Identifying Corporate Fraud: A Summary

Project submission for Udacity's Data Analyst Nanodegree Program
Completed by Narissa Kreutz, November 20th, 2017

Question 1:

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?

The goal of this project is to use Machine Learning Algorithms to identify potential "persons of interest", or "poi's" from the corpus of information available regarding the investigation of fraud at Enron in 2001. See the Enron Email Dataset, the Wikipedia page or watch the movie "Enron: The Smartest Guys in the Room" for more details on the case. As per SAS, Machine Learning algorithms iteratively learn from data, allowing computers to find hidden insights without being explicitly programmed where to look. Therefore, using machine learning is appropriate in this case as a "simple answer" is not easily determined by human intuition when looking at the data, or from more straight-forward methods like linear regressions.

There are 146 people and a total of 21 features in the dataset. Of those features, 14 are financial, such as salary and exercised stock options in US dollars, 6 are email features, such as how many emails were sent from a poi to this person, and the last feature is the label feature, where people who are identified as poi's are labeled as a 1, and non-poi's as a 0. The poi label feature was hand-generated by Udacity, and identifies individuals who were indicted, reached a plea deal or settlement with the government, or testified in exchange for prosecution immunity. There are 18 people in the dataset who are identified as poi's, and 35 poi's total identified by Udacity, therefore this dataset is incomplete.

This dataset also contains many missing values in the financial features, every person in the dataset has at least one "NaN" value, and it is unknown whether these missing values account for missing data, or simply that the value for that feature for that person is $0. In the processing of the data, all "NaN" values are converted to $0. There was also a large outlier in the financial features, which was the Total of all of the values in each feature, which I removed from the dataset.

Question 2:

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. 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.

In this project, I decided to custom engineer two features. The first is a combination (sum) of the total payments and total stock value for each person:

'combined_pay_and_stock' = 'total_payments' + 'total_stock_value'

The rationale behind this metric would be that poi's may have greater sums of payments and stock values than non-poi's.

The second custom feature is the proportion of payments made to a person in relation to their base salary:

'proportion_pay_to_salary' = ('total_payments' – 'salary') / 'salary'
The rationale behind this metric would be that poi's may receive payments on top of their salary that are many times higher than their base salary.

For the feature selection step, I decided to use a pipeline with SelectKBest, first applying a MinMaxScaler, as it improved my results. Using GridSearchCV, the best number of features to use was k=5, and the features selected were:

1. 'exercised_stock_options' (score=25.10)
2. 'total_stock_value' (score=24.47)
3. 'bonus' (score=21.06)
4. 'salary' (score=18.58)
5. 'combined_pay_and_stock' (score=17.19)

Question 3:

What algorithm did you end up using? What other ones did you try? How did model performance differ between algorithms?

In my investigation, I tried the GaussianNB, Support Vector Machines, Decision Tree Classifier, Random Forest Classifier and AdaBoost Classifiers. I ended up using the GaussianNB() Classifier algorithm in my model as it produced the best evaluation metric scores. Using different combinations of the above algorithms, some produced precision and recall scores as low as 0, others had higher precision scores,(up to 0.5) however had low recall scores and vice versa.

Question 4:

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?

I decided to use GridSearchCV to tune my parameters as I felt this was the most robust way to find the most effective parameters. However, with the scaler, dimension reduction and classifier algortithms that I ended up choosing for my final model, the only parameter that required tuning was the k parameter in SelectKBest(). For this, I chose a range of 2 through 7 features, and the optimal k was equal to 5.

Question 5:

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 the process of splitting your data into training and testing sets. By using only the training set to train and fit the classifier, and only the testing set to evaluate it's performance, you are able to estimate how accurate your trained classifier is on unseen, independent data. This serves as a check for over-fitting, which is a classic mistake that can be made if validation is done wrong.
in my analysis, because the data is highly skewed, and only has a few positive poi labels, I used a StratifiedShuffleSplit within my GridSearchCV with scoring set to f1_weighted (as f1 is the metric that best represents the true performance of my model).

Question 6:

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.

The tester.py script calculates the accuracy, precision, recall, f1 and f2 scores. My final model has an accuracy of 0.85800, which means that it is predicting the correct value ~85.8% of the time. While this seems high, in a dataset that is skewed, like this one, the algorithm can choose the more common class 100% of the time, and reach high accuracy rates. Since we care about "catching" True Positives (aka labeling poi's as poi's), this is not what we want our algorithms to do. So in this case, it is more applicable to look at other metrics, such as Precision and Recall. My final model has a Precision of 0.45536, which means that when my algorithm predicts that a person is a poi, it is correct ~45.5% of the time. The Recall was 0.33150, which means that my algorithm "caught" ~ 33.2% of the poi's in the test set. Since the dataset is limited in multiple ways e.g. missing data on multiple poi's, skewed data (not a lot of examples of poi's in the dataset), and many "NaN" values, we couldn't expect to get metrics that would be significantly more accurate.

References:

Sci-kit Learn: For the documentation on the Algorithms I trialed and implemented in my model, specifically Pipeline, GaussianNB Classifier, Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier, GridSearchCV, StratifiedShuffleSplit, StratefiedKFold, train_test_split, Support Vector Classifier, Principal Component Analysis, MinMaxScaler, Metrics, SelectKBest
Enron Email Dataset
Wikipedia
"Enron: The Smartest Guys in the Room"
SAS
Udacity's Data Analyst Nanodegree
Stack Overflow

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.