Zihe Xu
July 20, 2015

Enron POI Identifier Report

Zihe Xu

Introduction:

In this project, I will analyze the Enron fraud using email and financial data to identify persons of interests with some data processing, analyze and machine learning. I tested several algorithms like 'Logistic Regression ' 'Naive Bayes' 'Support Vecto' K-Neighbors' 'Decision Tree', 'AdaBoost' etc.

Enron Data Overview

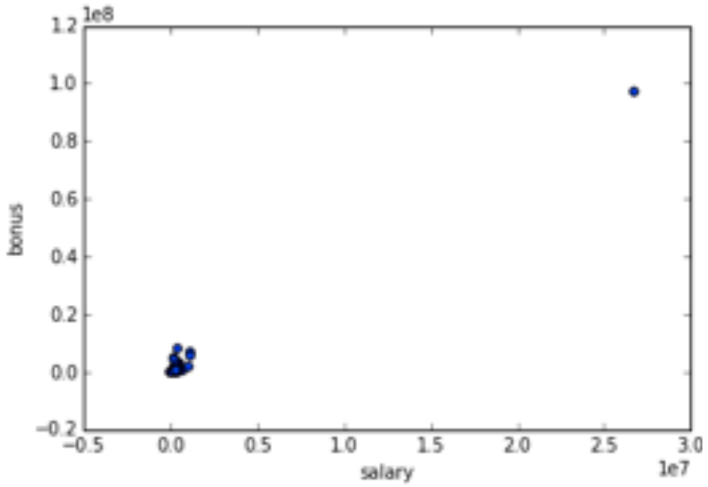
The dataset I had contains a table of those information for everyone related to Enron: There are 21 features in total, they are :['salary', 'to_messages', 'deferral_payments', 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'restricted_stock_deferred', 'total_stock_value', 'expenses', 'loan_advances', 'from_messages', 'other', 'from_this_person_to_poi', 'poi', 'director_fees', 'deferred_income', 'long_term_incentive', 'email_address', 'from_poi_to_this_person']

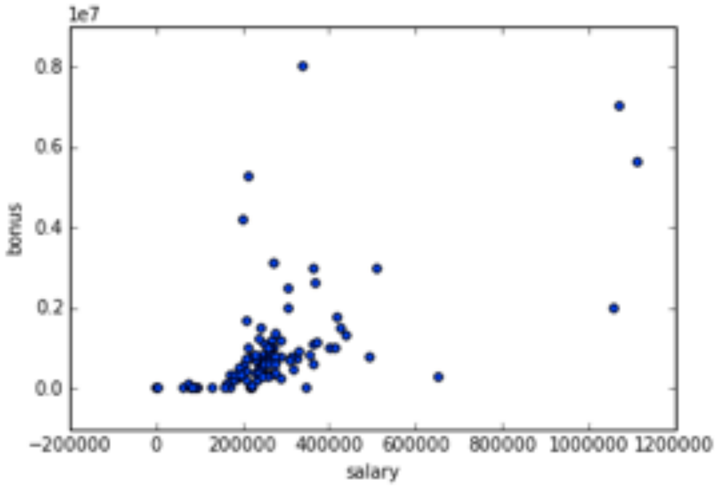
Of all those features numbers like 'salary', 'total_payments' would be very useful to identify the POI, also the relationship between one person to one POI is also important, I could find that information through how many email exchanges they have with the POI. Also, I noticed that the data is not perfect, there are some null values (NAN to be exact) in the dataset, which should be careful to deal with.

The total number of data is 146, 18 of them is POI, however not all data are meaningful. There are some outliers we need to get rid of before further analyze.

Feature Processing

As mentioned above, there are some outliers in the dataset, by plotting it out, I can see that there is a clear outlier :"Total" . By looking at the meaning of the name, I also found a name THE TRAVEL AGENCY IN THE PARK which is clearly not a person, Therefore, we can take them out. Moreover, there is a guy named LOCKHART EUGENE E who has null value for most of his attributes, there is no means to keep it.





before after

As you can see in the second graph, it seems there are 4, maybe 2 more outliers, however they are not, they are actually some people made into the top position in Enron and clearly the person of interest.

Adding new features

In this part, we I tested several different combinations of features, all new features are made by compare two original features. Here I generated new features on financial data as well as the email data. In order to determine how many new features, I test the features from 3 to 10, because we don't want to use too many features which will increase the running time and therefore not scalable. The idea is to select minimum features that can make a good prediction, in my project, I found when you get more than 5 features, the precision and recall gains little, as a results , From all these new created features and the original features (in total 29 features) I choose top 5 by using SelectKBest.

The features that get selected are 'salary', 'bonus', 'bonus_vs_salary', 'from_poi_to_this_person_vs_to_messages' 'from_this_person_to_poi_vs_from_messages

This make sense because the ratio between emails get and send to poi and the total number of email get and send shows how connected between this person and poi. If for a person he or she has a large portion of emails exchanged with poi instead of other people, that might indicate that this person is more likely to be a poi himself/herself.

Algorithm selection & Feature scaling:

In this task I tried 4 different algorithms with it's default setting. For LinearSVC and KNeighbors needs feature scaling because they both use distance to separate different cluster. Here I use MinMaxScaler. For salary and bonus which have a very large range, so we need to perform feature scaling on them. 'bonus_vs_salary', 'from_poi_to_this_person_vs_to_messages' 'from_this_person_to_poi_vs_from_messages' are percentages, they are already within (0,1) no need to scale them. Here is a result of different algorithms The results shows that although the KNeighborsClassifier has the largest max value, the Decision Tree is the only one that has both recall and precision larger than 0.3.Moreover it has the highest f1 score and f2 score. So I choose Decision Tree to be further optimized.

