Files

parse_out_email_text.py - from Text Learning module

vectorize_text.py - based on script from Text Learning module

email_text.pkl - output of vectorize_text.py

poi_id.py - final project

tester.py - test script (Note: I could not get this to run, there were multiple errors with StratifiedShuffleSplit that I did not want to take the time to try and troubleshoot. However, it did run to the point of pulling in my data and attempting to score it, so that was enough validation for me)

Questions

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 of this project is to use the given data that we have about the Enron employees financial and email data to try and determine which were/would have been persons of interest in the investigation. Machine learning is a very useful tool here, because there is a lot of data (especially in the emails) and a lot of patterns - the financial data alone has roughly 20 features. It is hard for humans to find patterns out of this much data, so we turn to machine learning to help.

There are multiple ways to use the data to determine if someone is a POI - I attempted two of them. The first is to use this financial + email dataset to see if there are any patterns in the financial or email numbers that correlate with someone being a POI. Another way of looking at this is to use the actual email text to find patters - when I did this in my vectorize_text.py script, I was getting a score of around 90, with a recall of .5 and a precision of .89 - which was a much simpler and quicker way of getting high accuracy compared to using this dataset.

The only outlier I noticed was the "Total" record, which had numbers way higher than any other record. Once I removed this record, I didn't see any other large outliers, so I left the rest of the potential outliers in.

Data exploration

The original dataset has 145 rows and 21 columns

The original dataset has 18 POI records and 127 non-POI records

The following table is the count of NaN values in the original dataset.

	0
salary	51
to_messages	59
deferral_payments	107
total_payments	21
exercised_stock_options	44
bonus	64
restricted_stock	36
shared_receipt_with_poi	59
restricted_stock_deferred	128
total_stock_value	20
expenses	51
loan_advances	142
from_messages	59
other	53
from_this_person_to_poi	59
poi	0
director_fees	129
deferred_income	97
long_term_incentive	80

	0
email_address	34
from_poi_to_this_person	59

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”]

I used an ExtraTreeClassifier to determine the features to use - I used a 60% percentile for the "standard" features (data + financial) and used a 0% percentile for my custom features - as I found that these were consistently marked as 0.0 performance.

I ran three different classifiers against four different datasets - each combination of the standard features vs standard plus my custom features, as well as scaled vs unscaled. The best performing classifier ended up using the unscaled dataset.

My custom features were generated using the text from the emails that each person sent - based on what was in the emails_by_address folder. I then ran these through a TfIdf vectorizer. My assumption was that this should give quite a bit more power to the classifier, as POIs would probably use some words more often than non-POIs. As we'll see below, the datasets with the custom features have a better average Recall score than the dataset that uses only the standard features.

The below stats are for the last run completed.

"The dataset with the highest performance used unscaled features. With this dataset, my custom feature's best classifier had an F1-score of 0.8947, while the standard feature's best classifier had an F1-score of 0.9189."

The best performing classifier used 6 features. Below are the most important features and their relative importance according to the ExtraTreeClassifier.

	index	0
0	exercised_stock_options	0.107956

	index	0
1	bonus	0.091950
2	total_stock_value	0.083077
3	long_term_incentive	0.070065
4	salary	0.066010
5	expenses	0.065296

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 found that the SVC and the Decision Tree Classifier gave me the best balanced score, on average. This being said, I wrote my script to return the classifier that gave me the best balanced score, so I never chose any one algorith per se, I let the metrics choose for me. Besides the SVC and Decision Tree, I also used a Gaussian Niave Bayes classifier, however this consistently performed much lower than the others.

The below stats are for the last run completed.

The best performing classifier was the DecisionTreeClassifier using unscaled features with an F1-score of 0.9189.

With this same set of features, the GaussianNB classifier had an F1-score of 0.4167, and the SVC classifier had an F1-score of 0.8205.

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"]

Tuning an algorithm is to find what parameters for that algorithm give you the best performance. Without tuning your algorithms, you can have much worse performance.

With the exception of the Niave Bayes classifier, I tuned all of my classifiers using the GridSearchCV estimator, passing in a defined list of parameters and potential values. For the SVC classifier the parameters that were tuned were kernel, C, random_state, gamma, and for the DecisionTreeClassifier classifier the parameters that were tuned were min_samples_split, max_features, random_state.

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”]

Validation is how you test that your model works on actual data. The most important thing here is to have, at the least, a separate training and testing data set. If you do your testing on your training data, you are not going to see how your model performs on real data, because it was already trained on the same data you are using for testing.

I validated my data by splitting it into separate training and testing data sets. The original dataset was resampled to add 50 POI records, to assist with the imbalanced classes. After being split into training and testing data the training dataset had 146 rows and the testing dataset had 49 rows.

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”]

There are two evaluation metrics I pay a lot of attention to, precision and recall. The reason I chose to ignore the general accuracy score is that it does not give a great representation when you have imbalanced classes, which is what we had in this dataset. However, the F1 score is another great evaluation metric, especially for imbalanced. It is the harmonic mean of precision and recall. Based on my research, this is a very good metric to use when classifying imbalanced classes with few positive records, like what we have here.

Precision is the ratio of how many times the model **accurately** predicted the record was a POI record, compared to all of the times it predicted the the record was a POI record in total. For example, if it predicted 3 non-POI records were POI records, and it predicted 7 POI records were POI records, this would be a precision rate of .7.

Recall is a bit more important in this model, in my opinion. Recall is the ratio of how many times, when encountering a POI record, did it **correctly** classify that it was a POI record.

Below I have the average for recall, precision, and F1 for the POI class, grouped by both the dataset and by the classifier, as well as by both.

The below stats are for the last run completed.

By feature type (unscaled/scaled, standard/custom)

	recall	precision	f1-score
adj_type			

adj_type	recall	precision	f1-score
scaled	0.705867	0.689667	0.614833
scaled - custom	0.980400	0.682300	0.783433
unscaled	0.745100	0.763867	0.718700
unscaled - custom	0.980400	0.655567	0.767833

By classifier

clf	recall	precision	f1-score
DecisionTreeClassifier	0.970600	0.806675	0.880425
GaussianNB	0.602925	0.542900	0.437525
SVC	0.985300	0.743975	0.845650

By both

adj_type	clf	recall	precision	f1-score
scaled	DecisionTreeClassifier	1.0000	0.7727	0.8718
	GaussianNB	0.1176	0.6667	0.2000
	SVC	1.0000	0.6296	0.7727
scaled - custom	DecisionTreeClassifier	0.9412	0.8421	0.8889
	GaussianNB	1.0000	0.3953	0.5667
	SVC	1.0000	0.8095	0.8947
unscaled	DecisionTreeClassifier	1.0000	0.8500	0.9189
	GaussianNB	0.2941	0.7143	0.4167
	SVC	0.9412	0.7273	0.8205
unscaled - custom	DecisionTreeClassifier	0.9412	0.7619	0.8421
	GaussianNB	1.0000	0.3953	0.5667
	SVC	1.0000	0.8095	0.8947