Identifying Fraud from Enron Dataset

By Ying Li

Introduction (Purpose of Project)

The Enron fraud investigation made available a significant amount of financial and communication data to public record. Through the review of dataset, we can have a glimpse at the complexity in the well-known scandal.

The goal of this project is to apply machine learning techniques to identify the potential Person of Interest (POI) in the scandal. The given dataset includes the records of selected group of individuals. Each record includes financial data, such as salary, bonus and stocks of and the email communication data with the key personnel.

The individuals in the dataset have been labeled as POI or non-POI. The associated financial and communication records are to be selected and manipulated as features to be fed into machine learning algorithms for training of classifiers. The resulting classifier is then tested, cross-validated and evaluated.

Data Exploration

Valid Features

In the original dataset there are 146 records (person) of 21 items (features) each. Among the records, 18 (12%) are labeled as 'POI'. The data has been imported into a python dictionary. A first look at the dataset reveals there are many missing information marked as 'NaN', which mean not all records might be useful. First, a counter was run to single out the features contains at least 30% valid information, i.e. less than 30% of 'NaN'. In Table 1, these 17 features are highlighted with bold face letters.

Table 1 'NaN' ratio of Features

Feature Name	'NaN' ratio
salary	**34.9%**
to_messages	**41.1%**
deferral_payments	73.3%
total_payments	**14.4%**
loan_advances	97.3%
bonus	**43.8%**
email_address	**24.0%**
restricted_stock_deferred	87.7%
total_stock_value	**13.7%**
shared_receipt_with_poi	**41.1%**
long_term_incentive	**54.8%**
exercised_stock_options	**30.1%**
from_messages	**41.1%**
other	**36.3%**
from_poi_to_this_person	**41.1%**
from_this_person_to_poi	**41.1%**
poi	**0.0%**
deferred_income	**66.4%**
expenses	**34.9%**
restricted_stock	**24.7%**
director_fees	88.4%

From the table, we can summarize two categories of potentially useful features as in Table 2.

Table 2 Valid Features

Financial Features	Email Features
salary	to_messages
total_payments	email_address
bonus	shared_receipt_with_poi
total_stock_value	from_messages
exercised_stock_options	from_poi_to_this_person
other	from_this_person_to_poi
expenses	
long_term_incentive	
deferred_income	
restricted_stock	

Outliers

By plotting out the features in scatter plot, such as the two examples shown in Figure 1, an outlier was identified in the financial feature group, the outlier was caused by input error including the "TOTAL" line in the data sheet. This outlier has been removed using numpy pop function.

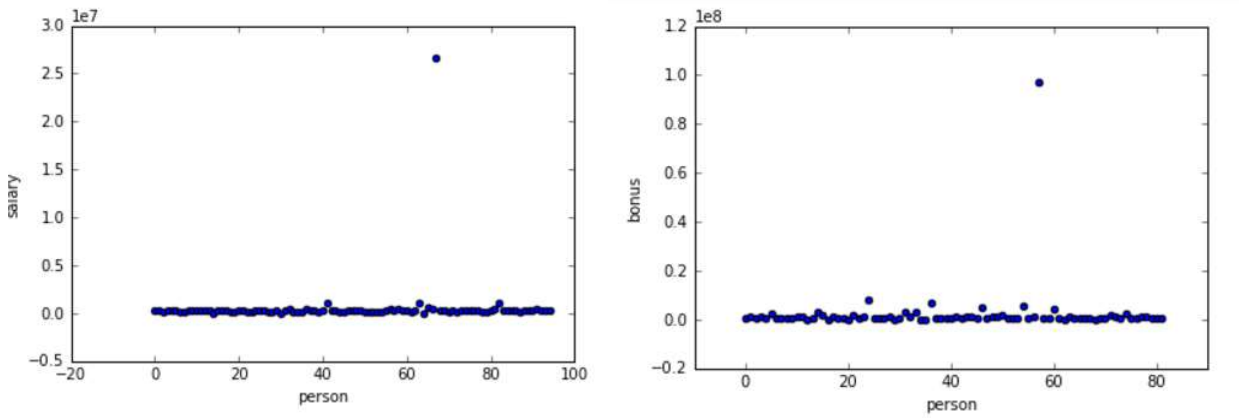

Figure 1 Identify Outlier

Additionally, the feature 'email_address' consists of email addresses of individuals, since in this project, the email texts are not used, this feature would not be useful for analysis, therefore it has been discarded as a whole.

