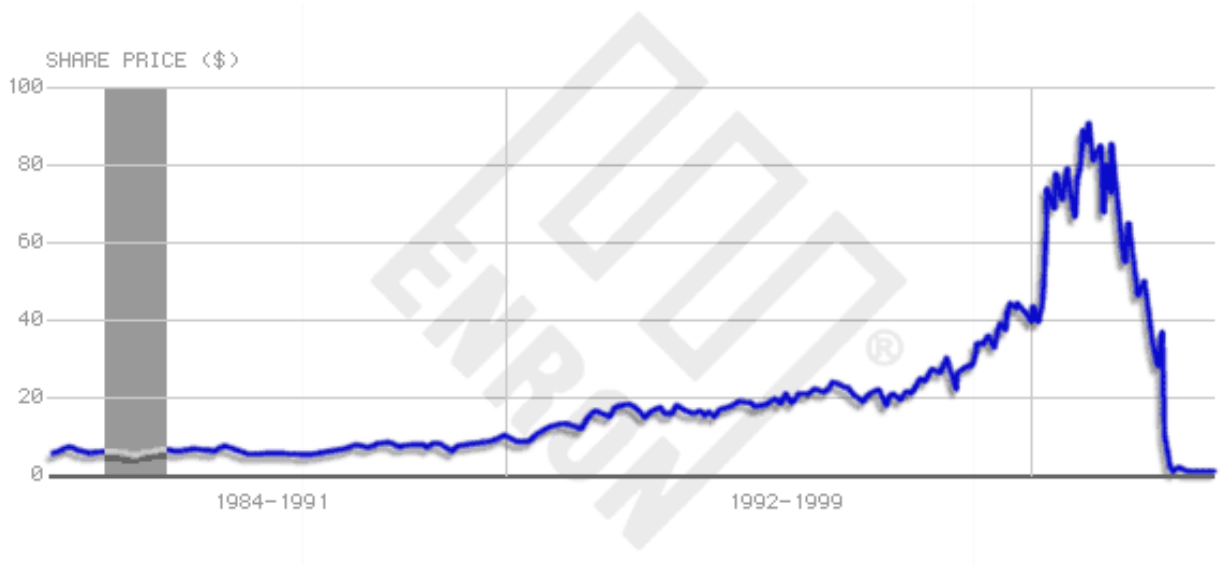
Quentin THOMAS

UDACITY DAN Intro to Machine Learning project

Identify Fraud from Enron Email



Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: "data exploration", "outlier investigation"]

Enron Corporation was an American energy, commodities, and services company based in Houston, Texas. It was founded in 1985 as the result of a merger between Houston Natural Gas and InterNorth, both relatively small regional companies. At the end of 2001, it was revealed that its reported financial condition was sustained by institutionalized, systematic, and creatively planned accounting fraud, known since as the Enron scandal. Enron has since become a well-known example of willful corporate fraud and corruption.(Source Wikipedia)

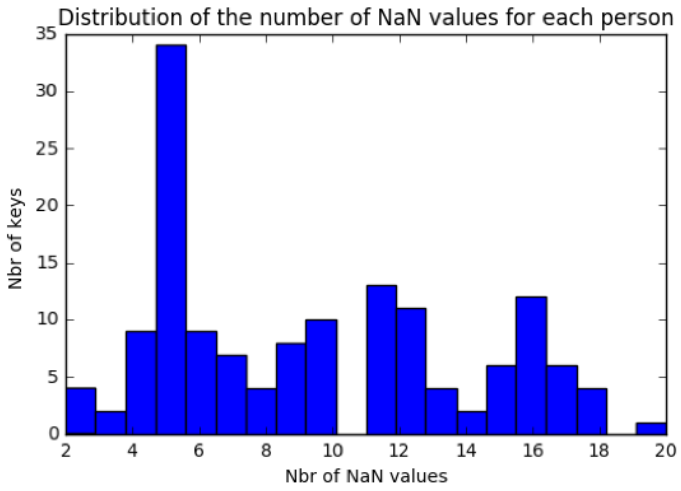
The goal of this project is to use machine learning skills for building a person of interest identifier based on financial and email data made public as a result of the Enron scandal.

The dataset provided is a dictionary where each of the 146 key corresponds to one person identified by its name. The key value corresponds to another dictionary which contain all the features:

- 1 label « poi » which we will try to predict
- 14 financial features
- 6 email features

Only 18 persons are labelled as a « POI » which is very few.

