Identifying Persons of Interest from the Enron Corpus

Nanodegree Udacity Project

(Machine learning techniques)

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 purpose of this project is to explore some of the tools and techniques of machine learning. Specifically, we use data from the Enron financial information and email dataset to identify 'persons of interest (Poi's)' - individuals whose pattern of remuneration and email activity suggest that they may have been involved in illegal activities and therefore warrant closer investigation. In a data set having several data points, spotting patterns that are indicative of potentially illegal behavior wouldn't be possible without the help of machine learning algorithm. This report documents the machine learning techniques used in building the POI identifier.

The Enron Dataset has 146 records. Of these 18 are labeled as Poi and the rest are labeled as non Poi. There are 21 possible features for each person. More than 85% missing are observed for the financial part of the fields like loan_advances (142), director_fees (129) and restricted_stock_deferred (128). 3 people had no values for the finanical features (Ronnie Chan, William Powers, and Eugene E. Lockhart, none of whom are Persons of Interest).

The above mentioned three people with no financial information are removed from the data set. Moreover, up on examining the Enroninsiderpay.pdf file the "Total" and the "THE TRAVEL AGENCY IN THE PARK", for which it is an agency and not one of the Enron Executives, are excluded from the data set.

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"]

I created three features namely, fraction_shared_receipt_with_poi, fraction_from_poi, and fraction_to_poi. The reason for creating these features was that POIs among themselves would be having a strong e-mail connection among each other and the fraction will give the percentage of the total e-mails received, sent or shared between the person and the POI. This could imply the biggest the fraction, the greatest the probability that the person has strong connection with the POI. Therefore, before doing algorithmic feature selection, I decided to take the new features and drop the original features that I originated my new features from (shared_receipt_with_poi, from_poi_to_this_person, and from_this_person_to_poi). [Since these three features are created based on the assumption that we know who the persons of interest are, inclusion of these features might cause a leakage of some knowledge available in the test data into the training data resulting to model over training. I learned later care should be taken with large datasets and high proportion of poi's].

To select features, I used the "Stability Selection" which applies a feature selection algorithm on different subsets of data and with different subsets of features. After repeating the process a number of times, the selection results can be aggregated, for example by checking how many times a feature ended up being selected as important when it was in an inspected feature subset. The sklearn randomized lasso algorithm attaches the following scores (%) for all features:

salary : 0.52

deferral_payments : 0.935

total_payments : 0.055

loan_advances : 0.11

bonus : 0.865

restricted_stock_deferred : 0.01

deferred_income : 0.93

total_stock_value : 0.315

expenses : 0.87

exercised_stock_options : 0.625

other : 0.315

long_term_incentive : 0.28

restricted_stock : 0.195

director_fees : 0.015

fraction_shared_receipt_with_poi : 0.215

fraction_from_poi : 0.33

fraction_to_poi : 1.0

I haven't done any scaling as scaling is not required for tree or tree-based ensembles. I tried three different cut off points (greater than 0.5, 0.3 and 0.24) and two algorithms to select my features. The 0.3 cut off point gave me the highest accuracy in both algorithms. Therefore, I ended up selecting the following 11 features.

- 'poi'
- 'salary'
- 'deferral_payments'
- 'bonus'
- 'deferred_income'
- 'expenses'
- 'total_stock_value'
- 'exercised_stock_options'
- 'other'
- 'fraction_from_poi'
- 'fraction_to_poi'

Stability Selection

Cut off points	No. of Features	Decision Tree	Random Forest
0.50	8	Accuracy: 0.81 Precision: 0.33 Recall: 0.33	Accuracy: 0.86 Precision: 0.50 Recall: 0.17
0.30	11	Accuracy: 0.80 Precision: 0.29 Recall: 0.28	Accuracy: 0.86 Precision: 0.50 Recall: 0.18
0.24	12	Accuracy: 0.79 Precision: 0.28 Recall: 0.28	Accuracy: 0.85 Precision: 0.48 Recall: 0.17

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"]

By just taking the default parameters, I tried Random Forest , DecisionTree , AdaBoost , Naïve Bayes, and KNeighbor classifiers (I tried SVM too, but my PC got frozen). As shown in the table below, AdaBoost achieved the best result according to each metric, but took almost 5 times longer duration.

