Identifying Fraud – Enron Dataset

Intro to Machine Learning

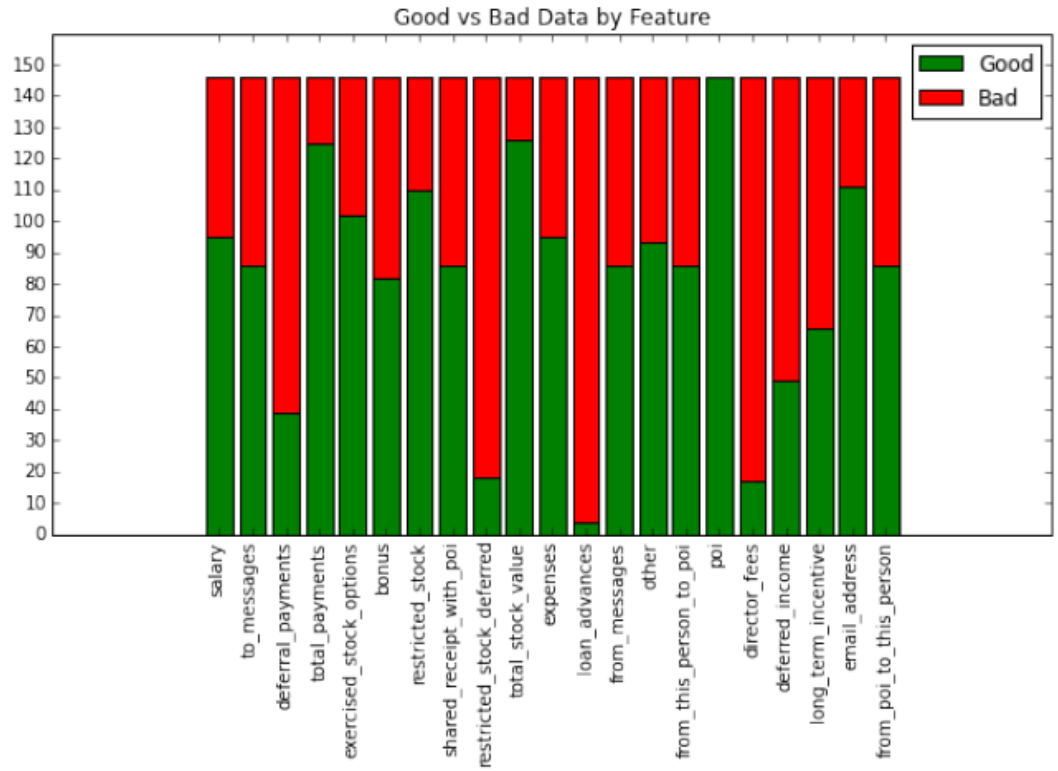
Edward Farrar

16 November 2015

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 this project is to identify persons of interest (POI) in the Enron fraud case. The data available consists of finance records and email from Enron. The finance data is "Payments to Insiders", Case No. 01-16034, EXHIBIT 3b.2 as provided from FindLaw. The data consists of 146 records, with each record containing 21 features. There are 128 non-POI entries and 18 POI entries in the dataset. Looking at each individual feature, 6 features had less than 50% valid data:



I identified 3 outliers in the data: TOTAL, THE TRAVEL AGENCY IN THE PARK, and LOCKHART EUGENE E. The TOTAL record was identified by creating histograms of each feature and looking for the person with the max value for each feature. This quickly showed there was a skew in the data centered around one person (i.e. one histogram bin isolated with only one entry). That person was "TOTAL". Further, since there were only 146 entries, a review of the names revealed a person named "THE TRAVEL AGENCY IN THE PARK". After reviewing the footnotes in the FindLaw data, I decided that this too should be removed since it is not a real person and all data but other and total_payments, were missing. Lastly, I decided to make sure every record had at least some valid data. Surprisingly, one record, LOCKHART EUGENE E, had no valid data, so it was also removed. The outliers were simply deleted from the dataset before additional processing.

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", "properly scale features", "intelligently select feature"]

I decided to use SelectKBest to pick features. I also wanted to explore PCA. Because PCA work better with scaling, I also experimented with scaling. This led to using pipelines to organize the flow. In the end, the best classifier was GaussianNB when used with these 5 features selected by SelectKBest: ['deferred income', 'bonus', 'total stock value', 'salary', 'exercised stock options']. The final selected classifier did not use PCA or require scaling. See poi_id.pdf for details.

I attempted to make 2 new features: bonusratio and poi_email_ratio. The bonus ratio was the percentage of bonus pay to salary. I reasoned that larger bonus percentages could indicate if someone was involved in the fraud versus a smaller ratio. The poi_email_ratio calculated the ratio of email to/from an POI in relation to all email sent by a person. The bonusratio variable showed up in many of the selected feature sets, but was not selected by the best classifier. The poi_email_ratio was also selected for several classifiers, but was not used by the final best classifier.

