**Identify Fraud from Enron Email**

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”]

Our goal here is to use the Enron email and financial database that are publically available, use machine learning algorithms, and identify Enron employees that were likely to be involved in the big scandals. The data set contains some financial information including salary, bonus, stock options, etc as well as email information including the number of emails sent by an employee, the number of emails sent/received from a person of interest, etc. In total the dataset contains the information for 146 employees (we’ll see later that this number is actually 144). There are 21 features present in the dataset. The features have a lot of missing values. Following is a list of features with more than 60 missing values:

loan\_advances: 142

director\_fees: 129

restricted\_stock\_deferred: 128

deferral\_payments: 107

deferred\_income: 97

long\_term\_incentive: 80

bonus: 64

from\_this\_person\_to\_poi: 60

from\_poi\_to\_this\_person: 60

from\_messages: 60

shared\_receipt\_with\_poi: 60

to\_messages: 60

We also have an imbalanced distribution of POIs vs non-POIs. There are 18 POIs and 128 non-POIs. Intuitively, one may think that people who received big money or people who communicated with other POIs could themselves be involved in the fraud.

Looking at the employees’ name, it turned out that there was an employee named “TOTAL”. This row contains the sum of all values from other employees’ data. There is another employee named 'THE TRAVEL AGENCY IN THE PARK' which is not an employee. These are our outliers and I removed them. There were some employees with a lot of missing values but I decided to keep them. Also, there were some other high values of salary/bonus but I decided to keep them as well because here were are looking for those big numbers and correlations with committing a fraud. In other words, our algorithm would perform poorly if I apply a cutoff to salary, stock values, or other financial information.

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”]

While I could intuitively/manually select some features, I decided to use SelectKBest module in sklearn to rank all of the features and get their scores. Here are the scores:

[ 21.32789041 0.20970584 11.73269808 2.0893099 25.3801053

6.37461449 0.15877024 5.44668748 2.47052122 7.30140665

10.22290421 4.26357664 9.4807432 0.06447703 18.86179532

8.90382156 1.75169428 8.96781935 24.75252302 16.87387026

3.29382863]

I then set different cutoff values and checked the performance of the algorithm:

Cutoff: 2, # of features: 18, precision: 0.36615, recall: 0.29750

Cutoff: 2, # of features: 17, precision: 0.36827, recall: 0.32850

Cutoff: 5, # of features: 13, precision: 0.46676, recall: 0.34050

Cutoff: 8, # of features: 10, precision: 0.42839, recall: 0.34100

Cutoff: 10, # of features: 7, precision: 0.45003, recall: 0.36700

Cutoff: 12, # of features: 5, precision: 0.47440, recall: 0.34750

Cutoff: 18, # of features: 4, precision: 0.43545, recall: 0.33900

We can see that the best accuracy was obtained with a cutoff value of 10 and 12. I decided to choose 10 as my cutoff. Using this method I kept 7 features. I looked at the selected features and they made perfect sense to me. Here’s the list of the selected features:

['bonus', 'deferred\_income', 'exercised\_stock\_options','long\_term\_incentive', 'salary', 'total\_stock\_value', 'to\_poi\_ratio']

I created two new features. 1) “to\_poi\_ratio”: the portion of emails that a person sent to a poi. This feature was calculated through dividing “from\_this\_person\_to\_poi” by “from\_messages”. This new feature shows the fraction of the emails that a person sent to a poi. While the absolute value of “from\_this\_person\_to\_poi” contains some information, it could be misleading because some people tend to send a lot of emails and some other people tend to send much less. The ratio here shows the fraction of emails that was sent to a poi. 2) “from\_poi\_ratio”: the portion of emails that a person received from a poi. This was calculated through division of “from\_poi\_to\_this\_person” and “to\_messages”. Again a person can receive a lot of emails. What matters is the fraction of those emails from a poi. Therefore, these new features could be pretty important in identifying a POI. These features got good scores in my feature selection step: 16.87387026 and 3.21076192 but “from\_poi\_ratio” was eventually dropped.

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 decided to use NaiveBayes in the end because it was the fastest method and it satisfied our target precision and recall values. I tried other algorithms such as DecisionTree, SVM, and RandomForest. SVM and RandomForest took a very long time to run (the testing didn’t finish within a reasonable time period) so I decided to ignore them. While DecisionTree (without rescaling the financial features) satisfied our target (Precision: 0.33205, Recall: 0.30100), it was slower and the performance was lower compared to NaiveBayes (precision: 0.45003, recall: 0.36700).

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 uses a method to optimize its performance. Performance in our case means making correct classification decision. Depending on the nature of the algorithms, there could be several tunable parameters to guide them how to make decisions (e.g. when to stop branching the data) or select boundaries. By tuning the parameters, we essentially test different parameters and pick the ones that gives the best performance. Our algorithm can perform poorly, run slowly, or it could be prone to over-fitting if we don’t do this well. NaiveBayes doesn’t have any tuning parameters but for the DecisionTree I used GridSearch to tune the parameters. I tuned it for 'min\_samples\_split': [2, 5, 10], and 'max\_features': [5, 10, len(features\_list)-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 means assessing whether our algorithms is doing what it is supposed to do. In other words, validation essentially means evaluating the performance of our algorithms. To do this, we split our data into training and testing subsets. This creates an independent dataset to test the performance of our algorithm, and it helps prevent overfitting. If we test and train on the same data, we will most definitely overfit our model. Another mistake that one can make is the case where you have imbalanced data (like in this project). If you have an imbalanced dataset, i.e. you have only a few items in one of your classes (POI) and many more items in your other class (non-POI), you can easily fool yourself by calculating the score (number of correctly classified items divided by total number of items). Since we only have a handful of POIs, our score will always be high no matter how poor our algorithm performs. Here I used precision and recall to evaluate the performance of my algorithms.

In this project, I used “train\_test\_split” in sklearn to split my data into training and testing subsets. This module randomly selects a fraction of the dataset (defined by “test\_size”) and assigns it to the testing subset. The rest goes to training subset. We then train our data using training subset and test it using testing subset.

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”]

Precision: 0.45003

Recall: 0.36700

Precision value of 0.45003 means: if my algorithm predicts that someone is a POI, there is 45.003% percent chance that the person is actually a POI.

Recall value of 0.36700 means: if a person actually is a POI, the probability of predicting that using my algorithm is 36.700%.