Identify Fraud from Enron Email
By Dogan Askan

Part 1 – Overview and Understanding the Dataset

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, there was a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for to executives.

The goal of the project is to construct a model to successfully identify the person of interests in the Enron dataset. There are 20 features in the dataset. Those are;

- 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)

There are also;
Total Number of Data Points: 146
POI: 18 Non-POI: 128
51 missing values in salary
107 missing values in deferral_payments
21 missing values in total_payments
142 missing values in loan_advances
64 missing values in bonus
128 missing values in restricted_stock_deferred
97 missing values in deferred_income
20 missing values in total_stock_value
51 missing values in expenses
44 missing values in exercised_stock_options
53 missing values in other
80 missing values in long_term_incentive
36 missing values in restricted_stock
129 missing values in director_fees
60 missing values in to_messages
60 missing values in from_poi_to_this_person
60 missing values in from_messages
60 missing values in from_this_person_to_poi
60 missing values in shared_receipt_with_poi
1323 total missing values in dataset for selected 19 features

To define the outliers, I used $mean \pm x * SD$ method. So I kept the values in 3 standard deviation range. I also tried 1, 1.5, 2, 2.5, 3.5, 4 and 4.5 as coefficient, I reached the best result with 3. Some outliers are below,

TOTAL
LAY KENNETH L
BHATNAGAR SANJAY
KITCHEN LOUISE
SHAPIRO RICHARD S
BELDEN TIMOTHY N
KEAN STEVEN J
LAVORATO JOHN J
DIETRICH JANET R
KAMINSKI WINCENTY J

However, it looks we should only remove the 'TOTAL" since others are real person and some are poi.

Part 2 – Features

I created two new features; 'from_poi_ratio' by dividing messages form poi to incoming messages and 'to_poi_ratio' by dividing messages to poi to outgoing messages. Because I thought more interactions with poi's indicate higher probability of being a poi. Then I chose 3 best features by using 'SelectKBest' algorithm, I also tried to choose 2, 4, 5, 6 and more but I reached the best results with 3. Some of the outputs are below.

The result with 3 best, **this is the best result**;

```
Accuracy: 0.82200       Precision: 0.41990      Recall: 0.41150 F1: 0.41566     F2: 0.41315
Total predictions: 13000        True positives:  823    False positives: 1137   False negatives: 1177   True negatives: 9863
```

The result with 4 best;

```
Accuracy: 0.78238       Precision: 0.28645      Recall: 0.27800 F1: 0.28216     F2: 0.27965
Total predictions: 13000        True positives:  556    False positives: 1385   False negatives: 1444   True negatives: 9615
```

The result with 5 best, 'from_poi_ratio' is included;

```
Accuracy: 0.78814       Precision: 0.27451      Recall: 0.29400 F1: 0.28392     F2: 0.28988
Total predictions: 14000        True positives:  588    False positives: 1554   False negatives: 1412   True negatives: 10446
```

The result with 6 best, 'from_poi_ratio' is included;

```
Accuracy: 0.78679       Precision: 0.23705      Recall: 0.22200 F1: 0.22928     F2: 0.22486
Total predictions: 14000        True positives:  444    False positives: 1429   False negatives: 1556   True negatives: 10571
```

Results with Engineered Features

The result with 3 best plus 'from_poi_ratio' and 'to_poi_ratio';

```
Accuracy: 0.80177       Precision: 0.35075      Recall: 0.33900 F1: 0.34477     F2: 0.34129
Total predictions: 13000        True positives:  678    False positives: 1255   False negatives: 1322   True negatives: 9745
```

The result with 3 best plus 'from_poi_ratio';

```
Accuracy: 0.79308       Precision: 0.33022      Recall: 0.33550 F1: 0.33284     F2: 0.33443
Total predictions: 13000        True positives:  671    False positives: 1361   False negatives: 1329   True negatives: 9639
```

The result with 3 best plus 'to_poi_ratio';

```
Accuracy: 0.80423       Precision: 0.35815      Recall: 0.34400 F1: 0.35093     F2: 0.34674
Total predictions: 13000        True positives:  688    False positives: 1233   False negatives: 1312   True negatives: 9767
```

It looks both new features lower the performance in both precision and recall wise. So there is no need to add the new features into our final features list. At the end, I reached the best performance by using features ['bonus' 'total_stock_value' 'exercised_stock_options']. Ranks of all numerical features are below;

---FEATURE SELECTION---
Feature Ranking:
1. feature exercised_stock_options (25.0975415287)
2. feature total_stock_value (24.4676540475)
3. feature bonus (21.0600017075)
4. feature salary (18.575703268)
5. feature from_poi_ratio (16.6417070705)
6. feature deferred_income (11.5955476597)
7. feature long_term_incentive (10.0724545294)
8. feature restricted_stock (9.34670079105)
9. feature total_payments (8.86672153711)
10. feature shared_receipt_with_poi (8.74648553213)
11. feature loan_advances (7.24273039654)
12. feature expenses (6.23420114051)
13. feature from_poi_to_this_person (5.34494152315)
14. feature other (4.2049708583)
15. feature to_poi_ratio (3.21076191697)
16. feature from_this_person_to_poi (2.42650812724)
17. feature director_fees (2.10765594328)
18. feature to_messages (1.69882434858)
19. feature deferral_payments (0.21705893034)
20. feature from_messages (0.164164498234)
21. feature restricted_stock_deferred (0.0649843117237)

Since the scales of features scales vary highly then I performed feature scaling. For example;

long_term_incentive	from_poi_to_this_person
max: 48521928.0	max: 528.0
min: 0.0	min: 0.0
range: 48521928.0	range: 528.0
SD: 4032191.77161	SD: 73.6476032167
mean: 664683.945205	mean: 38.2260273973

In some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. I tried MinMaxScaler, MaxAbsScaler and Normalizer algorithms in my pipeline. I reached the best result with MinMaxScaler. I also used PCA in the pipeline with decision tree classifier. PCA also improved the output, so the feature importances of the final components are below. Note that, the importances constantly change due to randomness but the ratio is approximate.

