Question 1

Dataset Background

The collapse of the Enron Corporation in 2001 led to the largest bankruptcy in US business history up to that point. Not only did Enron itself collapse, it also brought down its auditing firm, Arthur Andersen, with it. Named as "America's most innovative company" for six years running by Fortune magazine, Enron's greatest innovation turned out to have occurred in the world of creative auditing. In Enron's world, everything that glisters was by no means gold.

Dataset Particulars

The dataset is a dictionary of 150 keys. Each key is the name of an Enron employee, and the values for each respective key are features of that employee's behavior. There are financial values, like salary and bonus payments, and email values, such as the number of emails sent by the person, sent by this person to a person of interest, received from a person of interest, and so on.

The most important value in the dataset is `poi`. This is a Boolean, True/False, value – is this employee a person of interest as regards the investigation into whether fraud did or did not take place at Enron? The purpose of this project, then, is to find an algorithm that best identifies persons of interest through machine-learning techniques.

Why Machine Learning?

It's important to remember just how big the Enron corporation was, and just how big a wave a corporation of that size and with that level of influence can generate in the course of its doing business. This generates a huge amount of data, far more than could be quantifiable by human intellect. There are simply too many dots to join.

And it's in cases like this that machine learning comes into its own, because a terrifying computational task for a human being is a humdrum one for a correctly-instructed algorithm. Consider how technology was able to reduce audio files firstly to the size of a CD and latterly to the size of an .mp3. It is possible in the same way to reduce data to manageable proportions, using the algorithms discussed below.

Reducing the data doesn't alter the data. It just makes it easier to identify patterns that might not have been as identifiable prior to the reduction. But the patterns have been there all the time – it just took the work of the machine learning algorithms to allow us to notice them.

Question 2

Data Exploration

The dataset is incomplete. There is no employee for which we have concrete statistics for all criteria. There is one criterion, `loan_advances`, for which we have data for only three people. Some criteria are of obvious interest, such as salary or bonus, because these are the places where the fruits of fraud appear. Some are of less obvious interest, such as the relationship between the stock criteria.

Outliers

There are some considerable outliers in the data that is extant, to say nothing of what may or may be the figures that we don't have. However, a cursory inspection of salary alone reveals that the outliers contain the very data that we seek. These are the top salaries, with their poi status:

Employee	salary	poi
SKILLING JEFFREY K	$1,111,258.00	TRUE
LAY KENNETH L	$1,072,321.00	TRUE
FREVERT MARK A	$1,060,932.00	FALSE
PICKERING MARK R	$655,037.00	FALSE
WHALLEY LAWRENCE G	$510,364.00	FALSE
DERRICK JR. JAMES V	$492,375.00	FALSE
FASTOW ANDREW S	$440,698.00	TRUE
SHERRIFF JOHN R	$428,780.00	FALSE
RICE KENNETH D	$420,636.00	TRUE
CAUSEY RICHARD A	$415,189.00	TRUE

An Enron investigation without outlier data is a production of *Hamlet* without the Prince. The outliers are where the action is.

Identifying the POI values

The first thing to do was eliminate some of the data with missing values. To this this end, three different features lists were created. There was a list with the default two features, `poi` and `salary`; there was a list with six features, determined as those features with more than 100 valid data points: `poi`, `salary`, `total_payments`, `total_stock_value`, `exercised_stock_options` and `restricted_stock`; and finally there was list with fourteen features, determined as those features with more than eighty valid data points: `poi`, `salary`, `total_payments`, `bonus`, `total_stock_value`, `expenses`, `exercised_stock_options`, `other`, `restricted_stock`, `to_messages`, `from_poi_to_this_person`, `from_messages`, `from_this_person_to_poi`, and `shared_receipt_with_poi`.

The second thing to do was to test for collinearity within the data, and use that to eliminate and/or combine data fields. This was achieved by converting the

dataset to a pandas data frame, normalizing the values so that they would compare properly to each other, and then running pandas' `.corr()` function to check for co-linearity.

The `.corr()` function returns a matrix with Pearson's Coefficient for every data combination, ranging from -1, perfect negative correlation, to 0, utterly random connection, to 1, perfect positive correlation. These are the values for which there was a Pearson's Co-efficient greater than 0.9:

	salary	deferral_payments	total_payments	loan_advances	bonus	restricted_stock_deferred	deferred_income	total_stock_value	expenses	exercised_stock_options	other	long_term_incentive	restricted_stock	director_fees
salary														
deferral_payments				1.00										
total_payments				0.99										
loan_advances					0.97		1.00	0.96		0.96	1.00	1.00	1.00	
bonus														
restricted_stock_deferred														
deferred_income														
total_stock_value										0.96				1.00
expenses														
exercised_stock_options														
other														
long_term_incentive														
restricted_stock														0.97
director_fees														

As remarked earlier, `loan_advances` are so few they can be ignored. The features most worth combining were `total_stock_value` and `exercised_stock_options`.

