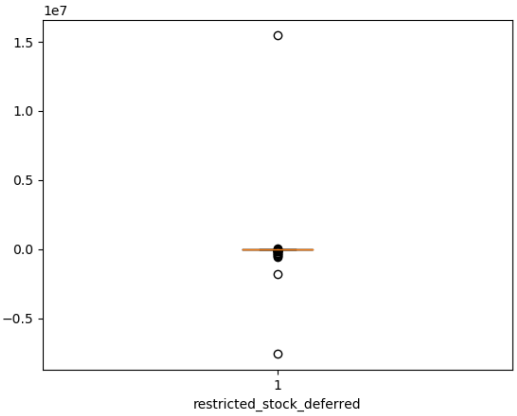

600
500
400
300
200
100
0
1
from_this_person_to_poi

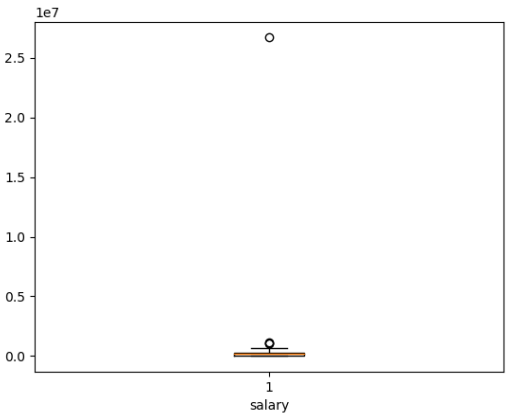

1e8
0.8
0.6
0.4
0.2
0.0
1
loan_advances

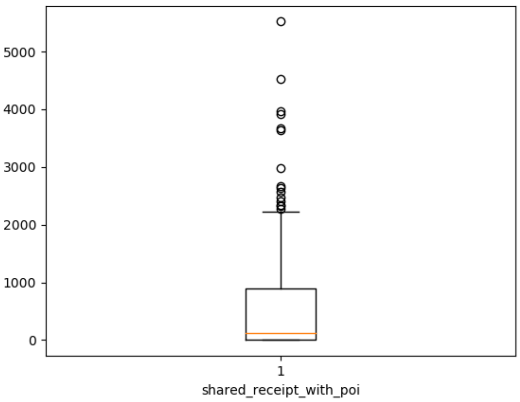

1e7
5
4
3
2
1
0
1
long_term_incentive

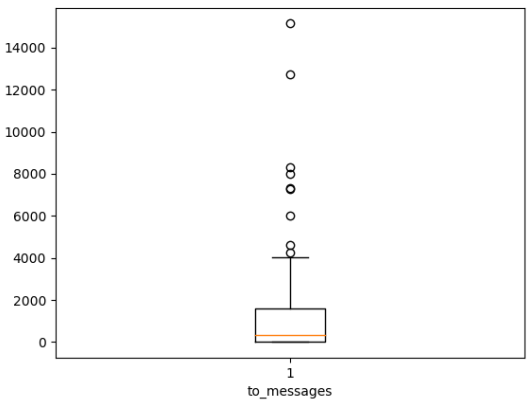

1e7
4
3
2
1
0
1
other

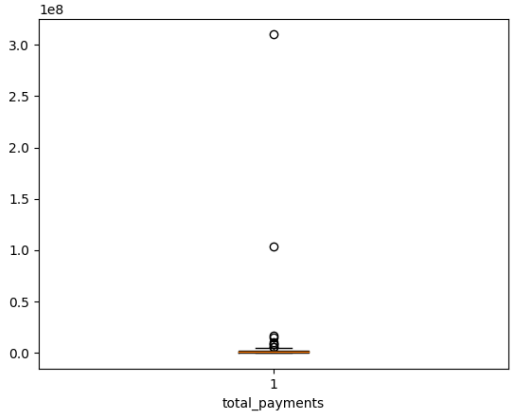

1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1
restricted_stock

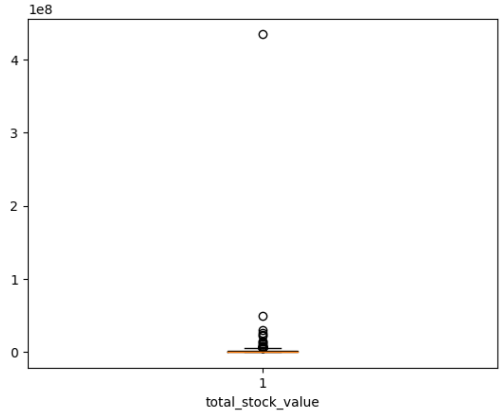

1e7
1.5
1.0
0.5
0.0
−0.5
1
restricted_stock_deferred


1e7
2.5
2.0
1.5
1.0
0.5
0.0
1
salary


5000
4000
3000
2000
1000
0
1
shared_receipt_with_poi


14000
12000
10000
8000
6000
4000
2000
0
1
to_messages


1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
total_payments


1e8
4
3
2
1
0
1
total_stock_value