Identifying Fraud from Enron Emails and Financial Data

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"]

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, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives. This project attempts to predict the likelihood of someone being a suspect of Enron fraud conspiracy by looking at given dataset. We call the suspects Person of Interest (POI). The dataset contains insider pays to all Enron executives as well as emails sent through their company accounts, and their POI status.

The dataset contained 146 records with 14 financial features, 6 email features, and 1 labeled feature (POI). Of the 146 records, 18 were labeled, a priori, as persons of interest. Fifty eight of the people in the dataset had no values for the email features (including 4 Persons of Interest) and 3 people had no values for the finanical features (Ronnie Chan, William Powers, and Eugene E. Lockhart, none of whom are Persons of Interest) and these rows were discarded. Through exploratory data analysis and more of manual inspection, I identified two outliers TOTAL This was an extreme outlier for most numerical features and THE TRAVEL AGENCY IN THE PARK.The final count of records in this relatively small dataset was 141.

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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

Feature selection process includes several iterations. First I used scikit-learn SelectKBest to choose best influential features, in order to use those features for all the upcoming algorithm.

10 most influential features. Their associated scores

Feature	Score
exercised_stock_options	24.25
total_stock_value	23.61
bonus	20.25
salary	17.71
deferred_income	11.22
long_term_incentive	9.62
poi_ratio_messages	9.61
restricted_stock	8.95
total_payments	8.57
shared_receipt_with_poi	8.27

I tried to combine those best features one by one adding to the given rank and applied GaussianNB classifier. In selectKbest we make an assumption that the features that have a high variability between categories will show up as important when we fit a model to the data. The scoring function that we are using on the data is univariate, so it doesn't take into account feature interactions, but it is relatively fast. It also cuts out worthless features that are irrelevant in a first step. Therefore from the result, the combination of top 7 features provided somehow better outcome in both precision and recall, afterwards it started to drop down in either of the evaluation metrics. Even though the classifier gives a higher F-score when k is 20, including all features will overfit the model, due to inclusion of features that might not be important and redundant, in addition the algorithm takes a long time to fit. After examining 5 classifiers on the top 7 features I ended up picking 4 of the top features. So my final feature choice is ['exercised_stock_options', 'total_stock_value', 'bonus', 'salary'].

Precision and recall using GaussianNB based on combined rank

k,	*Precision*	*Recall*	*F-score*
(1,	0.555555555	0.2777777777777778,	0.370)
(2,	0.416666666	0.2777777777777778,	0.333)
(3,	0.428571428	0.3333333333333333,	0.375)
(4,	0.461538461	0.3333333333333333,	0.387)
(5,	0.5,	0.3888888888888889,	0.437)
(6,	0.5,	0.3888888888888889,	0.437)
(7,	0.5,	0.3888888888888889,	0.437)
(8,	0.466666666,	0.3888888888888889,	0.424)
(9,	0.4,	0.3333333333333333,	0.363)
(10,	0.375,	0.3333333333333333,	0.352)
(11,	0.5,	0.3333333333333333,	0.4)
(12,	0.5,	0.3333333333333333,	0.4)
(13,	0.5,	0.3333333333333333,	0.4)
(14,	0.4615384615	0.3333333333333333,	0.387)
(15,	0.4615384615	0.3333333333333333,	0.387)
(16,	0.4285714285	0.3333333333333333,	0.375)
(17,	0.4285714285	0.3333333333333333,	0.375)
(18,	0.4,	0.3333333	0.363)
(19,	0.4,	0.3333333333333333,	0.363)
(20,	0.3793103448,	0.6111111111111112,	0.468)

I engineered a feature, poi_interaction which was a ratio of the total number of emails to and from a POI to the total number of emails sent or received. The reason behind creating Poi_interaction is that, for the potential POI, they will have more communication within their community than other normal people. Although new feature is added in my dataset, but it is not selected, due to the fact that the score of the feature is slightly lower, 9.61. I also checked the correlation between the newly created feature (poi_ratio_messages) and target label using f_regression and find out the score to be 6.75 which is even lower, indicating a weak correlation with poi-ness.

Min-Max scaling was used to also account for the large ranges of values within some of the features (particularly in the financial data). While this wasn't necessary for all of the machine learning algorithms used (i.e., Decision Tree), applying the scaling would allow for greater flexibility and performance in using other algorithms where feature scaling would have an impact (i.e., SVM).

3. What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]

Five different classification algorithms were tested for this effort: Gaussian Naive Bayes, Decision Tree, AdaBoost, KNN, SVM and Random Forest. Adaboost Classifier performed the best of all of the algorithms.

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? (Some algorithms don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Tuning the parameters of an algorithm means to pull some of the 'levers' to influence the results of the model. If you don't tune your parameters appropriately, you will end up using the defaults, which will likely not result in an optimized model. This means the accuracy, precision, recall or other performance measure is not as good as it could be because the model was not customized to the particular dataset.

In my case, GridSearchCV was used to tune Adaboost algorithm using stratified shuffle split cross-validation and I also used pipeline to allow me to do a more reliable GridSearchCV on my workflow.

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 refers to the process of training and testing a machine learning algorithm in order to assess its performance, and to prevent overfitting. Overfitting can occur when the algorithm is fit too closely to the training data, such that it performs really well on the training data, but poorly on any other new unseen data. This is why it is important to always set aside data for testing, since testing on the same data used for training can lead to a misleading assessment of an algorithm's performance. Because machine

learning is often used to try to form predictions about new data, a proper validation strategy is critical.

A single split was applied to train test split. Next I appliedGridSearchCV and StratifiedShuffleSplit to the cross validation set in order to perform parameter tuning. The best estimator is then used to submit to the grader for final testing.

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 main evaluation metrics utilized were Precision and Recall. Precision measures how many of the flagged items are relevant. In this case, precision equals the number of persons correctly identified as POIs (true positives) divided by the total number of persons identified as POIs (positives). With a precision score of 0.67, it tells us that if this model predicts 100 POIs, then the chance would be 67 people who are truely POIs and the rest 33 are innocent. On the other hand, recall will measure how many relevant items are flagged, that is the number of persons correctly identified as POIs (true positives) divided by the number of actual POIs (true positives + false negatives). With a recall score of 0.4, this model can find 40% of all real POIs in prediction. In the case of this project, recall is the most appropriate measure. Since our goal is to find out the real POIs, that is, the actual Person of interest who got involved in the fraud at enron. Due to the unbalanced nature of the dataset (many more negatives than positive) Accuracy is not a good measurement as even if non-POI are all flagged, the accuracy score will result that the model is a success.