At the suggestion of the reviewer, I modified the K values for the final pipeline (GaussianNB) to search over the full range of K. This increased the GridSearchCV time to 53 seconds. Interestingly, SelectKBest still only chose 5 features:

Feature(Score):
    to messages(1.646341) Not Selected
    deferral payments(0.224611) Not Selected
    expenses(6.094173) Not Selected
deferred income(11.458477) Selected
    long term incentive(9.922186) Not Selected
    restricted stock deferred(0.065500) Not Selected

shared receipt with poi(8.589421) Not Selected
loan advances(7.184056) Not Selected
from messages(0.169701) Not Selected
other(4.187478) Not Selected
director fees(2.126328) Not Selected
poi email ratio(5.399370) Not Selected
bonus(20.792252) Selected
bonusratio(10.783585) Not Selected
total stock value(24.182899) Selected
from poi to this person(5.243450) Not Selected
from this person to poi(2.382612) Not Selected
restricted stock(9.212811) Not Selected
salary(18.289684) Selected
total payments(8.772778) Not Selected
exercised stock options(24.815080) Selected

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"]

I ended up using the GaussianNB algorithm. The other algorithms I tried were LogisticRegression, SVC, DecisionTreeClassifier, and RandomForestClassifier. These performance varied greatly based on the parameters supplied. I used GridSearchCV to try various combinations of arguments. The search ran as quickly as 10 seconds for GaussianNB searching over 4 different K values up to 18.3 hours for the RandomForestClassifier over 4 n_estimators ranging from 10 to 10000. See poi_id.pdf for details on the algorithms, parameters, timing, and results of the various algorithms tested.

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? (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 item: "tune the algorithm"]

Tuning the parameters of an algorithm means to explore and fine tune the configuration of the algorithm. The objective of tuning is to find the optimal values for the given parameters of an algorithm with respect to a specific metric. For example, I first used recall as the evaluation metric for the tuning, but was unable to find a set of parameters that satisfied both the recall and precision above 0.3. When I changed the metric to f1, which is a measure that more balances the recall and precision rather than favoring just one, I was able to identify a set of parameters that allowed GaussianNB to perform above the minimal values for recall and precision. I used GridSearchC to iterate over many parameters and values for the pipelines tested. For each combination of parameters, grid search would run the 1000 train/test sets and average the results to find the best algorithm/parameter combination.

This made an impact of some of the classifiers that took parameters like SVC. As I changed the range of values for C, as an example, I could see hoe larger values impacted the precision and recall.

Interestingly, I started using recall as the score function in GridSearchCV, but I wasn't finding any algorithm that met the .3 specifications for both recall and precision. I then changed to the f1 score which gave a more balanced result for both precision and recall. When I did that, I was able to find the GaussianNB algorithm that worked best on 5 features.

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 item: "validation strategy"]

Validation is a method of determining if the algorithm is adequate to predict unseen data. The best way to validate is to split your data into training and testing sets. I used StratifiedShuffleSplit to create 1000 random train/test sets.

There were two main reasons for choosing StratifiedShuffleSplit in order to generate training and testing sets. First, our dataset is small, so choosing one training and testing set could lead to bad results. This allowed me to create 1000 sets with 70% in the training set and 30% in the test set. Secondly, I choose a stratified method to preserve the ratio of POIs in the test sets. Since we only have 18 POIs, a random split could easily end up with all the POIs in either the training or the testing set which would be catastrophic for the algorithm. The StratifiedShuffleSplit allowed for the creation of many train/test sets while preserving the ratio of POIs in each set to the whole.

A classic mistake of not doing validation well is overfitting. Overfitting is when the algorithm is very good at predicting the data it knows, but fails miserably on unseen data.

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"]

For evaluation, we used two scores to judge the algorithm: precision and recall. For our algorithm, we had to ignore the accuracy score and focus on precision and recall. Precision is true positives divided by the sum of true positives and false positives. Recall is true positives divided by the sum of true positives and false negatives. For my algorithm, the goal was to have both precision and recall greater than 0.3. The actual algorithm had a precision of 0.41322 and a recall of 0.35950. With a precision of 41%, our algorithm is capable of finding roughly 7.4 out of the 18 POIs in the data set. The recall of 36% means that roughly 6.5 out of the 18 POIs are correctly identified.

Conclusion

I enjoyed working on this project. It shows how machine learning techniques can be used on a complex, but small dataset. One challenge was balancing the desire to run many algorithms with many varying parameters, but my small home system began to struggle with the more complex algorithms and larger parameter values. In the end, the simple GaussianNB classified turned out to be the best of the ones tested.

References

Lutz, Mark. *Python Pocket Reference, Fifth Edition.* Sebastopol, CA: O'Reilly Media, Inc., 2014.

Project resources, classes mini-projects, course examples.

machinelearningmastery.com. (2015) Machine Learning Mastery [How to Improve Machine Learning Results]. Retrieved from http://machinelearningmastery.com/how-to-improve-machine-learning-results/.

matplotlib.org. (2015) MATLAB Plotting [API reference for plotting functions]. Retrieved from http://matplotlib.org/api/pyplot_api.html

scikit-learn.org. (2015) Machine Learning in Python [API Reference for syntax and explanation of various functions]. Retrieved from http://scikit-learn.org/stable/.