Scott Burns
Udacity – Data Analyst Nanodegree
Project 4: Identify Fraud from Enron Email

Enron Submission Free-Response Questions

Section 1: Summary

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?

Goals

In this project I will attempt to determine whether features of Enron employees' email correspondence and compensation can be used to accurately predict their participation in fraud, or at least their status as a fraud suspect, as determined by traditional investigative methods.

Machine learning can be helpful in identifying the employee characteristics that are indicative of fraud, by allowing us to more effectively analyze big datasets. Drawing on available computational power, we can test hypotheses about email and compensation evidence at a larger scale than possible using other approaches to investigation. Human intuition is an important part of this analysis, but machines' iterative parsing of available data can help us formulate, evaluate and discard hypotheses more rapidly than we could simply through more manual testing.

Outliers

I initially included all available features in my targeted feature list for 'poi_id.py'. In the beginning, I felt that all features could plausibly be associated with an employee's status as a POI, and seemed worth exploring. Furthermore, I recognized that I could later drop less useful features using functions like K-Best or consolidate features with PCA as needed.

Applying my own intuition to the dataset, I looked at potential outliers in the email records, which might hinder us from accurately predicting relationships between email features and fraudulent behavior.

I decided to plot all financial variables in a grid against salary, just to get a better sense of their distribution. Each subplot showed at least one massive outlier, which I suspected was a 'TOTAL' column, as we discovered in lesson 7 of the Intro to Machine Learning course.

In the script, I inserted a function to remove the 'TOTAL' item, and plotted the financial features again. Given that the subplots were used for initial exploratory data analysis, and not needed for final output, I have commented out this section of code in the submission.

After rescaling my features as described in Section 2, I drew on Sebastian's heuristic for handling outliers, as explained in lesson 7, to 'cleanse' the dataset, while reducing the percentage of outliers removed in light of the small total size of our dataset (e.g. – I removed only outliers in

the top percentile of the dataset for two key features based on regression predictions, rather than the 10% suggested in Lesson 7 as an outlier-removal rule of thumb). As a note, I also chose to rescale, then cleanse the data, because this allowed me to re-include outliers removed from the training set that have been re-scaled consistently with the full dataset back in the test set following training,

To cleanse, I fitted various compensation features on employees' salaries with a linear regression, then removed the top 1% of data points by their absolute residual of actual feature value minus predicted feature value. I tried the process iteratively over two variables I saw containing large outliers based on plotted figures. The removal of several employees with large residuals yielded updated scatterplots with a relatively low overall number of striking outlier points.

After several trials, removing outliers based on predictions of correlation with salary, I ultimately decided to remove outliers for algorithm training based on 'bonus' and 'exercised_stock_options' feature values, and how each value differed from a prediction based on the user's salary.

After removing all outliers from my training set, the total number of employees (datapoints) in the training dataset fell to 139 from 145 initially.

To clarify the characteristics of outliers removed from the training set and address previous reviewer questions on the values removed, below is a table of the data points removed from training data, along with summary statistics for key features of outliers and non-outliers from the two classes of label – 'POI' and 'non-POI'.

Table 1: Comparison of Outlier and Non-Outlier Characteristics

	Non-POIs			**POIs**	
	Outlier	**Non-Outlier**		**Outlier**	**Non-Outlier**
Total data points	2	125		4	14
Outlier names	'LAVORATO JOHN J', 'ALLEN PHILLIP K'			'RICE KENNETH D', 'HIRKO JOSEPH', 'LAY KENNETH L', 'BELDEN TIMOTHY N'	
Median value for group (in $)					
Bonus	6,087,500	700,000		5,249,999	1,200,000
Exercised Stock Options	2,944,268	940,257		25,280,120	1,963,175
Long-term Incentive	1,170,093	375,304		2,608,506	788,482
Median value for group (count)					
From POI to This Person	288	25		123	58
From This Person to POI	238	6		16	15

The removal of the data points for training only (note that I have also updated my tester.py to add back removed outliers to test data after calling StratifiedSampleSplit) seems to be a valid approach, as the outliers show feature values that are extremely different (within each same class - non-POI and POI) from those left in the training data. At the same time, feature values from the

outlier groups seem to be more similar for non-POIs and POIs than is the case for non-outlier data points, which suggests to me that these similarities may be making it more difficult for algorithms to identify the features that are largely distinct between the two classes for the bulk of the data points.

