Identify Fraud from Enron Email

Submitted By : Ashutosh Gupta
Date : 14th August 2017

Overview

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, you will play detective, and put your new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal. To assist you in your detective work, we've combined this data with a hand-generated list of persons of interest in the fraud case, which means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Q&A
Question1

Summarise for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: “data exploration”, “outlier investigation”]

The goal of this project is to use the user and financial data of enron to identify fraud i.e. person of interest(POI).
The dataset has 146 records, i.e. 146 peoples record including their name, email, salary information, other financial informations and their email records and etc.
Each record has 20 features, 1 poi label + 14 financial + 5 email.
Out of 146 records 18 are POI and 128 are Non POI.

Name	number of non NAN values
POI	146
salary	51
bonus	64
exercised_stock_options	44
total_stock_value	20
deferred_income	97
long_term_incentive	80
restricted_stock	36
total_payments	21
shared_receipt_with_poi	60
loan_advances	142
expenses	51
from_poi_to_this_person	60
other	53
from_this_person_to_poi	60
director_fees	129
to_messages	60
deferral_payments	107
from_messages	60
restricted_stock_deferred	128

As we can see there are few features for which there are very few number of non NAN values. Like
loan advances : 142
restricted_stock_value : 128
directors_fees : 129
I would like to remove these features as there are not much data available to be useful for my model.

There are many rows for which only few features have values,
- 90th record has 20 features as NAN value
- 101st record has 18 features as NAN value
- 141st record has 18 features as NAN value

We can delete those rows, but since we do not have a large dataset, I am keeping those records also.

Yes, we do have outliers in the dataset, since the dataset in very not very large I prefer to look into dataset manually and try to find out any outlier.
I found 2 outliers,

1. The travel agency in the park
2. Total

These 2 are clearly outliers, as The travel agency in the park is definitely can't be someone's name.
And total is the amount total row, which adds all the column value for all rows.
Removed these 2 outliers.

Above mentioned features and records are outliers and we can handle it in variety of ways, like

1. To ignore them, as dataset if limited,
2. replace them with mean, median or mode of the remaining column.

There are other ways to detect outliers in the dataset.

1. Using box plot to find the outliers
2. Plotting features and see are there points too above or too below the normal data points.

New Dataset statistics:
total records in dataset: 141
poi count: 18
non-poi count 123
total features: 19
Target Label : 1

Question2

What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

To investigate the scores of features I used selectkbest algorithm.
Below are the features and their importance score as provided by selectkbest:

Feature	Importance score
shared_receipt_with_poi	8.90
from_poi_to_this_person	5.45
loan_advances	2.52
from_this_person_to_poi	2.47
to_messages	1.75
director_fees	0.55
total_payments	0.35
deferral_payments	0.24
exercised_stock_options	0.23
deferred_income	0.22
total_stock_value	0.17
from_messages	0.16
bonus	0.10
other	0.07
restricted_stock	0.03
long_term_incentive	0.02

Feature	Importance score
expenses	0.01
restricted_stock_deferred	0.004
salary	0.00016

We can see the features score above.
There are many features having very low importance score, but I am not selecting or removing any of the features only based on the importance score. I will use some other measures like finding different metrics score for different number of features.

In next section I will use one of the studied algorithm and train it using different number of features, from k=1 to k= 19 features and will see at what number of feature I am getting maximum accuracy, F1-score, recall and precision score.

Feature Selection:For selecting what number of features would give me best result, I used selectkbest algorithm and GuassianNaiveBayes classifier, and starting from number of features from 1 and goes till all features. Results are below:

No of Feature	Accuracy	Recall	Precision	F1-score
1	0.82467	0.06967	0.02550	0.03734
2	0.81467	0.06954	0.03150	0.04336
3	0.81160	0.07684	0.03750	0.05040
4	0.82180	0.09311	0.03850	0.05447
5	0.79000	0.15403	0.12800	0.13981
6	0.84120	0.25130	0.09650	0.13945
7	0.83280	0.14023	0.04950	0.07317
8	0.79140	0.09765	0.06850	0.08052
9	0.77413	0.12364	0.11400	0.11863
10	0.76987	0.13554	0.13500	0.13527
11	0.77487	0.17292	0.18200	0.17734
12	0.81353	0.24921	0.19800	0.22067

No of Feature	Accuracy	Recall	Precision	F1-score
13	0.82400	0.29592	0.23200	0.26009
14	0.83260	0.34180	0.27600	0.30539
15	0.83473	0.35405	0.29050	0.31914
16	0.83293	0.34421	0.27950	0.30850
17	0.83220	0.34014	0.27500	0.30412
18	0.83500	0.34007	0.25250	0.28981
19	0.82827	0.32692	0.27200	0.29694

In the above table I have determined 4 different metrics: accuracy, F1score, Recall and Precision.
The scores for different metrics are for different number of features, starting from No of feature =1 to No of feature = 19.

