Project Summary

P5: Identifying Fraud from Enron Email

What is your name?

James McCabe

What E-mail address do you use to sign in to Udacity?

jm6819@att.com – also jamemcc@gmail.com in Piazza and Google +

Goal/use of machine Learning

The goal of the project was to develop a machine learning algorithm that can identify “persons of interest” POIs using email data. For the purpose of this algorithm, POIs are those individuals who are likely to be committing illegal business activities. In this project we built a dataset consisting of individual involved in the “Enron Crisis” of 2001 and then used various machine learning techniques largely depending on the scikit-learn resource. This data was largely extracted from the Enron Email Corpus. Given a large e-mail extract from some future company this Algorithm should be able to identify POIs there, thus helping internal or external investigators to quickly sort through the entire company's personnel using their e-mail.

What features did I pick?

The features given/developed in class in the final project dataset consisted of 2 general types , financial information about each Enron affiliate person and counts derived from their e-mail. The ways we sorted through and extended these features are described below.

The 'e-mail” category of features was increased using a Bag-of words derived algorithm I call weasel words. The Weasel words algorithm first organizes all e-mails by sender and then counted the # of times specific words from a list were used in that senders e-mails.

Both the financial as well as the e-mail related features were scaled. The financial data had vastly different scales e.g. Director fees with a maximum of $101,250 and Total Payments as high as $103,559,793. In order to have a more valid and regular relationship (slope) using SVM or K means clustering it's good to scale these. Bag of Words counts had a different scaling need in that for some subjects there were lots of e-mails compared to others and so their word counts tended to be higher. We needed to scale the word counts by # of e-mails in essence giving the % or e-mails in which the word appeared.

In looking through the data for outliers, the following outliers were observe and dealt with, some needed fixing as described and others were valuable in their own right.

1. Total Payments – Ken Lay had Total Payments feature with value vastly higher than anyone

else, +$100M. Of this $81M was in loan advances which no one else had. This feature of “loan advance” seems unusable due to this and was removed from the analysis. It called into question if Ken Lay as a sample was an outlier but it does not seem good to throw him out when as a POI his other features should be especially helpful to the machine learning process.

2. From looking at the Weasel word score it was identified that Mark Haedicke had a very high concentration of legal type words and yet this outlier is due to the fact that Marks role at Enron was as legal counsel and he was in a position to coordinate other outside counsel. Determined his scores were not useful and removed Mark entirely as an outlier.
3. From looking at the Weasel score for the word 'kill' realized that Jeff Skilling and those who sent to him had a very high value. Since “kill' is in the name “Skilling”, determined this feature was not useful and removed it as an outlier.

<u>What Algorithms did I try and how did I tune parameters?</u>

Both Decision Tree as well as a combination of Principle Component Analysis (PCA) and Support vector Classifier (SVC) machine learning algorithms were used. In general the decision tree made sense to sort between the fairly different financial versus word count type of features. It helped to better understand which features were most relevant. On the other hand and ultimately the better algorithm came from the PCA/SVC which is advised to give better results on training sets such as this where there are more features and fewer observations.

Decision tree: Decision tree was run resulting in a selection of top features and their feature importances as follows:

```
index 2 was 0.496972082702 for deferred_income (financial)

index 6 was 0.111674174174 for from_messages (e-mail count)

index 20 was 0.0434623813002 for just (e-mail weasel word count)

index 25 was 0.258552022484 for target (e-mail weasel word count)

index 28 was 0.0893393393393 for opportunity (e-mail weasel word count)
```

The main tuning parameters for the decision tree are:

- min_samples_split: ended at 2 this gave the most consistent results. When this was set too high (e.g. 5) it often produced a very different tree each run. -
- max_depth: ended at 3 as this gave us a few classifiers in the output. When setting to 1 it only gave 1 classifier which was not conducive to producing the features importances comparison. Notably the depth of 1 resulted in using only deferred income as the decision point.

PCA/SVC:

A combination PCA and then SVC was run.

The PCA was evaluated in reducing the # of features from 38 down to 1,2,3,5 and 10 principle components, evaluating each by it's accuracy. The highest accuracy came from using just the top 3 principle components .

The SVC was tuned using the Grid Search algorithm to find the best Kernal, C value and gamma. This ended up as:

```
kernel='poly', C=1000.0, gamma=0.01
```

When the more default linear and or rbf kernels were used the algorithm did not result in as good of performance. Interestingly for suboptimal PCA component amounts (1, 5 or 10) the optimal C values were 5000 and gamma lower at .005 to .0001. In more plain words it seems the SVC was over fitting to make up for the less helpful components.

How did I validate and what if I did not?

The performance was validated first via the accuracy, the fraction (default) of correct predictions. For this the Decision Tree scored 93.3% and the PCA/SVC scored 94.7%.

The performance of each was then validated via precision and recall scores. The decision tree would produce different results each run but as a high point produced this:

```
                    precision    recall  f1-score   support

         not a POI       0.93      0.96      0.95        27

      this is a POI       0.50      0.33      0.40         3

       avg / total       0.89      0.90      0.89        30
```

The PCA/SVC produced this:

```
                    precision    recall  f1-score   support

         not a POI       0.94      1.00      0.97        34

      this is a POI       1.00      0.50      0.67         4

       avg / total       0.95      0.95      0.94        38
```

The PCA/SVC scores better on all dimensions. Most notably the most desirable statistic is Precision for the situation of interest (when someone is a true POI), we would like it to always identify as a POI. It is less critical if we define someone else as a POI who is not as this algorithm would be followed up by much more investigation. That statistic that is less critical here is the Recall.

Method of Validation and danger if done wrong:

Validation of any machine learning developed algorithm needs to be done with a separate section of testing data. This means taking all the data your have for developing your algorithm and splitting it into testing and training sections.

For the PCA/SVC algorithm finalized in this exercise the a 1 pass approach using SKLEARN based cross validation test_train_split routine reserving 25% of the input data for testing was used.

In the final test of the algorithm the tester.py script uses the StratifiedShuffleSplit to test the accuracy using 1000 random passes through the data reserving 10% for testing each pass.

One of the larger dangers of a test train split is if the data is not in random order then test and train split may select in essence different categories of data for testing versus training. The stratified shuffle split at a high iteration count can help alleviate it this. Best case is ensuring data is in random order to begin with. In our data the POIs seem randomly dispersed already.

A second common issue with validation especially with a multiple step algorithm such as PCA/SVC is to ensure the testing data is not just applied in the final predict step but in all steps of the algorithm. In this case in validating the PCA and SVC on the testing data both steps were run on testing data and the final CLF is created as a Pipeline to ensure both PCA and SVC are fit with the testing data.

<u>Summary and next Steps:</u>

The quest for the best machine learning algorithm to use on company data including financial and e-mails in order to identify a person of interest (POI) in terms of fraudulent activity involves the steps of:

1. data investigation
2. creating additional data
3. identifying and deciding what to do with outliers
4. investigating what features of the data to use in the algorithm which also is very tied to the type of algorithms used.
5. Evaluating the accuracy and likely also the precision and recall of the results using K fold testing.

For this case a combination pipeline built using e-mail word count data fed to a pipeline of PCA and then SVC results in the highest precision results.