Enron fraud: development of a persons of interest identifier using financial information

1. Study context and objectives

1.1. Background

The American energy, commodities, and services company Enron which was one of the largest companies in the United States up to two years before, collapsed in 2002 into bankruptcy due to widespread corporate fraud. Although a huge one, the Enron fraud is one of the numerous (relatively) recent scandals which are bringing to light corporate malfeasance of various types, questioning the validity of the current watchdogs in place to prevent these immoral actions in the first place. The Enron email and financial datasets were made publicly available. The present project explores a collection of email and finance information preprocessed by Udacity into a single dataset including:

- 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 number of emails messages; with the exception of ‘email_address’, which is a text string)
- and POI label: [‘poi’] (boolean, represented as integer). The information used to create this variable was from a hand-generated list of persons of interest (POI) in the fraud case including individuals who were either indicted, reached a settlement or plea deal with US government, or testified in exchange for prosecution immunity.

1.2. Objectives

In this project, our main objective is to build a person of interest identifier using machine learning techniques on financial and email data made public during the Enron scandal investigations.

Features will be explored. Outliers will be removed during the data wrangling, and then, we will create new features based on our understanding of the problem at hand. Afterwards, we will try various classifiers with and without automated selection of (*i*) the best parameters for the classifiers and (*ii*) features.

2. Features

2.1. Feature selection, missing values and outliers’ removal

The total number of features available for each person was 21. The number of persons in the dataset was 146, with only 18 POI.

Missing values were removed. For most features, data were available for only 12-14 POI, with the exception of 3 features where all 18 were available: expenses, total stock value, and total payments.

Data exploration revealed/confirmed the presence of boolean (poi) information, and emails addresses as strings. An outlier was also detected and removed (‘TOTAL’ that included the total of data, probably brought in the dataset mistakenly).

Another awkward name found in the list of names of persons ('THE TRAVEL AGENCY IN THE PARK') was removed, as well as the person 'LOCKHART EUGENE E' for whom no data was available.

In addition, three features were removed because they had too many missing values and no or too few POI:

- loan_advances (142 missing, 1 POI and 3 non-POI remaining);
- restricted_stock_deferred (128 missing, 0 POI remaining);
- and director_fees (129 missing, 0 POI remaining)

Detailed information on data available after munging was stored in dataExplorationResults.txt.

POI were expected to have values slightly different than the majority of people in the dataset. The need to conserve as many POI as possible for each feature led to plotting data before deciding whether to run outliers’ removal algorithms on data (to prevent data loss associated with unnecessary filtering) and to run outliers’ removal on individual features.

2.2. New features

The following composite features were created from outliers-free data as relative numbers of emails received and sent :

- the ratio of emails from POI to a person (from_poi_to_person_value) to the total number of emails received by the person (from_person_to_poi_value);
- the ratio of emails from a person to POI (from_person_to_poi) to the total number of emails sent by the person (from_messages).

These features were created to assess whether the relative numbers of emails received from and sent to POI by a person can be of use to detect POI. Both features had all 18 POI over the 77 persons available (see dataExplorationNewFeatures.txt).

3. Train/test splitting and classifiers

Data were splitted in randomized train and test datasets (using train_test_split) for cross-validation to reduce the overfitting (and keep the appropriate bias to be able to detect accurately the label looked for in a comparable dataset). Then, StratifiedShuffleSplit method was used for the validation of the classifier, i.e. to make sure that the scores obtained using the classifiers were good enough.

3.1. Splitting and intuitive tuning

Train/test splitting was made automatically using SKlearn cross validation module (StratifiedShuffleSplit), with test size for train/test splitting set to 0.3. We obtained 7 POI and 43 non-POI obtained in test set, with a ramdom value set at 42% after testing with decision tree classifier. Six classifiers were tested using all features and default (and manually optimized parameters): decision tree, support vector machine (SVM), naive Bayes, adaBoost, random forest, and k nearest neighbors considering their precision and recall (See manualTuning.text). We obtained optimum accuracy for most of the classifiers, but not the latter parameters. The best and only interesting results in term of recall and precision were obtained with decision tree classifier using default settings (Accuracy: 0.79220, Precision: 0.24345, Recall: 0.26500, F1: 0.25377, F2: 0.26039).

