Identify fraud from Enron Email

Report

Overview

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. In this project, I will play detective, 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.

This project uses the tools and starter code of the Udacity's Intro to Machine Learning course.

As preprocessing to this project, it has been combined the Enron email and financial data into a dictionary, where each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name, and the value is another dictionary, which contains the names of all the features and their values for that person. 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', '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)

Enron Submission Free-Response Questions:

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? [relevant rubric items: “data exploration”, “outlier investigation”]

   The goal of is to identify the persons of interest in the Enron fraud case using machine learning methods along with financial and email data. The persons of interest mean individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity. **Dataset** The Enron data set is comprised of email and financial data. The dataset contains 146 entries and 21 features. 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']. email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', 'shared_receipt_with_poi']. 18 of these points is labeled as a POI and 128 as non-POI. **Outliers** The following was eliminated from the data set.

- TOTAL: This is not a person
- THE TRAVEL AGENCY IN THE PARK: because it is a company and does not represent a person
- FREVERT MARK A: Not a POI but has a very high salary

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. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importance 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. [relevant rubric items: “create new features”, “intelligently select features”, “properly scale features”]

I added three features to the dataset:

- fraction_from_poi
- fraction_to_poi
- bonus ratio

Scaling the 'fraction_from_poi' and 'fraction_to_poi' by the total number of emails sent and received, respectively, might help us identify those have low amounts of email activity overall, but a high percentage of email activity with POIs. Also, bonus to total payment is added since if the ration is high, it could lead us to identify POI.

I used SelectKBest algorithm to score the features. Here is the result:

FeatureName	**Score**
total_stock_value	24.182898678566879
bonus	20.792252047181535
bonus_ratio	20.715596247559954
salary	18.289684043404513
deferred_income	11.458476579280369
long_term_incentive	9.9221860131898225
restricted_stock	9.2128106219771002
total_payments	8.7727777300916756
shared_receipt_with_poi	8.589420731682381

FeatureName	Score
loan_advances	7.1840556582887247
expenses	6.0941733106389453
from_poi_to_this_person	5.2434497133749582
other	4.1874775069953749
from_this_person_to_poi	2.3826121082276739
director fees	2.1263278020077054
to messages	1.6463411294420076
fraction_to_poi	1.2565738314129471
fraction_from_poi	0.23029068522964966
deferral payments	0.22461127473600989
from messages	0.16970094762175533
restricted_stock_deferred	0.065499652909942141

Selected Features In order to choose the best K value for feature selection, I tested different k values (2, 3, 5, 6, 9). The best result is produced by K=6 that has relatively the best precision and recall.

the selected features:

- 'Exercised_stock_options'
- 'Total_stock_value'
- 'bonus'
- 'Bonus_ratio'

- 'salary'
- 'Deferred_income'

Hence, only one of the three engineered features (bonus_ratio) is in the selected list.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

Because we are handling a classification problem, I tried with some of the most used algorithms (Naive Bayes, Support Vector Machine and Decision Tree) with resulting metrics:

GaussianNB()
Accuracy: 0.73213 Precision: 0.23405 Recall: 0.44400 F1: 0.30652 F2: 0.37646
Total predictions: 15000 True positives: 888 False positives: 2906 False negatives: 1112 True negatives: 10094

Pipeline(steps=[('scaler', StandardScaler(copy=True, with_mean=True, with_std=True)), ('classifier', SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False))])
Accuracy: 0.86747 Precision: 0.58333 Recall: 0.02100 F1: 0.04054 F2: 0.02602
Total predictions: 15000 True positives: 42 False positives: 30 False negatives: 1958 True negatives: 12970

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
presort=False, random_state=None, splitter='best')
Accuracy: 0.82467 Precision: 0.33729 Recall: 0.32650 F1: 0.33181 F2: 0.32860
Total predictions: 15000 True positives: 653 False positives: 1283 False negatives: 1347 True negatives: 11717

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? 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). [relevant rubric items: "discuss parameter tuning", "tune the algorithm"]

The behavior/results of an algorithm can be modified changing the value of the parameters. The idea is to tune the parameters of the algorithm to obtain better results. Each dataset is different and the algorithm parameters try to perform the best for each one.

To obtain the better values of the parameters of the Decision Tree Classifier I used GridSearchCV obtaining some parameters for the Decision Tree Classifier that increase all the metrics used to qualify the algorithm.

5.What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric items: "discuss validation", "validation strategy"]

If you test a model on same data as used for training the learner, the model may appear to make overly accurate predictions. This is an example of overfitting. To reliably estimate the predictive power of a model, it should be tested on data that hasn't been used for training the learner. Cross validation lets you use all examples for both learning and testing without ever using the same sample for both training and testing.

The way I'm validating the analysis is with the function test_classifier defined in the tester.py script provided by the tools folder of the starter code. This script uses *stratified shuffle split cross validation*.

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. [relevant rubric item: "usage of evaluation metrics"]

The metrics used to evaluate the performance of the algorithm are **Precision** and **Recall**. In our problem these metrics are better than accuracy to evaluate the performance because of the content of the dataset. We have a very skewed classes (labels) and the accuracy metric is not the ideal one. The final results of the metrics are:

Metric	**Result**
Accuracy	0.87600
Precision	**0.54113**
Recall	**0.46050**
F1	0.49757
F2	0.47464
Total predictions	15000
True positives	921
False positives	781
False negatives	1079
True negatives	12219

In this case, the values of precision and recall means that a 46% of the real POIs are detected and a 54% of the detected POIs is really true.