Report

October 18, 2016

0.1 Identifying Fraud from Enron Emails and Financial Data

0.1.1 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 the public record, including tens of thousands of emails and detailed financial data for to executives.

Utilizing the classifiers and techniques taught in Into to Machine Learning Class, I built a classifier to detect if a person is guilty or not.The persons who are guilty are termed POI(Person of Interest) in the context of this problem.

0.1.2 Questions

> 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 identify fraud from email and financial data available about Enron employees. We are trying to identify each individual as person of interest or not. The dataset consisted of the following features: - **email features:** to_messages, email_address, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi - **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

Apart from the above features, I constructed some binary features that captured the information for missing values in the features. Also, I created a new feature that measured the interaction of an individual with the POIs. The decision to create binary features was taken while exploring the data and finding out that a lot of values for the features were missing. The missing value count for each column is displayed below:

Feature	Count
to_messages	60
deferral_payments	107
expenses	51
poi	0
long_term_incentive	80

Feature	Count
email_address	35
from_poi_to_this_person	60
deferred_income	97
restricted_stock_deferred	128
shared_receipt_with_poi	60
loan_advances	142
from_messages	60
other	53
director_fees	129
bonus	64
total_stock_value	20
from_this_person_to_poi	60
restricted_stock	36
salary	51
name	0
total_payments	21
exercised_stock_options	44

The data exploration was performed on a csv file generated from the data-dict supplied for the project. During exploration, the following two outliers were identified:

- **TOTAL** - This was the total of all the people in the dataset.
- **THE TRAVEL AGENCY IN THE PARK** - This does not belong to any of the employees.

There were only 146 records in total and after removal of above two outliers, only 144 remained. Also, there were only 18 POIs and rest were non-POIs. Also, the above two outliers were not POI.

> 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.

As discussed in the previous question, I created binary features for all features available in the data to capture the missing value information for these features. If the value of a feature was missing, the binary feature was populated with 1 and otherwise with 0. Also, a feature measuring the interaction of an individual with poi was created. This is the ratio of the messages to or from poi and total messages sent and received. Below is a list of all the engineered features:

- missing_bonus
- missing_from_messages
- missing_total_stock_value
- missing_exercised_stock_options
- missing_total_payments
- missing_director_fees
- missing_deferral_payments

- missing_to_messages
- missing_loan_advances
- missing_other
- missing_deferred_income
- missing_from_this_person_to_poi
- missing_salary
- missing_from_poi_to_this_person
- missing_restricted_stock_deferred
- missing_restricted_stock
- missing_expenses
- missing_shared_receipt_with_poi
- missing_long_term_incentive
- interaction_with_poi: (from_poi_to_this_person + from_this_person_to_poi) / (from_messages + to_messages)

I am using an L1 penalty based logistic regression as a POI Identifier with features selected based on the random forest classifier. The below features were selected in the final model based on random forest classifier:

Feature
to_messages
from_messages
from_this_person_to_poi
from_poi_to_this_person
shared_receipt_with_poi
salary
deferral_payments
total_payments
exercised_stock_options
bonus
restricted_stock
total_stock_value
expenses
other
deferred_income
long_term_incentive
missing_bonus
missing_exercised_stock_options
missing_to_messages
interaction_with_poi

Standard Scaling was performed for L1 regularized logistic regression. This was done because regularization imposes penalty on the size of the coefficients. If we have features on different scales, the corresponding weights would also be on different scales and hence, the shrinking of the weights would not be uniform.

> What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I tried 3 different algorithms for my POI Identifier. I used decision trees, Random Forest and Logistic Regression. However, I chose Logistic regression with L1 penalty with feature selection from the random forest classifier for the final model. The choice for Logistic regression came after validating the model with 10-fold cross validation and tuning the hyperparameters which resulted in a good enough precision and recall for the positive class (POI in this case). The decision tree had a high variance among the results. So, it was dropped and random forest was tested. However, random forest also did not give the desired precision and recall and hence, was discarded.

> 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?

It is very important to tune the parameters of an algorithm because tuning the parameters helps the algorithm to perform well and generalize well. In order to tune the hyperparameters for all the three models, I used the following approach: - I performed a 80-20 train-test split and kept the testing set aside. - I performed 5 fold cross validation for Decision Tree Classifier while searching for the best parameters using Grid Search. After the best model was identified based on 5-fold cross validation, I tested the model on the final test set. For the other two models, i.e. random forest and logistic regression, I perfomed 10-fold cross-validation and performed a final test on the test dataset with the best identified model.

> 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 performed to ensure that the machine learning model generalizes well. The most common issue that arises is overfitting wherein a model simply remembers the training set as much as possible and then it is not able to generalize to unseen cases well. When this happens, the performance of the algorithm drops drastically. I used K-fold cross validation and learning and validation curves to validate my analysis (all analysis present in the notebook, `model selection.ipynb`).

> 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.

In order to tune the algorithms, I used F1 score as the scoring function as it captures the precision and recall information in a single number. I wanted both my precision and recall to be higher; however, recall is much more important as it will help us tag POIs. We want to flag people who might be fraudulent and then pursue an investigation. In the context of our project, precision measures the accuracy of our POI identifier in identifying the POIs. It gives us the probability of an individual being POI if he has been identified as such by our model. On the other hand, recall measures the ability of our model to correctly identify the POIs. It is the probability of a POI being identified by our model. Evaluation metrics for all three models are displayed below:

** Decision Tree **

	precision	recall	f1-score	support
0.0	0.92	0.88	0.90	25
1.0	0.40	0.50	0.44	4
avg / total	0.85	0.83	0.84	29

** Random Forest **

	precision	recall	f1-score	support
0.0	0.87	0.80	0.83	25
1.0	0.17	0.25	0.20	4
avg / total	0.77	0.72	0.75	29

** logistic Regression (SelectKBest Feature Selector) **

poi	precision	recall	f1-score	support
0.0	0.84	0.64	0.73	25
1.0	0.10	0.25	0.14	4
avg / total	0.74	0.59	0.65	29

** logistic Regression (Random Forest Feature Selector) **

poi	precision	recall	f1-score	support
0.0	0.89	0.96	0.92	25
1.0	0.50	0.25	0.33	4
avg / total	0.84	0.86	0.84	29

In the above table, we are concerned with the precision and recall for row with poi value of 1. Hence, we have the precision and recall of the final model as 0.5 and 0.25 respectively. However, the cross-validated average precision and recall obtained was 0.46 and 0.55 respectively.

0.1.3 References:

- Advanced Machine Learning With Scikit Learn by Andreas Mueller
- Python Machine Learning by Sebastian Raschka
- Applied Predictive Modeling by Kuhn & Jhonson