DETECTING ENRON FRAUD – FREE RESPONSE

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 use the available dataset to investigate data about Enron employees for the purpose of identifying people who participated in the fraudulent financial practices at the company. Machine learning techniques can help by identifying potential similarities between people already known to be involved and other employees. The original dataset used in this investigation consist of 14 financial and 5 email-related parameters (features), combined with the name, email-address and known category (POI - person-of-interest (18 instances) or non-POI (128 instances)) for 146 employees which have been investigated by the authorities. A few major outliers are present for all parameters - the "TOTAL" and "THE TRAVEL AGENCY IN THE PARK" which are not employee entries and have therefore been removed. More specific anomalies (e.g very large payments to "LAY KENNETH L" or salary-to-stocks ratios for "BANNANTINE JAMES M") were also identified, but since the data appears correct these have not been removed in the initial cleaning. Unfortunately, multiple gaps are present in many of the features, for example 58 people don't have any of the email-related features. Some of the financial features also have significant gaps ("NaN" values), for example only 3 people have defined values for loan advances. Based on the primary source ("enron61702insiderpay.pdf"), the "NaN"s in the financial data can be interpreted as 0 and have been filled, eliminating the gaps. The gaps in email data have not been rectified.

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"]

The feature discreationary_cashout_ratio was engineered to reflect the amount of cash withdrawn from the company vs the total assets of the employee in the company. It is based on the thinking the persons-of-interest would be keen to "cash-out" their earnings (legal or not) as soon as possible to avoid losing money in potential bankruptcy proceedings or other. Two simpler features, 'poi_to_ratio' and 'poi_from_ratio' were also introduced to provide a normalized value for the emails involving persons of interest, idependent of the actual number of emails the person is sending or receiving.

None of the methods used (Naive Bayes, Random Forest and AdaBoost) needed scaling between the features so none was implemented. The features used in the were manually optimized by an initial "educated guess" based on human reasoning, and using the feedback for feature importance provided by the two ensemble methods. For these, all parameters were also evaluated as a reference point, together with the educated guess tuning. Interestingly, all email parameters seemed to result in a worse precision and recall for the algorithm. This is likely due to the multiple gaps in the email dataset, which leads to less values being available for the financial parameter evaluations as well. The final set of

parameters with peak performance for Naive Bayes was 'exercised_stock_options', 'salary', 'deferred_income'.

Table 1: Importance of various feature combinations

Features	**NaiveBayes**		**AdaBoost (best F1)**		**Random Forest**	
	Prec	**Recall**	**Prec**	**Recall**	**Prec**	**Recall**
All Original Feautres	0.2	0.405	0.408	0.255	0.414	0.12
All Features (original + custom)	0.2	0.405	0.179	0.585	0.519	0.135
Financial features (original)	0.22	0.425	0.466	0.345	0.617	0.185
Email features (original)	N/A	N/A	0.26	0.29	0.164	0.11
discretionary_cashout_ratio + poi_to_ratio + poi_from_ratio	0.181	0.085	0.134	0.86	0.33	0.16
exercised_stock_options +salary + deferred_income + poi_to_ratio + poi_from_ratio	0.583	0.37	0.34	0.275	0.427	0.22
exercised_stock_options + salary	0.59	0.24	0.466	0.24	0.462	0.335
exercised_stock_options + salary + deferred_income	0.606	0.4	0.439	0.255	0.552	0.265
exercised_stock_options + salary + deferred_income + discretionary_cashout_ratio	0.558	0.41	0.389	0.255	0.452	0.19

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"]

The final algorithm resulting in best performance was the NaiveBayes, with other tried including Random Forest, AdaBoost (with NaiveBayes or DecisionTree as base estimator) and SVM. SVM was rejected at the early stages due to none of the kernels being able to correctly identify any of the POIs in the training set. Random Forest and AdaBoost also performed well, however interestingly Random Forest was able to only successfully consider fewer features, but with higher scores for these reduced feature sets. AdaBoost on the other hand successfully dealt with a large number of features, especially when using the NaiveBayes for base estimator (see *Table 1*)..

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"]

Some algorithms require tuning, i.e. the determination of some coefficients to adapt the algorithm to the dataset. The AdaBoost algorithm for example can take different base estimators (in this case NaiveBayes or the default DecisionTree were used), as well as a different number of learning rates and estimators. GridSearchCV was used in combination with StratifiedKFold validation (see below) to optimize the parameters for AdaBoost. As demonstrated by the large difference in the scores for different parameter combinations, tuning the parameters is important since the wrong parameters can prevent an otherwise working model from providing good results (see *Table 2*).

Table 2 Grid Search results from parameter optimization

	base_estimator	n_estimators	learning_rate	fit_time	test_score (F1)
1	DecisionTree()	10	1	0.015946	0.316667
2	DecisionTree()	50	1	0.073982	0.297619
3	DecisionTree()	100	1	0.144631	0.275
4	DecisionTree()	10	5	0.014012	0.231477
4	DecisionTree()	50	5	0.070495	0.231477
4	DecisionTree()	100	5	0.138628	0.231477
7	GaussianNB()	100	1	0.129341	0.216667
8	GaussianNB()	10	0.1	0.01413	0.194444
8	GaussianNB()	50	0.1	0.062847	0.194444
8	GaussianNB()	100	0.1	0.125221	0.194444
8	DecisionTree()	10	0.1	0.015097	0.194444
8	DecisionTree()	50	0.1	0.065134	0.194444
8	DecisionTree()	100	0.1	0.14429	0.194444
14	GaussianNB()	10	1	0.012309	0.175926
15	GaussianNB()	10	5	0.013139	0.145833
16	GaussianNB()	50	1	0.07048	0.141667
17	GaussianNB()	50	5	0.048373	NaN
18	GaussianNB()	100	5	0.083023	NaN

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 an evaluation of the performance of the algorithm on new data which it hasn't seen before. A typical mistake which can happen if the data is not split like this is that the model may be "overfit" to the existing data, predicting very accurately for data points where it has already been shown the answer, but not capable of determining the right category for new data points. In this analysis, a simplistic split into a training set (to be used to fit the model) and test set (only used for evaluating the performance) was used, with 30% of the data reserved for test, as well as a stratified K-Fold split which splits the dataset into several subsets while maintaining the ratio of POI and non-POI in each subset. Due to the significant discrepancy between the scores of both methods, the reference implementation of Stratified Shuffle Split was also used (with a reduced number of folds to improve runtime).

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 two relevant evaluation metrics used were precision and recall.
Precision score - out of the items classified POI by the algorithm, 57.9% are indeed POI
Recall score: - out of the items labeled POI in the test set, the algorithm identified 35.5%.
As a reference point, 12.5% of the datasets are actual persons of interest, and the algorithm is doing significantly better than random chance.