Report - Enron Fraud Detection Project

Fernando Maciel Motta

Project Goal and Machine Learning Application

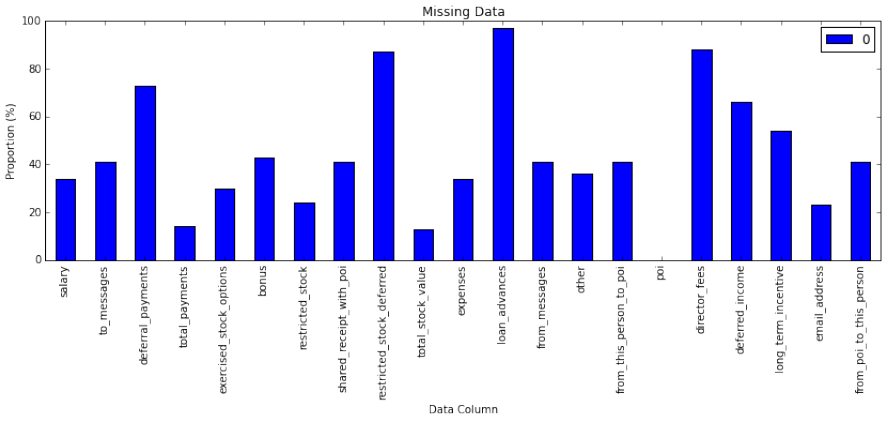
The goal of this project is to develop a tool that helps identifying financial fraud in companies. Machine Learning is used to achieve this goal by generating relationships between email and financial data and involvement in fraudulent practices.

This is achieved by generating models that learn these relationships (in this case, in a supervised manner) given data where it is known who the Persons of Interest are and the data is available.

The Enron dataset is useful to develop such tools, because it is publicly available and the people involved in the fraud are known and can be labelled, so the model can infer what patterns in the data imply fraudulent practices.

There were some outliers. I chose to remove the "Total" data, which seemed to be a visualization tool, and not representative of any single person. There was also a company which I decided to remove. Finally, the high executives which were not PoIs were removed because their high compensations might distort the analysis.

Using the .shape function from pandas, we see that there are 146 labels in the "poi" field. This implies there is data available for 146 people. This is also the only boolean field, all others are floats. Using the .info function, we can see that there are varying amounts of data for the different fields (which was to be expected, seeing as the original table gives meaning to NaNs).This can be visualized in the missing data chart that follows. The chart illustrates the amount of data present for a given field, relative to the number of possible points. There are 18 people with the PoI label in the dataset.



Features Used, Selection Process and Scaling

Scaling was used in the interest of normalizing the dataset, so that features with larger ranges and higher means won't be overvalued due to their inherently higher (or lower) values.

The SelectKBest function was used, and the scores for each feature are shown below:

1. from_poi_ratio: 0.3782
2. shared_receipt_with_poi: 0.2485
3. expenses: 0.2476
4. shared_poi_ratio: 0.0665
5. from_poi_to_this_person: 0.0592

All other features showed a score of zero. This intrigued me a bit, because upon using GridSearchCV to find the optimal parameters for the model, the suggested number of parameters used was 19.

Upon repeating my tests with GridSearchCV I kept obtaining oscillating results for the optimal number of parameters, so I decided to test by hand all values between the ones that were given (14 and 19).

- 14 parameters: F1 = 0.5665
- 15 parameters: F1 = 0.6055
- 16 parameters: F1 = 0.7611

- 17 parameters: F1 = 0.7899
- 18 parameters: F1 = 0.7848
- 19 parameters: F1 = 0.7623

Due to this result the number used in the final approach was 17 parameters.

The created financial features were the ratio of bonus to salary and to total payments for each person. This is because, in my opinion, people who receive larger and disproportionate bonuses raise suspicion. I also created features that sought to show communication and relationship with known PoIs, namely to_poi_ratio, from_poi_ratio and shared_poi_ratio, since I figured a lot of communication might mean involvement in something together.

Algorithm Choice

The algorithm I ended up using was the Decision Tree. The other algorithms tried were the naive Bayes classifier, the SVC with linear kernel and the KMeans classifier.

I chose the Decision Tree mainly because it outperformed all others in the F1 score metric, where it achieved an initial value of 0.47368. Naive Bayes achieved 0.4533, SVC achieved 0.2963 and KMeans achieved 0.16216.

After adding the engineered features I created, an enhancement in performance was perceived. F1 score rose to 0.49321 for Naive Bayes, 0.6 for Decision Tree, 0.37037 for SVC and 0.16666.

Parameter Tuning

Parameter tuning means setting the options within the machine learning algorithm so that it achieves optimal performance. This usually means an optimization process. Grid search was used to tune the parameters of my particular chosen model, and the optimal settings were classify_criterion = entropy, classify_max_depth and classify_max_features as none and classify_min_samples_split = 20.

Validation

Validation means to test your algorithm against a dataset so as to assess whether the training was effective. A classic problem is overfitting, which

means to fit your model too much to the training set, which causes it to become too specific and not generalizable, performing poorly on other datasets.

The validation done was implemented in the tester script and is cross-validation. This means that it performs splits on the dataset to generate different training and validation sets. The classifier is then trained and tested in each iteration. This seeks to provide different proper subsets and avoid overfitting.

Evaluation Metrics

Two of the metrics used were recall and F1 score. Recall represents the number of correctly identified positives among all positives in the dataset.

Recall is therefore defined as the amount of true positives divided by the sum of true positives and false negatives.

The optimized algorithm obtained a value of 0.84100 for recall.

The F1 score combines precision and recall by taking the harmonic mean between them .

Precision means the number of correct positives among all identified positives. F1 score is a form to blend this measure with recall by taking the harmonic mean between them.

The optimized algorithm obtained a value of 0.79246 for F1 score.