Adding a combined stock value proved difficult, however. The tricky point was the `NaN`, not-a-number, values. The `featureFormat()` function in `tester.py` drops `NaNs`, but it would be better to return them as `NoneType`, as this would allow the matching up of the new features with the keys in the original data dictionary.

This could be done by using the pandas data frame already created, running `fillna()`, calling a regression or a PCA on the two columns, mapping them to the data frame and then converting back. But it was decided to see if we could hit 30% in the recall and precision scores without going to the extra labour, as one of the secrets of life is the correct allocation of time to tasks.

Choosing the Algorithm

As we're trying to identify whether or not someone is a person of interest, we're looking for a classifier algorithm. The algorithms tried were Naïve Bayes, Support Vector Machines, AdaBoost, Decision Trees and Random Forest. Each was tried under different criteria – default settings, different feature lists, tuned with `sklearn.grid_search.GridSearchCV` and with and without Principle Component Analysis having being carried out on the data.

Question 4

Tuning the Parameters

Theory states that machine learning classifier algorithms seek to find a balance between bias and variance. That is to say, they want to find a model that is sufficiently flexible to return accurate data for all cases, but not so flexible that it ends up overfitting data and losing the general pattern. In general, a model's accuracy increases as features are added up to a certain point. Beyond that point, the model becomes over-laden with features and can no longer see the wood for the trees, as it were.

These are the results of the individual algorithms.

Naïve Bayes

Naïve Bayes achieved the highest recall (see below) score of any of the classifiers at 0.8. Unfortunately, a precision score of 0.18 meant that the result set returned was not very valuable and dragged down the overall accuracy of the algorithm.

Support Vector Machine

Support Vector Machine algorithms would not run on this dataset. An SVM with an rbf (radial basis function) kernel returned no true positive results, while a linear kernel seemed to stall, and was abandoned after running overnight for no result.

AdaBoost

The Adaboost Classifier did better than Naïve Bayes in terms of precision with a score of 0.25 but its recall score was a disappointing 0.16.

Decision Tree

Progress started being made with Decision Tree algorithms. A Decision Tree algorithm run on the basic feature set, salary v poi, returned recall and precision scores of 0.24. A Decision Tree run on the six-feature set on which a Principal Component Analysis had been performed first saw recall hold at 0.24 but

precision slip to 0.22. When the Decision Tree was run on the six-feature without the PCA being performed on it the algorithm returned 0.26 for both precision and recall.

Random Forest

Random Forest was the most successful algorithm run on the dataset. A Random Forest algorithm run on the six-feature dataset returned a precision score of 0.38 and a recall score of 0.22.

A Random Forest was then set up to optimise the classifier fit according to the following code:

```
my_rfc_parameters = {'criterion':['gini',
                     'entropy'],
                     'max_features':['auto',
                                     'sqrt',
                                     'log2']}
clf = GridSearchCV(RandomForestClassifier(),
                     my_rfc_parameters)
```

This returned a precision score of 0.36, and a recall score of 0.14 – a distinct dis-improvement.

Principle Component Analysis was then run on the features according to this code:

```
rfc = GridSearchCV(RandomForestClassifier(),
                     my_rfc_parameters)
clf = make_pipeline(PCA(), rfc)
```

This saw precision rise to 0.42, and recall rise to 0.22.

Question 5

Validation

The data was validated using k-fold cross-validation. The idea of a k-fold validation is to randomize the training and testing data, so that co-incidental values, like all the persons of interest being at the head of the data set, don't skew the results. The `sklearn.cross_validation.KFold` iterator sees every datum being used as both a test and a training case, and thus true randomization can be assured. In this case, there was no difference discernable in the setting the number of folds.

Question 6

Evaluation

A classifier that returns a Boolean (true/false) result will have four possible results, as per this table:

	POSITIVE	NEGATIVE
TRUE	True Positive	True Negative
FALSE	False Positive	False Negative

Two results are correct, two are incorrect. But they are incorrect in different ways – a false positive is a result that incorrectly classifies a positive result, and a false negative a result that incorrectly classifies a negative result.

In common language, a false positive would be an instance of an innocent person being convicted of a crime, while a false negative would be an instance of a guilty person being acquitted. Common law sees false positives as worse than false negatives – we would rather see a guilty person go free than imprison an innocent. However, if we are minesweeping, for instance, we have to prioritize the elimination of false negatives – it is better for us to blow up a rock than to let a mine through that may sink a ship.

Recall and **precision** are the metrics by which we measure accuracy. Recall, the ratio of true positives against the sum of true positives and false negatives, is a measure of the worth of the algorithm. Precision, the ratio of true positives against the sum of true positives and false positives, is a measure of the worth of the result set.

In this case, it was decided that precision was the better metric. There is a huge recall score (0.8) for Naïve Bayes, but this is almost certainly due to the huge preponderance of negative against positive values. With such a weight of negative results, how could an algorithm miss them?

In this light, the Random Forest Generator run on a six-feature dataset on which both a Principle Component Analysis had been run first was seen as the best classifier, as it had the highest precision score at 0.42. This returned the most useful result set.