Machine Learning Project Report

Purpose of Project and Data Background

The purpose of this project was to generate a model that could predict whether a person at Enron was a person of interest (POI) in the Enron scandal. I relied on e-mail and financial data from Enron in order to generate a machine learning model.

The dataset I was provided with was a combination of e-mail and financial data that was made available in the wake of a massive financial fraud perpetrated by some of the leadership at Enron through the early 2000s.The e-mails, also known as the Enron corpus, featured over 600,000 e-mails generated by 158 employees.

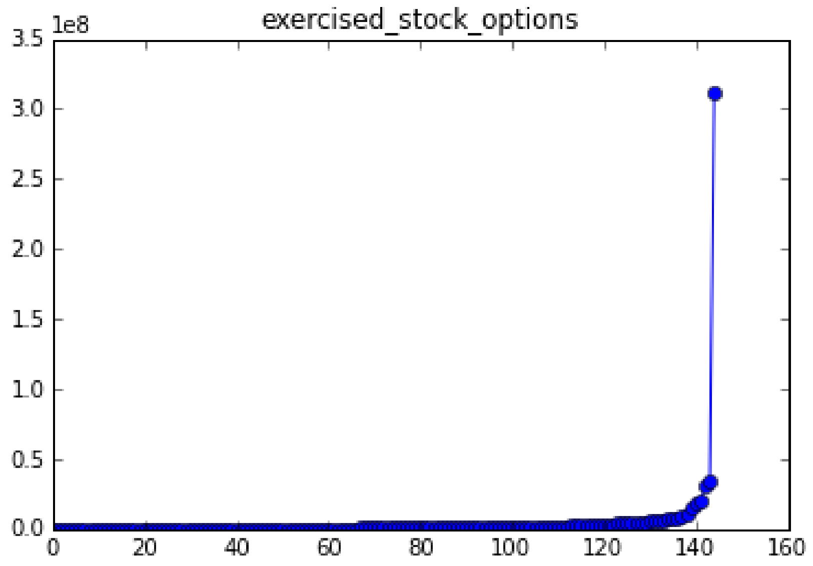
In the dataset I was provided, these e-mails had already been processed into a set of summary statistics that could easily be used by machine learning algorithms. In the end, I was provided with a dictionary containing information for 146 people, of which one was an aggregate. Initially, the following data was provided for each person in the dataset:

Feature Name	Number of Missing Values	Percent Missing Values
salary :	51	35%
to_messages :	60	41%
deferral_payments :	107	73%
total_payments :	21	14%
loan_advances :	142	97%
bonus :	64	44%
email_address :	35	24%
restricted_stock_deferred :	128	88%
total_stock_value :	20	14%
shared_receipt_with_poi :	60	41%
long_term_incentive :	80	55%
exercised_stock_options :	44	30%
from_messages :	60	41%
other :	53	36%
from_poi_to_this_person :	60	41%
from_this_person_to_poi :	60	41%
deferred_income :	97	66%
expenses :	51	35%
restricted_stock :	36	25%
director_fees :	129	88%

The table above shows that there are significant numbers of missing values in the data, particularly for some features. In the process of cleaning the data, the missing values were all converted to 0. While I left these zeros alone for the purpose of this analysis, substituting 0 for NaN may not be the best course of action for all of the features. For example, with loan_advances, where 97% of people had a value of NaN, it seems reasonable to assume that a missing value is the same as zero since people with NaN can be assumed to have not had a loan_advance, which is the same thing as saying that they had a loan_advance of $0. On the other hand, with a feature like to_messages and from_messages, where 60 people had values of NaN, it might not make sense to substitute a 0. It seems unlikely that these people actually didn't send or receive any e-mails. This data is truly missing, and so when we substitute 0 for the missing value, we will likely skew our analysis.

The data also included a dummy variable called ‘poi’ indicating whether a given employee was a POI. By using machine learning algorithms, I hoped to be able to detect patterns that predict whether a person is a POI or not. In the dataset overall, there were 18 POIs. Due to the small size of the dataset and very small number of POIs within the dataset, it was challenging to produce a model which could accurately classify POIs.

Finally, in order to get an understanding of outliers, I plotted all of the values for each feature from least to greatest. I ended up with a lot of graphs like the one below:



While doing this analysis, it became clear that there were clearly some outliers. A little search led to the revelation that there is a “TOTAL” entry in the data that aggregates some of the variables. This entry in the dictionary was removed. Upon repeating this graphical analysis, I did not find any other significant anomalies.

A Note on General Approach

In trying to find the best algorithm, I tried multiple approaches using multiple permutations of the original data. Below I describe the main approaches I took.

