

Final Project of Project 4 for Udacity Data Analysis Nanodegree

Identifying Fraud from Enron Emails and Financial Data Using Machine Learning

Zhongyuan Zhou

Introduction

Before bankruptcy, Enron was one of the world's major electricity, natural gas, communications, and pulp and paper companies based in Houston, Texas. It employed approximately 20,000 staffs and had claimed revenues of nearly \$111 billion during 2000 [1]. At the end of 2001, Enron collapsed into bankruptcy due to corporate fraud, which was known as Enron scandal. After that a significant amount of confidential information, including tens of thousands of emails and detailed financial data for top executives, entered into public record. Enron has since become a well-known example of willful corporate fraud and corruption.

Answers to short questions

1. *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?*

Goal: The goal of this project is to build a prediction model to identify “person of interest” (POI) based on emails and financial data in Enron scandal using machine learning skills.

Background of the dataset: I apply the pandas DataFrame to explore the structure of the dataset. The dataset has a form of dictionary data. In the dataset there are 146 persons (data points) in which there are 95 persons who have quantified salary, 21 persons who have NaN total payment, and 111 persons who have email address. There are 21 features in which there are 6 email features, 14 financial features, and 1 labeled feature (POI). Out of 35 listed POI names in poi_names.txt there are 18 persons who are finally labeled as POIs in the dataset.

Outliers: Visualization is one of the most powerful tools for finding outliers. The scatter plot of financial features, salary and bonus, are plotted first to identify outliers. Four possible outliers are found from this plot: 'TOTAL', 'LAVORATO JOHN J' (a top executive), 'LAY KENNETH L' (former Chairman and Chief Executive Officer), 'SKILLING JEFFREY K' (former Enron CEO). Apart from 'TOTAL', other three persons were former high-ranking executives. They are definitely important and should be kept in the investigation. Thus only one major outlier 'TOTAL' should be removed.

Usually a person's name contains three parts: first, middle, and last names. The number of words/characters of a person's name should be larger than 1 and less than 4 if suffixes such as Junior/Senior, Younger/Elder, or Ordinals are ignored. Based on this characteristic, the key names of the dictionary are checked and found another outlier 'THE TRAVEL AGENCY IN THE PARK' apart from 'TOTAL'. Because this record is not for a person it is an outlier and should be removed.

By checking 'NaN', another outlier 'LOCKHART EUGENE E' is found. This record does not contain any data and should be removed.

I also explore possible outliers from email features, in particular, "from_poi_to_this_person" and "from_this_person_to_poi", but do not find any obvious outliers.

Apart from the outliers above, I also find some bugs. Both the "restricted_stock_deferred" and "deferred_income" should be negative. I find that the values of "restricted_stock_deferred" of two records, "BHATNAGAR SANJAY" and "BELFER ROBERT", are positive. Obviously these data are not outliers but typos. These data can either be kept as is or be removed as outliers. I will keep these data as is because some other features are still correct and may contribute to the machine learning problem.

The outliers identified above have been removed using the `data_dict.pop(i_remove, 0)`. After removal of the 3 outliers a much clean dataset with 143 points is gained for further study.

2. *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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.*

Features: I end up using 14 features: 'exercised_stock_options', 'loan_advances', 'total_stock_value', 'final_total_stocks', 'bonus', 'ratio_to_poi', 'final_total_benefits', 'salary', 'final_total_payments', 'total_payments', 'long_term_incentive', 'shared_receipt_with_poi', 'other', and 'deferred_income'. These features contain 1 email-related feature and 13 finance-related features, in which there are 4 new features (see the *italics*).

New features: Five new features, two email-related and three finance-related features, are created based on analysis to the dataset. Usually POIs should have stronger connection between each other [2] and thus email communications between POIs should be more frequent than those between POI and non-POI. If a person receives more emails from POIs or a person sends more emails to POIs the person is probably a POI. Although this has been characterized by two features, "from_poi_to_this_person" (the number of emails from POIs to this person) and "from_this_person_to_poi" (the number of emails from this person to POIs), in the dataset, the description is not that rigorous and accurate. For example, an IT technician may receive/send more emails from/to POIs but may not be a POI. Thus in addition to the numbers of emails "from_poi_to_this_person" and "from_this_person_to_poi", two new features relevant to these two features are created: (1) *ratio_from_poi*: the ratio of the number of emails "from_poi_to_this_person" to the number of emails "to_messages" [2] and (2) *ratio_to_poi*: the

ratio of the number of emails “from_this_person_to_poi” to the number of emails “from_messages” [2].

