DAND Project 5: Machine Learning - Enron POI

Stephan Teodorovich
May, 2018

Q1: Project 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? [relevant rubric items: "data exploration", "outlier investigation"]

The Enron scandal is one of the most well-publicized accounts of corporate greed, fraud, corruption, and deceit in recent history and included price fixing, insider trading, embezzlement, accounting fraud, and lies.

This project is an attempt to take a dataset consisting of corporate emails (including To and From information, as well as text) and financial information (including salary, bonus, and stock value) and apply data wrangling techniques and machine learning algorithms as a means to determine the primary Person of Interest (POI) for the scandal by analyzing the patterns and habits exhibited from the email data.

The machine learning aspect of the project is vital in that it can take a large set of complex data (such as email headers and sender & recipient information) and efficiently determine both the relationship between disparate data points (i.e. email subjects and individual salary), as well as evaluating patterns, thereby allowing one to confidently identify and classify otherwise unknown data points.

1.1: Goals

The goals for this project are to:

- Be able to review a dataset to understand the information it contains
- Recognize which variables/features are relevant to the question asked
- Select the proper tools to manipulate and evaluate the data
- Prepare the data properly for Machine Learning

I utilized the following packages:

- **numpy** - efficient numerical computations
- **pandas** - data structures for data analysis
- **scikit-learn** - machine learning algorithms, dataset access
- **matplotlib** - plotting (both interactive and to files)
- **seaborn** - extra plot types, elegant and readable plot style

1.2: Data Exploration, Outlier Identification & Cleaning

All of the coding for this project is found in the poi_id_st.ipynb file accompanying this report. The data exploration followed this path:

1. Examine the structure of the dataset
2. Manipulate null entries to a usable data type
3. Review the data to determine the structure and possible presence of outliers
4. Confirm validity and usability of the data
    The results showed:

   - Data set n=146
   - Data set features=21
   - Data set POIs: 12.3% (18 of 146)
      One of the biggest concerns was the presence of null data, or NaN. The number of NaN data points varied from feature to feature. For example, the ~25% of the Restricted Stock feature contained NaN values (36/146), while the Loan Advances feature contained ~97% NaN (142/146)!

     During the initial data exploration, NaN values were converted to 0 in order to locate outliers, but as will be discussed, NaN was handled differently during the machine learning section.

     Simple scatterplot analysis was used to determine the presence of outliers. A quick review of the provided *enron61702inderpay.pdf* document showed that one of the outliers was the non-human data point **Total**, while the other outlier was identified as **Ken Lay**.

     Given the presence of a non-human data point, I also did a visual review of all of the entries in the pdf document and found one other easily identifiable non-human listed as **The Travel Agency In The Park**. Both **Total** and **Travel Agency** were removed from the data set, leaving a total of 144 entries.

     The last step was to ensure that the entries in the dataset were valid. This was accomplished by comparing the listed **Total Payments** and **Total Stock Values** entries with a calculated sum of their respective components for each line (the logic being that an error in data entry would result in the calculated totals and entered totals not matching).

     The review identified two lines with conflict: Sanjay Bhatnagar and Robert Belfer. A comparison of the dataset with the pdf document confirmed a data entry error, which was corrected.

     The data was now reviewed, confirmed, and cleaned and ready for the next step.

Q2: Feature Use and Creation

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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

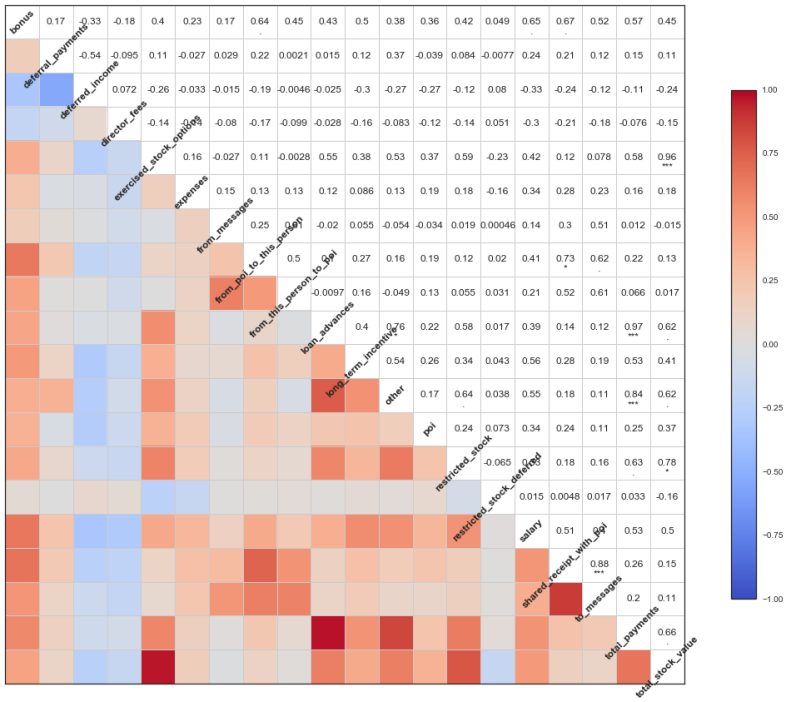
2.1: Features Used

In selecting the featured for use in this project, I employed the **SelectKBest** module from **scikit-learn** to find the 1o most relevant features.

