Identify Fraud from Enron Email

Darren Chiu

Note: All the wraggling on the data and process of experimenting with the machine learning algorithms are in the 'Enron_Data_Exploration' jupyter notebook. It serves as a record of stuff that I tried and some notes throughout the process. It was not intended to be use as documentation but a reference of the thought process.

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 the project is to classify if a person is Person of Interest(POI) in the Enron email and financial details that were released to the public space.
   The dataset was entered into the public record as a result of Federal investigations. The dataset contained information about the payoff of employees and email details of Enron employees.

   After I got the data, I did some examination and found that there is an entry "TOTAL" in the dataset which obviously didn't represent any employee. I removed this record. There was a record with no data at all (LOCKHART EUGENE E), so this is also removed. In each of the financial fields, there are outliers with extreme values, so to avoid the model being affected by these values, I removed the the top and bottom 2% of records of each fields. At the end, 56 records are classified with having at least 1 extreme value under the 2% indicator.

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"]

   There were 4 features being used in the final model - from_this_person_to_poi, from_poi_to_this_person, shared_receipt_with_poi, number_of_missing_fields. For all the selected fields, I did feature scaling using MaxMin scaling method. This is because I was planning to try out SVM algorithm and SVM is scale sensitive.

   I also added the new feature 'number_of_missing_fields' as I found that records with a lot of missing fields are generally not POI.

   For feature selection, I tried three different methods. Firstly, I tried using SelectKBest algorithm but the feature selected using this method are having a low precision score. I have tried select 3, 5, 7 best features, with Gaussian, Decision tree and

SVM, all resulted in a precision under 0.2. The best was able to achieve 0.2-0.24 precision. I did some research and found that SelectKBest assumes that the features are independent so I also tried ExtraTreesClassifier to find out the feature importance as I found on sklearn documentation. However, there were no luck as they are resulting a under 0.2 precision as well. So, I turned into handpicking features. I picked three features concerning the email correspondence with POI, which resulting in >0.3 precision for Gaussian, SVM and Decision tree. For SVM, the precision score hit 0.41. At last, I tried to put in my ‘number_of_missing_fields’ feature and precision score improved to 0.51985.

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 end up using SVM. I tried Guassian-bayes and decision tree as well. From my experiments, it seems that SVM performed the best and Guassian-bayes performed the worst (all using the setting tuned by GridSearchCV). An example with three features used(from_this_person_to_poi and from_poi_to_this_person and shared_receipt_with_poi), decision tree is having a precision score of 0.31790, SVM is having precision score of 0.41795 and Gaussian-bayes is having a score of 0.30.

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”]

   Each algorithm might have more than 1 parameter to set and these are all impacting the performance of a machine learning algorithm. If we didn’t tune it well, some might be causing overfitting or overgeneralization of the model which results in low precision/recall score.

   I used the GridSearchCV to tune the parameter for SVM and Decision tree. For SVM, 2 parameters were tuned, the kernel and C value. C value was tuned against the range of 1000 to 100,000, with stepping size of 1000.

   For decision tree, 5 parameters were tuned, criterion, splitter, max_features, max_depth and min_samples_split. Details of the tuning is available in the jupyter notebook.

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 the process of evaluating the performance of machine learning model. Some common mistakes are using same set of data as training set and test set. This could resulted in an extremely high accuracy and lead to overfitting if someone keep trying to improve the model without splitting data into training and test set.

   I mostly use accuracy, precision and recall evaluating my setup in the tester.py. In tester.py, it used the StratifiedShuffleSplit to break down the data into multiple train/test set for cross validation. Before I send my model to tester.py, I also printed out the precision score and accuracy score to get a rough understanding of the model performance.

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"]

   Two common evaluation metrics are accuracy and precision. My final model is having an average accuracy of 0.83675 and precision of 0.51985.

   For accuracy, it means that out of all the predictions that my model is giving, 83% of the prediction is correct.

   For precision, it means that out of all the prediction that the model classified as POI, 51% of those are real POI.