Project: Identify Fraud From Enron Email

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

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. The objective of this project is to build a 'person of interest' identifier based on financial and email data made public as a result of the Enron scandal. Person of interest is defined here as individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Machine learning algorithms are useful in trying to accomplish goals like this one because they can process datasets faster than humans and they can spot relevant trends that humans would have a hard time realizing manually. Below is given some background on the Enron financial and email dataset.

Data points and allocation across classes(poi/non-poi)

There are 146 Enron employees in the dataset. 18 of them are pois.

Features

There are fourteen (14) financial features. All units are US dollars.

financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)

There are six (6) email features. All units are number of emails messages, except for 'email_address', which is a text string.

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi']

There is one (1) other feature, which is a boolean, indicating whether or not the employee is a person of interest.

POI label: ['poi'] (boolean, represented as integer)

For the purpose of the exercise all the features except 'email_address' were initially chosen.

Missing Features

20 of the 21 features have missing values (represented as "NaN"), with the exception being the "poi" feature.

The missing financial features are imputed by featureFormat to zero (0). Imputing to zero makes sense for these features because we have a reasonably complete financial picture through the FindLaw "Payments to Insiders" document. As mentioned in one of the discussion threads, if a feature has a dash ('-'), one can assume that means it is zero.

For the missing email features each feature's mean was imputed. Imputing to zero does not make sense in this case because the email data appears incomplete. A missing feature likely means we could not find the data, rather than that the value is zero.

Outliers

Every financial feature had a huge outlier generated by the "Total" row in the FindLaw "Payments to Insiders" document. These were removed from the dataset. There was another non-employee entry in the dataset belonging to "The Travel Agency in the Park" that was removed as well. Finally, for the employee named ‘Eugene E Lockhart’, there did not appear to be any records in the ‘Payments to Insiders’ document. This particular employee was removed as well.

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. (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”, “properly scale features”, “intelligently select feature”]**

Features Engineered

Two new features were added to the dataset to bring the total number of features to 23:

- From this person to poi fraction
- From poi to this person fraction

Scaling the 'from_this_person_to_poi' and 'from_poi_to_this_person' by the total number of emails sent and received, respectively, might help us identify those who have low amounts of email activity overall, but a high percentage of email activity with ‘poi’s.

As can be seen from the ‘Features Selected’ section below, both these new features were selected by the chosen algorithm (decision tree). ‘Fraction to poi’ figured high compared to other features in terms of both feature importance (No 2) and score (13.714). ‘Fraction from poi’ was sixth in terms of feature importance and had a score of 5.906.

Scaling

The final algorithm chosen was a decision tree. As mentioned in a lesson, scaling is not required for tree-based algorithms because the splitting of the data is based on a threshold value. The decision made based on this threshold value is not affected by different scales.

Selection Process

I used a univariate feature selection process, select k-best, in a pipeline with grid search to select the features. Select k-best removes all but the k highest scoring features. The number of features, 'k', was chosen through an exhaustive grid search driven by the 'f1' scoring estimator, intending to maximize precision and recall.

Features Selected

The following 15 features were used in my POI identifier, which was a decision tree classifier that gave the highest precision and recall scores. The first number is feature importance (from the decision tree classifier) and the second is feature score (from select k-best). The order of features is descending based on feature importance.

```
The 15 features selected and their importances:
feature no. 1: expenses (0.374691075074) (0.197148817803)
feature no. 2: fraction_to_poi (0.246751266963) (13.7140212886)
feature no. 3: shared_receipt_with_poi (0.138338836161) (0.409359730015)
feature no. 4: total_payments (0.125043933251) (4.26289149582)
feature no. 5: total_stock_value (0.115174888551) (2.76789862612)
feature no. 6: fraction_from_poi (0.0) (5.90622307847)
feature no. 7: director_fees (0.0) (14.6913086526)
feature no. 8: restricted_stock (0.0) (5.30405931551)
feature no. 9: long_term_incentive (0.0) (0.763457106222)
feature no. 10: exercised_stock_options (0.0) (11.1294791511)
feature no. 11: deferred_income (0.0) (0.259376640855)
feature no. 12: bonus (0.0) (11.1962683054)
feature no. 13: salary (0.0) (1.53383619918)
feature no. 14: from_this_person_to_poi (0.0) (2.50658908505)
feature no. 15: from_poi_to_this_person (0.0) (0.153469841885)
```

Expenses, fraction to poi, shared receipt with poi, total payments and total stock value were the five most important features.

3. **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 focused on three algorithms, with parameter tuning incorporated into algorithm selection. These algorithms were:

- Decision Tree Classifier
- Naïve Bayes
- SVM

Here is how each performed:

- The decision tree classifier had a precision of 0.44194 and a recall of 0.46050, both above the 0.3 threshold.
- The Naïve Bayes classifier had a precision of 0.39542 and a recall of 0.328, both above the 0.3 threshold.
- The SVM classifier had a high precision of 0.53828 but a poor recall of 0.1125. Possibly this algorithm is not well-suited to a dataset with imbalanced classes.

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? (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 item: "tune the algorithm"]**

Tuning the parameters of an algorithm means adjusting the parameters in a certain way to achieve optimal algorithm performance. There are a variety of "certain ways" (e.g. a manual guess-and-check method or automatically with GridSearchCV) and "algorithm performance" can be measured in a variety of ways (e.g. accuracy, precision, or recall).

In the present exercise I used GridSearchCV for parameter tuning. As described in the relevant lesson, GridSearchCV allows us to construct a grid of all the combinations of parameters, tries each combination, and then reports back the best combination/model.

For the chosen decision tree classifier, apart from multiple parameters for the feature selection step, I tried out multiple different parameter values for criterion and min sample split. The different parameter values tried for the entire pipeline (with the optimal combination bolded) are given below.I used Stratified Shuffle Split cross validation to guard against bias introduced by the potential underrepresentation of classes (i.e. 'poi's).

- k = [2,3,4,5,10,**15**,20]
- criterion=['gini', **'entropy'**]
- min_samples_split=[2,4,10,**20**]

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 item: “validation strategy”]**

Validation is a way to substantiate the machine learning algorithm's performance, i.e., to test how well your model has been trained. A classic validation mistake is testing your algorithm on the same data that is was trained on. Without separating the training set from the testing set, it is difficult to determine how well your algorithm generalizes to new data.

In my poi_id.py file, I used 10-fold Stratified Shuffle Split cross-validation to tune the algorithms and extract the best parameters. Because the Enron data set is so small, this type of cross validation was useful because it essentially created multiple datasets out of a single one to get more accurate results.

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. [relevant rubric item: “usage of evaluation metrics”]**

The two notable evaluation metrics for this poi identifier are precision and recall. The average precision for my decision tree classifier was 0.44194 and the average recall was 0.46050. These metrics can be interpreted as follows:

- Precision measures what percentage of our prediction of the true class (in this case ‘poi’) actually contains the true class.
- Recall measures what percentage of the actual true class is captured in our prediction.
- In the context of our ‘poi’ identifier, it is arguably more important to make sure we don't miss any ‘poi’s, so we don't care so much about precision. In other words, the cost of a ‘false negative’ (not labelling somebody a ‘poi’ when he or she actually is one) is higher than that of a false positive (labelling somebody a ‘poi’ when he or she actually is not one). When we label somebody wrongly as a ‘poi’, we can always rectify our mistake later through due diligence. For our case, we want high recall: when somebody is a ‘poi’, we need to make sure as far as possible that we are capturing that information. The decision tree algorithm had the highest recall score among all the three algorithms tested and hence was chosen in the end.

References:

1. Sklearn documentations on algorithms and methods
2. Udacity DAND P5 discussion threads.
3. https://public.tableau.com/profile/diego2420#!/vizhome/Udacity/UdacityDashboard