Different metric shows different trend in the table, like accuracy starts with high value then decreases till No of features become 5, then increases for a while and then again start decreasing. And after No of features = 12, it stays almost constant till the end.
I get Max accuracy at No of features (k) = 5.

For Recall, Precision and F1-score the trends seems similar and increasing. Constantly increasing as K increases.
But All these 3 features gives max value at K = 15.

At K=5 where accuracy is max we are getting very low Precision and F1-score. But at K=15, where recall, precision and F1-score is maximum, accuracy is also quite high. Almost comparable to what it is at K = 5.

So based on the above table and my observation about maximum values of different metrics, I will choose K=15, No of features at which I am best result for all metrics.

The reason why I didn't choose K=5, is precision is very low, and precision is very important metric as it tells that the whatever result model is giving how much accurate it is. So in Enron perspective, I would say Precision is more important then accuracy and other metric as, I don't want to tag a Non POI as

POI, I am happy leaving some POI undetected, but prosecuting innocent Non POI as POI would be worse then that.
So precision is important. This is the reason I am not picking K=5 instead choosing K=15.

So from above discussion it is clear that by choosing top 15 features, I am getting max score for all the metrics.
I will choose top 15 features.

To get top 15 features I will use the table displayed earlier about feature and their importance score.
The table displayed the importance score for all the features, I will pick top 15 features.
Below is the table:

Feature	Importance score
shared_receipt_with_poi	8.90
from_poi_to_this_person	5.45
loan_advances	2.52
from_this_person_to_poi	2.47
to_messages	1.75
director_fees	0.55
total_payments	0.35
deferral_payments	0.24
exercised_stock_options	0.23
deferred_income	0.22
total_stock_value	0.17
from_messages	0.16
bonus	0.10
other	0.07
restricted_stock	0.03

Scaling of features :
Each feature has variety range of values, so I need to Scale (normalize) them. I used MinMaxScaler for scaling of features.

New feature :
I have created a new feature total_income, which is sum of salary and total_stock_value

total_salary = salary + total_stock_value

The reason behind creating total_salary as new feature is the person who is POI would have very high total_salary which comprises of salary and stock value. So i think this could be an important factor in determining the POI.

Now I will add the this new feature in my features list and see is there any improvement or decline in the metric score:
New features list :
[shared_receipt_with_poi,from_poi_to_this_person,loan_advances,from_this_person_to_poi,to_messages,director_fees,total_payments,deferral_payments,exercised_stock_options,deferred_income,total_stock_value,from_messages,bonus,other,restricted_stock, total_salary]

No of features	Accuracy	Precision	Recall	F1-score
Top 15 feature	0.83473	0.35405	0.29050	0.31914
Top 15 feature + new feature	0.84107	0.36992	0.27300	0.31415

There is not much difference in the overall metric score, there is slight increase in accuracy and Precision, And Recall and F1-score has reduced slightly.
The increase in the Precision is important here, as explained above how Precision plays important role in this Enron POI detection scenario.

So I am happy with the new created feature.

I end up using the below 16 features :
[shared_receipt_with_poi,from_poi_to_this_person,loan_advances,from_this_person_to_poi,to_messages,director_fees,total_payments,deferral_payments,exercised_stock_options,deferred_income,total_stock_value,from_messages,bonus,other,restricted_stock, total_salary, total_income]

Question3

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

I tried 5 algorithms total, all performs differently and I used accuracy, recall, precision and F1 score metrics to decide the final algorithm.

Algorithm	Accuracy	Precision	Recall	F1
NB	0.84107	0.27300	0.36992	0.31415
DT	0.78600	0.20020	0.20020	0.201010
LR	0.76720	0.17900	0.20364	0.19010
ADA	0.79967	0.18550	0.21236	0.19803
KNN	0.87820	0.23500	0.61278	0.33972

Naive Bayes with PCA performs exceptionally well, other algorithms like Decision Tree also did well in terms of all the metrics.
KNN was a surprise as its has very high accuracy and very high precision but average recall.

For final tuning of parameter I am choosing Naive Bayes and KNN as they have all 4 metrics reasonable values.

Question4

What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: "tune the algorithm"]

By tuning the parameter we mean that finding the best parameter for our model so that we get best result in the end.
Every algorithm has many parameter and each parameter can have many possible values, so we need to find those values at which the algorithm will perform best.

For performance tuning we use GridSeachCV function which takes algorithm to tune and all possible combinations of parameter. And in result it provides us with the best parameters.

If we don't tune the parameter we wont get the best results and our struggle for finding best result will increase as we will be trying every possible combination of parameter for the algorithm. We might find the best combination available but this will take a lot of time.

In this project I tried to tune 2 algorithms Naive Bayes and KNN.

For KNN I am tuning

- n_neighbors: [2,3,4,5,10] —Number of neighbors to use by default for neighbour query.
- algorithm: ['auto', 'ball_tree', 'kd_tree', 'brute'] —Algorithm used to compute the nearest neighbours.
- leaf_size: [20,30, 50,70,100] — This can affect the speed of the construction and query, as well as the memory required to store the tree

For Naive Bayes since there are no parameters to tune, I have used PCA in pipeline with Naive Bayes and used PCA's parameter for tuning

n_components : [None,1,2] — Number of components to keep

Final parameters that I have got after tuning are :
- n_neighbors = 3
- algorithm = auto
- leaf_size = 30
- PCA = None

Question 5
What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

Validation is the way to confirm the robustness of a classifier with given dataset and model. The class mistake is the over-fitting case. When a classifier is over-fitted, it cannot provide good performance on test dataset. To avoid this problem, usually dataset is partitioned to three set, (train set, validation set, test set). However, it causes the number of samples to be drastically reduced. Therefore, the results can depend on a particular random choice for the pair of (train, validation) sets.

Like in our case too, we have a very small dataset, so we cannot partitioned it into 3 sets.
One possible solution is that we can use cross validation. By using cross-validation, we can split whole data into k parts and train on first k-1 parts and test on remaining kth path. Next model will be train on another k-1 parts and tested on remaining kth part.
Ex : lets divide the data into 5 parts, names 1,2,3,4 and 5

Train	Test
1,2,3,4	5
2,3,4,5	1
3,4,5,1	2
4,5,1,2	3
5,1,2,3	4

This way our model will be trained and tested on all the dataset points. And we will have a good result.

In our case I am using the same strategy with slight improvement, since our dataset is small I am using StratifiedShuffleSplit function. It is a combination of StratifiedKFold and ShuffleSplit. What this does is first shuffle the data using the ShuffleSplit and then do the k-Fold validation and testing using StratifiedKFold.
The reason for using StratifiedShuffleSplit is that our dataset is small (only 146 records) and it highly imbalance, imbalance refers here the fact that we have only 21 POI out of 146 records, and if we use test train split to validate and test our model, this imbalance will affect our performance, as test dataset might have very few POI.

StratifiedShuffleSplit(n_splits=100, test_size=0.1, random_state=42)
test data size is 10% and number of folds in 100.
Same strategy is used in tester.py for validating our final model.

Question 6
Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says

something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]

In the end I am using 2 Algorithms with tuned parameters.
The evaluation metric I am using is Recall and Precision.

Recall : number of true positive values selected out of all the possible true values.
Recall = TP / (TP + FN)

Precision : Number of true positive values selected out of all the given true values.
Precision = TP / (TP + FP)

In very general term recall can be treated as metric which will decide how many POI can model is able to detect.

And Precision is out of recalled POI, how many are correct, i.e. actually POI and not Non-POI.

Algorithm	Recall	Precision	Accuracy
KNN	0.28100	0.54247	0.87253
PCA + Naive bayes	0.30050	0.31237	0.81853

For optimal performance of our model, I have passed F1 score as tuning parameter for score.
F1 score is weighted average of Recall and Precision, so this way both Precision and Recall will get tuned.

In the above table we can see that KNN performed better in terms of accuracy, precision. PCA + NB did better in recall.

According to my understanding of the Enron POI detection case, it is important to have an algorithm which is accurate and having high Precision rather then Recall. Because if my model is stating that Mr X is POI and action should be taken against him, I wanted to make sure Mr. X is actually a culprit not some one who is by mistake caught. Because that would be worse to punish an innocent rather then leaving few culprits.
If I choose high recall and low precision I might get many POI's but who actually are POI will be a dilemma.

Conclusion

KNN is the algorithm I would like to suggest due to above mentioned statistics and reasons. So using above methods and ways we can hope to detect Enron POI's.

What extra thing I tried in the dataset was to find the correlation between the features, since the number of data points are very less, we have the option to reduce the number of features as with larger number of features we need more data, which we didn't have. So using correlation we can reduce the redundant features, i.e. features having high correlation can be removed as one feature would be enough to give the effect of 2 highly correlated features. I didn't tried the strategy but didn't use it here, but I thing this can surely be used.

References:

1. StratifiedShuffleSplit
2. GridSearchCV
3. DecisionTreeClassifier
4. KNeighborsClassifier
5. GuassianNB
6. Pipeline