**Enron Submission Free-Response Questions**

A critical part of machine learning is making sense of your analysis process and communicating it to others.

[Link to the rubric](https://www.google.com/url?q=https://docs.google.com/a/knowlabs.com/document/d/17-JwNQH1aRxtqMkJ6zpCL_68kh5F6uSbDXcJS26vZWY/pub&sa=D&ust=1481913623103000&usg=AFQjCNE11svhW0PoD_P-nHRm_JyYm0RCkg)

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 flying high as one of the largest companies in the US of A. By 2002 though it had abruptly fallen into bankruptcy. During the ensuing Federal witch-hunt, a large amount of confidential information was dispersed into public record (i.e. the internet). This data included tens of thousands of emails and detailed financial data for top executives throughout the company. Udacity has been to kind as to find a dataset which organizes this info into an easy to understand form. This data set includes but is not limited to:

**financial features**: ['salary', 'deferral\_payments', 'total\_payments', 'loan\_advances', 'bonus', 'restricted\_stock\_deferred', 'deferred\_income', 'total\_stock\_value', 'expenses', 'exercised\_stock\_options', 'other', 'long\_term\_incentive', 'restricted\_stock', 'director\_fees'] (all units are in US dollars)

**email features**: ['to\_messages', 'email\_address', 'from\_poi\_to\_this\_person', 'from\_messages', 'from\_this\_person\_to\_poi', 'shared\_receipt\_with\_poi']

This dataset also includes a hand-generated list of 18 persons of interest in the fraud case out of a total of 146 persons in the data set. These persons of interest were selected based upon being indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

There was a lot of missing data within this dataset. Below are the total amounts of missing values for each feature.

{'salary': 51, 'to\_messages': 59, 'deferral\_payments': 107, 'total\_payments': 21, 'long\_term\_incentive': 80, 'loan\_advances': 142, 'bonus': 64, 'restricted\_stock': 36, 'restricted\_stock\_deferred': 128, 'total\_stock\_value': 20, 'shared\_receipt\_with\_poi': 59, 'from\_poi\_to\_this\_person': 59, 'exercised\_stock\_options': 44, 'from\_messages': 59, 'other': 53, 'from\_this\_person\_to\_poi': 59, 'deferred\_income': 97, 'expenses': 51, 'email\_address': 34, 'director\_fees': 129}

These missing values could lead to problems with our algorithm. For example, about 14.38% of have “NaN” written for their total payments. None of those with “NaN” written for their total payments were POIs. For this reason, a machine-learning algorithm would most likely associate a “NaN” value with non-POIs. This could lead to our algorithm associating NaN with non-POIs.

The goal of the project was to create a way to accurately predict if an Enron employee is a person on interest and should be investigated for fraud in the scandal. Since the dataset provides a large amount of features about each person and we have a list of persons of interest we can use supervised machine learning create a discrete output of whether a person should be labeled as a person of interest and burned at the stake.

In the dataset there was an outlier in the dataset. A so called person, “TOTAL”, had a ‘salary’ of $26,704,229. This was an extremely high salary, even for the witches practicing dark moneymaking magic at Enron. I found this data point to be an outlier based upon a spreadsheet quirk and removed it.

1. 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 importance 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”, “properly scale features”, “intelligently select feature”]

I ended up using the following features: 'poi','fraction\_to\_poi', 'fraction\_from\_poi', 'fraction\_bonus\_salary', 'fraction\_total\_stock\_value\_salary'. I use POI to allow for my supervised machine learning algorithms to check against the true answer. I created the fraction to poi and fraction from poi since I wanted to get a better understanding of the true amount a person had email contact with a poi. It would be a lot more incriminating if a person sent 30 emails out of 40 emails to poi rather than 30 emails out of 100 emails. I also created the fraction of bonus to salary and fraction of total stock value to salary based upon an assumption that those who helped orchestrate the Enron fraud would likely have lined there pockets with large bonuses and stock much greater than a normal salary would allow. Using these features instead of bonus, salary, total stock value, fraction this person to poi and from poi to this person alone increased my precision by about .06 with my final algorithm.

I used a MinMaxScalar since some of the fraction\_bonus\_salary and fraction\_total\_stock\_value\_salary were between 1 and 2. The scalar was not completely needed but helped to fine tune the algorithm.

I did not use a feature selection tool for my final algorithm I chose.

1. 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 used a support vector machine fine-tuned with a GridSearchCV function optimized to the F1 score to create the best performance overall. By best performance I mean a reasonably high recall but a very high precision since I wanted to get smallest amount of false positives as possible. I also tried Gaussian Niave Bayes algorithm which performed better than the support vector machine for recall but worse for precision. I also tried a decision tree which had a more even result for precision and recall but has worse precision than the svm. I also tried to apply a principal component analysis algorithm to the support vector machine suing pipline but the precision and recall was worse than the svm alone.

1. 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 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 item: “tune the algorithm”]

Tuning an algorithm means changing the parameters to create the best optimized prediction output for your particular set of data. If you do not tune the parameters of an algorithm well you could significantly decrease the performance of prediction power of your algorithm.

I tuned my support vector machine algorithm by allow the gridsearch cv to find the best parameters values optimized for the f1 score.

1. 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 tests if your algorithm is performing well and most people make a mistake by only running an accuracy test without running and recall and precision test.

Validation allows a data analyst to test whether or not an particular algorithm is performing well or is failing to produce useful results. Many people make the classic mistake of running validation testing on the exact same data that they trained their algorithm on. This can lead to much higher performance results during validation testing than what the actual algorithm should produce. I validated my analysis by running the tester.py file checking the accuracy, recall, precision and f1 score.

1. 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”]

I achieved a precision of about .47 and a recall of about .34 with an overall accuracy of .81 for my final svm algorithm. I preferred this algorithms performance over all the other algorithms I ran since it had the highest precision. I wanted to make sure that my algorithm had the least amount of false positives as possible since this algorithm would potentially be used to find persons of interest for detectives to investigate. I would not want to have a large amount of people investigated who were not involved in the fraud at all. The Gaussian Naive Bayes algorithm I employed had a much larger recall than the svm but a very low precision. For this reason it would be much more likely to find more true positives than my svm but also more likely to have false positives aswell.