UDACITY DATA ANALYST NANODEGREE

P5: Identify Fraud From Enron Email

Prepared by: Sheena Yu, Data Analyst

January 29, 2017

INTRO TO MACHINE LEARNING

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, 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.

Goals

In this project, I will put my machine learning 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. This data has been combined with 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.

Solution

I used scikit-learn to address this supervised classification problem. Scikit-learn is a free software machine learning library for the Python programming language. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

Project Outline

I will walk you through my project seamlessly by answering the following questions with reasonable logic.

- **Question 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. Were there any outliers in the data when you got it, and how did you handle those?

The goal of this project is to apply an appropriate supervised machine learning algorithm to predict whether an insider is a person of interest(POI) based on their available financial and email dataset that was made public after Enron bankruptcy. Machine learning is quite useful in such kind of situation because it gives us the power to more accurately classify data points into groups we are interested in. Although this is a supervised learning case, meaning we already know the results (who is POI of Enron scandal), the model built could be applied to other similar company bankruptcy scenarios.

Data Overview

After initial rough data investigation, I find that there are 146 insiders (data points) in the dataset, each of which contains the respective financial and email data. Out of 21 features related to each data point, 14 are financial features and 6 are email features, with one remaining being the label ("poi"—boolean format). There are 18 persons of interest in the dataset, and 128 innocent persons.

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']

Data Wrangling

Missing values:

I inspect the dataset to check the existence of any missing values. Through exploratory data analysis I identified top 5 persons with the most “NaN” values and the numbers of “NaN” values in each feature.

Top 5 persons with the most missing values: [(‘LOCKHART EUGENE E', 20), ('WHALEY DAVID A', 18), ('WROBEL BRUCE', 18), ('THE TRAVEL AGENCY IN THE PARK', 18), ('GRAMM WENDY L', 18)]

None of the features of Lockhart Eugene has any useful data values, while “The travel agency in the park” is not a valid person and could potentially introduce some bias to my prediction model, thus I choose to cut out these two data points.
“loan_advances”, “director_fees” and “restricted_stock_deferred” contain the least amount of meaningful values, however at this point I was not very sure about the significance of these features, so it would be better to keep them in the dataset.

Outliers:

Next step is to find any outliers that could potentially distort the true picture of the dataset. My method is to draw box plots and visually identify potential outliers. The only outlier I deleted is “Total”, which was accidentally included in the dataset as this is an Excel artifact. The other data points look like outliers, but due to the particularity of the Enron case, these persons may bring important information into the prediction model, so I will also keep them.

Inconsistent data:

I also find out that ‘BELFER ROBERT’ and ‘BHATNAGAR SANJAY’ has inconsistent data values. The miscellaneous payment items do not add up to the “total_payment”, same as “total_stock_value”. So I correct these two data points.

- **Question 2:** 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.

Feature Engineering and Feature Selection

According to Wikipedia, feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. In addition to the existing features, I created another two new features based on my accounting experience. The first one I created is "controllable_gain", meaning the items included in the controllable_gain are more prone to manipulation on the financial statements by executives and management. In a word, the controllable gain is the sum of the following items:
controllable_gain= salary + bonus + loan_advances + other + expenses + director_fees + exercised_stock_options
The other new feature is called "poi_pctg", as I'm more interested in the level of interaction with POIs, which I view as a key metric to accurately classify POIs and non-POIs. The formula is:
poi_pctg = (from_poi_to_this_person + from_this_person_to_poi) / (from_messages + to_messages)

In machine learning and statistics, feature selection is the process of selecting a subset of relevant features for use in model construction. Irrelevant features may harm the robustness and generalization power of model. After doing an extensive research, I choose SelectKBest in the sklearn package to rank the features based on their scores. "SelectKBest" belongs to univariate feature selection and works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. So I take a deep look into the features list and decide to cut out the string feature "email_address" and keep the features with scores greater than 5. Note here that I applied the feature selection on the entire dataset instead of the training dataset, because I would just like to know the features with higher scores and conduct preliminary feature selection for computational efficiency. Later in the phase of tuning classifiers, feature selection combined with other parameters will be conducted again to tailor to each classifier.

Features with scores greater than 5

total_stock_value	22.51
exercised_stock_options	22.35
bonus	20.79
salary	18.29
controllable_gain	17.94
deferred_income	11.42
long_term_incentive	9.92
total_payments	9.28
restricted_stock	8.83
shared_receipt_with_poi	8.59
loan_advances	7.18
expenses	5.42
poi_pctg	5.40
from_poi_to_this_person	5.24

Now the final features list I will feed into my selected supervised classification models consists of 14 features.

As I just mentioned, the scientific way of picking features will come into play when I build the pipeline for each algorithm. According to scikit-learn documentation, the purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. The first step of my pipeline is feature scaling using *MinMaxScaler*. The main advantage of scaling is to avoid features in greater numeric ranges (financial features $) dominating those in smaller numeric ranges (email features). For the applicable models(e.g. KNN, SVM), scaling ensures that all features are weighted evenly. The second step is using *selectKBest* to allow each algorithm to keep the k highest scoring features suitable for them.