POIs should be strongly associated with Enron's finance issues. In fact, a person who is identified as a POI must illegally benefit from Enron greatly. POIs may benefit in different ways. For example, in order to deceive other people, POIs may adopt “*deferred_income*” and “*restricted_stock_deferred*” to reduce “*total_payments*” and “*total_stock_value*”, respectively. However, the POIs should be identified based on their total benefits in which “*deferred_income*” and “*restricted_stock_deferred*” are included. Because “*deferred_income*” and “*restricted_stock_deferred*” are excluded from “*total_payments*” and “*total_stock_value*”, respectively, both “*total_payments*” and “*total_stock_value*” cannot play a role of total benefits. In order to compensate such kind of flaw in dataset, I introduce three new financial features: (1) *final_total_stocks*: the sum of “*exercised_stock_options*” and “*restricted_stock*” = the sum of “*total_stock_value*” and absolute value of “*restricted_stock_deferred*”; (2) *final_total_payments*: the sum of “*total_payments*” and absolute value of “*deferred_income*”; and (3) *final_total_benefits*: the sum of “*final_total_stocks*” and “*final_total_payments*”.

As shown in Table 1 that based on the scores of features there are 4 new features in the first 10 features. This demonstrates that 4 out of 5 new features play an important role. This is further justified by the metrics obtained from the models with different new features in Table 5. In particular, the email-related new feature ‘*ratio_to_poi*’ always has large effect on the results, while precision and recall reach their maximum values when the email-related new feature is combined with all the 3 finance-related new features simultaneously.

Feature scaling: Before performing feature selection all the features including new created ones are scaled to the range of [0, 1] using MinMaxScaler. This is a very important step because the original features have different units and vary significantly by several orders of magnitudes. In particular, some features such as ‘*deferred_income*’ and ‘*restricted_stock_deferred*’ are negative. Scaling of features has obvious impact on the feature selection and on the results of some classifiers such as SVM with rbf kernel [2]. For some algorithms, such as Decision Tree, scaling of features may be unnecessary, but the scaled features are not going to hurt their performance.

Feature selection: The univariate feature selection method that is based on univariate statistical tests is used to select the best features. The module *selectKBest* together with univariate statistical χ^2 tests [3] are applied to the feature optimization. Given in Table 1 are the scores of features and p-values of the feature scores. The scores of features are listed in descending order, which correspond the p-values in ascending order. The procedure of feature selection includes (1) calculating a score for each feature and (2) selecting features that have higher scores (lower p-values) [4].

Number of features: There are many systematic methods to optimize features and select the number of features, such as recursive feature selection method based on principal component analysis (PCA) [10]. However in this project I select the features based on the order of features in Table 1 which are reordered by the scores of features. First, K features with high scores of features are chosen from Table 1, then the features together with dataset are fed into Stratified ShuffleSplit cross validation iterator [11], *StratifiedShuffleSplit*, with 2000 folds to split the dataset into training and testing groups, and finally the training and testing groups are fed into various algorithms to compute evaluation metrics. By comparing the evaluation metrics from different algorithms and different number of features, the algorithms are optimized and the

numbers of features are determined. As shown in next section, in the final model the optimal algorithm is Decision Tree classifier and the best features are the first 14 features in Table 1.

Table 1. Scores of features and p-values of feature scores.

