Identify Fraud from Enron Emails

Project for - Udacity Data Analyst Nanodegree

By Harish Garg

Table of Content

Understanding the dataset

Summary

In this project, I analyze the enron emails to identify Enron employees who may have committed fraud. The dataset I am using a processed dataset provided by Udacity as part of the project. This dataset belongs to the famous case of American corporation, Enron, Inc. which defrauded it's shareholders and ultimately went bankrupt. I am using python along with scikit-learn machine learning libraries for this analysis.

Data Exploration

Total number of data points

From the initial analysis, I find that there are 146 records in this dataset.

Allocation across classes (POI/non-POI)

Out of this, 18 are labelled as POIs(Person of Interest) and 128 Non POIs. POI means the ones who may have been party to the fraud and the criminal activity based on the feature data we have for these individuals.

Number of features

There are a total of 21 features…

- 14 of these are of financial nature with all values in USD.
- 6 of these are email related features.
- And the last on is the POI label, a boolean value.

Are there features with many missing values? Etc.

Most of the features at least 1 value missing. However, there are some features which have lot of missing values. For Example…

- With the exception of POI, all features have some missing values.
- Director fees has only 16 non-null values out of 146.
- Loan advances has only 3 non-null values
- Restricted stock deferred has only 17 non-null values

Missing values are handled in 2 ways...

1. Impute the the financial features by featureFormat to 0.
2. And Impute the missing email features to each feature mean.

Outlier Investigation

As part of the analysis, I discovered there is a record by the name of "TOTAL". This is obviously not a person and hence I removed it before proceeding with the analysis. There is one more record which doesn't stand for a person. It's "THE TRAVEL AGENCY IN THE PARK". This also had to be removed. That leaves us with 144 records.

Optimize Feature Selection/Engineering

Create New Features

Then, I created 3 new features...

- Bonus to Salary ratio - Outliers here on both higher and lower ends would be suspicious(a potential POI) and hence this was feature was created.
- Percentage of from_this_person_to_poi to total number of emails sent
- Percentage of from_poi_to_this_person to total number of emails received

And these 2 new email features would show who had lot of email activity with POIs and hence maybe a potential POI themselves.

Properly scale features

Scaling was performed for k-nearest neighbours algorithm and the support vector machines (SVM) algorithm. However, no scaling needed while using a decision tree.

Intelligently select feature

Below features were selected for the model using Decision Tree Classifier.

#	Feature	Feature Importance Score (DT Classifier)	Feature Score(select k-best)
1	ratio_of_bonus_salary	0.6585959527	22.1067164085
2	shared_receipt_with_poi	0.1802701987	6.1299573021
3	total_stock_value	0.1611338486	16.8651432616
4	exercised_stock_options	0.00	16.9328653375
5	bonus	0.00	34.2129648303
6	salary	0.00	17.7678544529

Pick an Algorithm

I ended up picking Decision Tree Classifier based on the evaluation metrics results.

Model performance of 3 algorithms i initially focussed on...

Algorithm	Precision	Recall
SVM	0.83333	0.06500
k-nearest neighbors	0.32247	0.29200
Decision Tree Classifier	0.31336	0.59100

Tune the Algorithm

Tuning a Machine learning algorithm means finding the best combination of parameters for a particular problem. Tuning a machine learning algorithm is important as can have a huge effect on its performance.

I performed automatic parameter tuning using scikit-learn GridSearchCV during the algorithm selection process.

Validate and Evaluate

Validation Strategy

Validation is the training and testing of a ML algorithm to assess its performance, and to prevent overfitting. One of the classic mistake one can make is to not split the testing and training sets adequately. For my analysis, I relied on the implementation provided by Udacity in their testing feature.

I validated by

- separating the data into training and testing sets, and
- using Udacity's provided tester.py's Stratified Shuffle Split cross validation on the chosen algorithm.

Evaluation Metrics

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"]

Usage of Evaluation Metrics

We consider 2 evaluation metrics and how they affect our choices here...

1. Precision

Precision measures the correctness of class prediction. Decision tree classifier gives us 0.31336 as average precision.

2. Recall

Recall measures how often we guess the class when the class actually occurred. I got 0.59100 as average recall.

This model has a higher recall and lower precision. That means we are casting a wider net i.e. more false positives to make sure we don't miss any POIs. Now, this will flag some as POIs but in fact they are not. However, isn't this how the investigators work. They go through a wider suspect list with a rigore to arrive at much short list of actual potential perpetrators of a crime.