Machine Learning Project – Identifying Fraud from Enron Emails

Author: Tanbir
Status: Complete
Completion Date: January 2017

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? [relevant rubric items: “data exploration”, “outlier investigation”]

The goal of this project is to use machine learning to identify people of interest in the Enron scandal by looking over their employment history. What makes machine learning really useful for this is because it saves time and money by having an automated system to identify the person of interest. If humans were to do this purely then it would definitely take longer.

From the initial run of gathering the dataset we can see that there are a total of 146 employees which includes 18 person of interest. The rest 128 are non-person of interest.

There are a total of 21 features within the dataset. Each one is categorized in to three categories. One is financial feature which consist of salary, deferral_payments, total_payments, loan_advances, bonus, restricted_stock_deferred, deferred_income, total_stock_value, expenses, exercised_stock_options, long_term_incentive, restricted_stock, director_fees, and other.

The other category is email feature which consist of to_messages, email_address, from_poi_to_this_person, from_messages, from_this_person_to_poi, and shared_receipt_with_poi.

Final category is poi label which has the poi feature.

After seeing some of the data and using some features I plotted in scatter plots using matplotlib I saw some exponentially high value in each plots. There was a value always on the top right and based on the pdf file that had the list of keys, it seemed like the TOTAL value, which is the summation of every person in the data, is being displayed. So first step, I removed this outlier from the dictionary. I decided to stick with scatter plots due to the fact histogram didn’t show NaN values. When looking at the financial and email feature, “LOCKHART EUGENE E” has NaN on everything when you look at the data. NaN data for all values is useless so I removed him. There was also another value called "THE TRAVEL AGENCY IN THE PARK" which isn’t a name so I removed that as well.

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. (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", "intelligently select features", "properly scale features"]

One of the features that I made is percentage of receipt share with poi. If the person has a higher percentage of receipt then most likely he can be a poi and also a person heavily involved in the scandal. I also made a bonus_to_salary_ratio because this will show the gap in money between the employers. Finally I made two additional features involving the ratio of mailing from and to between the poi and non-poi. This should also play a similar role as to the percentage of receipt. However the receipt targets people who also hasn't emailed or perhaps were quiet but involved secretively.

Afterwards I decided to see if these new features that I made would be considered useful with the SelectKBest algorithm. So I decided to use the algorithm to find the score for the top 7 features. My decision for the top 7 came based on the K score and surprisingly the top 7 were the following:

```
('exercised_stock_options', 24.815079733218194),
('total_stock_value', 24.182898678566879),
('bonus', 20.792252047181535),
 ('salary', 18.289684043404513),
('to_poi_ratio', 16.409712548035792),
 ('deferred_income', 11.458476579280369),
('bonus_salary_ratio', 10.783584708160824)
```

Two of the features I made were top 7 but the other two weren't. from_poi_ratio and shared_receipt_percent features were the other two that were not on top. However, the interesting thing was the K-score for shared_receipt_percent was the same as shared_receipt_with_poi K score meaning that by the number of receipt that algorithms can determine it either way without a comparison to the total number of receipt. I ended up checking some of the accuracy score using Naives Bayes first. So I compared the accuracy for full feature of 25 and accuracy with the top 7 K score. Obviously it showed that the top 7 had a slight lower accuracy while the all the features combined had a higher one:

```
Accuracy of entire 25 features: 0.883720930233
Accuracy of top 7 K-score features: 0.880952380952
```

After also tampering with the number of K-score I decided to pick 7 and add two of the features I created which didn't change much in the accuracy in Naïves Bayes and will come in handy for choosing my algorithm. For scaling, Naïves Bayes does not naturally need scaling because the algorithm behind it already has inbuilt scaling by design. Decision Tree also somewhat does not require scaling because the algorithm involves measuring the distant between the points and does it by threshold value which is not affected by the features. However I did scaling for it just to be sure and results were the same. For the other two algorithms, scaling and tuning had to be done because both measures by distant between each points and if any features were to be big in large scale it will dominate and can over shadow the small features.

3. 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 ended up using Naives Bayes due to the fact it showed a good accuracy in my earlier checks. Then I decided to try SVM, Decision Tree, and Kneighbors algorithm. There were slight differences among the accuracy for each but the most significant metric that differed between each drastically were precision and recall. Naïves bayes also had a higher accuracy, precision and recall than most of them.

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? (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 items: "discuss parameter tuning", "tune the algorithm"]

Tuning algorithm basically means adjusting the parameters until you get the best performance from that algorithm. You can do it manually by adjusting certain parameters based on the documentation for each algorithm or you can do it automatically using other algorithms such as GridSearchCV. You can choose to not to tune your algorithm but in order to get the best results tuning maybe required. It would be a job of a person to engineer and tune to test out the best results or else the predictions will not work well on other data. For this project I utilized GridSearchCV for tuning my parameters. It tries every combination of parameters and then shows the result for best combination. This also saves lots of time especially for SVM algorithm because SVM algorithm can take significantly long especially when using different parameters. I tried for each classifier algorithm and tuned it based on the following:

- Naïves Bayes algorithm is good as the way it is since it's designed for best performance automatically.
- For Decision Tree, it was the same as Naives Bayes as in the default parameters are the best.
- SVM the best parameters are: {'C': 1.0, 'kernel': 'linear', 'degree': 3, gamma: 'auto', 'coef0': 0.0, 'shrinking': True, 'probability': False, 'tol': 0.001, 'cache_size': 200, 'class_weight': None, 'verbose': False, 'max_iter': -1, 'decision_function_shape': 'ovr', 'random_state': None}

- Finally for Kneighbors the best parameters are: {'n_neighbors': 4, 'n_jobs': 1, 'algorithm': 'auto', 'metric': 'minkowski', 'metric_params': None, 'p': 2, 'weights': 'uniform', 'leaf_size': 1}

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 items: "discuss validation", "validation strategy"]

Validation is to test your machine learning algorithm's performance on independent data in order to see how well it was trained. A mistake one can do on validation is to test the algorithm on the same data that it was trained on. The way I dealt with this naturally is make test size split only by 30 percent in the code give as shown: "train_test_split(features, labels, test_size=0.3, random_state=42)" while the training was 70 percent. This is within the evaluate_clf function In addition in the function the iteration is set to 100 times so you'll have the data test 100 times with cross validation technique by splitting the precision, recall and accuracy then grabbing the mean of each metric. This also ensures the numbers when testing on independent data.

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"]

The two evaluations metric and average for Naives Bayes are the following:
Accuracy: 0.86050 Precision: 0.51572 Recall: 0.38550

Accuracy is 0.86 meaning it can classify POI vs Non-Poi at 0.86 correctly. Precision on the other hand is how accurate can the algorithm identify POI when an actual POI is given. In this case it's highly important we get a high value for this in all cases. As Sebastian has stated in precision and recall lesson that getting a high value of precision is important and having 0.51572 means that 51 percent is classified as actual true POIs. Recall refers how many were classified as POIs from the dataset and it was 0.38 from the dataset.