Intro to Machine Learning

Project: Identify Fraud from Enron Email

Marvin Lüthe

Data Analyst Nanodegree 2018/2019

Table of Contents

1 Introduction

1.1 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.

The goal for this project is to create a machine learning software which identifies former Enron employees who may have committed fraud based on the public Enron financial and email dataset. Machine Learning helps us identify patterns in the data, select the most powerful pieces of information (features) and pick an appropriate classification algorithm which we can use to predict the Persons of Interest (POIs).

Enron Dataset:

- Total number of people: 145
- Allocation across classes: $\frac{POI}{Non\_POI} = \frac{18}{127} = 0.14$
- Number of features used: 4
- Amount of 'NaN' values:
    - 'salary': 51
    - 'total_payments': 124
    - 'bonus': 64
    - 'exercised_stock_options': 44
    - 'total_stock_value': 20

1.2 Were there any outliers in the data when you got it, and how did you handle those?

I have identified one significant outlier and deleted it from the dataset. It was the datapoint 'TOTAL' containing the cumulated values for the whole Enron dataset.

2 Features

2.1 What features did you end up using in your POI identifier, and what selection process did you use to pick them?

I used the SelectKBest Method and selected the feature *bonus* since it best represents the pattern in the data.

2.2 Did you have to do any scaling? Why or why not?

I did not have to do any rescaling for the algorithms because I focused on the following algorithms: decision tree, Naïve Bayes, ensembles (Adaboost and RandomForrest). None of them require feature scaling.

However, I used Principal Component Analysis (PCA) to engineer a new feature and therefore, I used StandardScaler to scale the features for PCA.

2.3 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.)

I used PCA to engineer my own feature based on all features. The idea to use PCA was to reduce dimensionality by creating new combinations of attributes. I wanted to determine if only a few PCA features could explain most of the patterns in the data and then take the feature which explains most of the variance in the data (which is basically the first element of the PCA transformed features array). This *PCA_feature* explains 33.8% of the variance in the data. If I took this feature, the performance of the algorithm (tuned decision tree classifier) would decrease:

Tuned decision tree classifier:

Precision: 0.08312

Recall: 0.04950

Due to these bad performance indications, we do not consider the *PCA_feature* as an important feature. The table below in 2.4 shows that the *PCA_feature* has been taken into account in the feature selection process but does not have an above average SelectKBest Score (*PCA_feature* score = 1.46). This is why *PCA_feature* was not selected for the final algorithm.

2.4 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.

During the feature selection phase we have to ensure the right balance between bias (few features) and variance (many features). For the Enron dataset we have to specifically draw attention to the problem of overfitting the algorithm because we only have a very limited number of data points. Moreover, the decision tree algorithm is classically an algorithm that is prone to overfitting. I have selected one feature: bonus. This table shows the SelectKBest scores:

Feature	Selector score
salary	11.78
bonus	23.82
deferral_payments	0.04
deferred_income	7.51
director_fees	1.70
exercised_stock_options	18.06
expenses	3.02
from_messages	0.44
from_poi_to_this_person	4.28
from_this_person_to_poi	0.10
loan_advances	6.50
long_term_incentive	5.87
other	3.80
restricted_stock	6.78
restricted_stock_deferred	0.75
shared_receipt_with_poi	7.99
to_messages	3.42
total_payments	8.09
total_stock_value	17.32
PCA_feature	1.46

The following table shows how the performance of the tuned decision tree classifier (final algorithm) differs for varying *k*-values (SelectKBest parameter). We can see that both precision and recall values decrease with increasing amounts of features. This is why I only selected one feature for the algorithm.

k-value	Features	Precision	Recall
1	bonus	0.55591	0.35050
2	bonus, exercised_stock_options	0.39750	0.36550
3	bonus, exercised_stock_options, total_stock_value	0.38430	0.32300
4	bonus, exercised_stock_options, total_stock_value, salary	0.37026	0.31250
5	bonus, exercised_stock_options, total_stock_value, salary, total_payments	0.31876	0.27700
10	salary, bonus, deferred_income, exercised_stock_options, loan_advances, long_term_incentive, restricted_stock, shared_receipt_with_poi, total_payments, total_stock_value	0.27394	0.23600
15	poi, salary, bonus, deferred_income, director_fees, exercised_stock_options, expenses, from_poi_to_this_person, loan_advances, long_term_incentive, other, restricted_stock, shared_receipt_with_poi, to_messages, total_payments, total_stock_value	0.23335	0.20150
20 (all)	poi, salary, bonus, deferral_payments, deferred_income, director_fees, exercised_stock_options, expenses, from_messages, from_poi_to_this_person, from_this_person_to_poi, loan_advances, long_term_incentive, other, restricted_stock, restricted_stock_deferred, shared_receipt_with_poi, to_messages, total_payments, total_stock_value, PCA_feature	0.22886	0.19350

3 Algorithms

3.1 What algorithm did you end up using?

Finally, I used the decision tree algorithm.

3.2 What other one(s) did you try?

I tried Gaussian Naïve Bayes, Adaboost and the Random Forest Algorithm.

3.3 How did model performance differ between algorithms?

Algorithm	Precision (untuned)	Recall (untuned)	Precision (tuned)	Recall (tuned)
GaussianNB	0.57980	0.17800	Not applicable	Not applicable
Decision Tree	0.42869	0.26450	0.55591 (final algorithm)	0.35050 (final algorithm)
Adaboost	0.44627	0.28450	0.33929	0.08550
Random Forest	0.42267	0.29650	0.53704	0.30450

4 Evaluation

4.1 What does it mean to tune the parameters of an algorithm, and what can happen if you do not do this well?

Parameter tuning means that we provide the algorithm with combinations of different parameters and choose those parameters where the algorithm best predicts the outcome. If parameter tuning, also called **hyperparameter optimization,** is not done well, the performance of the algorithm will decrease. The performance of the model will decrease because in Machine Learning we have to make sure that we set good parameters which are appropriate both to the learning model and the data pattern.

The same kind of learning model can require very different input parameters to generalize different data patterns. The parameter tuning process identifies a tuple of parameters which enables the best learning experience for the selected model and therefore, increases the quality of the resulting predictions / clustering.

4.2 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).

I used GridSearchCV to tune the parameters of the algorithms that I tried out (except for GaussianNB). For the decision tree classifier I tuned the parameters max_features and min_samples_split.

4.3 What is validation, and what is a classic mistake you can make if you do it wrong?

Validation means to test the trained algorithm and check the performance. A common mistake is to not separate between training and testing data.

4.4 How did you validate your analysis?

I split the dataset into training (70%) and testing (30%) data and checked the performance of my trained algorithm with the *tester.py* module. The *test_classifier* function uses a cross-validation method called *StratifiedShuffleSplit.* The basic idea of cross validation is the k-fold cross validation which separates the dataset in k bins. In k-fold cross validation we run k separate learning experiments. *StratifiedShuffleSplit* is a cross-validation method and is derived from *ShuffleSplit*. *ShuffleSplit* is an iterator which generates a user defined amount of independent train and test dataset splits. *StratifiedShuffleSplit* means that these folds preserve the same percentage of samples for each class.

This validation method is used since we do not have many POIs in the Enron dataset (only 18). The assessment of the POI identifier becomes more accurate since we average many experiments (= 1000) and use all the data for training and testing.

4.5 Give at least 2 evaluation metrics and your average performance for each of them.

Please refer to performance table in 3.3.

4.6 Explain an interpretation of your metrics that says something human-understandable about your algorithms performance.

The decision tree algorithm has a precision of 55.59% and a recall value of 35.05%. The precision indicator means that out of all the people that were identified as POIs, 55.59% truly were POIs (and may have committed fraud). On the other hand, 35.05% of all the POIs could be detected by the fraud software. I personally would argue that the precision value is more valuable in this case as it is more important, from my point of view, to avoid putting somebody into jail who is innocent of a crime. This means that I will miss more real POIs as I am reluctant to pull the trigger on edge cases.

Bibliography – Index of Literature and Sources

http://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html

http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

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

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

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

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

https://en.wikipedia.org/wiki/Hyperparameter_optimization