Identify Fraud from Enron Email

Cora Ou

Table of Contents

1. Understanding Dataset and Question

This project uses machine learning tools to identify Enron employees (Persons of Interest -POI) who have committed fraud. Machine learning tools help us select the best features that can predict POI. We then apply those features to algorithms, tune those algorithms and compare their performances by evaluation metrics. Hopefully, by predicting POI, the best algorithm can help avoid collapses of companies due to fraud.

Data exploration:

General description:

The dataset of this project is a nested dictionary containing Enron email and financial data. 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 in the data fall into three major types, namely financial features, email features and POI labels.

The important characteristics of the data:

The dataset has information of 146 users. Financial features include 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 and director fees. Email features include to messages, email address, from poi to this person, from messages, from this person to poi, shared receipt with poi. Among those features, all of them are numeric except email address. As email address is useless information, it is excluded in this project.

```
In [55]: print enrons.isnull().sum()

salary                       51
to_messages                  60
deferral_payments           107
total_payments               21
exercised_stock_options      44
bonus                        64
restricted_stock             36
shared_receipt_with_poi      60
restricted_stock_deferred   128
total_stock_value            20
expenses                     51
loan_advances               142
from_messages                60
other                        53
from_this_person_to_poi      60
poi                           0
director_fees               129
deferred_income              97
long_term_incentive          80
from_poi_to_this_person      60
dtype: int64
```

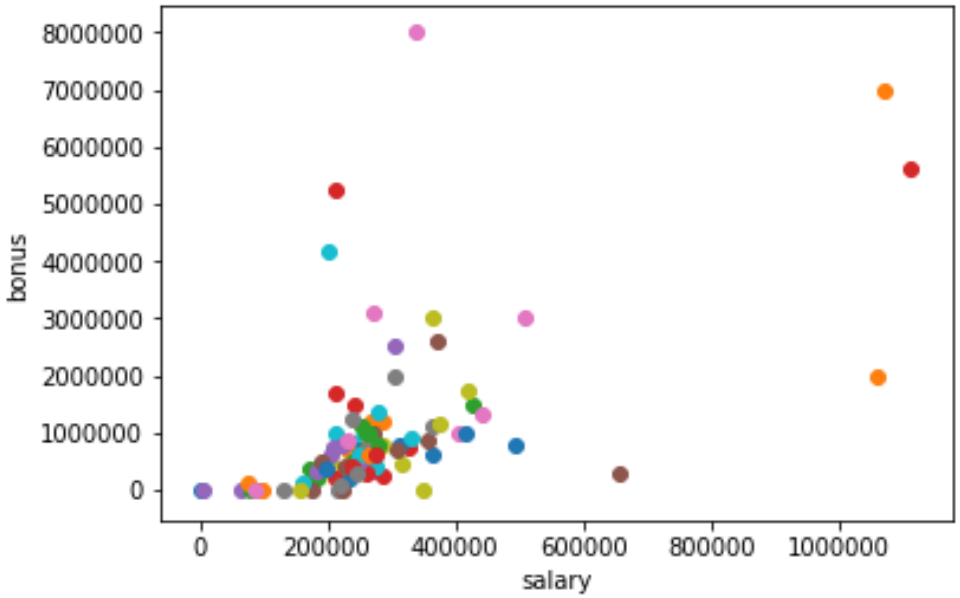
Dealing with missing values:

All of the features have missing values, except POI. To increase credibility, this project exclude features whose missing values are over 70. Also, the missing values of features that are included in the project are filled in with 0.

As for the allocation across POI, among 146 users, 18 of them are considered POI, while the rest are not POI.

Outlier Investigation:

When I plotted bonus vs. salary, there was an outlier representing the "TOTAL", "THE TRAVEL AGENCY IN THE PARK" and "LOCKHART EUGENE E"columns. I removed them.



Here are the codes:

```
my_dataset.pop("TOTAL", 0 )
my_dataset.pop("THE TRAVEL AGENCY IN THE PARK", 0 )
my_dataset.pop("LOCKHART EUGENE E", 0 )
```

2. Optimize Feature Selection and Engineering

Create new features:

I created a new feature called “with_poi_proportion” that represents the ratio of messages that this person sent and received against the total number of messages that he or she sent and received. I think that this new feature can help predicting POI as it shows the intimacy of this person with POI.

```
for en_row in my_dataset:
    from_this_person_to_poi = my_dataset[en_row]['from_this_person_to_poi']
    from_poi_to_this_person = my_dataset[en_row]['from_poi_to_this_person']
    from_messages = my_dataset[en_row]['from_messages']
    to_messages = my_dataset[en_row]['to_messages']
    if (from_messages != 0) and (to_messages != 0):
        my_dataset[en_row]['with_poi_proportion'] = float(from_this_person_to_poi + from_poi_to_this_person)/(from_messages + to_messages)
    else:
        my_dataset[en_row]['with_poi_proportion'] = 0
```

Intelligently select features:

After creating my new feature, in order to observe the effect of it, I looked at the

performance of each algorithm with the original best features (excluding new features) and compared it with the performance while adding the new features. Also using SelectKBest, I compared the performances of different algorithms when different numbers of features are used, hoping to find out the best K. I eventually took the first 8 features.

