Identify Fraud from Enron Financial Data

by Inessa Prokofyeva

Results Analysis

Introduction

Enron Corporation was an American energy, commodities, and services company based in Houston, Texas. The Enron scandal, revealed in October 2001, eventually led to the bankruptcy of the Enron Corporation. By 2002, it had collapsed 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 the main goal is to build a person of interest (POI) identifier based on financial data of Enron that became public after scandal. To perform better identifier results it's also necessary to select proper combination of features to analyse. Using my machine learning skills I'll try several classifiers and different data features to create the model that will allow to find maybe more persons of interest that already known.

Exploring financial data I've found that there are some outliers: "TOTAL" and "THE TRAVEL AGENCY IN THE PARK". That's obviously not proper personal data points so they must be removed before analysis or they'll make a mess in final results.

Also according initial analysis I've discovered several features that have more than 80% of missing values: "loan_advances", "restricted_stock_deferred", "director_fees".

Features selection

One of the most important steps is feature selection. Proper feature selection and knowing relationships in data can be very useful for algorithm selection and tuning. The final list of features is: *['bonus', 'expenses', 'exercised_stock_options', 'from_messages_ratio']*.

After analysis I've discovered that features *"to_messages", "from_messages", "from_poi_to_this_person", "from_this_person_to_poi"* probably can not be very representative as soon as some people can send many messages according their work duties, so more important ratio of sent and received messages to and from POIs. That's why I've created 2 new features: *"to_messages_ratio", "from_messages_ratio"*, and used them in future feature selection.

As a first step of feature selection I've used SelectKBest algorithm. There are top-5 rated features:

- *salary : 4.45863623849*
- *exercised_stock_options : 5.74035718147*
- *bonus : 4.95809486065*

- *deferred_income : 3.03525178831*
- *from_messages_ratio : 4.07647319113*

According these results it is noticed that the most valuable features are *"salary"* and *"exercised_stock_options"*. And it's expected as soon as most of POIs were among leaders of company and had significant values in these fields. Surprisingly there's also highly rated "from_messages_ratio" that can show that people involved in scandal received more letters from POIs that usual employees.

SelectKBest algorithm as univariate feature selection works by selecting the best features based on univariate statistical tests. But it doesn't really see the regularities in data. As a second step I've used RFE and DecisionTreeClassifier. Running RFE and estimator I've got the following top ranked features:

- *exercised_stock_options : 1*
- *expenses : 1*
- *other : 1*
- *from_messages_ratio : 1*

Here we can see mostly confirmed results as there were with SelectKBest. But also *"expenses"* and *"other"* features were highly rated. It was confirmed that the new created feature *"from_messages_ratio"* is pretty valuable. *"Other"* feature has a lot of missing values, so I didn't include it into my final list. So I've tried several combinations of this features to get better accuracy in identifier. As a result the final list of features described earlier helped to achieve the accuracy of 86% with GaussianNB classifier.

Algorithm selection

I've started with GaussianNB classifier, and as it was described earlier there were pretty good results achieved - 86%. But I was really curious about algorithms and decides to try several just to see what results I can get. Here you can see the results of different algorithms used just "from the box" (tester.py is used to get these results):

- GaussianNB - Accuracy: 0.85571, Precision: 0.49157, Recall: 0.29150, F1: 0.36598, F2: 0.31733
- LinearSVC - Accuracy: 0.71569, Precision: 0.20617, Recall: 0.29750, F1: 0.24355, F2: 0.27329
- AdaBoostClassifier - Accuracy: 0.85221, Precision: 0.47582, Recall: 0.33950, F1: 0.39626, F2: 0.36014
- DecisionTreeClassifier - Accuracy: 0.83529, Precision: 0.17447, Recall: 0.04100, F1: 0.06640, F2: 0.04841

As we see all of them are giving not bad results according accuracy value. The lowest accuracy has LinearSVC classifier and it's explainable because SVC works bad with data with a lot of noise, and we know that this data has several "outliers" that are not actually outliers but a valid values for top managers in Enron (salary, bonus, etc.).

As soon as other classifiers showed not bad results in accuracy, but the precision value is not very impressive. Only AdaBoostClassifier showed satisfied results. Finally I've stopped my choice on KNN classifier (final results with this classifier):

- KNeighborsClassifier(algorithm='auto', leaf_size=10, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=5, p=2, weights='uniform')
    - Accuracy: 0.88893, Precision: 0.73645, Recall: 0.34650, F1: 0.47127, F2: 0.38754
    - Total predictions: 14000, True positives: 693, False positives: 248, False negatives: 1307, True negatives: 11752

This classifier works pretty well with this data and achieves 89% of accuracy. Also it worked fast: only a little more than a second to train it (LinearSVC was working with the same data almost 5 minutes).

Algorithm tuning

Working with KNeighborsClassifier it was really fast to get good results even without tuning it. But I wanted to get the best performance that possible. I've used GridSearchCV to try several different parameters for KNN. As a final result I've got:

- Best parameters: {'n_neighbors': 5, 'weights': 'uniform', 'leaf_size': 10, 'algorithm': 'auto'}

Using GridSearchCV helped to improve the accuracy from 86% to almost 89%. Not so much but there is still an improvement.

Validation

The first validation value was always accuracy, this value is really important to show how accurate the identifier predicts values. But it's not the most important in my analysis.

Speaking about initial goals of this project: we are trying to create an POI identifier that can tell us is some specific person involved in Enron scandal or not. So more important validation value is precision - out of all items labeled as positive how many of them are truly belongs to positive class. In other words: how many people marked as POIs how many of them are really involved in fraud. And this is most important value, because saying that some person is involved is really strong statement, so we must be pretty sure.

As we saw earlier a lot of classifiers showed good accuracy value. And it would be a really big mistake to trust just only this validation. So in case with KNN classifier I've got really good results in precision 74%. Also it not bad results in recall value: 35% (out of all items that are truly positive, how many of them were correctly classified as positive).

There must be a trade-off between recall and precision depending of what is more important in specific case. In this case these two metrics are even more important than accuracy, as soon as we have really not so much people marked as POI.

Conclusion

Working with this project I've completed the end-to-end process of investigating data through a machine-learning lens. Started with the very beginning of cleaning data and selecting features to analyse I've created a prediction model, a POI identifier.

The most interesting part was trying algorithms and studying every aspect of them, how good are they working with specific data, how fast they are, what parameters can I use to achieve more performance.

As soon as I've worked with real-world dataset there was a good lesson to find out that nothing is perfect especially when there are no right answers and you free to choose your own features, algorithms. The main thing I've learned is that you need to be very critical about decisions you make and results you get. Careful and diverse look on a problem can help to achieve great results.

References

- https://en.wikipedia.org/wiki/Enron
- https://en.wikipedia.org/wiki/Enron_scandal
- http://usatoday30.usatoday.com/money/industries/energy/2005-12-28-enron-participants_x.htm
- http://scikit-learn.org/stable/
- http://en.wikipedia.org/wiki/Enron:_The_Smartest_Guys_in_the_Room