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.

The goal of this project is to develop a method to identify which Enron top executives may have committed fraud based on the data provided. The 21 features in the data include financial features, email features and a person of interest (POI) label. A POI is identified as someone who was indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity – our indicator of someone who may have committed fraud. Out of the 146 people in the data set, 18 people are positively identified as a POI.

To narrow down the number of features to further investigate, the percent of all people, POIs, and non-POIs with missing values for each feature are calculated. The financial features that have <50% missing for all people and POIs are: salary, total_payments, bonus, total_stock_value, expenses, exercised_stock_options, other, and restricted_stock. Additionally, the deferred_income and long_term_incentive features are included since they are missing for more than 50% of the non-POIs, but less than 50% were missing for POIs. An entry for these features may be an indicator of a POI. The deferred_income feature is one of the top features (see question on feature selection). The actual fractions of NaNs for each feature is shown in the table below:

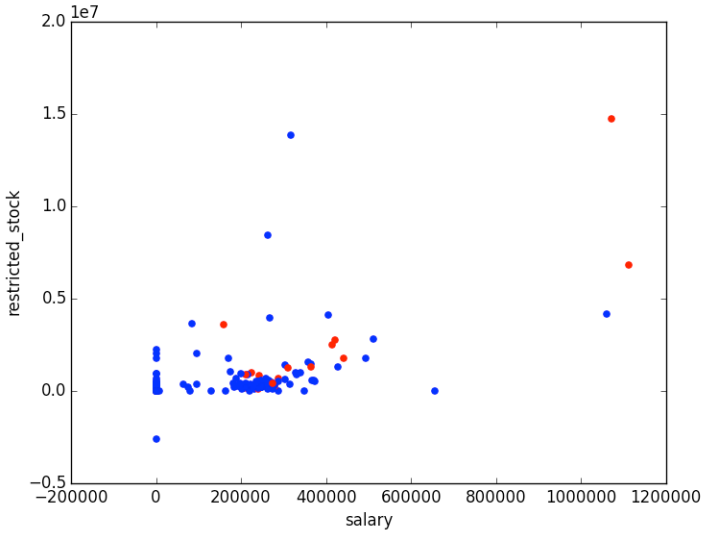
feature	fraction NaNs for all	fraction NaNs for POI	fraction NaNs for non-POI
salary	0.35	0.06	0.39
deferral_payments	0.73	0.72	0.73
total_payments	0.14	0.00	0.16
exercised_stock_options	0.30	0.33	0.30
bonus	0.44	0.11	0.48
restricted_stock	0.25	0.06	0.27
restricted_stock_deferred	0.88	1.00	0.86
total_stock_value	0.14	0.00	0.16
expenses	0.35	0.00	0.40
loan_advances	0.97	0.94	0.98
other	0.36	0.00	0.41
director_fees	0.88	1.00	0.87
deferred_income	0.66	0.39	0.70
long_term_incentive	0.55	0.33	0.58

In summary, the following financial features are investigated further:

1. salary
2. total_payments
3. bonus
4. deferred_income
5. total_stock_value
6. expenses
7. exercised_stock_options
8. other
9. long_term_incentive
10. restricted_stock.

Were there any outliers in the data when you got it, and how did you handle those?

By printing out the keys in the data set, It was found “TOTAL” and “THE TRAVEL AGENCY IN THE PARK” are listed as individuals. As they are groups, they are removed. Other outliers were searched by plotting each feature against salary and coloring POIs as red dots and non-POIs as blue dots. The restricted stocks for BHATNAGAR SANJAY is the negative value found in enron61702insiderpay.pdf. This was found when plotting salary vs restricted_stock, as seen in the figure below. The entry is kept, but fixed. Additionally, using the featureFormat function removes seven people that had zeroes for all features.



As part of the assignment, you should attempt to engineer your own feature that doesn't come ready-made in the dataset--explain what feature you tried to make, and the rationale behind it.
Two new features are created, using the email data: fraction_to_poi and fraction_from_poi. The fraction of a person's emails related to another POI might be more useful than the pure number of e-mails related to a POI since some people may e-mail a lot or very little. The fraction_to_poi feature is one of the top features (see question on feature selection). Without the fraction_to_poi feature, final classifier with scaling produces the following metrics:
Accuracy: 0.85467 Precision: 0.43333 Recall: 0.29250 F1: 0.34925
With the fraction_to_poi feature, the final classifier with scaling produces the following metrics:
Accuracy: 0.87080 Precision: 0.52214 Recall: 0.36550 F1: 0.43000

It is clear that the addition of fraction_to_poi helps the final significantly increases precision, recall, and f1 scores. It also slightly increases accuracy.

What features did you end up using in your POI identifier, and what selection process did you use to pick them?
Features used for the POI identifier are salary, bonus, deferred_income, total_stock_value, expenses, exercised_stock_options, and fraction_to_poi. Three feature selection algorithms (SelectKBest, RandomizedLasso, and RFE) are used to select ten important features from the ten listed financial features above and the two new features, fraction_to_poi and fraction_from_poi. There is some disagreement, but the seven kept are agreed to be amongst the most important features. The table below lists the results from the tests. The seven features that were selected amongst the top ten in all three feature selection tests are highlighted. The scores from SelectKBest and RandomizedLasso also show the disagreement in ranking each feature.