In the training data only, the percentage of all data points marked as POIs is reduced slightly, with more POIs removed than non-POIs, from 12% to 10%.

Looking at the non-POIs removed from the training dataset, it is clear that these are massive outliers, with Thomas Lavorato showing over $4 million in exercised stock options versus a median exercised stock option value of non-outlier, non-POIs under $1 million. Similarly Phillip Allen and Lavorato registered bonuses of over $4 million and $8 million, respectively, where the median non-outlier non-POI bonus was only $700,000.

For each employee datapoint in the trimmed set, some attributes were typically missing, with the 'deferred_income' being the most complete, where all 139 data points showed values for this feature. As another example, only 61 records showed values for 'long_term_incentive' - one of the more incomplete features. Based on a spot-check comparison with the enron61702insiderpay.pdf, it looks like the missing values in the dataset line up with the missing values denoted by ' – ' in the pdf.

The outlier removal approach follows from suggestions in Lesson 7 lectures of the Intro to Machine Learning course, and techniques required in exercises for the lesson. Based on a search of online literature regarding outliers in machine learning, the removal strategy seems to be consistent with widely recommended practices. An example of discussions on outlier detection and removal can be found at http://machinelearningmastery.com/how-to-identify-outliers-in-your-data/.

Also, as noted in Section 3, I ultimately chose to classify data using an AdaBoost classifier, and according to literature on the topic, the algorithm seems to be highly sensitive to outlier feature values. One example paper on the topic can be found here: http://www.ism.ac.jp/~eguchi/pdf/KANAMORIetal.pdf. Given this sensitivity, it seems prudent to have removed the given outliers from training datasets.

Section 2: Features

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.

Initial broad feature list

I initially started with all the given features in my model, deciding that they each had plausible correlation with an Enron employee's POI status. After observing a few generated scatterplots of variables versus salary values for employees, re-assessing features' theoretical impact on classification, I decided to drop the features 'directors_fees', 'restricted_stock_deferred' and 'loan_advances'. Following suggestions from a reviewer, I also examined the 'total_payments' feature and noticed that it actually seemed to incorporate 'total_stock_value', regardless of whether the stock had been exercised, which seemed fundamentally incorrect. As a result I decided to remove it from the feature list as well.

Beyond this, I decided that another email-related feature implicit in the dataset, but not directly calculated, may be helpful in improving classification. In the dataset, we have 'to_messages' and 'from_this_person_to_poi', as well as 'from_messages' and 'to_this_person_from_poi'. I hypothesized that the most predictive indicator of being a POI was likely to be the emails received from or sent to POIs as a percentage of total emails received or sent.

Creating 'percent_of_emails' features

Accordingly, I included the features 'percent_of_emails_from_poi' and 'percent_of_emails_to_poi' in the features list. Initially, I played with the idea of removing other email-related features due to their correlation with the new features, but found that their inclusion increased recall and precision in subsequent classification test runs, and that features like 'from_this_person_to_poi' were also routinely included in feature lists as part of using the KBest features selection tool.

Rescaling features

Because I was incorporating features with wide differing absolute variances, I also decided that it would be helpful to rescale my features, using MinMaxScaler in the preprocessing module of Scikit learn.

I created a deep copy of the 'data_dict' dictionary, then for all features in my features list (other than 'POI'), rescaled the feature according to its position relative to the range of values for the feature in the dataset – adding it to a corresponding value entry in my new copied data dictionary, which I entitled 'rescaled_data_dict'.

