Enron POI Detector

1. Project Overview

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. Using the emails and financial data, this project aims to build a person of interest (POI) identifier which detect persons who were indicted, reached a settlement or plea deal with government, or testified in exchange for prosecution immunity.

2. References

- Enron email dataset:

  https://www.cs.cmu.edu/~./enron/enron_mail_20150507.tgz

- Enron insiderpay (financial data for top executives at Enron):

  enron61702insiderpay.pdf

- Udacity's Intro to Machine Learning Course

3. Resources from Udacity Course Material (preprocessing details)

- enron_dataset.pkl

  As a preprocessing to this project, the Enron email and financial data were combined into a dictionary (dumped into 'enron_dataset.pkl'), where each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name, and the value is another dictionary, which contains the names of all the features and their values for that person. The features are listed as following:

  ['salary', 'to_messages', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'email_address', 'restricted_stock_deferred', 'deferred_income',

'total_stock_value', 'expenses', 'from_poi_to_this_person', 'exercised_stock_options', 'from_messages', 'other', 'from_this_person_to_poi', 'poi', 'long_term_incentive', 'shared_receipt_with_poi', 'restricted_stock', 'directror_fees']

These fall into three major types of features, namely financial features, email features and POI labels.

financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is ‘email_address’, which is a text string)

POI label: ['poi'] (boolean, represented as integer)

- emails_by_address

  This directory contains many text files, each of which contains all the messages to or from a particular email address. It is for reference, if more advanced features need to be created, there is no need to process the email corpus.

4. Data Exploration

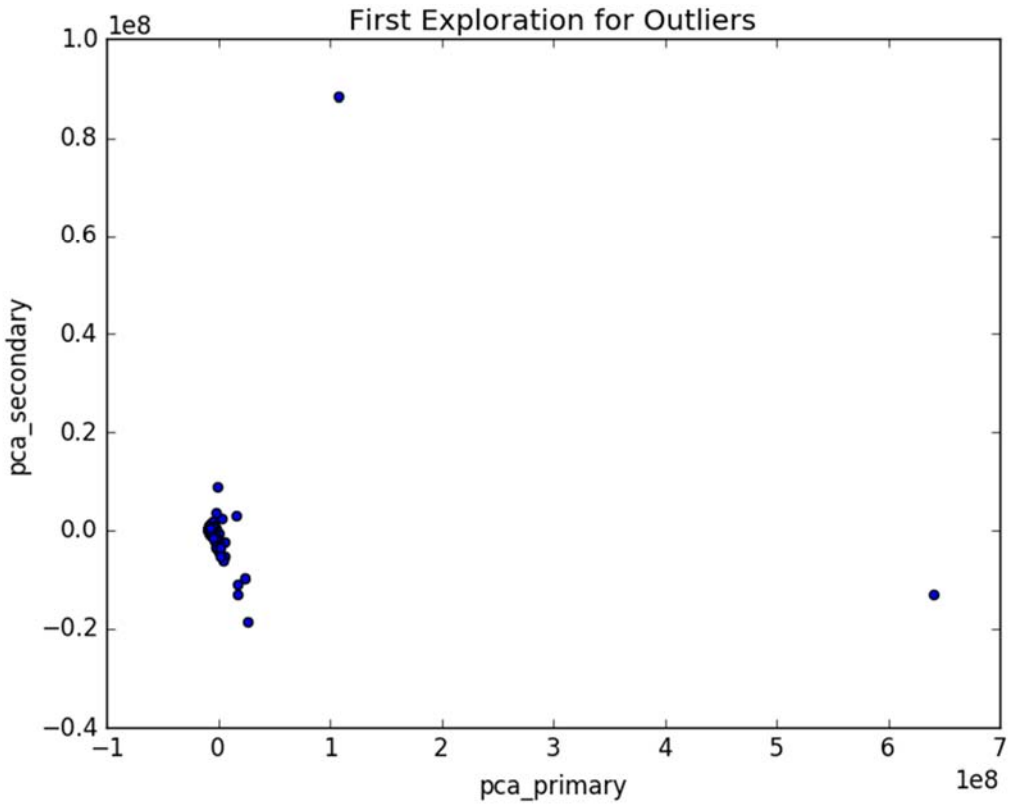
data_exploration.py was used to investigate the dataset and check for outliers for the financial data.