Feature	SelectKBest	Score	RandomizedLasso	Score	RFE
salary	T	18.29	T	0.45	T
total_payments	T	8.77	F	0.14	F
bonus	T	20.79	T	0.81	T
deferred_income	T	11.46	T	0.90	T
total_stock_value	T	24.18	T	0.42	T
expenses	T	6.09	T	0.92	T
exercised_stock_options	T	24.82	T	0.62	T
other	F	4.19	T	0.75	T
long_term_incentive	T	9.92	T	0.54	T
restricted_stock.	T	9.21	F	0.24	F
fraction_from_poi	F	3.12	T	0.31	T
fraction_to_poi	T	16.41	T	1.00	T

Did you have to do any scaling? Why or why not?
Scaling was used on the features prior to feature selection since the values for the features span many orders of magnitudes (10^{-2} to 10^{7}). Scaling allows the proper relative significance of each feature.

If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.
A decision tree was not used to determine the important features.

What algorithm did you end up using? What other one(s) did you try? The AdaBoost, RandomForest, Bagging, ExtraTrees, AdaBoost with ExtraTrees, and AdaBoost with RandomForest classifiers were tried out of the box using the test_classifier function. The seven features are scaled before the classifiers were tested. The AdaBoost classifier is chosen for tuning it has a high F1 score and the recall and precision scores are already above 0.3. Below are the metrics from the default classifiers using the selected features:

AdaBoost
Accuracy: 0.86553 Precision: 0.49370 Recall: 0.33300 F1: 0.39773
RandomForest
Accuracy: 0.86833 Precision: 0.51739 Recall: 0.18600 F1: 0.27363
Bagging
Accuracy: 0.85200 Precision: 0.38565 Recall: 0.18550 F1: 0.25051
ExtraTree
Accuracy: 0.85953 Precision: 0.42721 Recall: 0.15700 F1: 0.22962
AdaBoost with Random Forest
Accuracy: 0.86300 Precision: 0.46165 Recall: 0.16550 F1: 0.24365
AdaBoost ExtraTree
Accuracy: 0.85987 Precision: 0.42956 Recall: 0.15550 F1: 0.22834

What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well?
To tune the parameters of an algorithm is to find a set of parameters that optimize an algorithm to a given data set. The parameters can change the complexity of the algorithm. Without tuning, an algorithm might not perform as well as it could since the initially chosen parameters may not be suited for the problem.

How did you tune the parameters of your particular algorithm? (Some algorithms don't 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 if you used, say, a decision tree classifier).
Attempts are made at using GridSearchCV for parameter tuning. However, the resulting parameters from this function (using the precision, recall, or f1 score as the performance metric) actually resulted in lower precision and recall scores. Consequently, I created a version of this that loops through the AdaBoost classifier parameters, and calls the test_classifier function (with only 250 folds) used for the final analysis. From the suite of simulations, I choose the set that produces the highest F1 score, given that the precision and recall scores are at least 0.3.

The parameters tuned for AdaBoost are:

1. n_estimators: The maximum number of estimators at which boosting is terminated.
    a. From 50 to 70
2. learning_rate : Learning rate shrinks the contribution of each classifier by learning_rate.
    a. From 1 to 0.9
3. algorithm:
    a. Stayed with default = SAMME.R

The spacing for n_estimators is +/- 5 and for learning_rate is +/- 0.05. Finer values were not used to prevent overfitting. This improved the precision, recall, and F1 scores from (0.49370, 0.333, 0.39773) to (0.52214, 0.36550, 0.43). The base_estimator was not changed since two other base estimators were tested previously.

What is validation, and what's a classic mistake you can make if you do it wrong?
Validation is testing an algorithm to ensure it's doing what is intended. By testing, one can check the performance of the algorithm and if it is overfitting the data. One classic mistake is to train an algorithm on the same data used for testing. This would not be an independent validation and may cause overfitting. Instead, a data set should be

split into a training set and a test set. For this project, the intended purpose of the classifier is to be able to predict a POI from the selected features. The best metrics for this would be precision and recall. A higher precision would mean whenever a POI gets flagged in the test set, it's very likely to be a real POI and not a false alarm. A high recall suggests that a large fraction of the real POIs are identified. The F1 score can be considered as a good metric to consider both recall and precision. A high F1 score would mean that a POI is identified accurately and reliably and one can have confidence that real POIs are identified, and non-POIs are not flagged.

How did you validate your analysis?
Using the test_classifier function in the tester script, the data was split into training (90%) and test (10%) sets using the StratifiedShuffleSplit algorithm with 1000 folds. StratifiedShuffleSplit is a merge of StratifiedKFold and ShuffleSplit, which randomly splits the data set into the training and test sets. However, the sets are generated in such a way that the mean POI label is equal in the training and testing sets, which is useful since there are an abundance of non-POIs in the data. The classifier is fitted with the training data and validated with the test data. The shuffle, splitting, and testing is done 1000 times, and the average metrics are used.

Give at least 2 evaluation metrics, and your average performance for each of them.
The precision, recall, and F1 scores are performance metrics used since the data is skewed (many more non-POIs than POIs). Accuracy is not as a good metric for this problem because even if everyone is predicted as a non-POI, accuracy can still be pretty good. Also, if everyone is labeled as a non-POI, the classifier is not doing the job of identifying POIs. As mentioned previously, precision is the fraction of positively identified POIs out of all the predicted POIs. Recall is the fraction of positively identified POIs out of all the true POIs.

The untuned AdaBoost classifier provided the following average statistics:
Precision: 0.49370 Recall: 0.33300 F1: 0.39773
The tuned AdaBoost classifier provided the following average statistics:
Precision: 0.52214 Recall: 0.36550 F1: 0.43000

Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance
The tuned AdaBoost classifier has better precision than recall. That means that whenever a POI gets flagged in the test set, it's likely to be a real POI and not a false alarm. However, sometimes real POIs, are missed.