Precision measures how many true positives in all predicted value. It measures the accuracy of an algorithm. The higher the precision means we are more certain about a positive prediction. Recall measures the true positives in total number of real positive ones. A high recall means most of positive value existed were predicted, it might produce many fake positives but as long as it covers most real positive values, it will have a high recall. F1and F2 score are measurements that combines the precision and recall. Therefore use f2 or f1 is a good way to evaluate the performance of the algorithms.

f1 = 2.0 * true_positives/(2*true_positives + false_positives+false_negatives)

f2 = (1+2.0*2.0) * precision*recall/(4*precision + recall)

recall = 0.0565 precision = 0.275609756098 f1 score = 0.09377593361 f2 score = 0.0671819262782 max = 0.332109756098

the method I use is LogisticRegression

recall = 0.416 precision = 0.240671102112 f1 score = 0.304929448415 f2 score = 0.343096796718 max = 0.656671102112

the method I use is LinearSVC

recall = 0.196 precision = 0.573099415205 f1 score = 0.292101341282 f2 score = 0.225702441271 max = 0.769099415205

the method I use is KNeighborsClassifier

recall = 0.3565 precision = 0.351058591827 f1 score = 0.353758372612 f2 score = 0.355398265377 max = 0.707558591827

the method I use is DecisionTreeClassifier

Tune the algorithm

Why we need t tune the algorithms? First, we want to have a better predictions, also we want to balance the recall and precision. How to tune the algorithms depends on the algorithm of choice, here, we have the decision tree, the main properties are the max depths, min split and max_features. From the algorithm we know a higher value of max_depths, min split and max_features will increase the complexity of the algorithm and therefore might get overfitting, so in the training process, I start them with lower value and increase them in the for loop. The reason I didn't use the build in Grid search is because I want to get more detailed feed back, and the example of Grid search seems only able to adjust one parameter. With my code, I could add unlimited number of combinations of different parameters, it might take a lot of resources for a huge dataset. But for a small project even my laptop can handle it.

For the decision tree I also looked up other parameters, like gini and entropy. I tested both of them, for the max_feature, I tested auto, sort, log2 and None, (because we already did some feature selection, I prefer None, which give no limits for the total number of features. max_dep I set to choose from 2 to 10 which considering our dataset is not very large, that should be enough. The first round test I found only use gini I can get both recall and precision larger than 0.3, also I noticed that by increasing the max_depth, I will get higher recall , but lower precision. Also the larger the max_featuers the better overall performance. Because we already eliminate some of features and only select best 5, so as I predicted all those features are important as a result increasing the max_features would give a better results. The final parameter I choose for the decision tree is criterion = gini max_features is None and max_depth =2 which has the maximum precision and overall value.

```
recall =  0.3065 precision =  0.364880952381 max =  0.671380952381
criterion =  gini max_features =  auto max_depth =  6

recall =  0.318 precision =  0.35530726257 max =  0.67330726257
criterion =  gini max_features =  auto max_depth =  7

recall =  0.3275 precision =  0.351772287863 max =  0.679272287863
criterion =  gini max_features =  sqrt max_depth =  8

recall =  0.3345 precision =  0.354718981972 max =  0.689218981972
criterion =  gini max_features =  sqrt max_depth =  9
```

Data Validation and Evaluate

"Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set." Usually we have a finite data set for the training, perform learning algorithms on specific data set might get overfitting which makes the tuned algorithm not able to generalize to use on other data. Validation uses a random part of data set (some times many different part of dataset) as the training set and the rest as testing set. Perform the learning algorithms on the different training set and optimize the algorithm. Without data validation one might get overfitting In this project. To avoid overfitting on the training set. I split the data to different set for training, use StratifiedShuffleSplit to split, the fold is set to be 1000, so it has enough testing samples for the averaging. Use the validation process could eliminate possible overfitting and therefore for all those data above is calculated through the averaging.

Evaluation

I use precision and recall to evaluate the results.

The precision P is the ratio between true positive and the sum of true positive and false positive. So high precision means of all those predicted POIs I have a high confidence that most of them are labeled correct. However, there might be some POIs that I missed (labeled as non-POIs)

The recall R is the ratio between true positive and the sum of true positive and false negatives, higher recall means of all potential POIs I have recognized most of them, however, the tradeoff is that I might miss labeled some of them.

To measure the total performance algorithm I use f1 score, which defined as 2(P*R)/(P+R)

The final results of the decision tree is

recall = 0.3265 precision = 0.498473282443 max_f1 = 0.394561933535

the parameters I use is :

max_features = 0.3505 max_depth = None min_samples_split = 2

Discussions and Conclusions

It is very interesting to make some predictions with existing data, this project enhanced my ability to perform such a task. Some important idea like identify the outliers, creating new features, and choose machine learning method are fundamental and need some intuitive. Although I find sometimes we can automated this process simply by trying everything thing out, however, it is impossible if we want to scale. The problem with this project is the useful data is too little, as a results when I am tuning the algorithms, I only get a limit ways, and the features have been selected before the tuning process, so that there is little left to try. If we can get more data and with more attribute, we can have a better rate of recall as well as precision. Another thing I noticed when tuning the algorithms is that some parameters like the number of max_depth in Decision Tree will have a positive impact on recall but might have a negative impact on precision. We could have two different tuned program to making prediction based on our preference : the false positive or true negative.

Overall it is a well designed project and I enjoy learning how to explore the data.