- **Question 3:** What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I ended up using **Decision Tree algorithm** in a supervised classification context. I tried several algorithms including k Nearest Neighbors(kNN), Logistic Regression, Random Forest classifier, Support Vector Machine, and Gaussian Naive Bayes classifier.
Since the objective of this project is binary classification (POI/non-POI), and we have an unbalanced and sparse dataset, with only 18 being POI, so using accuracy score as benchmark to compare prediction performance of different algorithms would yield poor decision. F1 score, however, can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0. So F1 score is favored over accuracy when we have an unbalanced dataset. From the table below, I finally choose Decision Tree algorithm as its precision, recall and F1 scores are all above 0.3. It also selected only 5 features, including ***'salary', 'exercised_stock_options', 'bonus', 'total_stock_value', 'controllable_gain'.***

Algorithm	Accuracy	Precision	Recall	F1 Score	Features
KNN	0.86200	0.46377	0.22400	0.30209	3
Logistic Regression	0.86207	0.44812	0.14900	0.22364	3
Decision Tree	0.84900	0.43352	0.43200	0.43267	5
Random Forest	0.86460	0.48707	0.29200	0.36511	4
SVC	0.87607	0.60383	0.20500	0.30608	5
GaussianNB	0.85013	0.41218	0.29100	0.34115	5

I also investigated the effect of engineered features on the predictive performance of my selected algorithm Decision Tree classifier. I removed two engineered features from the above features list and rerun the "evaluate_algorithms" function. Surprisingly, I find that the performance became worse without the engineered features.

	Precision	Recall	F1 score	Features
Decision Tree Algorithm with engineered features	0.43263	0.4335	0.43307	total_stock_value, exercised_stock_options, bonus, salary, controllable_gain
Decision Tree Algorithm without engineered features	0.29688	0.32850	0.31189	total_stock_value, exercised_stock_options, bonus

- **Question 4:** 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?

Tuning the parameters is essentially selecting the best parameters for an algorithm to optimize its performance. If not tuning parameters and just using the default values for those parameters, we are more likely to end up with an unsuitable algorithm for a particular case and the performance will be poor.
I adopted the method of grid search provided by scikit-learn's *GridSearchCV* instance. GridSearchCV exhaustively considers all parameter combinations. It implements the usual estimator API: when "fitting" it on a dataset all the possible combinations of parameter values are evaluated and the best combination is retained.
For the algorithm(Decision Tree) I choose, the parameters I tuned are k best features, decision tree's criterion(function to measure the quality of a split) and min_samples_split(The minimum number of samples required to split an internal node). As a result, the best combination of parameters are selecting 5 features (mentioned in question 3) with decision tree's criterion being entropy and min_samples_split being 8.

- **Question 5:** What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is a process of making sure the machine learning model can generalize well to unseen data. Given a dataset, we need to split it into training and test dataset. After we trained the model on the training dataset and get satisfied evaluation metrics, we can test the generalization power of the model on the independent test dataset. If the performance is much poorer on the test dataset, a common mistake known as "overfitting" could have occurred during the training procedure.
I validated my analysis with the help of sklearn's *StratifiedShuffleSplits* function, which provides train/test indices to split data in train/test sets. This cross-validation object 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. I pass it as an argument of GridSearchCV (cv=sss where sss is a StatifiedShufleSplit).

- **Question 6:** 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.

The main evaluation metrics I used are **Precision**, **Recall and F1 score**.
Precision is true positives divided by true positives plus false positives, that is the ratio of correct positive predictions made out of the total positive predictions. All the algorithms have very good precision greater than 0.3.

Recall is true positives divided by true positives plus false negatives, that is the ratio of correct positive predictions made out of total true positive events. The Decision Tree algorithm had the best recall (43.2%) with 864 true positives and 1136 false negatives.

F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

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

I passed F1 score into GridSearchCV object in order to avoid boosting one metric at the expense of the other. So I choose to use the F1 score to balance them.

In this Enron case, I do not encourage the usage of Accuracy as a evaluation metric. Because we have a quite sparse and unbalanced data, with few individuals being POI. Consequently, we will have very close accuracy scores for all the algorithms.

However, the most important evaluation metric, from my point of view, is recall. The purpose of the predictive classification model is to identify whether or not an insider is a person of interest. Therefore we are more interested in maximizing the recall metric to capture as many persons of interest as possible, who will then be taken to face further human-led investigation. We do not care if he is innocent or not, as we will only need to leave it to judicial investigation.

References

Scikit-learn Documentation

Udacity P5 Discussion Forum

Udacity Intro to Machine Learning

Feature Selection for Machine Learning

Model Selection and Feature Selection