Here is an overview of the dataset from enron_dataset.pkl

```
Total number of persons in dataset: 146
Total number of POI in dataset: 18
Total number of features in dataset: 21
```

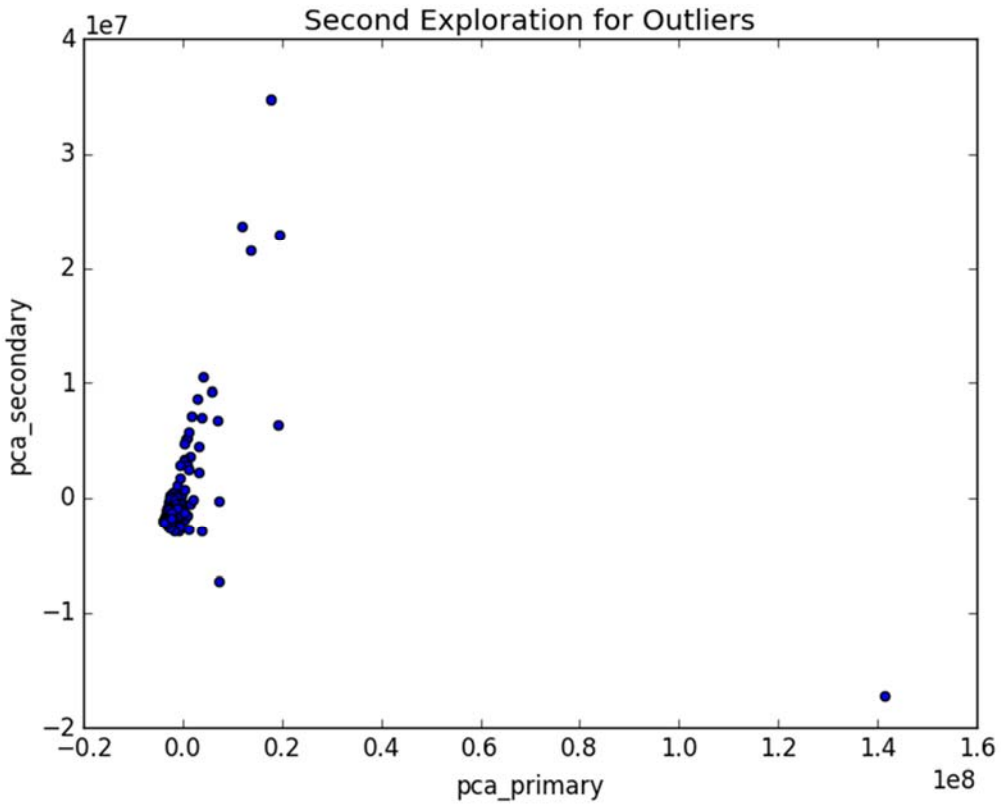
```
Number of missing values for each feature in dataset:
{'bonus': 64,
 'deferral_payments': 107,
 'deferred_income': 97,
 'director_fees': 129,
 'email_address': 35,
 'exercised_stock_options': 44,
 'expenses': 51,
 'from_messages': 60,
 'from_poi_to_this_person': 60,
 'from_this_person_to_poi': 60,
 'loan_advances': 142,
 'long_term_incentive': 80,
 'other': 53,
 'poi': 0,
 'restricted_stock': 36,
 'restricted_stock_deferred': 128,
 'salary': 51,
 'shared_receipt_with_poi': 60,
 'to_messages': 60,
 'total_payments': 21,
 'total_stock_value': 20}
```

To check for outliers in financial data, all financial features were extracted from the dataset. Using dimension reduction (PCA), the multidimensional financial features were projected to two dimensions and the mapping was visualized (saved to 'figure_1.png').



This visualization clearly indicate two outliers. A further exploration of the outliers revealed one is from "TOTAL" and the other is from financial data of "LAY KENNETH L". The formal outlier was removed as it was not a valid data point, whereas the latter was kept because it is a valid input, and it is actually coming from a "POI"

After removing the invalid data point, the outlier checking process was repeated (extract all financial features and apply PCA to project to two dimensions) and several more outliers were observed (saved to 'figure_2.png'), the new outliers were all valid financial data from 'HIRKO JOSEPH', 'RICE KENNETH D', 'SKILLING JEFFREY K' and 'PAI LOU L'. Therefore they are kept without change.



