Identifying fraud from Enron Email

Contents

[Introduction 3](#_Toc427539703)

[Enron Submission Free Response Questions and Answers 3](#_Toc427539704)

[Conclusion 6](#_Toc427539705)

[References 7](#_Toc427539706)

# Introduction

Enron Corporation was one of the world’s major electricity, natural gas, communications, and pulp and paper companies, with claimed revenues of nearly $111 billion during 2000. At the end of 2001, it was revealed that Enron’s reported financial condition was sustained substantially by an institutionalized, systematic, and creatively planned accounting fraud, known since as the Enron scandal. By 2002, it collapsed into bankruptcy due to this widespread corporate fraud. In the resulting Federal investigation, a large dataset containing a variety of information such as tens of thousands of emails and detailed financial data of Enron executives was made publicly available.

# Enron Submission Free Response Questions and Answers

As per the project specifications, here are the questions and their answers

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 utilize the features from financial and email data from Enron to build a predictive model that could identify whether an individual could be considered a "person of interest" (POI). Since the dataset contains labeled data i.e. whether a person is a POI or not, it is an example of Supervised Classification Problem.

The dataset contained 146 records with 14 financial features, 6 email features, and 1 labeled feature (POI). Of the 146 records, 18 were labeled POI and rest were non-POI.

Through exploratory data analysis, I was able to identify 2 records which could be classified as outliers. First one was “TOTAL” which was a sum of the features for the entire dataset. Another one was “THE TRAVEL AGENCY IN THE PARK” which definitely didn’t resemble a person. Later on while, splitting the dataset into labels and features, I came across another outlier “Eugene E. Lockhart” for which all the features were 0. So, I decided to remove this record as well.

1. 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 doesn’t 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.)  If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.  [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]

I created a GridSearchCV pipeline to scale features, select features and reduce dimensionality. To scale features, the pipeline used a MinMaxScaler because:

1. Features varied vastly on scales, email features were ranging in hundreds while financial features were ranging in millions of dollars
2. PCA and a number of algorithms such as Logistic Regression etc. perform optimally with scaled features.

To select features SelectKBest was used in the pipeline which used Anova F-valuecalssification scoring function to select K Best features. Before selecting the features, I added a number of features such as Totals and ratios for both email and financial data. However, the total features consistently increased the precision and accuracy of most of the machine learning algorithms tested.

Finally to reduce dimensionality, I used PCA (Principal Components Analysis) in the pipeline. Though it was a small dataset and there were limited number of features, but even then it makes perfect sense to use PCA since it improves the performance of a number of machine learning methods.

1. What algorithm did you end up using?  What other one(s) did you try? [relevant rubric item: “pick an algorithm”

Since in the dataset labels were already identified and of two types – True and False. So, it was a classic case of binary classifiers. I tried a number of algorithms such as Logistic Regression, K Nearest Neighbors, Support Vector Machine and Random Forest.

With Logistic Regression, K Nearest Neighbors and Support Vector Machine, I was able to achieve both Precision and Recall better than 0.3. However, for Random Forest with a number of optimized parameters, I was able to achieve Recall around 0.28 only.

Based upon the best scores of these algorithms, I finally ended up using Support Vector Machine. I have tabulated the scores of each algorithms in question 6 and the optimized parameters in question 4.

1. 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 don’t 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 if you used, say, a decision tree classifier). [relevant rubric item: “tune the algorithm”]

Each algorithm is a mathematical model which is a function of several inputs which are called parameters. These parameters depend upon the structure of the data and a number of other factors. For each algorithm these parameters need to be tweaked, before an optimal model can be designed. This process of tweaking the parameters for the optimal model is called tuning the parameters of an algorithm. If tuning of parameters is not done, or done poorly then your algorithm will not perform optimally and the results will not appropriate.

Each algorithm was tuned using GridSearchCV from sklearn over any major tunable parameters, over 1000 randomized stratified cross-validation stratified splits. The parameters which gave the highest average score were selected for the final model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Final Optimized Parameters** | **Logistic Regression Classifier** | **K Nearest Neighbors Classifier** | **Support Vector Machines Classifier** | **Random Forest Classifier** |
| SelectKBest - k | all | 4 | 17 | 4 |
| PCA - n\_components | 3 | 0.5 | 2 | 0.25 |
| C: Regularization constraint | 0.1 |  | 0.01 |  |
| tol: Tolerance for stopping criteria | 0.01 |  | 0.001 |  |
| class\_weight | auto | uniform | auto |  |
| kernel |  |  | rbf |  |
| N Neighbors |  | 1 |  |  |
| Leaf size |  | 64 |  |  |
| max\_depth |  |  |  | 25 |

1. 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 process of ensuring that your prediction model will generalize well. A classic mistake would be to train your model in such a way that it performs really well on the training dataset, but worse on a new, unseen data. This is known as over-fitting. Cross-validation can be used to avoid this problem.

Cross-validation is the process of splitting the data into training and testing data. Then the model can be trained on the training data, and validated on the testing data.

I validated my model using 1000 randomized stratified cross-validation splits and selecting the parameters which performed best on average over the 1000 splits. A similar validation process was used in tester.py to evaluate the performance of the chosen model.

1. 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”]

There are a number of evaluation metrics – accuracy, precision, recall, F1 and F2 score. I will focus on precision, recall and F1 here. Accuracy will not be the best evaluation metric for our model because of the sparsity of the poi labels. For example, if a model which always predicts a label as non-POI then it would have been around 87% accurate which clearly we don’t want since we will ignore all the POIs.

Precision is the ratio of True Positives to the sum of True Positives and False Positives. So in our case, Precision is the ratio of POIs which were correctly identified by our model to the records which were actually POIs. This metric helps us describe how often False Alarms are not raised i.e. how relevant the retrieved results are.

Recall is the ratio of True Positives to the sum of True Positives and False Negatives. So in our case, Recall is the ratio of POIs which were correctly identified by our model to the total records which are identified as POI. This metric helps us describe sensitivity i.e. did the system retrieve many of the truly relevant documents.

There is a trade-off between Precision and Recall and sometimes these are equally important for a prediction model, so another evaluation metric was created known as F1 score which equally weighs Precision (P) and Recall (R). It is calculated as the harmonic mean of Precision and Recall i.e. 2\*P\*R/(P+R).

While tuning my parameters for GridSearchCV pipeline I chose F1 as my scoring metric for the evaluation of my model which gave me both precision and recall better than 0.3.

Here are the metrics for each of my algorithms. So, finally I ended up choosing Support Vector Machines as my final algorithm. Though, Logistic Regression came pretty close.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metric** | **Logistic Regression Classifier** | **K Nearest Neighbors Classifier** | **Support Vector Machines Classifier** | **Random Forest Classifier** |
| Accuracy | 0.7615 | 0.8214 | **0.744** | 0.8301 |
| Precision | 0.3194 | 0.3263 | **0.3141** | 0.3363 |
| Recall | 0.698 | 0.319 | **0.777** | 0.282 |
| F1 | 0.4383 | 0.3226 | **0.4473** | 0.3068 |

# Conclusion

The most challenging aspect of this small and yet complex dataset was that it was very sparse in nature. I started the project selecting my features, optimizing parameters for algorithm and evaluating the results manually but later on, deploying pipeline from sklearn module simplified the job and finally, I was able to create a predictive model with Support Vector Machines Classifier with a precision of 0.31 and recall of 0.78 which is not bad.

# References

I relied on Wikipedia, scikit-learn, python documentation and Udacity classroom notes. Wherever I got stuck, I referred to stackoverflow and other blogs for resolving those errors.