Background and Goals of the Project

The goal of the project is to produce a classifier, with acceptable precision and recall levels, to identify persons of interest (POI) based on data extracted from public materials associated from the Enron case.

Briefly, the data derives from three sources:

- Fourteen financial features extracted from a *Payments to Insiders* document which contains an itemized record of payments from Enron Corp. to its insiders in both cash and stock.
- Six email features extracted from the Enron Corpus, a database of email of all Enron Corp. employees, as related to the insiders named in the Financial data. For the purposes of this project, the data points include the total number of emails sent and received by the person in question, and how many of them were sent to, received from, or have shared receipt with a POI.
- A list of POI.

This, in total, contains a dataset involving 146 insiders with 21 features for each insider.

Outlier investigation

During the course, it has already been noted that a grand total line from the *Payments to Insiders* document was inadvertently included into the dataset. This is removed from the dataset immediately.

Due to the large level of intra-individual variances within the data set, and the fact that some high-deviation data points within the data set cannot be excluded just because of this (as mentioned in the class, this would have removed the infamous Kenneth Lay and Jeffrey Skilling owing to the fact they were paid highly), no unsupervised outlier identification has been performed. Instead, I check for the sanity of several data points, for example, restricted_stock must be larger than restricted_stock_deferred, as no person can defer something that don't own. Similarly, all financial features are either positive or 'NaN' except for those ending with "_deferred," which are always negative.

Through this, I have identified two suspicious data points. Sanjay Bhatnagar is recorded to have $-2,604,490 in restricted stock, of which $15,459,290 was deferred, which is absurd, as there stocks does not negative value. Similarly, Robert Belfer has a total stock value of $-4,4093, which is similarly absurd. Cross-checking with the *Payments to Insiders* document indicates these are errors potentially introduced by the dataset by the OCR software that digitalized the document by left or right-shifting several fields. The financial data for these two people were corrected.

Other investigations of the data

While this data set includes data from 146 insiders, only 18 of them has been identified as a POI, which indicates an unbalanced response label.

On the other hand, many features have missing values for more than half of the insiders. In descending order, these fields include loan_advances (>98%), directors_fees (>90%), restricted_stock_deferred (>88%), deferral_payments (74%), deferred_income (36%), and long_term_incentive (55%). However, these do not appear to be essential features for an employee's remuneration package, so I decided to not act on those except treating them as zeroes.

Feature Processing

Field "email_address" was never imported in any analysis, due to the fact that I do not plan to perform any text analyses on the email messages, and its string nature poses problems for the other algorithms used in machine learning.

I added five new features to this data set:

Feature Name	Definition	Justification
proportion_from_this_to_poi	The proportion of emails from this person to one or more POI to all emails he sent	The level of email communication between an insider to a known POI is indicative of that insider's involvement of the fraud in question. However, I consider the proportional email communication to POI relative to the person's total email count to be more predictive than the raw email number.
proportion_from_poi_to_this	The proportion of emails from one or more POI to this person to all emails he received	
proportion_shared_receipt_with_poi	The proportion of emails to one or more POI and this person to all emails he received	
proportion_of_stock	The proportion of stock for that person's remuneration package.	The use of stock options is seen by critics to encourage risk-taking behavior for the recipient, while deferred compensation is seen to increase corporate debt[1]. Thus I'd want to like to see whether the absolute and relative values of these measures affects the mindsets of insiders.
proportion_payments_deferred	The proportion of payments this person was entitled to received, but was deferred.	

[1] https://www.newyorkfed.org/medialibrary/media/research/staff_reports/sr445.pdf

Due to the choice of these features, the original email features (to_messages, from_messages, from_poi_to_this_person, from_this_person_to_poi and shared_receipt_with_poi) were removed from the data set prior to feature selection.

While the Lasso regression is a common way to perform feature selection, scikit-learn documentation recognizes that if would only select one out of several correlated features[2]. Since the nature of this dataset means that there may be serious correlations between different financial features, Lasso alone may limit the predictive power of the models trained from the selected features.

Randomized bootstrapping of the Lasso regression, as implemented by sklearn.linear_model.RandomizedLasso, is recommended by scikit-learn for such situations, especially when the "ground truth" has a large fraction of zeroes, as in this case. Based on the lasso model, RandomizedLasso, in the default setting, performs 200 of 75% stratified subsets of the normalized data, ran Lasso on each subset, and counts the frequency a feature has a non-zero coefficient. Features having a non-zero coefficient in more than 25% of the subsets were selected for further algorithm testing and training[3]. Eleven features were eventually selected, including salary, deferral_payments, bonus, restricted_stock_deferred, deferred_income, total_stock_value, expenses, exercised_stock_options, other, long_term_incentive, and proportion_from_this_to_poi.