Feature Selection and Manipulation

New Features

It is observed there can be a wide range for the number of emails each person sent or receive. Some people might be prone to communicate more than others. The number of emails to and from POI might not indicate the importance of POI communication. Therefore, three new features have been added to the dataset to rationalize the communication with POIs:

1. Ratio of received messages from POIs ('from_poi_fraction')

2. Ratio of sent messages to POIs ('to_poi_fraction')
3. Ratio of received message shared with POIs ('share_receipt_poi_fraction')

Feature Selection

Select K best has been used to obtain the scores of potential useful features identified. The results are shown in the Table 3.

Table 3 SelectKBest scores for Features

Financial Features	SelectKBest Score	Email Features	SelectKBest Score
bonus	20.79	to_messages	0.29
salary	18.29	from_messages	0.47
total_payments	8.77	from_this_person_to_poi	1.09
exercised_stock_options	24.82	shared_receipt_with_poi	4.62
total_stock_value	24.18	from_poi_to_this_person	2.43
expenses	6.09	from_poi_to_fraction	0.93
restricted_stock	9.21	to_poi_fraction	10.98
other	4.15	share_receipt_poi_fraction	2.79
long_term_incentive	10.07		
deferred_income	11.60		

In selecting the top 10 features with higher scores, 'from_poi_to_this person' is selected instead of 'shared_receipt_poi_fraction' even though the score of 'from_poi_to_this person' is slightly higher. This was because the 'shared_receipt_with_poi' is with much higher score and have been selected, and 'shared_receipt_poi_fraction' was then ignored to avoid overfitting with highly correlated feature.

Further exploration reveals a close-to-linear relationship between 'exercised_stock_options' and 'total_stock_value' as seen in Figure 2. This implies that PCA might be helpful in reducing the dimension of the features.

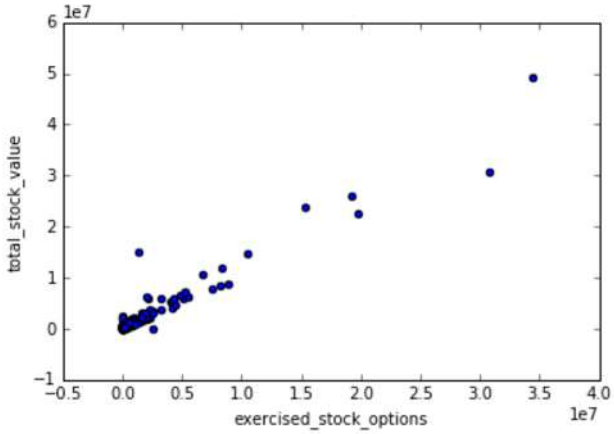

Figure 2 Relationship between 'total_stock_value' and 'exercised_stock_options'

In summary, the top 10 features have been selected (as shown as red text in Table 3) :

- Bonus
- Salary

- Exercised_stock_options
- Total_stock_value
- Restricted_stock
- Long_term_incentive
- Deferred_income
- Shared_receipt_with_poi
- To_poi_fraction
- From_poi_to_this_person

Feature Transformation

The following feature manipulation has been applied to the selected features:

1. Scaling
2. PCA transformation (n_components)

These two steps have been combine with classifier with *Pipeline* in sklearn. The parameter for PCA is included in the parameter tuning process. This was to avoid overfitting of excessive features.

Machine Learning Algorithms

Selection of Classifiers

There are two steps in defining a classifier: selecting an algorithm and selecting parameters. The following algorithms in the sklearn package have been applied with selected parameters shown in parentheses:

- **Gaussian Naïve Bayes**
- Supported Vector Machine (C, kernel)
- Adaboost (n_estimator)
- Multi-layer Perception (hidden_layers, solver)

Additionally, the parameter shown in Feature Transformation, n_components for PCA are also included in the parameter tuning. GridSearchCV has been used to obtain the optimal parameter settings for the highest F1 score.

All these classifiers have been optimized and tested. **Gaussian Naïve Bayes** was the one that was finally chosen to satisfy the requirement of precision and recall. The results will be presented in the Evaluation section of this report.