Model	Time	F1	Precision	Recall
Random Forest	55.685s	0.27451	0.50133	0.18900
Decision Tree	2.42s	0.28607	0.28818	0.28400
AdaBoost	259.081s	0.47600	0.54255	0.42400
Naïve Bayes	2.28s	0.31550	0.41363	0.25500
K Nearest Neighbor	2.598s	0.33872	0.68024	0.22550

I tried different combination of tuning parameters using GridSearchCV for Random Forest, k-Nearest Neighbors and AdaBoost so as to find out which classifiers achieve better result for the POI identification. Below are the outcomes of few combinations of RandomForest Classifier out of the many I tried:

max_depth=4 min_samples_leaf=3 min_samples_split=2 n_estimators=10	Accuracy: 0.86857 Precision: 0.61765 Recall: 0.21000
max_depth=4 min_samples_leaf=1 min_samples_split=2 n_estimators=10	Accuracy: 0.86143 Precision: 0.53191 Recall: 0.25000
max_depth=4 min_samples_leaf=2 min_samples_split=4 n_estimators=20	Accuracy: 0.86357 Precision: 0.57377 Recall: 0.17500
max_depth=4 min_samples_leaf=1 min_samples_split=3 n_estimators=20	Accuracy: 0.86714 Precision: 0.58140 Recall: 0.25000

Since, for cross-validation we are using StratifiedSuffleSplit and the results for the three algorithms vary slightly for each run. I took the maximum readings of F1 score for each classifier.

The parameters that yield the best F1 score for RandomForest Classifier are : (the F1 score is a combination of the two metrics,

F1 = 2 * (precision * recall) / (precision + recall).

max_depth=4 min_samples_leaf=1 min_samples_split=3 n_estimators=20	Accuracy: 86.8% Precision: 61.1% Recall: 22% F1: 32.3%

The parameters that yield the best F1 score for k-Nearest Neighbors Classifier are:

leaf_size=30 n_neighbors=5	Accuracy: 87.4% Precision: 68% Recall: 22.5% F1: 33.8%

Likewise, the parameters that yields the best F1 score for AdaBoost Classifier are:

min_samples_leaf=1 min_samples_split=2 n_estimators=100	Accuracy: 88.4% Precision: 62..8% Recall: 46.5% F1: 53.4%

My final model was built using the AdaBoost algorithm. This algorithm uses a group of weak decision trees. Each of the trees splits the data based only on a single variable (assuming max_depth=1). At classification, each of these trees 'votes' for its preferred classification and the decision is based on a weighted total of the votes. AdaBoost also weights the importance of classifying a given point correctly during iterations of the algorithm so that the points that early iterations do a poor job classifying are considered more important in later iterations.

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 the process of making adjustments when trailing the classifier to improve the fit on the test data. If not done well, poor predictions can happen as a result of overfitting or underfitting. Holding a subset of the data to test the predictions against is essential to avoid such problems.

For adaboost, the main parameters for tuning are the number of weak estimators and

their complexity. In this case the estimators are decision trees, so I tune the number of them and their maximum depth. Adaboost achieved an F1 score of 0.53448.

For random forest I tuned the number of trees built and the number of parameters evaluated at each node. Random Forest improved from F1 score of 0.26820 to 0.32353.

F1 score for k-Nearest Neighbors showed a very slight difference only after the n_neighbor parameter is tuned. The best achieved F1 score was 0.33964

Adaboost achieved the best score compared to Random Forest and k-Nearest Neighbors.

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"]

The aim of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (unknown dataset). One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (the training set), and validating the analysis on the other subset (validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

One of the main reasons for using cross-validation instead of using the conventional validation (e.g. partitioning the data set into two sets of 70% for training and 30% for test) is that there is not enough data available to partition it into separate training and test sets without losing significant modeling or testing capability. In these cases, a fair way to properly estimate model prediction performance is to use cross-validation as a powerful general technique.

Since our dataset is small in size, we used StratifiedShuffleSplit cross validation technique in order to hold a test set for final evaluation. StratifiedShuffleSplit is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.

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 AdaBoost model had accuracy = 0.88429, precision = 0.62838, recall = 0.46500, and F1 = 0.53448.

Accuracy gives the percentage of correctly classified persons that is 88.4% of the persons in the dataset are correctly classified as either poi or non-poi. However, we can't rely on this result as it can be misleading since the number of POIs is much less than

non POIs in our dataset. The most important metrics to consider are Precision and Recall.

Precision determines the probability that a person is identified as POI by the classifier he/she truly is POI. This means that out of 100 people the model predicted as POIs, 62% of them are correctly classified as POIs. Whereas a 46% recall indicates that out of a total of 100 POIs, this model identifies 46 of them correctly as POIs. This means that given a Person of Interest, the classifier will recognize it most of the time 46%. For our project the most important metric is Recall because we are interested to find out how accurately our model identified actual POIs in the dataset (persons of interest who have actually involved in the fraud of Enron scandal).

References:

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Stability Selection http://stat.ethz.ch/~nicolai/stability.pdf

https://www.youtube.com/watch?v=Gol_qOgRqfA

http://blog.datadive.net/selecting-good-features-part-iii-random-forests/

http://blog.datadive.net/7-tools-in-every-data-scientists-toolbox/

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation