Udacity Data Analyst Nanodegree
P5: Identify Fraud from Enron Emails

Megan O'Neil
2017-02-16

Goal of Project

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.[1]

The goal of this project is to build a person of interest (POI) identifier based on public email and financial data to identify individuals in the Enron scandal that can be classified as POIs, defined as those who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity. In the dataset, the public financial and email data was combined with a hand generated list in the fraud case of the names of the identified POIs. In this project I will train a machine learning algorithm that is designed to identify POIs based on features of our data.

Dataset

There are 145 individuals included in the dataset, with 21 features, though not all individuals had data for each feature. The features come from either financial data (14 features), emails (6 features), or manually included 'poi' designation for persons of interest (1 feature).

Of 145 individuals in the dataset, 18 were POIs (12.4%), the remaining were non-POIs.

The financial features include:

- salary
- loan_advances
- deferral_payments
- total_payments
- exercised_stock_options
- bonus
- long_term_incentive
- restricted_stock
- restricted_stock_deferred
- total_stock_value
- expenses
- other
- director_fees
- deferred_income

The email features include:

- to_messages
- from_messages
- from_this_person_to_poi
- from_this_person_to_poi
- email_address
- shared_receipt_with_poi

[1] This section of summary is from Udacity Project Overview.

Six of the financial features had no values (NaN) for more than 50% of the individuals in the database. These features are listed with the proportion of NaN values below:

- deferral_payments': 0.74
- 'deferred_income': 0.67
- 'director_fees': 0.89
- 'loan_advances': 0.98
- 'long_term_incentive': 0.55
- 'restricted_stock_deferred': 0.88

While I did not remove these features, none of them were selected as useful features. As discussed in the 'Feature Selection' section below, I ultimately used four features in my POI identifier.

Outliers

There were several outliers in the data that I removed. One clear outlier in the financial data was for 'Total', which summed the benefits received by all listed individuals. I identified this outlier manually by plotting the salary and bonus data and finding a clear outlier, identifying that it was the 'Total' rather than an individual, then removing it.

Another outlier was all data for the individual, 'LOCKHART EUGENE E', for whom the value of all features was 0. I identified this outlier by employing an automatic process to run through all of the features for each individual and identifying if any individual had 'NaN' for every feature. One individual was identified and removed.

Feature Selection

I ended up using the following four features in my POI identifier, with feature scores of each following:

- bonus: 30.08
- total_stock_value: 15.96
- salary: 19.03
- fraction_to_poi: 15.60

I used 'SelectKBest' to select these features out of all of the financial and email features provided, excluding 'email_address'. I used GridSearchCV to try the estimator with multiple parameter values for K, from 1 to 14, in order to find the optimal number of features to include. The other parameter I passed for SelectKBest was the socre function 'f_classif', which computes the ANOVA F-value for the provided sample.

I used the linear scaling function, 'MinMaxScaler' in my pipeline operator. I used feature scaling because I used the Gaussian Naïve Bayes algorithm, which would be affected by features that have much larger numeric values. Since SelectKBest uses ANOVA to gauge the potential of efficacy that each feature will have during the model-fitting stage, linear changes to feature values from MinMaxScaler do not affect the ANOVA F-statistics computed by SelectKBest.

New Features

I included three new features in the dataset, 'fraction_to_poi', 'fraction_from_poi', and 'avg_fraction_to_from_poi'. These features were based on the fraction of emails sent between the individual and POIs; with the average fraction being a simple average of the fraction of emails to POIs and the fraction of emails from POIs.

I decided to include these features because I thought that if an individual has high correspondence with POIs, he or she may be more likely to be a POI him or herself. However, the number of emails alone to or from a POI could be misleading as an indicator of an individual's relationship with another POI. If someone sends or receives a higher than average number of emails, it is likely that by extension they will have a higher than average number of emails to or from a POI.

By using the ratio of emails to or from a POI, we can control for the email volume factor and tease out specifically what fraction of total emails an individual is sending to or receiving from a POI. I created the third feature, 'avg_fraction_to_from_poi', so that I could have a single feature that served as a proxy an individual's level of communication with POIs.

