Intro to Machine Learning Project report

1. Introduction

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, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives. In this project, we will play detective, and put our new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal.

2. Enron Dataset exploration

As raw input data, we have a hand-generated list of persons of interest in the fraud case, which means individuals who were indicted, reached a settlement, or plea deal with the government, or testified in exchange for prosecution immunity.

Total number of data points are 146, allocation across classes (POI/non_POI) is about 12%, number of available features are 19. The features in the data fall into three major types, namely financial features, email features and POI labels. 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', '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)

There are samples with many missing features. Missing features is set as 0. If a sample’s features are all zeros, it is removed.

There are two kinds of outliers in the dataset. One is entry error, the “Total” sample is an excel importing mistake so this outlier is removed. There are also other outliers, but they are samples which we interested in and are retained. Outliers are detected by visualization, pandas description over the dataframe, or outlier detector like covariance.EllipticEnvelope.

3. Optimize Feature Selection/Engineering

I end up with using below features in my POI identifier:
'exercised_stock_options', 'fraction_to_poi', 'from_poi_to_this_person'
The selection process I used is forward feature selection. forward feature selection works as below

1. Enumerate all available features, select one at a time and use them to learn and predict, finally pick the feature with best score.
2. From the remaining features, enumerate them , select one at a time, combine it with feature(s) already selected, use this new feature combination to learn and predict
3. Pick the feature combination with best scores
4. Repeat step 2 until an optimal feature combination is found

As SVM is used, features are scaled before being plugging into the learning algorithm.
New feature “fraction of emails from/to POI” is added. This is based on the intuition that if a person has a lot of email exchange with a POI, he himself might as well likely be an POI.

As an experiment, both SelectKBest and Decision tree are used as preliminary steps which helps assess relative importance of the available features, and the result is as below:
decision tree importance rank: ['exercised_stock_options' 'other' 'fraction_to_poi' 'expenses'
'shared_receipt_with_poi' 'restricted_stock' 'long_term_incentive' 'bonus'
'total_payments' 'deferral_payments' 'deferred_income' 'fraction_from_poi'
'restricted_stock_deferred' 'from_messages' 'loan_advances' 'salary'
'director_fees' 'total_stock_value' 'from_poi_to_this_person'
'from_this_person_to_poi' 'to_messages']

anova f test importance rank: ['exercised_stock_options' 'total_stock_value' 'bonus' 'salary'
'fraction_to_poi' 'deferred_income' 'long_term_incentive'
'restricted_stock' 'total_payments' 'shared_receipt_with_poi'
'loan_advances' 'expenses' 'from_poi_to_this_person' 'other'
'fraction_from_poi' 'from_this_person_to_poi' 'director_fees'
'to_messages' 'deferral_payments' 'from_messages'
'restricted_stock_deferred']

4. Pick and tune Algorithm

The algorithm I end up using in the POI identifier is SVM. I also tried decision tree. Generally speaking, SVM has slightly better scores, and Decision tree is much faster.

Cross validation techniques are used to fine tune parameters for algorithms. Parameters tuning makes significant difference to the final result and is very important. Specifically, sklearn.grid_search.GridSearchCV is used to search for optimal hyper parameters.

5. Validate and Evaluate

Validation is about evaluating different settings (hyper parameters) for estimators, such as C setting that must be manually set for an SVM. A classic mistake would be to tweak these hyper parameters on test set. This way, knowledge about the test set can “leak” into the model and evaluation metrics no longer report on generalization performance.

One approach to this problem is to set aside a “validation” set just for hyper parameters tuning. In our case, the overall data we have is very limited (<145 valid samples). As a result, the validation strategy we used is cross validation. Considering the fact the label distribution is imbalanced, Stratified k-fold cross validation is used.

The evaluation metrics used is F1, precision and recall. F1, as a single metrics, is used to pick best features/algorithms during cross validation. Precision and recall statistics is also provided to allow for more insight about the evaluation result. In our case, precision refers to the correctly labeled POI samples versus all the predicted POI, and recall refers to the correctly labeled POI samples versus all true POI.

6. Final Result

Final algorithm: SVM ({'kernel': 'rbf', 'C': 45, 'gamma': 120})
Final selected features: 'exercised_stock_options', 'fraction_to_poi', 'from_poi_to_this_person']
Final performance:
F1:0.4718958399491902
Precision: 0.6466492602262838
Recall 0.3715
All intermediate result are in experiment_result folder