Q1. 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”]

This project is to identify Enron employees who may have committed fraud based on the public Enron financial and email dataset. The machine learning algorithms are suitable to detect implicit patterns in a large amount of data without explicit methodology. With given the data on which the fraud is marked, we have to classify unknown fraud from the data. This corresponds to the supervised machine learning.

The data has 146 data points, and each data point consists of 21 features including POI mark. It requires attention that the amount of data is insufficient, and also lots of points have ‘NaN’ values. When dividing the dataset to training and test set, and selecting the validation metrics, it should be considered that the data point marked as a POI is only 18 points. There were three outliers, ‘TOTAL,’ ‘LOCKHART EUGENE E,’ and ‘THE TRAVEL AGENCY IN THE PARK.’ They were removed before any other processing.

Q2. 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 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”, “intelligently select features”, “properly scale features”]

At first, features with more than 100 non-NaN values are selected. They are 'total_payments', 'total_stock_value', 'exercised_stock_options', and 'restricted_stock.' At a first glance, 'from_poi_to_this_person' and 'from_this_person_to_poi' seemed to be important features. However, if a person is a heavy email user, he/she might have high 'from_poi_to_this_person' or 'from_this_person_to_poi' value without relevance to POI. Rather than the absolute values, the ratio of messages from/to POI is more important. Therefore, I made new features, 'fraction_from_poi', 'fraction_to_poi.’

I applied the SelectKBest to all data points with all features except ‘email_address.’ The score for features all listed below in Table 1. The features with more than a score of 10 are also added to the feature list. The updated feature list contains ‘salary,’ ‘bonus,’ ‘deferred_income,’ and 'fraction_to_poi.’

Table 1

salary	18.2897	exercised_stock_options	24.8151

to_messages	1.64634	from_messages	0.169701
deferral_payments	0.224611	other	4.18748
total_payments	8.77278	from_this_person_to_poi	2.38261
loan_advances	7.18406	long_term_incentive	9.92219
bonus	20.7923	shared_receipt_with_poi	8.58942
restricted_stock_deferred	0.0654997	restricted_stock	9.21281
deferred_income	11.4585	director_fees	2.12633
total_stock_value	24.1829	fraction_from_poi	3.12809
expenses	6.09417	fraction_to_poi	16.4097
from_poi_to_this_person	5.24345		

A random separation of the dataset to training and test set could lead to poor performance of an algorithm. Because the portion of the POI is too small, the test set could have a too small number of POI points. To preserve the portion of POI in training and test set, I used StratifiedShuffleSplit.

Q3. 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 several supervised classification algorithms, Gaussian Naive Bayes, decision trees, random forests, and AdaBoost. Among them, My pick is Gaussian Naive Bayes, because it shows good performance with a small number of data points. Other algorithms tended to overfit the data.

Q4. 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”]

Gaussian Naive Bayes showed satisfactory result at the first trial. Though it does not have parameters to tune the performance, the model can be enhanced even more by preprocessing procedures. Some feature has a range of ~1e6, on the other hands, some feature has a range of ~1. To prevent the algorithm is dominated only by the features with a

big range, the all features should be normalized. Although the Gaussian Naive Bayes shows good performance for a small dataset, I should be cautious about the possibility of overfitting. I could reduce the risk of overffing by dimensionality reduction.

I chose MinMaxScaler for a feature scaling and PCA for dimensionality reduction. The optimized parameter for PCA, i.e., n_components is searched using GridSearchCV. By using the MinMaxScaler and optimized PCA(n_components=5) together, the precision score increased by 0.009 and the recall score increased by 0.017.

Q5. 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"]

By evaluating the trained model on the test set, I can get an estimation of the error rate of my model. If the training and test datasets overlap each other, the model will be overfitted and show low performance on the real field. If the separation of the dataset is done manually, the model can be trained in a biased way. I avoided these issues by using the test_classifier in tester.py. One more benefit of using this method is that the evaluation result will be the same with the project evaluation. One thing to note is that we can not use accuracy for this project, because the data set is skewed. POI is only 18/143=13% of whole data points, so even the estimator classify all data points to non-POI, the accuracy score will be 0.87.

Q6. 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"]

A precision, recall, and f1 is suitable metrics for this project. My final model has 0.43 for a precision score, and it means that about 43% of the predicted POI is the actual POI. Also, my model has 0.33 for a recall score, and it means that it detects about 33% of the actual POI.