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", "properly scale features", "intelligently select feature"]

K-Best features

After some trial and error, I determined that filtering features automatically with Scikit-learn tools would be helpful in increasing my model's predictive power, and ultimately decided on

capping the total number of features selected at 5, as with higher K values I was seeing multiple features with 0 relative importance in the model, and with smaller values for K I was seeing the predictive power of my models decline.

I used Chi-2 tests of feature significance when filtering features with the SelectKBest function from the Scikit-learn feature_selection module.

The 5 features automatically selected using the K-Best process were:

'salary', 'bonus', long_term_incentive', percent_of_emails_to_poi', 'percent_of_emails_from_poi'

The relative importance of each feature are highlighted below (in parenthesis), based on the 'feature_importances_' attribute from the AdaBoost classifier.

Feature Importance (*importance score*):

```
1. feature percent_of_emails_from_poi (0.300000)
2. feature salary (0.250000)
3. feature percent_of_emails_to_poi (0.150000)
4. feature long_term_incentive (0.150000)
5. feature bonus (0.150000)
```

Other ideas (for future exploration)

Aside from the given features I felt that the following features could also potentially be associated with fraud:

1. **exclusive_poi_exchange**

This feature would be a count of emails sent exclusively to a POI, or emails received, with no cced addressees, from a POI. The feature is similar to both the from_poi_to_this_person and the from_this_person_to_poi, but it excludes emails where the author had addressed a group of people or received a message with many others included. I hypothesize that POIs may have been more likely to leave other recipients out of email traffic if discussing matters related to fraudulent activity.

2. **exclusive_exchange**

Thinking about my hypothesis above, I decided that in general, an employee participating in questionable business practices may be more likely to focus his correspondence, taking pains to limit the number of people on his email chains. I decided it would be useful to create a general record of total emails sent and received by each employee in which there were only two parties in email chain.

3. **spv_mention**

From what I understand of the Enron fraud case, most of the company's schemes to illegitimately present its financial position related to the use of 'special purpose vehicles' or SPVs. I decided to test whether the presence of the strings 'special purpose vehicle' or 'spv' was associated with an author being a POI. This variable would track the number of emails sent by each employee with either of these strings.

I ultimately decided to not implement these features, as I didn't have a full list of the email addresses associated with each individual name in the keys of the data_dict dataset. I contemplated trying to build this set, but finally decided that this was going to require more time than I have available to spend on the project. This is an area I'd like to revisit and explore further in the future.

Section 3: Algorithm

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 used an AdaBoost classifier to identify POIs from the dataset.

I selected this algorithm as it was demonstrating the highest F1 scores in the given tester from a list of classifiers I tried, including SVM, Naïve Bayesian, Decision Trees, RandomForest and K-nearest Neighbors.

When testing the algorithm directly in my 'poi_id.py' script, I was able to train the classifier with outliers excluded, then add back outliers for the test set. This approach yielded a predictive classifier, with F1 score of 0.5, and an improvement on the F1 value of the classifier trained and tested in my 'poi_id.py' script without excluding outliers.

I modified the 'tester.py' script to also allow training with data cleansed of the 2 non-POI outliers, but tested with outliers included.

Using this approach with the modified 'tester.py' script, my F1 score for predictions using this estimator was 0.47, as seen in tester output below.

Evaluation of AdaBoost Classifier in 'tester.py'