---FEATURE IMPORTANCES---
Feature Ranking:
1. component (0.569566634083)
2. component (0.232472876917)
3. component (0.197960489)

Part 3 – Algorithm

I tried several algorithm with their default settings at first. Outputs are;

RandomForestClassifier()

```
Accuracy: 0.86000       Precision: 0.59036      Recall: 0.29400 F1: 0.39252     F2: 0.32681
Total predictions: 13000        True positives:  588    False positives:  408   False negatives: 1412   True negatives: 10592
```

SVC()

```
Precision or recall may be undefined due to a lack of true positive predicitons.
```

tree.DecisionTreeClassifier()

```
Accuracy: 0.80385       Precision: 0.36792      Recall: 0.38300 F1: 0.37531     F2: 0.37988
Total predictions: 13000        True positives:  766    False positives: 1316   False negatives: 1234   True negatives: 9684
```

Since I reached the best F1 score with valid precision and recall with Decision Tree Classifier, I chose it to continue.

Part 4 – Tune the Algorithm

In addition to features and labels, most of the algorithms have some parameters directly affect the algorithm performance. For example, tree number for Random Forest, split or leaf numbers for both Decision Tree and Random Forest, C or Kernel type for Support Vector Machine. So changing these parameters to reach the optimal result is called tuning the parameters of an algorithm. Since those are directly related to metrics such as accuracy, f1, wrong choices in these parameters may drastically lower the performance. In my case, I tried several parameters with great range of values and I reached the best performance as below. I have used both GridSearchCV() and manually picking technique since GridSearchCV() greatly lowers the running time of tester.py. In addition to GridSearchCV trials, I manually tried 350 combinations. Some of the parameter combinations I tried can be seen below with their results as ordered by 'f1',

criterion	max_depth	min_samples_split	min_samples_leaf	recall	precision	f1
gini	100	3	1	0.371	0.387872452	0.379248658
gini	20	3	1	0.372	0.385492228	0.378625954
gini	20	2	1	0.3915	0.365546218	0.378078223
entropy	10	3	1	0.376	0.37979798	0.377889447
entropy	20	2	1	0.391	0.365079365	0.377595365
entropy	20	3	1	0.3755	0.378910192	0.377197388
entropy	50	3	1	0.373	0.381390593	0.377148635
entropy	100	1	1	0.393	0.362378976	0.377068841
entropy		3	1	0.3745	0.379240506	0.376855346
gini	20	1	1	0.388	0.36552049	0.376424933
entropy	100	3	1	0.375	0.377833753	0.376411543
gini	10	3	1	0.368	0.383933229	0.375797804
entropy		2	1	0.387	0.363892807	0.375090865
gini		2	1	0.387	0.363721805	0.375
gini		3	1	0.3685	0.381469979	0.374872838
entropy		1	1	0.39	0.360277136	0.37454982
gini	10	1	1	0.3875	0.361642557	0.37412503
gini	100	2	1	0.3855	0.362652869	0.373727581
entropy	50	2	1	0.3885	0.359888837	0.373647511
gini	100	1	1	0.3835	0.364197531	0.37359961

Part 5 – Validation

In machine learning, the dataset is split to three in general to train, validate and test the algorithm performance. This is called validation.

- Train set: Here, we train the algorithm and try to understand the best approach or approaches.
- Validation set: Here, we validate the algorithm (i.e. tune the algorithm).
- Test set: Here, we test the performance our algorithm.

If we do not use any validation technique and we just the entire dataset for both train and test, we basically end up with overfitting and the result in actual data would be very bad.

In my analysis, I spilt the data as train set (70%) and test set (30%).

Part 6 – Evaluation

There many metrics such as 'accuracy', 'recall', 'precision' and 'F1' to quantify the model performance in machine learning. For example, 'accuracy' is basically the ratio of true predictions over all predictions. However, 'accuracy' is not a good metric in our case because one type of labels is very very few. There are 18 poi and 128 non-poi in our initial dataset. If we would use 'accuracy' as a performance metric we easily reach 0.88 'accuracy' rate just by flagging every data points as non-poi. So F1 score is definitely a better metric in the cases like this.

$$F_1\ Score = 2\frac{PR}{P+R}$$

Where,

P is for Precision
How often does the algorithm cause a false alarm?
=TP/(TP + FP)
High precision is good

R is for Recall
How sensitive is the algorithm?
=TP/(TP + FN)
High recall is good

Where,

TP: True positive (we guessed 1, it was 1)
FP: False positive (we guessed 1, it was 0)
TN: True negative (we guessed 0, it was 0)
FN: False negative (we guessed 0, it was 1)

In consequence, in addition to my final algorithm I added a scaler (MinMaxScaler) and a decomposition tool (PCA) into my pipeline. And, I got the below final result.

```
Accuracy: 0.81923       Precision: 0.40602      Recall: 0.37800 F1: 0.39151     F2: 0.38329
Total predictions: 13000        True positives:  756    False positives: 1106   False negatives: 1244   True negatives: 9894
```

In other words, the final algorithm identifies poi's when it sees one with the accuracy 37.8% which is recall. And, when the final algorithm flags an instance as a poi it is 40.6% correct which is precision.