New Feature Creation

In order to improve my models, I decided to create two new variables:

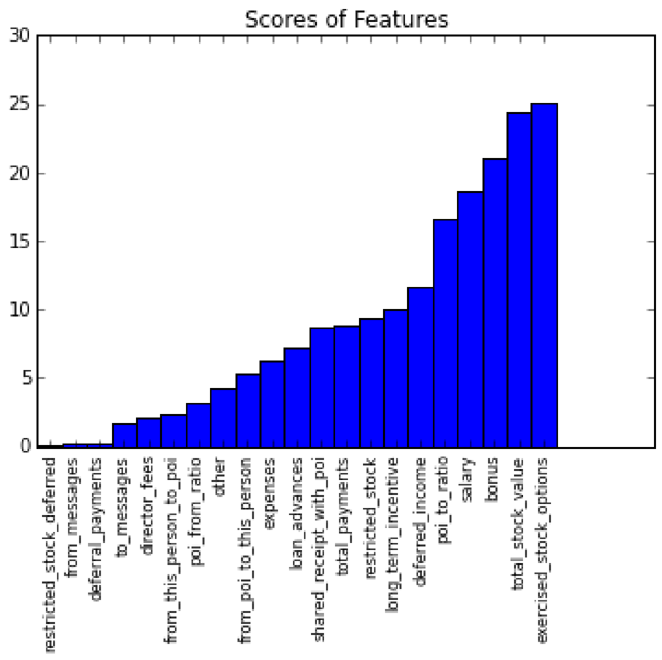
POI_from_ratio = from_this_person_to_poi / from_messages
POI_to_ratio = to_this_person_from_poi / to_messages

Both of these variables were meant to capture the relative importance of communication with POIs relative to all e-mail traffic related to a person. For example, if someone has higher e-mail traffic, we might not be surprised that they naturally have a higher number of incoming and outgoing e-mails to POIs in general.

More on the value of these variables in prediction will be discussed in the feature selection section.

Feature Selection

I started off by using the SelectPercentile function with the ANOVA F-value selection criterion and a threshold of 10% before quickly realizing that 10% of 19 features does not leave very many features. Below is a graph of the F-scores of the different features that were available for use.



In my approach, I ordered the variables in descending order by F-score. From there, starting from the variables with the highest F-score, I added back variables one at a time to see which number of variables would result in the highest f1 score.

In the end, I was left with the following variables:

exercised_stock_options
total_stock_value
bonus

It's important to note that neither of the variables that I constructed were picked in the final analysis. A chart outlining the results of these algorithm-feature combinations will be presented in the *Algorithm Used* section below.

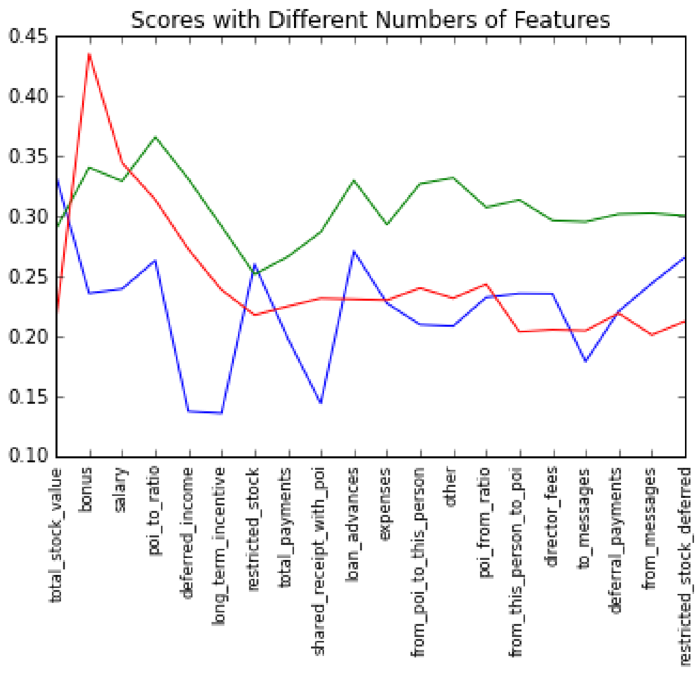
Scaling Data

I scaled the data where scaling was appropriate. I scaled in these cases because the algorithms are sensitive to the variance values of the data. Since data with larger values will mathematically tend to have larger variance values, it's important to place all features on the same scale to ensure an undistorted comparison of features.

Algorithms Used

I focused on three different algorithms, running each on 20 sets of data. The algorithms I utilized were SVM, decision tree, and random forest. I applied each of these algorithms to the data sets presented in the *Feature Selection* section.

The F1-scores of the 60 algorithm-feature list combinations are shown in the plot below. In the plot, the red line represents the Random Forest algorithm, the blue line represents the SVM algorithm, and the green line represents the Decision Tree algorithm. The variables on the x-axis are variables added to the previous variables in that particular iteration of features. On the first iteration, I chose to include exercised_stock_options and total_stock_value, and from there I added one feature at a time. So, when reading the F1-scores above bonus, they would be for the respective algorithms run on the features exercised_stock_options, total_stock_value, and bonus.



