Identify Fraud from Enron Email

Chengyuan Yang 16.07.2018

1. ***Summarize 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?***

This project serves the goal of creating a predict model, which studies the patterns of emails and financial data in order to identify the Person of Interest (POI) in the Enron case, which is one of the most famous financial scandal in history and leads to the bankruptcy of the Enron Corporation. In this Project, the dataset published by US Federal Energy Regulatory Commission was used. In the dataset there are 146 records with 1 labeled feature (POI), 14 financial features, 6 email features. In total 18 people in the dataset are labeled as POI.

Through the check on the name of the 146 names, I found two possible outliers:

- TOTAL, which is the sum of all the financial data in the dataset.
- THE TRAVEL AGENCY IN THE PARK, which contains only in "total_payments" and "others".

So, I deleted these two outliers from the dataset.

2. ***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.***

In this project, I created extra two features for analysis. One is "ratio_from_poi_to_this_person" which measures how frequently this person receives mails from POIs. The other one is "ratio_from_this_person_to_poi" which measures how frequently this person sent mails to POIs. The reason for making these two metrics is that some people, for example the secretary of POIs, may write or receives a lot mails to and from POIs but they may not POIs. It is therefore more reasonable to measure the metric in percent.

After I created and added the new features into the feature list, the next step is to select the proper features to build identifier. By checking reference book I decided to use "SelectKBest" function to select the features with the most influence on identifying the POIs. "SelectKBest" selects the top k features that have maximum relevance with the target variable. The scores of all the features are as followed:

'exercised_stock_options'	24.81507
'total_stock_value'	24.18289
'bonus'	20.79225
'salary'	18.28968
'deferred_income'	11.45847
'long_term_incentive'	9.922186
'restricted_stock'	9.212810
'total_payments'	8.772777
'shared_receipt_with_poi'	8.589420
'loan_advances'	7.184055
'expenses'	6.094173
'from_poi_to_this_person'	5.243449
'ratio_from_poi_to_this_person'	5.123946
'other'	4.187477
'ratio_from_this_person_to_poi'	4.094653
'from_this_person_to_poi'	2.382612
'director_fees'	2.126327
'to_messages'	1.646341
'deferral_payments'	0.224611
'from_messages'	0.169700
'restricted_stock_deferred'	0.065499

As we can see from the scores, almost all of the top 50% features with the most influences are financial features together with the only email-related feature. At the beginning, I decided to use the first 10 features on the list but in the final test I reduce the number of features and only kept the top 3 features, namely “exercised stock options”,” total stock value”, and “bonus”. The reason for reducing the feature number is that the precision& recall performance didn't meet the requirement (the detailed process is discussed in part 4)

The scales of the features in use vary vastly, for example, the bonus ranges from 5-digit numbers to 7-digit numbers, while "the shared receipt with poi" ranges from a 1-digit number to a 4-digit one. So I decided to use the sklearn's MinMaxScaler() to standardize the features..

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

Since this project involves a supervised learning, I tried four types of classifiers: Decision Tree Classifier, Random Forest Classifier, SVC and Logistic Regression Classifier. I created a function called “evaluate_clf” to test the performance of the four classifiers. Without setting any parameters, I got the following results:

Classifier	Precision	Recall
Decision Tree	0.24	0.20
Random Forest	0.33	0.14
SVC	0.0	0.0
Logistic Regression	0.25	0.40

The SVC performed very badly, while the other three classifiers had the potential to be used. So, I decided to tune these three.

4. ***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? What parameters did you tune?***

Tuning the parameters means finding optimal hyperparameters for a Machine Learning Algorithm. This work is important because the performance of an algorithm depends on what the hyperparameters are. Since hyperparameters are set before any Machine learning algorithm is run, it becomes very essential to set an optimal value of hyperparameters as it effects the convergence of any algorithm to a large extent. Since I didn't apply any settings on the hyperparameters used in the classifiers above in the first place, I applied "GridSearchCV" to find the optimal hyper-parameter.

Classifier	Parameter 1	Parameter 2	Parameter 3
Decision Tree	min_samples_split	min_samples_leaf	max_features
Random Forest	min_samples_split	min_samples_leaf	n_estimators
Logistic Regression	C	tol	…

After I reran the tuned algorithms, I got the following results:

Classifier	Precision	Recall
Decision Tree	0.29418	0.31300
Random Forest	0.40569	0.19250
Logistic Regression	0.50932	0.20500

As we can see in the table above, the Decision Tree classifier had a balanced good performance. If I did not tune the algorithm, I could have missed an algorithm that could actually do the best job. But unfortunately, the performance could barely meet the requirement. I tried a variety of parameters by using "GridSearchCV", the Precesion& Recall were still blow 0.3. So, I decided to reduce the number of features. By reducing the number of features to three, I got the following result with "tester.py":

Classifier	Precision	Recall
Decision Tree	0.40661	0.41800

5. ***What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?***

Validation comprises set of techniques to make sure our model generalizes with the remaining part of the dataset. A classic mistake, which was briefly mistaken by me, is over-fitting where the model performed well on training set but have substantial lower result on test set. In order to avoid this mistake, the validation is used to separate the dataset into training and testing sets. The training dataset is used to train the algorithm and to find the optimal hyperparameters, and the testing dataset is to do the final check of the performance. In this project I used the train_test_split from the sklearn to do the job by randomly separate 30% of the dataset for testing and the rest for training. The randomicity is important especially by unbalanced dataset like the data in this case because there are way more non-POIs in the dataset.

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.***

Normally, the typical metric for evaluation is accuracy. Since the dataset has way for people who are not POI, then a high accuracy score can be achieved by simply predicting all labels to be the most common label in the dataset. In this context, it would be predicting all labels to be non-POI, which will still result in an accuracy score above 80% despite not actually predicting anything. However, by using precision and recall, we can get a better evaluation on the performance. With a precision score of 0.40., it tells us that if this model predicts 100 POIs, then the chance would be 40 people who are truely POIs and the rest 60 are innocent. On the other hand, with a recall score of 0.42, the classifier has correctly identified 42 of the POI.