The modified dataset were dumped into ‘enron_dataset_modify.pkl’

5. Feature Selection

feature_selection.py was used to select features used for POI identifier.

Before conducting feature selection, the financial data (from 'enron_dataset_modify.pkl') was split into training set (80%) and testing set (20%) (saved as 'enron_dataset_train.pkl' and 'enron_dataset_test.pkl'.) using StratifiedShuffleSplit from sklearn, which help to maintain percentage of samples for each class (considering the highly skewed distribution of the two classes in Enron dataset). All the feature selection and subsequent algorithm selection and cross validation were performed on training set. The testing set was hold out and was used to test the selected classifier and features at the last step.

First, irrelevant feature "email_address" was unselected. Features which have more than 50% of missing values were removed. After the two steps, there are 13 features left over.

Second, an ANOVA test was used to rank the 13 features left over. The F-score and P-value were listed below. From the result, for email features, other than "shared_receipt_with_poi", most of them have very small F-score, meaning they are irrelevant to poi. Considering the "from_this_person_to_poi" and "from_poi_to_this_person" didn't take all "from_messages" and "to_messages" into account, two new features "fraction_from_this_person_to_poi" and "fraction_from_poi_to_this_person" were added and saved to 'enron_dataset_train_modify.pkl'.

First Time ANOVA Test:

feature name	F_score	P_value
from_messages	0.040	0.842
to_messages	2.671	0.105
from_this_person_to_poi	3.550	0.062
other	4.679	0.033
from_poi_to_this_person	6.719	0.011
expenses	9.564	0.003
total_payments	10.016	0.002
shared_receipt_with_poi	10.910	0.001
salary	18.214	0.000
restricted_stock	20.800	0.000
bonus	21.916	0.000
exercised_stock_options	23.257	0.000
total_stock_value	26.598	0.000

After addition of the two new features, ANOVA test was conducted again. The results were listed below. Clearly, the two new added features still rank pretty low.

Second Time ANOVA Test:

feature name	F_score	P_value
from_messages	0.040	0.842
fraction_from_this_person_to_poi	1.942	0.166
to_messages	2.671	0.105
from_this_person_to_poi	3.550	0.062
other	4.679	0.033
fraction_from_poi_to_this_person	5.248	0.024
from_poi_to_this_person	6.719	0.011
expenses	9.564	0.003
total_payments	10.016	0.002
shared_receipt_with_poi	10.910	0.001
salary	18.214	0.000
restricted_stock	20.800	0.000
bonus	21.916	0.000
exercised_stock_options	23.257	0.000
total_stock_value	26.598	0.000

Based on the above ANOVA tests, features which have P level below 1% were selected as candidates for further algorithm selection.

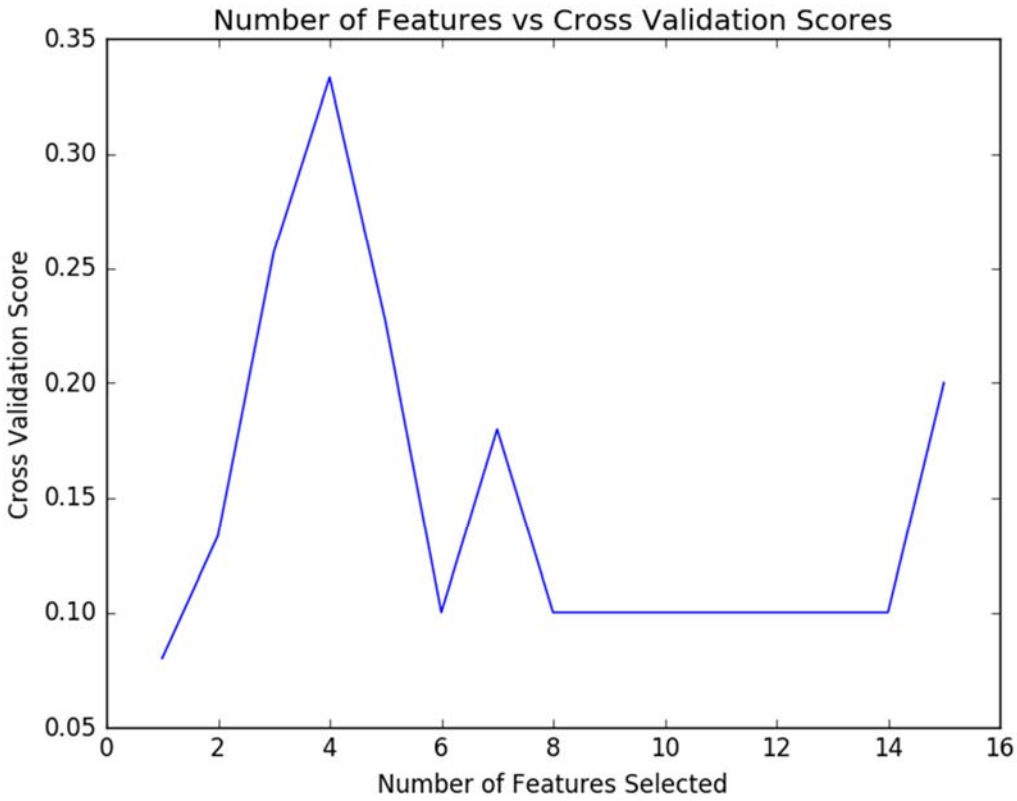
sf1 = "poi", "total_stock_value", "exercised_stock_options", "bonus", "restricted_stock", "salary", "shared_receipt_with_poi", "total_payments", "expenses"