Along with **SelectKBest**, I have also created a heatmap and a correlation matrix to try and zero-in on the best features to use. Ultimately, the heatmap proved to be of very little value, while the correlation matrix did seem to provide a bit of clarity:

In [1]:
```
from IPython.display import Image
Image(filename='attribute_correlations.png', width=800, height=600)
```

Out[1]:



Eventually I identified what I considered the 10 most important features to retain using the **SelectKBest** module focusing on both the overall Score and number of Valid (i.e. non-NaN) entries (shown in Table 1, below)

Table 1: Selected Features

Feature	Score	Valid
exercised_stock_options	24.815	101
total_stock_value	24.183	125
bonus	20.792	81
salary	18.290	94
deferred_income	11.458	48
long_term_incentive	9.922	65
restricted_stock	9.212	109
total_payments	8.772	123
shared_receipt_with_poi	8.589	86
loan_advances	7.184	3

A point of interest is that **loan_advances** has a comparably high Score, even with only 3 non-NaN entries.

The K-Best approach automates the univariate features selection algorithm. However, the lack of email feautres in the list is of some concern. Therefore I created three new features,**poi-ratio**, **fraction_to_poi**, and **fraction_from_poi** (described in more detail in section 2.2, below) to help increase the precision and accuracy of the algorithms tested.

2.1.1: K-Best Results

The table below shows that even before we get into the tuning and validation of our algorithm (DecisionTreeClassifier), using the 10 features identified with K-Best has better performance than using all the features (values rounded to 3 decimals):

Table 2: Performance Comparison using K-Best selected features

DTC	Precision	Recall	Accuracy
All Feautures	0.257	0.261	0.74
10 K-Best	0.293	0.284	0.77
11 K-Best	0.293	0.299	0.77
12 K-Best	0.287	0.303	0.81

Notice that increasing the features size to 11 or 12 selected by K-Best did not provide significantly better performance.

The next table shows the results when also adding the three newly created features (rounded to three decimals):

Table 3: Performance Comparison using K-Best selected features plus the three new features

DTC	Precision	Recall	Accuracy
All Feautures (incl. new)	0.369	0.213	0.84
10 K-Best (incl. new)	0.418	0.452	0.84
11 K-Best (incl new)	0.485	0.450	0.86
12 K-Best (incl. new)	0.478	0.452	0.86

First, we can see an immediate improvement in scores for all options when the three new features are added, with significant increases across the board. Also, like the run without the new features, there isn't a meaningful improvement in scores when we move from the top 10 K-Best features (plus the three new feautres) to either 11 or 12.

Therefore, I will continue using the 10 -Best features along with the three new features.

2.2: Feature Creation & Engineering

However, this list is comprised almost entirely of financial features. Therefore, I went ahead and created three new features with an email focus:

- **poi_ratio**: The proportion of messages connecting with a POI compared to all messages<.li>
- **fraction_to_poi**: The proportion of messages sent to a POI over all messages sent
- **fraction_from_poi**: The proportion of messages received from a POI over all messages received

The idea behind these new features was to focus on the connection in any communications two and from each individual and identified POIs, to try and determine any new POIs. The logic is that POIs would almost certainly have greater communication between themselves than they would with non-POIs.

2.3: Feature Scaling

Feature Secaling helps control for the influence of overly-strong features in order to help ensure an accurate result.

For this project I used Scikit-learn's **MinMaxScaler()** function to normalize all features to reduce the impact of the value of the financial variables (like **Salary**) on the email features (like **T0 Messages**) which can be several orders of magnitude lower.

Q3: Algorithm Choice

What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: “pick an algorithm”]

2.1 Algorithms Considered

There were several relevant algorithms from which I could select for the Machine Learning portion of the project:

- GaussianNB (GNB)
- DecisionTreeClassifier (DTC)
- SupportVectorClassifier (SVC)
- KNeighborsClassifier (KNC)
- RandomForestClassifier (RFC)
- AdaBoostClassifier (ABC)
- ExtraTreesClassifier (ETC)
   To help determine which would be the most effective choice, I reviewed each to learn their relative performance in terms of Accuracy, Precision, and Recall

  The project rubric specified a minimum score of 0.3 for both Precision and Accuracy. The table below indicates the test results (rounded to the third decimal):

  Table4: Algorithm Review Results

Algorithm	Accuracy	Precision	Recall
GNB	0.25	0.154	0.702
DTC	0.84	0.310	0.296
KNC	0.77	0.215	0.100
SVC	0.84	0.0	0.0
RFC	0.80	0.227	0.088
ABC	0.75	0.285	0.243
ETC	0.82	0.298	0.139

  As is clear, only DTC had both Precision and Recall close to (or above) the rubric minimum. Therefore my choice was to use DecisionTreeClassifier.