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=20, random_state=42)
Accuracy: 0.75475    Precision: 0.51096   Recall: 0.44275
F1: 0.47442  F2: 0.45490
Total predictions: 16000   True positives: 1771  False positives: 1695
False negatives: 2229 True negatives: 10305
```

Section 4: Tuning

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? (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).

'Tuning parameters' refers to adjusting the arguments passed into classifier algorithms in order to ensure that models fitted from training data achieve the desired predictive power, striking the right balance between variance and bias in the model.

To clarify, bias is the tendency of a model to 'under-fit', not generalizing sufficiently from training data and thus not providing useful predictions of test labels.

Variance is the tendency of a model to 'overfit', generalizing too extensively from training data and unable to accurately predict based on test data that differ from training.

In my exercises I tuned a number of parameters in my tested classifiers, including the criteria for my DecisionTreeClassifier and RandomForestClassifier (switching from 'gini' to 'entropy', the kernel and C value for my SVC Classifier ('rbf' to 'linear' and 1 to 100), as well as other parameters, such as n_neighbors in the K-Neighbors Classifier and min_sample_split in my DecisionTree Classifier. I found the variations to have occasionally substantial impact, but still found that none of this tuning allowed other classifiers to surpass the predictive power of my AdaBoost classifier.

Also, following optional suggestions from my reviewer, I used GridSearchCV to optimize my parameters for the selected AdaBoost model. In GridSearch, I tried a range of parameter values of 'n_estimators' from a set of values including 10, 20, 30, 40 and 50, and found a value of 20 to be optimal.

Section 5: Validation

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 item: "validation strategy"]

Validation is the use of separate data to train and evaluate the performance of a classifier. A classifier is only as useful as its ability to correctly classify new data outside those it used in training, so in building algorithms, we must be careful to partition our dataset for training and testing.

If a classifier is not correctly cross-validated, it is possible for it to yield a seemingly predictive model which may not be useful at all for classifying outside the current sample. Overfitting is a likely pitfall of insufficient cross-validation.

To initially validate my classifiers, I split the dataset using the Scikit-learn module 'cross_validation.train_test_split', keeping 30% of elements from the total dataset for testing. Using this approach allowed me to assess how accurate my classifier's predictions were on data outside the training sample. I also assessed how my classifiers' predictive power shifted with changes to my split of training and testing data. After validating using test_size values from 0.1 to 0.5, I found that 0.3 helped me identify the most predictive models.

Section 6: Evaluation

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.

Key evaluation metrics can be seen in the output from my 'tester.py' script below – as run to assess the AdaBoost algorithm I ultimately selected. In the 'tester.py' output, model Precision is 0.51 with Recall 0.44. Note that the 'tester.py' script was modified to allow training with data clean of outliers, but testing with these outlier data points included.

To confirm, Recall is the likelihood that an actual Enron POI was correctly classified as a POI by my algorithm, and Precision is the likelihood that an employee classified as a POI by my algorithm was actually a POI.

F1 is a measure of predictive power that incorporates both Precision and Recall – for which my classifier generated a score of 0.47.

Evaluation of AdaBoost Classifier in 'poi_tester.py'

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=20, random_state=42)
Accuracy: 0.75475    Precision: 0.51096   Recall: 0.44275
F1: 0.47442  F2: 0.45490
Total predictions: 16000   True positives: 1771  False positives: 1695
False negatives: 2229 True negatives: 10305
```

Section 7: References

Section 1:

http://stackoverflow.com/questions/7941207/is-there-a-function-to-make-scatterplot-matrices-in-matplotlib
http://matplotlib.org/examples/pylab_examples/subplot_demo.html
https://plot.ly/matplotlib/subplots/
http://matplotlib.org/api/pyplot_api.html?highlight=subplot#matplotlib.pyplot.subplot
http://machinelearningmastery.com/how-to-identify-outliers-in-your-data/
http://stackoverflow.com/questions/8347636/drop-in-single-breakpoint-in-ruby-code

Section 2:

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest
http://scikit-learn.org/stable/modules/feature_selection.html
https://discussions.udacity.com/t/creating-new-feature-in-dataset/8082/3
https://docs.python.org/2/library/copy.html

Section 3:

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
http://scikit-learn.org/stable/modules/neighbors.html#classification
http://scikit-learn.org/stable/modules/tree.html#classification

Section 4:

https://www.udacity.com/course/viewer#!/c-ud120-nd/l-2286088539/m-2466048540
https://www.udacity.com/course/viewer#!/c-ud120-nd/l-2252188570
https://www.udacity.com/course/viewer#!/c-ud120-nd/l-2258728540/m-2428648555
http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html

Section 5:

https://www.udacity.com/course/viewer#!/c-ud120-nd/l-2960698751/m-2944138932

Section 6:

https://www.udacity.com/course/viewer#!/c-ud120-nd/l-2945338635