Algorithm Selection

The random forest created with the 3 feature data set resulted in the highest F1 score in this case and as a result I selected it. In general, the random forest algorithm worked less well as more features were added. The decision tree algorithm performed relatively well for most feature set. With higher numbers of features, SVM and random forest were roughly equivalent in F1 score. Note that scores were lower with all three types of algorithms with the addition of *poi_to_ratio* and *poi_from_ratio* compared to the selected algorithm-feature set combination.

Algorithm Parameter Tuning

Tuning the parameters of an algorithm involves changing the parameters of an algorithm in order to increase its usefulness. In the specific case of the decision tree algorithm that I selected, tuning the parameters involved maximizing the algorithm's ability to classify POIs using the data. While maximizing an algorithm's ability could mean many things, in this case I used the F1 score as a metric describing the effectiveness of the algorithm. Without tuning the algorithm, I would not have been able to utilize the full potential of the decision tree algorithm that I eventually selected.

In order to tune the algorithm, I used GridSearchCV which automatically ran through a battery of potential parameters and selected the one with the best F1 score obtained through k-fold cross validation. In particular, when tuning the algorithm, I provided GridSearchCV with the following parameters and values to try:

n_estimators: This parameter describes the number of different decision trees that the random forest should produce.

values: range(10,100,10)

criterion: This parameter determines how the algorithm determines splits in the trees.

values: gini, entropy

GridsearchCV returned the following optimal parameters:

n_estimators:	40
criterion:	entropy

Validation

Validation is the process used to determine how well a model will perform when used on data that was not used to generate it. Without validation, there is a good chance that a model will over fit the data, making it far less useful when used with new data.

When validating, it is vital that the correct metric is used. For example, one might look only at recall, defined as the number of true positives divided by the total number of points that should be classified as positives. Unfortunately, in this case, it would be possible to have a perfect recall score of 1 by having an algorithm that classifies all points in the data as positive. By doing this, all of the points that should be classified as positive would be classified properly resulting in a high recall score. On the other hand, there would also be a very high number of false positives. In fact, the number of false positives would be so high that the algorithm would be completely useless. It's necessary, then, to also look at a metric that measure false positives to generate an effective model.

In order to validate my models, I used the F1 score, which combines metrics for both precision and recall with cross validation.

Validation Scores and Interpretation

In order to get to select my final machine learning algorithm from all of the potential choices, I used the F1 score, which is constructed from the precision and recall metrics for validation. I used the F1 score twice to get to the final algorithm.

In the first case, I used 5-fold cross validation which was conducted automatically by GridSearchCV as it searched for the optimal parameters in the process described above. 5-fold cross validation means that the data was put into 5 bins with a different fifth being held out for testing during five different model fits.

When comparing the algorithms against each other, I cross validated using the StratifiedShuffleSplit function used by the tester. In this case, stratifiedshufflesplit shuffles the data 1000 times, holding out 10% of the data to test with. Both the test and the train set are selected to mimic the distribution of the overall dataset. To select the best algorithm, I used this validation method with the F1 score which is a combination of precision and recall.

For the random forest algorithm that performed best, the precision and recall scores were as follows:

Precision:	0.667
Recall:	0.330
F1:	0.355

In this context, a precision of 0.667 means that of all of the people classified as POIs, 66.7% of them were actually POIs. A recall of 0.330 means that of all of the POIs actually in the data, the algorithm identified 33% of them as POIs. Finally, the F1 measure combines precision and recall in order to produce a measure that takes both into account in a single metric.

Sources:

Feature Selection

http://scikit-learn.org/stable/auto_examples/feature_selection/plot_feature_selection.html#example-feature-selection-plot-feature-selection-py

http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html

http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif

Cross Validation

https://randomforests.wordpress.com/2014/02/02/basics-of-k-fold-cross-validation-and-gridsearchcv-in-scikit-learn/

http://stats.stackexchange.com/questions/49540/understanding-stratified-cross-validation

MatplotLib

https://plot.ly/matplotlib/bar-charts/

http://stackoverflow.com/questions/8202605/matplotlib-scatterplot-colour-as-a-function-of-a-third-variable

http://codeyarns.com/2014/10/27/how-to-change-size-of-matplotlib-plot/

Basic Python

http://stackoverflow.com/questions/3179106/python-select-subset-from-list-based-on-index-set

http://stackoverflow.com/questions/2732994/python-sort-a-list-and-change-another-one-consequently

http://stackoverflow.com/questions/13283689/python-sum-the-values-of-lists-of-list

Numpy

http://docs.scipy.org/doc/numpy/reference/generated/numpy.logspace.html

http://stackoverflow.com/questions/18313322/plotting-quantiles-median-and-spread-using-scipy-and-matplotlib

Gridsearch CV

http://abshinn.github.io/python/sklearn/2014/06/08/grid-searching-in-all-the-right-places/

http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html

Pipeline

http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

Background

https://en.wikipedia.org/wiki/Enron_Corpus

Machine Learning Algorithms:

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC