Enron Submission: Responses to questions

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 the given financial and email information for selected Enron employees to generate a model that will help predict if an individual is a "person of interest" (POI for short) or not. A person of interest in the Enron fraud case refers to an individual who was found guilty of wrongdoing – reaping financial gain by accessing or hiding important financial or market-related information, hiding from or misinforming stakeholders on financial information, or colluding with such persons or not disclosing information related to such individuals to further their interest. The goal of this project is to analyze the given data set to see if there are any "markers" that can be isolated for persons of interest in order to distinguish them from other, non-POIs. Machine learning is useful here in two ways: (1) It provides us with different tools that allow for a large number of "features" or attributes, as well as large volumes of data to be ingested, in order to help identify important features or markers and build models to aid our investigation in a quick, automated, scientific manner. Doing the same by hand is a time-consuming and error-prone task. (2) Given a set of training data, machine learning tools "learn" and store these markers, which can then be applied to test, or unseen data to help validate these rules as well as to predict who maybe potential persons of interest. (3) Machine learning tools allow for more than one way to analyze the data by providing different kinds of models with flexibility of tuning parameters, that can then be used together to produce the best results.

   The data set has 146 data records or rows, and 21 columns. This is a small data set. The row identifier is the full name of the Enron individual and the columns represent their financial data (bonus, deferral payments, deferred income, director fees, exercised stock options, expenses, loan advances, long term incentive, other expenses, restricted stock, restricted stock deferred, salary and total stock value) and email meta-data (email address, no. of from messages, no. of to messages, no. of messages from POIs, no. of messages to POIs, no. of shared receipts with a POI) and finally, the POI/non-POI "label". Every feature includes 'NaN' values as seen with the describe() function, except POI which is either true or false. Counts of nans for each feature is given below:

Feature	Count of NaNs (unavailable values)
Salary	51
To messages	60
Deferral payments	107
Total payments	21
Loan advances	142
Bonus	64
Email address	35
Restricted stock deferred	128
Deferred income	97
Total stock value	20
Expenses	51
From poi to this person	60
Exercised stock options	44
From messages	60
Other	53
From this person to POI	60
POI	0
Long term incentive	80
Shared receipts with POI	60
Restricted stock	36

Director fees	129

There are 128 non-POIs and 18 POIs in the data set which means the data set is highly skewed towards non-POIs. This also means that the Stratified Shuffle Split method of cross-validation provided will be effective in maintaining the ratio of POI vs. non-POI records during training and testing. Empty values or NaN values were replaced with 0 so as to avoid guessing and biasing the analysis. In order to examine outliers, I plotted a scatter plot for salary vs. bonus and immediately noticed a record with a massive bonus value. This outlier record happened to be a “TOTAL” figure and was eliminated from the analysis. Other scatter plots of salary vs. other financial features did not reveal any outliers. Empty values or NaN values were replaced with 0.

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”]*

I created 4 new features:

(1) Fraction of messages from POI – Messages to this person from a POI/Total number of this person’s To messages. This feature makes sense because a higher fraction increases the possibility that this person may be a POI themselves.
(2) Fraction of messages to POI – Messages from this person to a POI/Total number of this person’s From messages. This feature makes sense because a higher fraction implies increased correspondence with a POI and thus increases the possibility that this person may be a POI themselves.
(3) Stock vs. salary – Total stock value/salary. A higher value here implies this person has a higher vested interest in ensuring that the Enron stock value stays as high as possible, and therefore increases the possibility of cooking or helping to cook the books for maximum financial gain!
(4) Other vs. salary – Other payments/salary. Similar rationale as above. If an individual is paid large sums of non-expense money as a fraction of their base salary, it might imply some of these payments are likely to be towards highly sensitive, confidential tasks including financial misappropriation or market misinformation. A high risk carries an abnormally high reward.

In addition, I dropped “director fees” and “loan advances” because they were too sparse or because they were accounted for in other features I used, or because I gleaned from their definition (given in the data dictionary) that they could not be reasonably connected to POI identification. After splitting the data set into train and test, I used the “SelectKBest” algorithm on the training set to pick the 10 best features and the f_classif parameter. 10 appeared to be a good starting point to pick up the statistically significant features. If all 10 features had high enough scores or low enough p-values, I would increase k, but this was not the case. I was looking for p-values no greater than 0.1 for statistical significance and the top 10 features ordered by increasing values of p-value satisfied this criteria. The f-classif parameter takes the ANOVA F-value between label/feature for classification tasks. The 10 features were identified as follows, and I can see here that only 1 of my 4 newly engineered features was ranked high enough by SelectKBest:

	Feature	Score	p-value
1.	bonus	11.09	0.001
2.	deferred income	10.18	0.002
3.	salary	9.34	0.003
4.	fraction_messages_to_poi	8.19	0.005
5.	shared_receipt_with_poi	5.29	0.024
6.	total_stock_value	3.74	0.056

7.	long_term_incentive	3.43	0.067
8.	exercised_stock_options	3.05	0.084
9.	expenses	3.04	0.084
10.	from_poi_to_this_person	1.85	0.176

I attempted 3 algorithms – Gaussian Naïve Bayes, SVM and Decision Tree classifiers. Standard scaling was used for SVM. This is important as all the features have a different range and different order of values. In order to ensure that all features contributes equitably to the classification process, we scale them so that the feature values are scaled relative to a mean of 0 and standard deviation of 1. The first two algorithms do not require scaling.