3.2. Classifier tuning

3.2.1. Best parameter selection

Parameter tuning is important to find the best parameters to obtain the best score possible in less time. The best parameters for the decision tree classifier were selected using GridSearchCV (from sklearn.grid_search). GridSearchCV tries different combinationo of parameters up to where the response is max (we use the f1 score as metric for parameter selection).

3.2.2. Feature selection

The best features were selected using either K-best or Principal components analysis (PCA) selectors in pipelines with the decision tree classifier. K-best selection of the features with the best SelectKBest score improved the scores to a precision and recall higher than 0.3 (Accuracy: 0.81680, Precision: 0.3278, Recall: 0.35600, F1: 0.34132, F2: 0.34998) unlike PCA selection (Accuracy: 0.80260, Precision: 0.26821, Recall: 0.27800, F1: 0.27302, F2: 0.27599). For details on classifier parameters see pca_vs_kbest.txt. Data preprocessing with min-max scaling improved the score of the best features and of the classifier (Accuracy: 0.83107, Precision: 0.37225, Recall: 0.38900, F1: 0.38044, F2: 0.38553).

3.2.3. Best features

Features with SelectKBest scores of at least 30 after scaling were selected. These 3 features were :

- 'exercised_stock_options' (SelectKBest score before scaling 18.06, after scaling 33.46)
- 'total_stock_value' (SelectKBest score before scaling 17.32, after scaling 32.28)
- 'bonus' (SelectKBest score before scaling 23.82, after scaling 30.09)

SelectKBest scores of the other features after scaling were (for scores before scaling see score_labels_kbest.txt):

- 20.159242339717046, 'salary';
- 16.680570532394171, 'long_term_incentive';
- 11.48546392067748, 'total_payments';
- 11.027816980321198, 'restricted_stock';
- 7.5761647353819068, 'from_poi_to_this_person';
- 5.6046411985192108, 'shared_receipt_with_poi';
- 4.357891542573328, 'other';
- 4.227541562441135, 'deferred_income';
- 3.7495636059001312, 'from_this_person_to_poi';
- 2.6873736283436087, 'expenses';
- 1.260569265962652, 'index';
- 0.95195888036223275, 'to_messages';
- 0.86433480241821226, 'deferral_payments';
- 0.62498403302111638, 'ratio_to_poi_to_all_sent_emails';
- 0.56674548581874329, 'ratio_from_poi_to_all_from_emails';
- 0.015396577650012906, 'from_messages'.

As shown above, unfortunately, both our new features ('ratio_to_poi_to_all_sent_emails' and 'ratio_from_poi_to_all_from_emails', which are the relative numbers of emails sent to and received from POI by a person, respectively) had a poor SelectKBest score, indicating that they were not good enough to help in the detection of POI.

4. Concluding remarks

The recall is the probability to predict correctly a given label, i.e. out of all the items that are truly positive, how many were correctly classified as positive, or simply, how many positive items were 'recalled' from the dataset. While the precision is the probability to be right when the algorithm selects a given label as true. Thus, out of all the items labeled as positive, how many truly belong to the positive class.

Testing the precision and recall as metrics for assessing the performances of various classifiers confirmed the existence of a tradeoff between precision and recall. Despite the small size of the dataset and the low number of POI, we managed to improve a decision tree classifier up to a precision and recall higher than 0.3. Such improvement was achieved via data normalization (using min_max scaler), best feature selection (using K-best method), and tuning of the decision tree classifier (determining the best parameters using GridSearchCV).