Project 5: Identifying Fraud from Enron Emails and Financial Data

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.

In this project, I tried to use machine learning algorithm that I have learned from class to identify persons of interest. There were so many features and I were really lost at first and don't know what to do. After lots of efforts, I finally get some results.

1. Project goals

The goal of this project is to build a predictive model that can identify persons of interest based on features included in the Enron dataset. Such model could be used to find additional suspects who were not indicted during the original investigation, or to find persons of interest during fraud investigations at other businesses.

- **total number of data points: 146 (I have deleted 3 point)**
- **allocation across classes (POI/non-POI): POI 18 non-POI**
- **number of features used: 5**
- **are there features with many missing values?**

{'bonus': 62, 'deferral_payments': 105,'deferred_income': 95, 'director_fees': 127, 'email_address': 32, 'exercised_stock_options': 42,'expenses': 49, 'from_messages': 57, 'from_poi_to_this_person': 57, 'from_this_person_to_poi': 57, 'loan_advances': 140, 'long_term_incentive': 78, 'other': 52, 'poi': 0, 'proportion_from_poi': 0, 'proportion_to_poi': 0, 'restricted_stock': 34,'restricted_stock_deferred': 126, 'salary': 49, 'shared_receipt_with_poi': 57, 'to_messages': 57, 'total_payments': 20, 'total_stock_value': 18}

2. Feature selection

At first, I used all feature in my algorithm
I got the result from Gaussian NB
Precision 0.095
Recall 0.278

After I used the selectKbest to check precision and recall on different K in selectKbest I also created two new feature which is the proportion of message to /from poi. I want to know if someone had really strong connect to other poi.

(K, precision, recall, f1_score)
(1, 0.6396807297605474, 0.2805, 0.3899895724713243)
(2, 0.437597503900156, 0.2805, 0.34186471663619744)
(3, 0.5164257555847569, 0.393, 0.4463373083475298)
(4, 0.515748031496063, 0.393, 0.4460839954597049)
(5, 0.5087378640776699, 0.393, 0.44344146685472496)
(6, 0.5087378640776699, 0.393, 0.44344146685472496)
(7, 0.4763636363636364, 0.393, 0.43068493150684933)
(8, 0.4763636363636364, 0.393, 0.43068493150684933)
(9, 0.40790273556231005, 0.3355, 0.3681755829903978)
(10, 0.4666203059805285, 0.3355, 0.3903432228039558)
(11, 0.5052710843373494, 0.3355, 0.4032451923076923)
(12, 0.46824842986741105, 0.3355, 0.3909117390037868)
(13, 0.46824842986741105, 0.3355, 0.3909117390037868)
(14, 0.46824842986741105, 0.3355, 0.3909117390037868)
(15, 0.506797583081571, 0.3355, 0.4037304452466907)
We can see that when k = 3, we have highest F1_score, so I choose three feature into my algorithm.
['total_stock_value', 'exercised_stock_options', 'deferred_income']

3. Algorithm selection

I have used three algorithms. Unfortunately, support vector machine seems not work in this case. So I chosed GaussianNB as my final algorithm.

DecisionTree:
Precision: 0.2
Recall: 0.25

SVM:
Precision: 1.2
Recall: 2.5

GaussianNB:
Precision: 0.5
Recall: 0.25

4. Algorithm tuning

The algorithms are parameterized so that their behavior can be tuned for a given problem. It's important to perform parameter tuning here to adjust the precision and recall. In order to get the best result in identifying poi I used scikit-learn GridSearchCV to tune my algorithm parameter and get the best parameter.
I used the parameter range below

```
parameters = {'tree__criterion': ('gini','entropy'),
                        'tree__splitter':('best','random'),
                        'tree__min_samples_split':[2, 10, 20],
                        'tree__max_depth':[10,15,20,25,30],
                        'tree__max_leaf_nodes':[5,10,30]}
```

And the best parameter for decision tree’s parameter are

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
max_features=None, max_leaf_nodes=30, min_samples_leaf=1,
min_samples_split=10, min_weight_fraction_leaf=0.0,
presort=False, random_state=None, splitter='best')

5. Validation

Validation

The cross validation process helps us to get the estimate performance of a independent data and serves as a check on overfitting. The goal is to define a dataset to "test" the model in the training phase. Because of the small size of the dataset, stratified shuffle split cross validation is used. StratifiedShuffleSplit also keeps the ratio of POI/non-POI the same as it was in the original data set. A classic mistake to make a wrong validation is to split the sorted data without shuffling.
Here are my final output of NB:

Tester GaussianNB report
Pipeline(steps=[('scaler', MinMaxScaler(copy=True, feature_range=(0, 1))),
('GaussianNB', GaussianNB())])
Accuracy: 0.86400 Precision: 0.53338 Recall: 0.38350 F1: 0.44619 F2: 0.40634
Total predictions: 14000 True positives: 767 False positives: 671 False negatives: 1233 True negatives: 11329 Evaluation

Precision:

Avg: 0.533
Precision is also referred to as positive predictive value. It's calculated as True Positive/(True Positive + False Positive). In here it means the proportion of the correct prediction of all the people who are predicted to be poi.

Recall:

Avg: 0.384

Recall is also referred to as the true positive rate or sensitivity. It's calculated as True Positive/(True Positive + False Negative). In here it means the proportion of the poi the model can detect of all the poi. For fraud prediction models, higher recall is generally preferred even if some precision is sacrificed.

This project is a classification and we have a little poi here, so it’s an unbalanced data. The accuracy will be so high even if we all predict negative.