The new feature, 'fraction_to_poi' representing the fraction of emails to POIs was selected as a feature using 'SelectKBest' and included in the POI identifier. This indicates that the fraction of emails an individual sends to POIs is an indicator for if the individual is a POI, and that this feature helped the model.

Algorithm Selection and Performance

I used the algorithm Gaussian Naïve Bayes. I also tried SVC, but model performance was lower, holding all other factors constant. I used the same approach of applying the MinMaxScaler and SelectKBest, using the GridSearchCV to optimize K from a range of values from 1-14. Using SVC, only 'bonus' was selected. While precision and recall were high for the non-POI class (0.93 and 0.96 respectively), they were 0 for the POI class.

Algorithm Tuning

Machine learning algorithms have parameters so that they can be tuned for a specific problem. Finding the best combination of parameters to optimize an algorithm is tuning. If not done well, the algorithm may not generate the best classifer for the data.

Gaussian Naïve Bayes has limited parameter tuning capability, so I did not tune parameters. I did use GridSearchCV to tune the parameters of SelectKBest.

With SVC, there would be multiple parameters to tune:

- Kernel: enables decision boundary to be linear or non-linear.
- Gamma: which defines how much influence a single training example has, the larger the gamma, the closer other examples must be to be affected.
- C: controls tradeoff between a smooth decision boundary and classifying training points correctly; low C makes decision surface smooth, high c aims at classifying all training examples correctly.

I would include multiple options for each parameter, then used GridSearchCV() to optimize them.

Validation

Validation is used to give an estimate of the performance of your estimator on an independent dataset, this is when the estimator that you have trained on the training portion of your dataset is applied to the 'test' portion of your dataset to determine how accurate the predictor is. A classic mistake is to validate your estimator on the entire dataset, or on the same data that was used to train the estimator. This will give inflated accuracy results, because you are using the exact same data that was used to train the estimator, so it should be very accurate.

I validated my results by splitting my data into training and testing datasets using sklearn cross_validation.train_test_split. The test size was 20% of the dataset. I also used StratifiedShuffleSplit to create an iterator for cross-validation in the GridSearchCV parameter optimization step.

Ultimately, I used my classifier to predict the class of the test data (pred = clf.predict(features_test).

Evaluation Metrics

My final algorithm returned the following Classification Performance Report:

Class	Precision	Recall	F1-score	Support
0.0	0.96	0.96	0.96	27
1.0	0.50	0.50	0.50	2
Avg/total	0.93	0.93	0.93	29

Where:

- Class refers to POI (0.0) or non-POI (1.0)
- Precision is equal to the ratio of true positives to all predicted positives
- Recall is equal to the ratio of true positives to all actual positives
- f1-score measures the accuracy using precision and recall, weighting them equally
- Support refers to the number of samples of the true response that lie in that class

For non-POIs, precision, recall and f1-score were all 0.96 which means that there is a 96% chance that the algorithm was correct when it predicted someone was not a POI, and a 96% chance that it would detect a non-POI. For POIs, precision, recall and f1-score were all 0.5, which means that there was a 50% chance that the algorithm was correct when it predicted someone was a person of interest, and 50% change it would detect a person of interest.

References

I used the following references in developing this project:

- Stratified Shuffle Split: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
- GridSearchCV: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- Classification Report: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html,
- http://stats.stackexchange.com/questions/117654/what-does-the-numbers-in-the-classification-report-of-sklearn-mean
- f_classif: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif
- f1-score: https://www.kaggle.com/wiki/MeanFScore
- cross-validating: http://nbviewer.jupyter.org/github/jming/cs109/blob/master/lec_10_cross_val.ipynb

- Udacity forum posts to develop code:
  - https://discussions.udacity.com/t/gridsearchcv-and-kfold-validation/32459/6
  - https://discussions.udacity.com/t/dont-know-how-to-validate/201521
  - https://discussions.udacity.com/t/feature-selection-vs-pca/34958
  - https://discussions.udacity.com/t/valueerror-the-least-populated-class-in-y-has-only-1-member-which-is-too-few/44984
  - https://discussions.udacity.com/t/final-project-order-of-pipeline-operation-with-minmaxscaler-and-svc/164802/2

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.