Nevertheless, ANOVA test considers all features independent, while in reality, features could have interactions with each other. Therefore, recursive feature elimination (RFE) was used to do feature selection. Specifically, random forest classifier was used to do the selection since it is robust (not necessarily to be the final algorithm). RFE was used with cross validation to rank features and select the optimized number of features. The f1 score was used as scoring strategy in cross validation. The RFECV was also nested with GridSearchCV to optimize some of the parameters in random forest classifier. The feature

ranking was shown below. The cross validation score vs number of feature selected was also shown below (saved to 'figure_3.png').

Feature Ranking from RFECV nested with GridSearchCV:

bonus	1
exercised_stock_options	1
expenses	1
from_poi_to_this_person	1
total_stock_value	2
fraction_from_poi_to_this_person	3
salary	4
other	5
restricted_stock	6
total_payments	7
shared_receipt_with_poi	8
from_messages	9
to_messages	10
from_this_person_to_poi	11
fraction_from_this_person_to_poi	12



The optimized number of features is 4, they are "bonus", "exercised_stock_options", "expenses" and "from_poi_to_this_person". Among the four features, the first three also have high F-score in ANOVA test, whereas the last one didn't rank high in ANOVA test. This suggest an interaction between "from_poi_to_this_person" and other features, leading to an increased importance. Based on the RFECV reault, the first 4 features were selected for further test.

sf2 = "poi", "bonus", "from_poi_to_this_person", "expenses", "exercised_stock_options"

Also, another combination of features were selected by combining sf2 and the top features in sf1 ("total_stock_value", "exercised_stock_options", "bonus"). (several different combinations were tested, this one proven to be most effecient.)

sf3 = "poi", "bonus", "from_poi_to_this_person", "expenses", "exercised_stock_options", "total_stock_value"

The three different combinations of features (sf1, sf2, sf3) were saved to 'initial_three_combinations_of_features.pkl' for further testing on algorithm.

6. Algorithm Selection

algorithm_selection.py was used to do algorithm selection.

First, several algorithms were selected for classification, including GaussianNB, DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, KNeighborsClassifier and SVC from sklearn. Each combination of the classifiers and selected features (sf1, sf2 and sf3) were evaluated using cross validation (the cross validation function method ('estimator_cv') could be find in 'helper_functions.py').

Specifically, in cross validation, 100 splitting iterations were used (n_iter=100). In each iteration, the training data was further split into sub-training set (90%) and sub-validation set (10%) using StratifiedShuffleSplit from sklearn. The classifier train on the sub-training set and predict on the sub-validation set. The False-Positive (FP), False-Negative (FN), True-Positive (TP) and True-Negative were counted (TN). After all 100 splitting iterations, the total numbers of FP, FN, TP, and TN were used to calculate accuracy, precision, recall, f1, and f2.

Performance for each combination of features and classifiers (default parameters were used. In case of knn and svc, feature scaling was conducted before use) were shown below.