A lot of these features have NaN values and I decided to plot the distribution of the number of NaN values for each key (person):



As we have a very few amount of data, I did not want to remove too much keys. The best cut I could see was about at 16 NaN values (without include POI).
This means that 11 non-POI keys (included THE TRAVEL AGENCY IN THE PARK) have been removed, which seems to be a good compromise.

I have also removed the « TOTAL » key which is obviously not a person.
All in all, 12 keys have been removed which represents about 8% of the data.

List of keys removed:

Name	Number of NaN values	POI ?
LOCKHART EUGENE E	20	False
WHALEY DAVID A	18	False
WROBEL BRUCE	18	False
THE TRAVEL AGENCY IN THE PARK	18	False
GRAMM WENDY L	18	False
WODRASKA JOHN	17	False
CLINE KENNETH W	17	False
WAKEHAM JOHN	17	False
SCRIMSHAW MATTHEW	17	False
GILLIS JOHN	17	False
SAVAGE FRANK	17	False
TOTAL		

What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: "create new features", "intelligently select features", "properly scale features"]

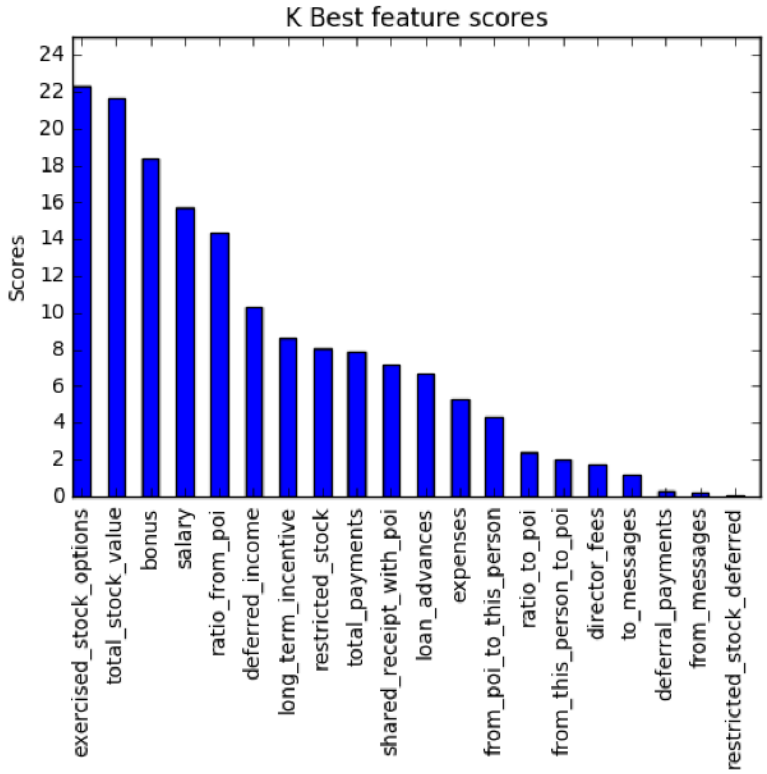
I have removed by instinct the « email » feature as I do not figure how it can help the prediction and the « other » feature which seems to be useless.

Two new features have been created:

- « ratio_from_poi » which is the number of email received from a poi divided by the total number of email received (by person).
- « ratio_to_poi » which is the number of email sent to a poi divided by the total number of email sent (by person).

Then all the remaining features have been scaled by the MinMaxScaler function from the SCIKIT Learn preprocessing package. This function transform feature values into a value include in a 0 (minimum value) to 1 (maximum value) range.

I used the SelectKBest function (with the f_classif algorithm) to select the best features:



I have noticed a significant gap after the top 5 feature scores and have decided that it was a good cutoff for keeping the bests of them.
The 5 retained features are highlighted in green:

exercised_stock_options: 22.329
total_stock_value: 21.670
bonus: 18.389
salary: 15.719
ratio_from_poi: 14.330
deferred_income: 10.268
long_term_incentive: 8.578
restricted_stock: 8.076
total_payments: 7.875
shared_receipt_with_poi: 7.189

loan_advances: 6.656
expenses: 5.275
from_poi_to_this_person: 4.342
ratio_to_poi: 2.404
from_this_person_to_poi: 1.993
director_fees: 1.750
to_messages: 1.193
deferral_payments: 0.302
from_messages: 0.226
restricted_stock_deferred: 0.077

I noticed that only email feature kept is the ratio of email coming from a poi.