Index	Features	Scores	p-values
1	'exercised_stock_options'	6.85414	0.00884
2	'loan_advances'	6.77150	0.00926
3	'total_stock_value'	5.49662	0.01905
4	'final_total_stocks'	5.43752	0.01971
5	'bonus'	5.27474	0.02164
6	'ratio_to_poi'	4.64457	0.03115
7	'final_total_benefits'	3.74916	0.05283
8	'salary'	3.05446	0.08052
9	'final_total_payments'	2.96987	0.08483
10	'total_payments'	2.78492	0.09516
11	'long_term_incentive'	2.65941	0.10294
12	'shared_receipt_with_poi'	2.43471	0.11868
13	'other'	1.71595	0.19021
14	'deferred_income'	1.66742	0.19660
15	'expenses'	1.48491	0.22301
16	'director_fees'	1.48153	0.22353
17	'from_poi_to_this_person'	1.37006	0.24180
18	'restricted_stock'	1.32812	0.24914
19	'from_this_person_to_poi'	1.00081	0.31712
20	'ratio_from_poi'	0.82723	0.36307
21	'to_messages'	0.43623	0.50895
22	'restricted_stock_deferred'	0.34203	0.55866
23	'deferral_payments'	0.07647	0.78215
24	'from_messages'	0.07244	0.78782

* *italics represent new features*

3. What algorithm did you end up using? What other one(s) did you try?

I have tried 5 different algorithms: GaussianNB [5], Support Vector Machines (SVMs) [6], Decision Tree [7], K Neighbors [8], and Logistic regression [9]. By computing and comparing the evaluation metrics, precision and recall, for different algorithms at different parameters with different features, the algorithms and parameters are optimized, and the features and the number of features are determined.

In Figure 1 the evaluation metrics (precision and recall) versus the number of features K are plotted for the 5 algorithms at the tuned parameters. It is shown that overall the evaluation metrics change with K largely for K in range of 1~14. This means that increasing features has obvious impact on the evaluation metrics. For K in range 15~24, the overall trend of the metrics changes slowly. This means that increasing features does not improve the results that much. Thus 14 features are chosen in my final model. Obviously, the metrics reach their maximum when 14 features are used in DecisionTree classifier. In Table 2 the metrics (precision and recall) are listed for 5 different algorithms with 14 features at tuned parameters that are given in Table 3. There are two algorithms, DecisionTree and LogisticRegression, their precision and recall are larger than 0.3. Because the recall from DecisionTree is larger than that from LogisticRegression and, as demonstrated below, the recall is more significant than the precision

in this project, thus I end up use DecisionTree classifier. The precision and recall are *0.51487* and *0.51075*, respectively, for this classifier at tuned parameters.

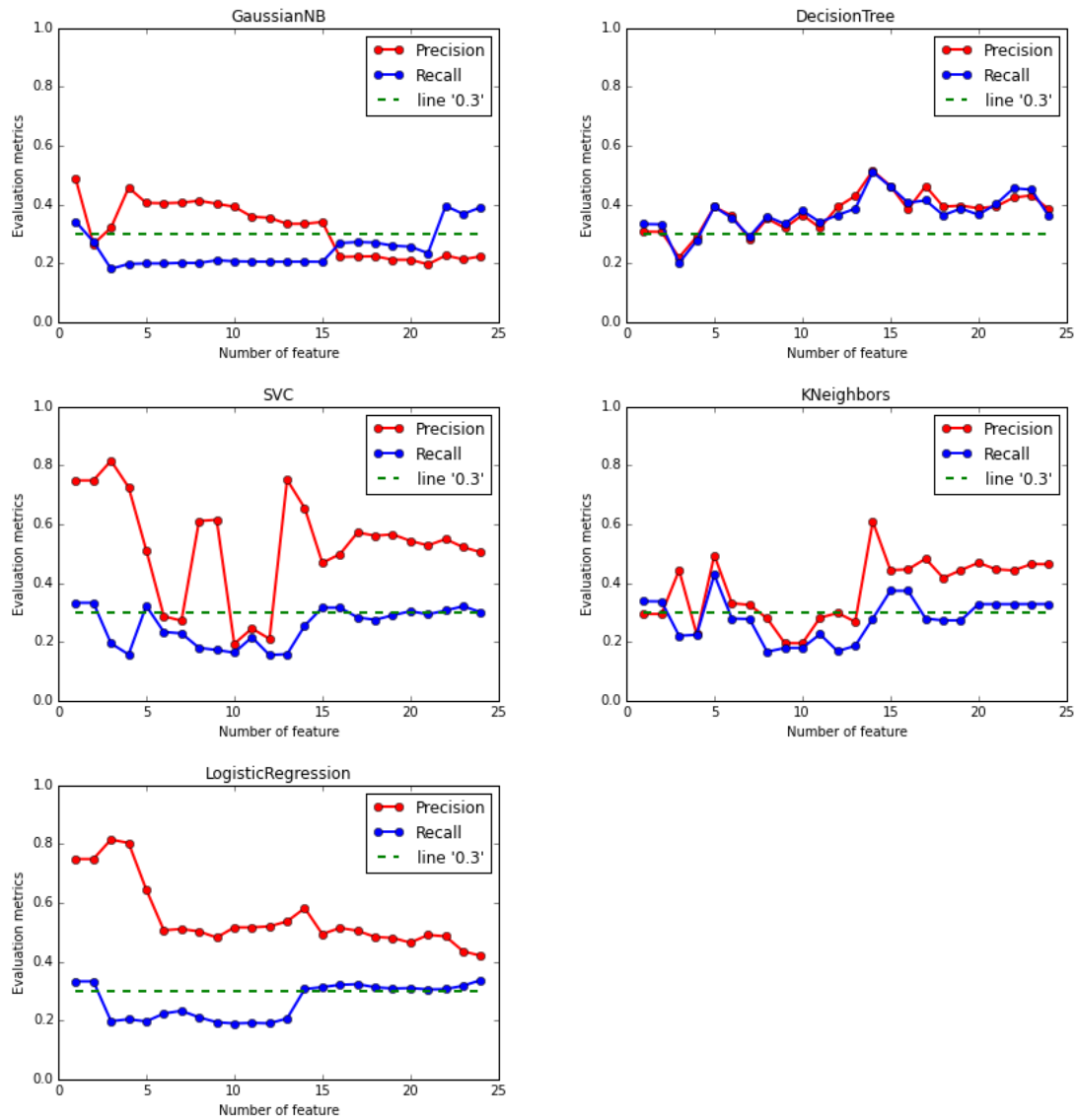


Figure 1. The evaluation metrics (precision and recall) versus the number of features at tuned parameters for 5 different algorithms

Table 2. The precisions and recalls for different algorithms at tuned parameters with 14 features

Algorithms	Precision	Recall
GaussianNB	0.33496	0.20600
DecisionTree	<i>0.51487</i>	<i>0.51075</i>
SVMs	0.65493	0.25575
KNeighbors	0.60948	0.27975
LogisticRegression	0.58246	0.30725

4. 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? (Some algorithms don't 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 if you used, say, a decision tree classifier).

Tune parameters: Every algorithm is parameterized by some tunable parameters and thus the performance (speed, accuracy, robustness, etc.) of the algorithm can be optimized by changing the parameters. The process of systematically searching the combination of appropriate parameters so as to gain the best performance of the algorithm is called “tuning the parameters”.

Improper parameters: If the parameters of an algorithm are not well tuned, improper parameters will be achieved. When these parameters are used in the algorithm it may cause (1) bad performance with low metrics (precision and recall) and (2) mistakenly excluding a good algorithm.

How to tune parameters: The module *GridSearchCV* [12] is a grid search approach for parameter tuning that methodically builds and evaluates a model for each combination of algorithm parameters specified in a grid [2]. I use *GridSearchCV* to tune and optimize major tunable parameters of an algorithm over 2000 randomized folds of Stratified ShuffleSplit cross validation iterator [11], *StratifiedShuffleSplit*. The final result is averaged scores over all 2000 folds. The parameters producing the highest averaged score are selected for the final model. The optimized parameters are dependent of algorithms and features. In Table 3, the input values and tuned values (underlined & bold) of tuning parameters are listed for 4 algorithms with 14 features. Note that *GaussianNB* does not have tuning parameters.