Screening Different Classifiers:

classifier	feature	accuracy	precision	recall	f1	f2
RandomForestClassifier()	sf2	0.894	0.404	0.360	0.381	0.368
RandomForestClassifier()	sf3	0.840	0.545	0.240	0.333	0.270
RandomForestClassifier()	sf1	0.883	0.222	0.160	0.186	0.169
GaussianNB()	sf2	0.875	0.317	0.320	0.318	0.319

GaussianNB()	sf3	0.848	0.578	0.315	0.408	0.347
GaussianNB()	sf1	0.892	0.341	0.310	0.325	0.316
SVC()	sf2	0.909	0.000	0.000	0.000	0.000
SVC()	sf3	0.834	1.000	0.005	0.010	0.006
SVC()	sf1	0.917	0.000	0.000	0.000	0.000
AdaBoostClassifier()	sf2	0.897	0.434	0.430	0.432	0.431
AdaBoostClassifier()	sf3	0.852	0.602	0.325	0.422	0.358
AdaBoostClassifier()	sf1	0.849	0.114	0.120	0.117	0.119
KNeighborsClassifier()	sf2	0.884	0.274	0.170	0.210	0.184
KNeighborsClassifier()	sf3	0.807	0.288	0.105	0.154	0.120
KNeighborsClassifier()	sf1	0.892	0.032	0.010	0.015	0.012
DecisionTreeClassifier()	sf2	0.868	0.331	0.440	0.378	0.413
DecisionTreeClassifier()	sf3	0.843	0.534	0.465	0.497	0.477
DecisionTreeClassifier()	sf1	0.858	0.246	0.340	0.286	0.316

Note that both f1 and f2 were calculated, with f1 measure the balance between precision and recall and f2 weighs recall higher than precision. In this specific POI identifier, we would like to weigh recall higher than precision, because we want to be able to identify all real POIs, even if the sacrifice is that we might identify some innocent people as POI (because we will always further check on them to see if they committed crime). The bottom line is that we don't want to miss any real POIs as innocent people. That's why we care about f2 score.

Several conclusions could be drawn from the above classifier screening. (1) sf2 and sf3 perform better than sf1 in almost all cases. sf3 is slightly better than sf2 in many cases. (2)

GaussianNB, AdaBoostClassifier and DecisionTreeClassifier perform better than other three classifiers in that they give all scores better than 0.3. (3) SVC and KNN are inferior to others, giving low precision, recall, f1 and f2 in most cases. Although the best precision was obtained from combination of SVC and sf3 (precision=1), the recall is pretty low (recall=0.005), meaning FP=0 (precision=TP/(TP+FP)=1), but FN is much larger than TP (recall=TP/(TP+FN)=0.005, FN=199TP). In another word, in this algorithm, almost all real POI were identified as non-POI. It is not a effective algorithm at all. (4) it could be noted that in all cases, the accuracy is high (larger than 0.8), that doesn't mean the algorithm is good, because accuracy is not a very effective measure for skewed classification (much smaller number of POI than non-POI).

Based on the result, AdaBoostClassifier was chosen for further tuning. AdaBoost is an ensemble method works by fitting a sequence of weak learners on repeatedly modified versions of the data. While random forest works well for complex systems to reduce variance, AdaBoost usually works well on weak systems to reduce bias. Considering the small amount of samples in Enron dataset and the highly skewed class, AdaBoost seems like a good choice. Also, AdaBoost performs well in initial screening of algorithms.

The parameters tuned in AdaBoostClassifier are 'base_estimator' and 'n_estimators'. 'n_estimators' with value 30, 50, 80 and 100 were evaluated. 'base_estimator' with value 'None' and 'DecisionTreeClassifier' were evaluated, since 'DecisionTreeClassifier' itself gives good result. Furthermore, "min_samples_split" and "max_features" in 'DecisionTreeClassifier' were evaluated with value 2, 5, 8, and 0.5, 0.8, 1.0, respectively. The evaluation method used is still the cross validation method talked above. Classifiers which give the best accuracy, best precision, best recall, best f1 and best f2 were obtained as below.

AdaBoost Tuning:

Note: if 'min_samples_split' and 'max_featues' are None, base_estimator = None.