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 tried using three algorithms: (1) Gaussian Naïve Bayes (2) SVM, and (3) Decision tree. The Gaussian Naïve Bayes and Decision Tree algorithms performed better than the SVM classifier as they both produced better than the required threshold parameters for recall and precision of 0.3. The model performance for each was as follows (using the test_classifier method from tester.py):

	Model	Performance
1.	Gaussian Naïve Bayes	Accuracy: 0.849; Precision: 0.463; Recall: 0.367; F1: 0.409; F2: 0.382
2.	Decision tree	Accuracy: 0.826; Precision: 0.384; Recall: 0.356; F1: 0.37; F2: 0.362
3.	SVM	Accuracy: 0.752; Precision: 0.25; Recall: 0.367; F1: 0.297; F2: 0.335

   If I were to pick one algorithm, I would choose the Gaussian Naïve Bayes classifier, as it has the better overall F1 and F2 scores.

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"]*

   To tune the parameters means to specify certain behavioral parameters for the algorithm. For example, parameter tuning could involve setting certain threshold values, specifying the linearity or degree of non-linearity of the decision boundary, specifying how many features should be considered, specifying the min/max no. of iterations for the model to converge, min/max no. of leaf nodes, penalization parameters etc. Tuning the parameters of an algorithm is vital so as to get the optimum performance from the algorithm for the data set in question. The aim is to get the best trade-off for speed and accuracy of the model. My chosen Gaussian Naïve Bayes algorithm did not require parameter tuning. One of the other algorithms I tried, the SVM classifier, did.

   In order to tune the parameters for the SVM, I first read up the sklearn documentation to see which values might make sense given that we have a highly skewed number of POI/non-POI labels. I also understood what the separation boundary might look like given various parameter values. To choose the optimum parameters, I used GridSearchCV with an SVC classifier which was initialized with a kernel = rbf, class_weight=balanced and random_state=20. A class_weight = balanced, according to the sklearn documentation, "uses the values of y to automatically adjust weights inversely proportional to class frequencies". This is desirable in our case due to the small proportion of POI vs. non-POI individuals. The random state parameter is a fixed but random number that enables reproducible results after shuffling. The GridSearchCV was configured to achieve the best results based on different input values of C and gamma, yet the SVM classifier did not produce a high enough value for recall.

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 how well our model performs on “unseen” or test data, i.e. data that has not been used for training or preparing the model in any way. A model will be deemed “valid” if it produces better than (one or more) pre-defined threshold parameter values such as accuracy, recall, precision, F1 score, F2 score etc. when used on unseen test data. A classic mistake is that the same data gets used for both, training and testing. Because the model has already “seen” the testing data as part of its creation, the consequence is that the model now overfits that data, i.e. we have configured the model to have a very high classification accuracy on the data. As a result, it performs much worse on completely unseen or new test data. In order to avoid this mistake, given a data set, we must first split it into a training set and test set, fit the model algorithm on the training data only, and validate it by using the test data previously unseen by the model. Further, when fitting the model to the training data, we aim for the decision boundary on the training data to be a good trade-off between accurate and simple, so that the most optimum performance is carried over to test data sets as well.

   I validated my algorithm by (i) dividing the original data set into training (70%) and test (30%) data, (ii) Using the training data only to build the model, and (iii) Checking model performance on the test data using the given test_classifier method from tester.py. This method internally used 1000-fold cross validation via Stratified Shuffle Split, which evaluated the model on multiple different sets of data taken from the original data set, and averaged the model results to yield final performance results.

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 – The Gaussian Naïve Bayes algorithm delivered a precision of 0.463. Precision is defined as:

     Correct positive classifications/Total positive classifications = TP/TP+FP. This means that out of all individuals that the model classified as POI, just over 46% of these were correct classification (Here, correctness is validated by comparing the model results with the actual labels for these individuals).

   - Recall – The Gaussian Naïve Bayes algorithm delivered a recall of 0.367. Recall is defined as:

     Correct positive classifications/Total true positive classifications = TP/TP+FN. This means that out of all POIs in the data set, the algorithm correctly identified nearly 37% as POI. Again, correctness is validated by comparing the model results with the actual labels for these individuals.