Importance of Parameter Tuning

The parameter tuning plays an important role in testing results. For example, in SVC classifier, different C value and kernel type can lead to different accuracy in testing, and even it can determine whether there are valid F1 score (Table 4).

Table 4 Different results from different SVC parameters (tested with tester.py)

C	kernel	Accuracy	F1
1.0 (default)	rbf (default)	0.86	Error
10.0	linear	0.85	0.15

Optimized Classifiers

The parameter tuning results from GridSearchCV are shown in Table 5.

Table 5 Classifiers with optimized parameters

CLASSIFIER TYPE	PCA PARAMETER	CLASSIFIER PARAMETERS	
GAUSSIAN NAÏVE BAYES	n_components=5	NA	
SUPPORTED VECTOR (SVC)	n_components=8	C=10.0	kernel='linear'
ADABOOST	n_components=2	n_estimators=30	
MULTILAYER PERCEPTTION (MLP)	n_components=10	hidden_layer_size=5	Solver='lbfgs'

Validation and Evaluation

Cross-Validation

Since the distribution of POI in the population is skewed (only 12% of POIs), it is a challenge to train a classifier with a limited number of data. Cross-validation is necessary to ensure the classifier would work for general population of data.

During the parameter optimization process using GridSearchCV, K-fold validation had been applied. Several values for the parameter 'cv' have been tested. Finally cv=10 had been chosen to produce the best performed classifier.

Once Gaussian Naïve Bayes Classifier has been defined as the final classifier (Table 6), K-fold validation has been applied to ensure the robustness of the classifier.

Table 6 K-fold validation results for Gaussian Naïve Bayes

K-fold	Overall Accuracy	F1-score of POI
Fold 1	0.77	0.33
Fold 2	0.83	0.40
Fold 3	0.91	0.57
Fold 4	0.85	0.29

Evaluation

The performances of different classifiers have been tested using the tester.py program provided by this project. The results are presented in Table 7.

Table 7 Testing Results for Different Classifiers

Classifier	Accuracy	Precision	Recall	F1-score
Gaussian Naïve Bayes	0.84	**0.44**	**0.39**	0.41
SVC	0.85	0.44	0.09	0.15
Adaboost	0.79	0.19	0.14	0.16
MLP	0.82	0.29	0.25	0.26

Discussion of Results

It is shown that Gaussian Naïve Bayes classifier provides the best F1 score in addition to satisfy the requirements of greater than 0.3 precision and recall. Gaussian Naïve Bayes is the winner in this project. For SVC, the precision is much higher than recall. Low recall means it is prone to miss identifying a POI (ignore criminals). If otherwise, the recall is much higher than precision, this implies it tends to mistaken a non-POI as POI (wrong an innocent person). In this project, MLP produces similar precision and recall, but either is good enough to meet the requirement of this project. Adaboost is the worst performed classifier. This might have been due to the decision-tree nature of the algorithm.

Concluding Remarks

Summary

In this project, a machine learning process has been applied in the Enron fraud dataset, which includes: Data Exploration, Feature Selection, Algorithm Selection and Evaluation. The key steps are:

1. Removing outlier and adding new features
2. Selecting top 10 features based on SelectKBest scores
3. Select and tuning 4 different types of classifier using F1 weighted GridSearchCV
4. Validate and evaluate classifiers with testing program

Lessons Learned

The most time-consuming parts in this process is the feature selection and parameter tuning. Several iterations between these two stages have been conducted. Initially, less features were selected due to the significant amount of 'NaN' in the data. However, if those features with more than 50% 'NaN's are discarded, none of the classifier was able to produce satisfactory results. In the end, the NaN threshold was raised to 70% and 10 features was selected to be fed into the classifier. The lesson learned from this project was that it is better to include as many features as possible at the beginning. PCA with tuning of paremeter 'n_components' can help get rid of the overfitting caused by excessive features.

Future Work

Due to the scope of this project, not all information has been utilized. Future work may include:

1. Apply text search across the massive email corpus.
2. Apply unsupervised machine learning techniques without using the POI labels in the main dataset.

Reference

- Sklearn documentation http://scikit-learn.org/
- https://en.wikipedia.org
- http://stackoverflow.com
- Udacity Forum