Else, base_estimator = DecisionTreeClassifier(min_samples_split = min_samples_split,

max_features = max_features)

	n_estimator	min_samples_split	max_features	accuracy	precision	recall	f1	f2
best_accuracy	80	None	None	0.854	0.607	0.355	0.448	0.387
best_precision	80	None	None	0.854	0.607	0.355	0.448	0.387
best_recall	100	2	0.8	0.846	0.541	0.495	0.517	0.504
best_f1	100	2	0.8	0.846	0.541	0.495	0.517	0.504
best_f2	100	2	0.8	0.846	0.541	0.495	0.517	0.504

From the result above, two classifiers were selected. One gives best accuracy and precision, using base_estimator=None, n_estimators = 80 (CLF1).The other classifier gives best recall, f1 and f2, using base_estimator = DecisionTreeClassifier(min_samples_split = 2, max_features = 0.8), n_estimators = 100 (CLF2).

PCA were also used on sf3 and CLF2 to check via cross validation to see if better result could be obtained. Number of components from 1 to 5 were tested. The results were list below. Compared with CLF2 without PCA, inferior results were obtianed.

PCA Analysis:

CLF2: AdaBoostClassifier(algorithm='SAMME.R', base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=0.8, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=42, splitter='best'), learning_rate=1.0, n_estimators=100, random_state=42)

n_components	accuracy	precision	recall	f1	f2
1	0.829	0.484	0.385	0.429	0.401
2	0.828	0.479	0.400	0.436	0.414
3	0.805	0.384	0.280	0.324	0.296
4	0.818	0.446	0.370	0.404	0.383
5	0.809	0.408	0.320	0.359	0.334

After all the tests, two best classifiers were selected, they are CLF1 and CLF2 from AdaBoost tuning, the features used were sf3. To make it clear, the features used, classifier parameters and cross validation scores were listed below for one more time. The features and classifiers were saved in 'final_clfs_features.pkl'

sf3:

"poi", "bonus", "from_poi_to_this_person", "expenses", "exercised_stock_options", "total_stock_value" ("poi" is the label)

CLF1:

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=80, random_state=None)

CLF2:

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=0.8, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=42, splitter='best'), learning_rate=1.0, n_estimators=100, random_state=42)

classifier	accuracy	precision	recall	f1	f2
CLF1	0.854	0.607	0.355	0.448	0.387
CLF2	0.846	0.541	0.495	0.517	0.504

7. Testing

In order to test the selected feature (sf3) and classifiers (CLF1, CLF2), the whole training set was trained by CLF1 or CLF2. After conducting predictions on the testing set, accuracy, precision, recall and f1 were obtained. The scores were shown below. (code is in final_test.py)

sf3:

"poi", "bonus", "from_poi_to_this_person", "expenses", "exercised_stock_options", "total_stock_value" ("poi" is the label)

CLF1:

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=80, random_state=None)

CLF2:

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=0.8, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=42, splitter='best'), learning_rate=1.0, n_estimators=100, random_state=42)

Results on testing set:

clf	accuracy	precision	recall	f1
CLF1	0.769	0.333	0.500	0.400
CLF2	0.731	0.286	0.500	0.364

Compared with cross validation score, the scores for testing set are all slightly lower. This is expected, since for a good fit, testing error would be slightly higher than the training error.

8. Conclusion

In conclusion, sf3 and CLF1/CLF2 were used as features and classifiers respectively for POI detection. CLF1 is slightly better than CLF2, with accuracy equals 0.769, precision equals 0.333, recall equals 0.5 and f1 equals 0.4. This means using this POI identifier, if a person is real POI, there are only 50% chance that this identifier will recognize him or her. And if this identifier recognize a person to be POI, there are only 33% chance that this person is really a POI. Although the accuracy is not too bad (77%), this doesn't mean much for a highly skewed classification (much less POI compared with non-POI).

The low efficiency are mostly due to limited samples and highly skewed class. There are about 140 samples with only less than 15% of people being POI, with this small amount of data, it tend to underfit. The train-test splitting is also challenging because there is a tradeoff

between training error and testing error. Furthermore, the financial data is not completed, with a large amount of missing values. Last but not least, this dataset is not likely to be representative for all Enron employees. Since this dataset comes from top executives in Enron, their financial status would be very different from regular Enron employees, and it is very likely that the percentage of POI from top executives would be much higher than the percentage of POI from regular employees.