Features were normalized by Normalize if they were used as an input for algorithms that are sensitive to unscaled features, such as SVC and Naïve Bayes. Otherwise, algorithms were trained with unscaled data.

Algorithm Selection and Tuning

Most algorithms taught in this course, including Gaussian Naïve Bayes, SVM (using sklearn.SVM.SVC), Decision Tree, AdaBoost and Random Forest, classify data points into one of the several classes, and would in theory applicable for this project. All of the above were tested to identify the best classifier based on their F1 score.

Furthermore, most of the algorithms above have large number of parameters, most of which would have serious implications to the models fitted to the algorithms as they can be one of the following:

- Constants used in the underlying equation, such as the C and coef parameters in SVM[4],
- The underlying function used, such as the kernel parameter in SVM[5] or the criterion parameter in RandomTreeClassifier[6], or

[2] http://scikit-learn.org/stable/modules/feature_selection.htm. In particular, see Chapter 1.13.4.2, *Randomized sparse models*.
[3] http://blog.datadive.net/selecting-good-features-part-iv-stability-selection-rfe-and-everything-side-by-side/
[4] http://scikit-learn.org/stable/modules/svm.html
[5] http://scikit-learn.org/stable/modules/svm.html

- Classification rules, such as min_samples_split and min_samples_leaf parameters in RandomTreeClassifier.

Because these parameters can directly impact on the performance on the resulting model, it is important for any data analyst to tune their preferred algorithm, i.e. adjust the algorithm's parameters to optimize the algorithm's performance. This is to ensure the trained algorithm is not underfitted due to the algorithm underestimating the underlying complexity of the training set, while on the same time avoiding overfitting, i.e. the algorithm overestimating the training data's complexity such that it cannot be generalized on the production data[7].

In this project, a systematic parameter tuning regime is used for most classification algorithms to select a combination of parameters that has the best performance for that algorithm.

First, 40% of the data set was set aside for testing using train_test_split. Bearing in mind of the imbalance of the response variable, the stratify parameter is used to ensure the composition of the response labels is the same between training and testing groups.

All the algorithms listed above were tuned using GridSearchCV using a large variety of parameters, with the exception of Naïve Bayes, which doesn't have parameters; the F1 score were used as the scoring function.

The F1 scores from the best model for each algorithm are:

Algorithm	F1 score
Random Forest	0.586
Decision Tree	0.578
AdaBoost	0.485
SVC	0.239
Naïve Bayes	0.195[8]

Based on these results, the model based on Decision Tree algorithm was selected for having the highest score of the five used.

The chosen Random Tree classifier assigned zero importance to most of the features used with the exception of the following five:

Feature	Importance
Other	0.4581
Expenses	0.2387
Total_stock_value	0.1432
Deferred_income	0.0983

[6] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[7] Shane Settle, *Automated Parameter Tuning for Machine Learning*, http://theorycenter.cs.uchicago.edu/REU/2014/presentations/settle.pdf
[8] Since GridSearchCV cannot be applied on the Gaussian Naïve Bayes algorithm, this F1 score calculated by comparing the predicted and true POI labels from the testing set.

Feature	Importance
proportion_from_this_to_poi	0.0617

Algorithm Validation

Classifier models were trained for a subset of all data points, and thus, logically speaking, applicable only for that particular subset and not necessarily for the entire data set. As a result, validation is required to ensure the models' general applicability for all data points to be predicted using the same model. A common issue arising from a mal-validated model is that of overfitting, i.e. the estimated model is highly performant for the training data, but of questionable power as general applicability is concerned.

The most common validation metrics for classifier models that predict discrete class labels are precision and recall. Precision is defined as the probability that a data point predicted by model to be a member of class *x* actually belongs to class *x*, while recall is defined as the probability that a member of class *x* is predicted by the model as class *x*.

On the other hand, due to the fact that the response variable is highly imbalanced in this data set, the standard metric of accuracy score would be considered inappropriate. For example, if all the POI in this data set were misclassified as non-POI were correctly classified as such, the accuracy score would still be 0.855, because 124 of 145 points were predicted correctly. However, in this case the recall is 0 and the precision undefined.

In this instance, I used the test_classifier subroutine from tester.py to perform validation. This subroutine performs 1,000 rounds of train-test splitting with 90% used for training and ensures the POI label ratios in each subset is roughly equal to the POI/non-POI ratio in the original dataset. Precision and recall metrics were calculated from the aggregated results from the 1,000 bootstrap trials. This subroutine estimated the precision and recall of this model to be 0.31965 and 0.47500 respectively.