Table 3. Input values and tuned values of tuning parameters for 4 algorithms with 14 features

algorithms	Tuning parameters					
	Parameter 1		Parameter 2		Parameter 3	
	Name	values	Name	Values	Name	Values
DecisionTree	criterion	gini, <u>entropy</u>	max_features	None, <u>auto</u>		
SVMs	kernel	linear, <u>rbf</u>	C	1, 20, 150, <u>300</u> , 500	gamma	<u>0</u> , 5, 10, 50
KNeighbors	Weights	<u>uniform</u> , distance	n_neighbors	1, 2, <u>3</u> , 4, 5, 6		
LogisticRegression	Penalty	l1, <u>l2</u>	C	10, 20, <u>50</u> , 100, 150		

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation: validation, in this course, is a process to check the performance of a model built on a known dataset (training group) over an unknown dataset (testing group) for validation so as to well generalize the model to all the other datasets. A classic mistake if doing the validation wrong is overfitting, that is, the model is trained to make prediction very well on the known dataset, but it makes worse predictions on the unknown datasets.

Cross-validation: In my analysis, I use cross-validation method to validate my model. The cross-validation is a process that randomly splits the data into training and testing groups, and

then the model is trained and built on the training group, and validated on the testing group. Due to small dataset, in this project, instead of the simple cross-validation method such as `TrainTestSplit`, the randomized Stratified ShuffleSplit cross validation [11], *StratifiedShuffleSplit*, with 2000 folds is used in the process of parameter tuning and model validation [13]. It will make randomly chosen training and testing datasets multiple times and average the results over all the tests. The data is unbalanced with many more non-POIs than POIs. *StratifiedShuffleSplit* can make sure that the ratio of non-POI:POI is the same in the training and testing datasets as it was in the larger data set. The training/testing datasets are set to 90%/10% of the total dataset.

6. *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.*

Due to the unbalanced nature of the dataset (more non-POIs than POIs), accuracy is no longer a good metric for evaluation. I will adopt precision and recall as evaluation metrics. They are defined by [14]

$$Precision = \frac{Tp}{Tp + Fp},$$

and

$$Recall = \frac{Tp}{Tp + Fn},$$

respectively, where, Tp (= *True positives*) is the rate that an actual positive is truly predicted as a positive, Fp (= *False positives*) is the rate that an actual negative is falsely predicted as a positive, and Fn (= *False negatives*) is the rate that an actual positive is falsely predicted as a negative. The precision is the ratio of the true positives against total predicted positives (true positives and false positives). The recall is the ratio of the true positives against total actual positives (true positives and false negatives). A high precision means less Fp (false positives) and a high recall means less Fn (false negatives).

In POI detection, a positive is a POI and a negative is a non-POI. Thus Tp is the rate that a POI is truly predicted as a POI, Fp is the rate that a non-POI is falsely predicted as a POI, and Fn is the rate that a POI is falsely predicted as a non-POI. For a high-precision prediction, the rate that a non-POI is wrongly identified as a POI is low, which can make sure that an actual non-POI is not treated as a POI. While for a high-recall prediction, the rate that a POI is wrongly predicted as a non-POI is low, which can make sure that an actual POI is not excluded as a non-POI. Thus a high-recall prediction is more appropriate than a high-precision prediction in POI investigations. A high-recall prediction can make sure that no POIs escape from the investigations even if some non-POIs may accidentally be included in the investigations in case the precision is low.

High-recall predictions have many applications in business activities [15]. For instance, in giving discounts to the true customers with frequent buyer status, the high-recall predication can make sure that all the true customers get discounts when the true customers are treated as positives even if some people who don't have frequent buyer status may get the discounts in case the precision is low. High-precision predications are important in many security activities [15]. For instance, in security credential scanning, if authorized people are treated as positives, the high-precision predication can make sure that no unauthorized people get into a secured facility even

if some authorized credentials are rejected a couple of extra times and have to re-scan their credentials in case the recall is low.

From Figure 1 (where the precision and recall versus the number of features are plotted for different algorithms at tuned parameters) and Table 2 (where precision and recall are given for different algorithms with 14 features at tuned parameters), the optimal model that I end up use for this project is DecisionTree classifier with 14 features. The precision and recall are 0.51487 and 0.51075, respectively.

