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 for this project is to identify "persons of interest" in investigating corporate scandals. Specifically, to identify the people who were involved in committing fraud as part of the Enron scandal. The dataset consists of email metadata and financial data for about 146 employees of Enron (mostly those high up in the corporate ladder). Only 18 of the 146 employess are labeled persons of interest, representing 12.3% of the dataset. This makes the dataset fairly unbalanced, which could present difficulty when training a classifier on it.

   Upon investigation into the dataset's 146 "employees", two were found to be outliers / invalid for our investigation, namely 'Total' and 'THE TRAVEL AGENCY IN THE PARK'. For example, the 'Total' category is an outlier across most features. After removing these, we now had 144 persons in the dataset.

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"]**

   To select the best 6 features of the original features, I manually experimented with the features that I intuitively guessed to be significant factors in determining person of interest. The decision to use only the following features was strengthened by manual testing, in which I found that the addition of new features either (1) did not improve measures of success or (2) actually decreased measures of success.

   Manual testing of combinations:

   features_list = ['poi','salary','exercised_stock_options', 'bonus', 'deferred_income', 'long_term_incentive']

```
Accuracy: 0.87371       Precision: 0.62636      Recall: 0.28750 F1: 0.39411     F2: 0.32238
Total predictions: 14000        True positives:  575    False positives:  343   False negatives: 1425   True negatives: 11657
```

features_list = ['poi','salary', 'expenses', 'loan_advances', 'from_this_person_to_poi', 'from_poi_to_this_person']

```
Accuracy: 0.82158       Precision: 0.41516      Recall: 0.17250 F1: 0.24373     F2: 0.19533
Total predictions: 12000        True positives:  345    False positives:  486   False negatives: 1655   True negatives: 9514
```

features_list = ['poi','salary', 'shared_receipt_with_poi', 'restricted_stock_deferred', 'from_this_person_to_poi', 'from_poi_to_this_person']

```
Accuracy: 0.80342       Precision: 0.31552      Recall: 0.15350 F1: 0.20653     F2: 0.17107
Total predictions: 12000        True positives:  307    False positives:  666   False negatives: 1693   True negatives: 9334
```

features_list = ['poi','salary', 'director_fees', 'long_term_incentive', 'from_this_person_to_poi', 'from_poi_to_this_person']

```
Accuracy: 0.80831       Precision: 0.23261      Recall: 0.10700 F1: 0.14658     F2: 0.11996
Total predictions: 13000        True positives:  214    False positives:  706   False negatives: 1786   True negatives: 10294
```

features_list = ['poi','total_stock_value', 'expenses', 'loan_advances', 'from_this_person_to_poi', 'from_poi_to_this_person']

```
Accuracy: 0.85536       Precision: 0.48285      Recall: 0.17600 F1: 0.25797     F2: 0.20163
Total predictions: 14000        True positives:  352    False positives:  377   False negatives: 1648   True negatives: 11623
```

features_list = ['poi','salary','exercised_stock_options', 'bonus', 'from_this_person_to_poi', 'from_poi_to_this_person']

```
Accuracy: 0.88554       Precision: 0.74710      Recall: 0.38700 F1: 0.50988     F2: 0.42829
Total predictions: 13000        True positives:  774    False positives:  262   False negatives: 1226   True negatives: 10738
```

I therefore used the last combination for my selection of original features: 'poi','salary','exercised_stock_options', 'bonus', 'from_this_person_to_poi', 'from_poi_to_this_person'

Next, I used 2 of the original features to create a new feature that I added myself: 'restricted_stock_ratio' ← 'restricted_stock' divided by 'total_stock_value'

The reasoning behind this originates from my prior knowledge that restricted stock is a beneficial form of compensation for executives as a consequence of its modified tax regulations. So, a higher proportion of total stock value being in the form of restricted stock should indicate (1) an aversion to tax policy and (2) compensation of executives, both of which are likely helpful in determining persons of interests.

The addition of this new feature improve accuracy and recall by almost 1% at the expense of slightly lower precision:

```
Accuracy: 0.89257       Precision: 0.72794      Recall: 0.39600 F1: 0.51295     F2: 0.43574
Total predictions: 14000        True positives:  792    False positives:  296   False negatives: 1208   True negatives: 11704
```

Therefore, I used a total of **7 features**: 'poi','salary','exercised_stock_options', 'bonus', 'from_this_person_to_poi', 'from_poi_to_this_person', 'restricted_stock_ratio'

Finally, I should mention that since the selected being K nearest neighbors necessitated the use of scaling, since said algorithm utilizes Euclidean distance in order to classify data points. To do this, I used preprocessing.MinMaxScaler().

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 K nearest neighbors algorithm proved most effective. I also experimented with the Gaussian naïve Bayes, random forest, and the gradient boosting algorithms, but these were not as effective and often took much longer to run (e.g. random forest).

   Results:

```
Accuracy: 0.89257       Precision: 0.72794      Recall: 0.39600 F1: 0.51295     F2: 0.43574
Total predictions: 14000        True positives:  792    False positives:  296   False negatives: 1208   True negatives: 11704
```

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 of an algorithm means to set those parameters such that the algorithm produce the optimal values given the particularities of the learning task. Too little parameter tuning can result in suboptimal training given the learning task, while too much parameter tuning can result in overfitting to the particular details that are present in the training set and not the general population / test set.

   I'll go through the several ways that I tuned the parameters for my implementation of the K nearest neighbors algorithm.

   - N_neighbors: I first tried k = 2, but quickly found that higher values of k gave better results. I ended up using k = 6.
   - Weights: Using weights = 'distance' gave the best results (as opposed to uniform).

   I also tuned parameters for the random forest (n = 10) and gradient boosting classifiers (n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0).

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"]**

A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's hyperparameters. The validation dataset is different from the test dataset that is also held back from the training of the model, but is instead used to give an unbiased estimate of the skill of the final tuned model when comparing or selecting between final models. Validation helps prevent overfitting on the training set.

I performed validation in this project by using train_test_split as imported from sklearn.cross_validation.

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"]**

Accuracy: The ratio of correct predictions to total predictions. In other words, accuracy measures how often the classifier makes a correct prediction.

Precision: The ratio of correct positive predictions to total positive predictions. In other words, precision measures how often the classifier is correct when it thinks that it has identified a person of interest.

Recall: The ratio of correct positive predictions to total positive instances. In other words, recall measures how often the classifier "catches" a person of interest when it comes across an actual person of interest.

The classifier I trained appears to have a much better accuracy (89.3%) than its recall (39.6%). Moreover, its precision (72.8%) is significantly better than its recall (39.6%). This means that the classifier is much better at avoiding falsely asserting that an Enron employee is a person of interest than it is at actually catching a person of interest in a positive sense.