Scores of different features:

	Features	features-scores	p-scores
5	exercised_stock_options	25.097542	5.797364
3	total_stock_value	24.467654	5.676582
2	bonus	21.060002	5.013117
0	salary	18.575703	4.518014
7	restricted_stock	9.346701	2.573495
1	total_payments	8.866722	2.466492
13	shared_receipt_with_poi	8.746486	2.439567
4	expenses	6.234201	1.864131
8	with_poi_proportion	5.518506	1.694767
10	from_poi_to_this_person	5.344942	1.653242
6	other	4.204971	1.375257
12	from_this_person_to_poi	2.426508	0.915337
9	to_messages	1.698824	0.710966
11	from_messages	0.164164	0.163701

My final feature list : ['salary','total_payments','bonus','total_stock_value','exercised_stock_options', 'restricted_stock', ‘expenses’, 'shared_receipt_with_poi']

Properly scale features:

As some of the features have very big values, I need to transform my data. I searched the Internet, discovering two other algorithms---StandardScaler and RobustScaler. But as I didn’t know the distribution of my data and there were not many outliers in my data, I chose MinMaxScaler, the one I learned in the course. And I used Pipeline to combine the feature scaling and model training process.

3. Pick and Tune an Algorithm

Pick an algorithm:

I have learned 5 modeling algorithms in this course---Naive Bayes, Support Vector Machines (SVMs), Decision Tree, Linear Regression and k-nearest neighbours. But I didn’t try Linear Regression which is applied to continuous data and k-nearest neighbours which is not suitable in this project. As for the three tested algorithms, I used precision, recall and F1 scores to evaluate their performances. The result is as

follows:

```
            splitter='best'))])
        Accuracy: 0.86260       Precision: 0.48164      Recall: 0.40000 F1: 0.43704     F2: 0.41404
        Total predictions: 15000        True positives:  800    False positives:  861   False negatives: 1200   True
 negatives: 12139

  tol=0.001, verbose=False))])
Precision or recall may be undefined due to a lack of true positive predicitons.
        Total predictions: 15000        True positives:  474    False positives:  845   False negatives: 1526   True
negatives: 12155
```

As Decision Tree has the highest scores, I ended up using it.

Discuss parameter tuning and its importance:

The parameters of an algorithm influence the performance a lot. Tuning the parameters of an algorithm means finding out the combination of parameters when the performance of the algorithm is the best. I used Grid_search to tune SVMs and Decision Tree.

Tune the algorithms:

Example of parameter Tuning:

For Decision Tree, I tuned a lot of parameters such as “criterion”, “splitter”. The code is as follows:

```
pipeline1 = [('scaler', MinMaxScaler()),('clf1', tree.DecisionTreeClassifier(random_state=42))]
pipe1 = Pipeline(pipeline1)
param_tree = dict(clf1__criterion=['gini', 'entropy'],
                  clf1__splitter=['random','best'],
                  clf1__max_depth=[1,10],
                  clf1__min_samples_leaf=[1,10],
                  clf1__min_impurity_decrease=[0,10],
                  clf1__min_samples_split=[2,40]
                  )
#from sklearn.model_selection import StratifiedShuffleSplit
#cv = StratifiedShuffleSplit(n_splits=1000)
grid_search1 = GridSearchCV(pipe1, scoring = 'recall', param_grid=param_tree)
grid_search1.fit(features_train, labels_train)
clf_tree = grid_search1.best_estimator_
pred1 = clf_tree.predict(features_test)
```

4. Validate and Evaluate

Validation and its importance:

Validation is a process in which we test our training models on testing dataset. A classic mistake is testing training models on the training dataset, leading to overfitting. As the dataset is imbalance, I used Cross Validation KFold function to split my dataset into testing and training datasets.

Evaluation metrics and their meanings:

I used precision, recall and F1 scores to evaluate the performance for these algorithms.

Precision score: True Positive / (True Positive + False Positive)

In this project, it represents the ratio of the real POIs against all the POIs that the algorithm picks out.

Recall score: True Positive/ (True Positive + False Negative)

In this project, it represents the ratio of the POIs picked out by the algorithm against all the POIs.

As for F1 score, it is the harmonic average of the precision and recall.

The performance of the selected algorithm is as follows:

```
Pipeline(memory=None,
     steps=[('scaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('clf1', DecisionTreeClassifier(class_weight=N
one, criterion='entropy', max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0, min_impurity_split=None,
            min_samples_leaf=10, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=42,
            splitter='best'))])
        Accuracy: 0.86260       Precision: 0.48164      Recall: 0.40000 F1: 0.43704     F2: 0.41404
        Total predictions: 15000        True positives:  800    False positives:  861   False negatives: 1200   True
negatives: 12139
```