Identify Fraud from Enron Email

Jill Andersen
Student ID: 001374500

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 accounting scandal was publicized in 2001. The dataset is a subset of Enron employees, which identifies known persons of interest (POI). The goal is to identify if there are any correlations to POI's to suggest that some of the employees not currently listed as POI's could also be added to the list of POI's.

Below are a list of the features in the dataset:

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)

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

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

Data Exploration of the Dataset

Number of people: 146
# of features for each person: 21
Count of POI's: 18
Count of Non-POI's: 128
Number of features for each person: 21

Outlier Investigation

"TOTAL" is an aggregate, so it is clearly an outlier. Therefore, it was removed.

Optimize Feature Selection

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"]

New Features

I chose to analyze the significance of whether or not a non-POI sending emails to a POI or from a POI is cause to be suspect.

- 'to_poi_message_ratio'
- 'from_poi_message_ratio'

Selection Process

Using scikit-learn's SelectKBest feature selection these are the top 10 features:

1.	salary:	18.58
2.	total_payments:	8.87
3.	loan_advances:	7.24
4.	bonus:	21.06
5.	total_stock_value:	24.47
6.	shared_receipt_with_poi:	8.75
7.	exercised_stock_options:	25.1
8.	deferred_income:	11.6
9.	restricted_stock:	9.35
10.	long_term_incentive:	10.07

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 tried all four of the algorithms listed below.

Prior to new feature:

GaussianNB:
Accuracy: 0.37620	Precision: 0.15599	Recall: 0.83400

SVM:
('Got a divide by zero when trying out:',

Means:
Accuracy: 0.84720	Precision: 0.13131	Recall: 0.02600

Decision Tree:
Accuracy: 0.79040	Precision: 0.20270	Recall: 0.19500

After new feature:

GaussianNB:
Accuracy: 0.74713	Precision: 0.23578	Recall: 0.40

SVM:
Accuracy: 0.83753	Precision: 0.27310	Recall: 0.13150

KMeans:
Accuracy: 0.83040	Precision: 0.21186	Recall: 0.10000

Decision Tree:
Accuracy: 0.82293	Precision: 0.32737	Recall: 0.31100

I chose to tune GaussianNB and Decision Tree since they had the best precision and recall after creating the new feature. I used GridSearch to search for the best parameters. The CV was set at 3 and iid as false. Ultimately the best algorithm found was Decision Tree with entropy as the parameter for criterion. The entropy of a random variable is the average amount of information or uncertainty in the variable's possible outcome. GaussianNB had a high precision, but the recall was low. There is a disproportionate amount of POI's to non-POI's so it is likely that there were sets of testing data that did not include many, or possible no POI's. I chose to study the parameter var_smoothing. This is the calculation of the distance of the samples to the sample mean. If there are minimal or no POI's in the dataset the precision will be high because there is not a lot of variation in the dataset,.

After hyperparameter tuning with GridSearch

GaussianNB:

- Parameter range value tested was var_smoothing: 0 to -9.
- The best parameter found was: var_smoothing = 0.6551285568595508
- Accuracy: 0.88093 Precision: 0.72495 Recall: 0.18450

Decision Tree:

- Parameter range value tested was criterion: 'gini' & 'entropy'
- The best parameter found was: entropy
- Accuracy: 0.82367 Precision: 0.34274 Recall: 0.33400

Scaling

Additional scaling was not necessary since the number of emails was fairly standard across all employees.

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"]

The purpose of tuning the parameter is to increase the precision and recall of your algorithm. You tune the parameter by scaling the features using classifiers such as:

- Decision Tree Criterion
  - Option for measuring how to split are 'gini' and 'entropy'
  - max_depth = If the depth is too high you will overfit the tree
  - min_samples_split = Minimum number of splits required to continue splitting. If set to too few you could overfit.
- Support Vector Machine Criterion
  - C = A low value with create a smoother/straighter decision boundary. A high value will likely result in overfitting.
  - Gamma = A high value will also result in overfitting.

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 tests the quality of your algorithm against a test data set to verify your results have a high precision and recall. Splitting your data two separate data sets, one or training your algorithm and one for testing the algorithm. The larger dataset becomes you training set as you need to have as many examples as possible to increase successful tuning of your algorithm. The smaller dataset will be come your testing dataset and will be used to validate your training classifier.

StratifiedShuffleSplit (SSS) was used in tester.py to validate the model. SSS is a cross-validation method that splits the train and tests sets by shuffling randomized folds. You can specify the number of reshuffles and splits. The default number shuffles is 10. The enron dataset is a small dataset that is skewed with many fewer POI's than non-POI's, so the purpose of reshuffling the testing and training sets is to retain as closely as possible, the same perecentage of POI's to non-POI's as in the original dataset.

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"]

Accuracy rate measures the number of correctly labeled observation by the total number of observations. However, if you have a dataset that is skewed by having a small number of a particular class. This dataset is skewed so using Precision and recall are a better measurement of how often your algorithm is returning a true positives and low false positive and low false negatives.

Precision = true positive / (true positive + false positive)
Recall = true positive / (true positive + false negative)

Precision is measuring how often, when a POI is flagged, that is indeed a POI. Recall measures when a POI is run through the algorithm it will be flagged and not missed. It is better to have false positives (POI flagged is non-POI) then to have missed flagging a POI. We can rule out a non-POI through additional investigation. It is much harder to discover that a POI was missed, which means someone who is potentially guilty of fraud would go uninvestigated.

References

- scikit-learn.org (on many topics covered in Udacity Intro to Machine Learning)
- https://rahbaran.github.io/data-analyst-udacity-nanodegree/p5-identify-fraud-from-enron-email/p5_final_rahbaran.html (As an outline since no sample project was provided.)
- https://www.quora.com/How-do-I-properly-use-SelectKBest-GridSearchCV-and-cross-validation-in-the-sklearn-package-together