Discussion

Feature importance: Feature importance is a quantity used to explore the importance of a feature in some classifier, such as Decision Tree. Theoretically, the higher the importance of a feature, the more important the feature [7]. In Table 4 the feature importance of the 14 features are listed for DecisionTree classifier. It is shown that the trends of the feature importance and scores of features are not always consistent to each other. Some of the feature importance are zero. One of the very interesting results is that none of the feature importances of the new features are zero, demonstrating the new features have obvious effects on the metrics.

Table 4. Feature importances of DecisionTree classifier with 14 features

Index	Features	Feature importance
1	'exercised_stock_options'	0.21566
2	'loan_advances'	0.
3	'total_stock_value'	0.03189
4*	'final_total_stocks'	0.04072
5	'bonus'	0
6	'ratio_to_poi'	0.08705
7	'final_total_benefits'	0.20722
8	'salary'	0.
9	'final_total_payments'	0.05569
10	'total_payments'	0.
11	'long_term_incentive'	0.
12	'shared_receipt_with_poi'	0.08151
13	'other'	0.20212
14	'deferred_income'	0.07812.

* *italics represent new features*

Effects of new features: It is shown from Table 1 and Table 4 that there are 4 new features in the final model with 14 features. The number of new features account for 28.6% of the total number of features used. Do the new features really affect the results? What are the effects of the new features on the results? How do the new features impact the evaluation metrics? To explore the effects, the precision and recall are computed by using the DecisionTree classifier with different features. The results are given in Table 5, where OFs is a collection of the 10 original features which are the features in the final model listed in Table 4 apart from the 4 new ones, and Ps1 = {criterion='entropy', max_features='auto'}, Ps2 = {criterion='gini', max_features='auto'}, and Ps3 = {criterion='entropy', max_features=None } are tuned parameters in the models. The order of features in computation is the same as that in Table 4.

Table 5. Precisions and recalls obtained from models with different new features

Models	No. of new features	Features used	Tuned parameters	Precision	Recall
M1	0	F1 = OFs	Ps1	0.29645	0.28200
M2	1	F2 = F1 + 'final_total_stocks'	Ps2	0.26745	0.26725
M3	1	F3 = F1 + 'ratio_to_poi'	Ps3	0.40638	0.38200
M4	1	F4 = F1 + 'final_total_benefits'	Ps2	0.26792	0.27100
M5	1	F5 = F1 + 'final_total_payments'	Ps3	0.28504	0.27250
M6	2	F6 = F2 + 'ratio_to_poi'	Ps3	0.40474	0.38450
M7	2	F7 = F2 + 'final_total_benefits'	Ps3	0.27363	0.25550
M8	2	F8 = F2 + 'final_total_payments'	Ps2	0.28036	0.26200
M9	2	F9 = F3 + 'final_total_benefits'	Ps3	0.39519	0.37375
M10	2	F10 = F3 + 'final_total_payments'	Ps3	0.42419	0.38050
M11	2	F11 = F4 + 'final_total_payments'	Ps2	0.28282	0.28650
M12	3	F12 = F6 + 'final_total_benefits'	Ps2	0.39414	0.42400
M13	3	F13 = F6 + 'final_total_payments'	Ps1	0.40112	0.41225
M14	3	F14 = F7 + 'final_total_payments'	Ps2	0.29236	0.28125
M15	3	F15 = F9 + 'final_total_payments'	Ps1	0.39424	0.39000
M16	4	F16 = F12 + 'final_total_payments'	Ps1	0.51487	0.51075

It is shown that: (1) Both precision (=0.29645) and recall (=0.28200) are less than 0.3 when the model (M1) does not include new features; (2) neither precision nor recall are improved when the models only include finance-related new features (models with one new financial feature: M2, M4, and M5; models with two new financial features: M7, M8, and M11; and models with three new financial features: M14), manifesting that the finance-related new features themselves have little effect on the results; (3) both precision (=0.40638) and recall (=0.38200) increase greatly when the model (M3) includes one email-related new feature, demonstrating that the email-related new feature has large impact on the results; (4) both precision and recall are greatly improved when the models include one email-related new feature and any one (models: M6, M9, and M10) or two (models: M12, M13, and M15) finance-related new features, but the results are very closed to those from the model (M3) with only one email-related feature, illustrating that including one and two finance-related features in any models does not affect the results that much; and (5) both precision and recall reach their maximum values when the model includes all the 4 new features, demonstrating that combination of the email-related new feature and the 3 finance-related new features has the greatest influence on the results.

To conclude, (1) the email-related new feature has large effect on the metrics; (2) the finance-related new features and their combinations have little impact on the metrics; (3) the finance-related new features have very small influence on the metrics in the models with the email-related new feature and any one or two finance-related new features; and (4) the model with the 4 new features has the maximum effect on the metrics. Thus all the 4 new features and their combination are indispensable to achieve the maximum values of precision and recall. Therefore the new features introduced in this project play a significant role in the process of optimizing the metrics.

Summary and conclusion

In this project, a prediction model to identify “person of interest” (POI) is built based on emails and financial data in Enron scandal using machine learning skills. There are 146 data points, 20 features, and 1 label in original dataset. After removing 5 outliers and 1 email feature

('email_address'), and creating 5 new features, the dataset contains 143 data points, 24 features, and 1 label. This dataset is used in this project. The new created features play a significant role in the process of optimizing the metrics.

All the features are scaled to the range of [0, 1] by *MinMaxScaler* before reordering the features by using *SelectKBest*. The scaled and reordered dataset is then used to perform feature selection, parameter tuning, cross validation, algorithm optimization, and model evaluation together. The *GridSearchCV*, *StratifiedShuffleSplit*, and two evaluation metrics are applied to the parameter tuning, cross validation, and model evaluation, respectively. Five algorithms have been explored and finally the DecisionTree classifier is found to be the best. The 14 features are selected in the final model. The best values of the evaluation metrics obtained are precision = 0.51487 and recall = 0.51075.

The most challenging part of the project is the search and removal of outliers. And the most important steps are the feature selection together with parameter tuning. This project is tough also because the original data is imperfect and messy, and the data points are very limited. In the future, the following work may be considered to further improve the performance of the model.

1. Try to systematically optimize the model by combining parameter tuning of an algorithm with (recursive) feature selection [16] using *GridSearchCV* [12] together with pipeline [17]. For instance, try to chain a *PCA* [10] to a logistic regression with a pipeline [18].
2. Try to tune the parameters using more parameters and parameter grids.

I love this project and thoroughly enjoyed exploring the data using machine learning techniques!

Scripts and source codes

<i>algorithms_parameters_tune.py</i>	Parameter tuning and algorithm optimization
<i>dataset_exploration.py</i>	Exploration of dataset structures
<i>features_selections.py</i>	Feature scaling and reordering
<i>feature_format.py</i>	Tool for converting data to other forms
<i>outliers_treatment.py</i>	Outlier searching and removing
<i>poi_email_addresses.py</i>	Email address list of POIs
<i>poi_id.py</i>	Main code of POI classification models
<i>tester.py</i>	Script for importing student's identifier and checking results

References

- [1] <https://en.wikipedia.org/wiki/Enron>
- [2] Intro to Machine Learning, <https://www.udacity.com/course/viewer#!/c-ud120/l-2948958577/e-3052588588/m-3071078557>
- [3] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest
- [4] <https://msdn.microsoft.com/en-us/library/ms175382.aspx>
- [5] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- [6] <http://scikit-learn.org/stable/modules/svm.html>
- [7] <http://scikit-learn.org/stable/modules/tree.html>
- [8] <http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
- [9] http://scikit-learn.org/stable/modules/linear_model.html
- [10] <http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
- [11] http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html
- [12] http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html
- [13] The module `tester.py` provided in final project already includes a code for this.
- [14] https://en.wikipedia.org/wiki/Precision_and_recall
- [15] http://fch808.github.io/Identifying_Fraud_at_Enron.html
- [16] <http://scikit-learn.org/stable/modules/pipeline.html>
- [17] <http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>
- [18] http://scikit-learn.org/stable/auto_examples/plot_digits_pipe.html#example-plot-digits-pipe-py