> 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"]

I used roughly 4 algorithms with the default parameters (except for the number of clusters in K Means and the random parameter for the random forest)

Algorithm	Accu.	Preci.	Recall	F1	Tpos	Tneg	Fpos	Fneg
KMeans(n_clusters=2)	0,758	0,259	0,305	0,280	611	9253	1747	1389
GaussianNB()	0,853	0,537	0,327	0,407	655	10435	565	1345
DecisionTreeClassifier()	0,776	0,296	0,330	0,312	660	9430	1570	1340
RandomForestClassifier(random_state = 42)	0,843	0,480	0,210	0,292	420	10545	455	1580

Because we look for guilty people, we want that the POI predicted are as much as possible real POI. In this case the most important score to look at is the precision.
The Gaussian naive bayes algorithm has the higher precision, accuracy and has also a decent recall score.

> 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 items: "discuss parameter tuning", "tune the algorithm"]

Tuning consists in changing the chosen algorithm parameters in order to increase its performance. Each algorithm has a different amount and type of parameters. A bad tuning can lead to poor results or a performance issue (too much time to process the algorithm).
The idea behind tuning a machine learning algorithm is not to reach at all cost the best prediction score on the training data. First because we want to avoid overfitting and we prefer to have a generalize algorithm that will work above all on unseen data. Secondly accuracy score is not the only validation score and we could want to priories an other score (such as precision) in function of the initial problem.

Gaussian naive bayes algorithm already gives results which meet requirements (precision and recall scores > 0.3) however these scores cannot be improved by tuning the parameters as there are no parameters to tune.
In this case I decided to take the Random Forest algorithm and see if by tuning its parameters I can beat the Gaussian naives bays.

According to the SCIKIT Learn documentation, Random Forest parameters such as n_estimators, max_features and min_samples_split should be tuned for a classification use.

I have used grid_search function in order to automatically combine the different parameters:

n_estimators	max_features	min_samples_split	min_samples_leaf
10	50 %	1	1
20	100 %	2	2
30	Square root(number features)	3	3
40	Log2(number features)	5	5
50		8	8
100		10	10

This table show the parameter values submitted to the grid_search function. Cells highlighted in yellow has been retained by the function.

The score obtained with these parameters is:

Algorithm	Accu.	Preci.	Recall	F1	Tpos	Tneg	Fpos	Fneg
Not Tuned Random Forest	0,843	0,480	0,210	0,292	420	10545	455	1580
Tuned Random Forest	0,847	0,507	0,163	0,247	326	10684	316	1674

Tuning has slightly improved the precision and accuracy score to the detriment of the recall score.

> 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 items: "discuss validation", "validation strategy"]

Validation is used to mesure the performance of an machine learning algorithm when confronted with unseen data. It allows to compare the performance of different algorithm and parameters tuning on the same data set and to choose the best.

Several type of score can be used:
- *Accuracy: pure prediction result*
- *Precision: quality of prediction (is the number of True Positives divided by the number of*
- *True Positives and False Positives)*
- *Recall: quality of prediction (is the number of True Positives divided by the number of True Positives and the number of False Negatives)*
- *F1: Combination of precision and recall score*

Notion of true/false - positive/negative is very important (source :http://www.uta.fi/sis):

	Prediction of model: positive	Prediction of model: negative
Truth: positive	TP	FN
Truth: negative	FP	TN

Validation is never operated on the train data in order to detect overfitting behavior which could happen if you do not split the data into train and test set.

When the number of data is very low some function can shuffle and split original data set into training and test subsets in order to exploit the maximum of them.

The project script in tester.py uses a stratified shuffle split cross validation in order to address two specific problems of this dataset:
- It is a small dataset which requires to create a lot of training subsets to use the maximum of it.
- As they are few POI, it is a very imbalanced (and small) dataset. The validation used guarantees that the percentage of the target class in each subset is as close as possible to the one we have in the complete dataset

> Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]

Algorithm	Accu.	Preci.	Recall	F1	Tpos	Tneg	Fpos	Fneg
GaussianNB()	0,853	0,537	0,327	0,407	655	10435	565	1345
Tuned Random Forest	0,847	0,507	0,163	0,247	326	10684	316	1674

The final result below shows that I will keep the initial Gaussian naive bayes algorithm instead of the tuned Random Forest.
They look pretty close but the recall score make the difference This score means that the Tuned Random Forest algorithm proportionally misses much more POI than the Gaussian NB does (Number of True Positives divided by the number of True Positives and the number of False Negatives).

Due to the nature of the prediction which involve guilty and innocent people the most important score remains the precision score (Number of True Positives divided by the number of True Positives and False Positives) as we we wish to be sure that our flagged POI are indeed POI. Once again the Gaussian naive bayes is the best here.