Q4: Algorithm 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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

4.1 What is Algorithm Tuning?

Machine learning algorithms are parameterized just like functions, and can be modified and tweaked to improve performance and the reliability of the learning process. "Tuning" is simply making these small changes and modifications after an initial run using the default values.

The only area of caution is to avoind over-tuning the algorithm, as this may end up overfitting, negatively affecting results.

4.2 Algorithm Tuning

The scikit-learn's **grid_search.GridSearchCV** module to determine a good algorithm configuration. I was then able to take use these results in an optimization algorithm to find the best performance, using the following parameters:

```
parameters = {'max_depth': [1,2,3,4,5,6,8,9,10],'min_samples_split':
[1,2,3,4,5],'min_samples_leaf':[1,2,3,4,5,6,7,8], 'criterion':('gini', 'entropy')}
```

After applying these to DTC I obtained these results:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=2,
max_features=None, max_leaf_nodes=None, min_samples_leaf=2, min_samples_split=2,
min_weight_fraction_leaf=0.0, random_state=None, splitter='best') 0.657142857143 Processing
time: 14.54 s
```

Q5: 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"]

5.1: What is Validation

Validating is the process by which one can determine how well the algorithm will fit, and how well it can generalize, any data beyond the initial dataset on which the algorithm was trained.

For the purposes of this project, we used an initial dataset with a very small size (n=144), and so we needed to split the data in order to train, test, and validate.

5.2: Mistakes

Perhaps the most common mistake during validation is to overfit the model. This problems is almost counter-intuitive, as over-fitting ends up having the initial training data perform exceptionally well with the algorithm, but at the expense of generalization. The result is that new data will actually permform poorly.

Overfitting is the situation when a model learns and adapts to the nopise and variation of the training data so well that it ends up not being able to perform well with new data, because the variation in the test data will not apply to the new data, meaning the model is too-specific and unable to generalize.

The corrolary to this is he mistake of underfitting the data, where the model hasn't been trained long enough to be able to generalize to new data.

5.3: Analysis Validation

Validation was performed by using the provided **evaluate.py** and **tester.py** scripts.

- **evaluate.py** reported the average Precision and Recall from 1,000 randomized trials, with the data split into a 3:1 training to test ratio
- **tester.py** used StratifiedShuffleSplit with folds=1,000, to provide the metrics of the classifier.

Q6: 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. [relevant rubric item: "usage of evaluation metrics"]

6.1 Evaluation Metrics

The project rubric specified that **Precision** and **Recall** both have a score above 0.30, and so those are the metrics I will use.

Precision is the “fraction of relevant instances among the retrieved instances”, while **Recall** is the “fraction of relevant instances that have been retrieved over the total amount of relevant instances”. In practical terms, this means as **Precision** approaches 1.0, the more true positives have been identified without any false positives. Similarly, as **Recall** approaches 1.0 more true positives are identified without any false negatives.

For this project **Recall** seems to be the more critical value, as the search for POIs would benefit more from a lack of false negatives than false positives – especially given that the Enron scandal was, at its core, a criminal activity.

DTC Average Performance (rounded to three decimals)

Validation 1 (StratifiedShuffleSplit, folds = 1000)

Precision	Recall	Accuracy
0.483	0.561	0.862

Validation 2 (Randomized, partitioned trials, n=1000)

Precision	Recall	Accuracy
0.332	0.390	0.82

6.2: Interpretation

As mentioned above, I find **Recall** to be more important in this analysis than **Precision**.

Since the point of the project was to identify additional *potential* POIs, one would think that being over-aggressive in identification would be better than being too restrictive. Thus, maximizing the number of True Positives while minimizing False Negatives would be preferred. This would certainly lead to false positive POI idenfication, however, as the initial step in a hypothetical investigation, those falsely identified POIs could be removed through additional work. The alternative of **Precision** would be far more likely to allow legitimate POIs to avoid initial detection, and escape completely.

Resources and References

- Introduction to Machine Learning (Udacity) (https://www.udacity.com/course/viewer#!/c-ud120-nd)
- scikit-learn Documentation (http://scikit-learn.org/stable/documentation.html)
- RandomForest Wordpress: K-Fold Cross Validation and GridSearchCV in Scikit-Learn (https://randomforests.wordpress.com/2014/02/02/basics-of-k-fold-cross-validation-and-gridsearchcv-in-scikit-learn/)
- gadatascience (http://www.gadatascience.com/modeling/logistic.html)
- Machine Learning Mastery: Overfitting and Underfritting with Machine Learning Algorithms (https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/)

In [ ]: