**Project: Identify Fraud from Enron Email\_Report**

**Author: Ashish Sharma**

**Date: Mar 16, 2018**

**Abstract**

Enron's financial scandal in 2002 led to the creation of a very valuable dataset for machine learning, one where algorithms were trained and tested to be able to find fraudulent employees, or persons-of-interest (POIs). In this project, a merged dataset of financial and email data will be used to go through the entire machine learning process.

First, the dataset will be manually explored to find outliers and trends and generally understand the data we're working with. Certain useful financial or email-based features will be chosen (manually and automatically using Sklearn functions) and ensemble features created from those available, and then put through appropriate feature scaling. Then, numerous algorithms with parameter tuning will be trained and tested on the data, with the results of a Decision Tree Classifier, an ensemble classifier named Adaboost, and SelectKBest classifier being presented.

The detailed results of the final algorithm, a Decision Tree Classifier, is shown. The validation and evaluation metrics are shown and the reasoning behind their choice and its importance explained. Finally, other ideas involving feature selection, feature scaling, other algorithms and usage of email texts are discussed.

**Introduction**

In late 2001, Enron, an American energy company, filed for bankruptcy after one of the largest financial scandals in corporate history. After the company's collapse, over 600,000 emails generated by 158 Enron employees - now known as the Enron Corpus - were acquired by the Federal Energy Regulatory Commission during its investigation. The data was then uploaded online, and since then, many people and organizations have graciously prepared, cleaned and organized the dataset that is available to the public today (a few years later, financial data of top Enron executives were released following their trial).

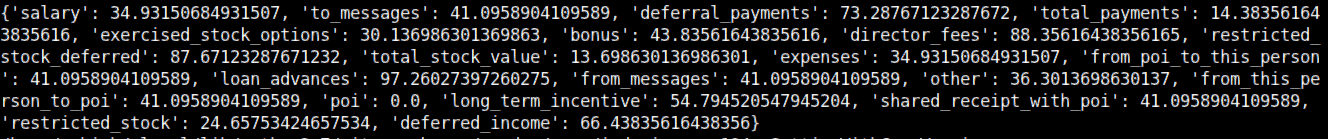
Today, the Enron Corpus is the largest and one of the only publicly available mass collections of real emails easily accessible for study. This excerpt from an [article](https://www.technologyreview.com/s/515801/the-immortal-life-of-the-enron-e-mails/) in MIT Technology Review summarizes the value of such a dataset:

“This corpus is valuable to computer scientists and social-network theorists in ways that the e-mails’ authors and recipients never could have intended. Because it is a rich example of how real people in a real organization use e-mail—full of mundane lunch plans, boring meeting notes, embarrassing flirtations that revealed at least one extramarital affair, and the damning missives that spelled out corruption—it has become the foundation of hundreds of research studies in fields as diverse as machine learning and workplace gender studies.”

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”]**

1. The goal of this project is to use machine learning to detect fraudsters from the Enron dataset. The Enron dataset contains a large quantity of emails between Enron employees along with data about employee financial compensation, e.g. salary, bonuses, etc. Each person of interest is labelled so we have a supervised learning problem.

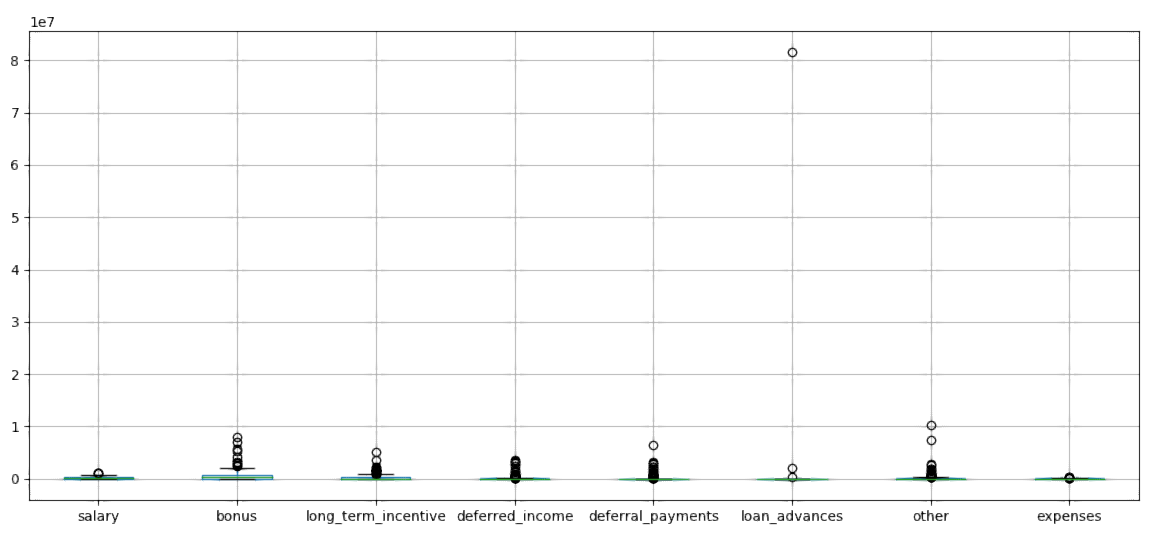
The Enron data set contains 146 persons, 18 of which are poi's. Each person in the dataset has 21 features. Across the total dataset, the percentage of NaN's for each feature are as follows:



1. Financial features were updated to 0 for unknown values
2. For email features, Imputation transformer was used for completing missing values. Since data is skewed, median was used as strategy than mean, and was calculated separately for poi and non-poi data.
3. Finally, I validated if sum of all payment equals to total payments. These errors were corrected using enron61702insiderpay.pdf using manual\_adjustment function. Same process was followed for stock data.
4. Outlier's were investigated visually using a pairwise scatter plot matrix. The pairwise plot matrix shows a clear outlier in each of the plots.
   * Further top 3 data point were explored for each feature to identify and check any non- person entity.
   * TOTAL was removed as it was summary statistic total only.
   * 'THE TRAVEL AGENCY IN THE PARK’ was removed which was a company co-owned by Enron's former Chairman's sister and is clearly not an individual that should be included in the dataset. Further LOCKHART EUGENE E was dropped from dataset as all financial features were 0 and data was non-relevant.
   * deferred\_income, deferral\_payments and restricted\_stock\_deferred were updated to their absolute values.
   * As observed in enron61702insiderpay.pdf, 'Director Fees' is essentially salary paid to Director.

Hence salary features were updated with director fees wherever salary was not defined.

The data was further explored for outliers once above steps were complete using Inter Quantile Range and boxplot.



**A Rest of the outliers were kept in the dataset for further processing.**

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 importance’s of the features that you use. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]**
2. **Create new features:**

The below new features were created using existing features:

* **New email features**: to\_poi\_ratio, from\_poi\_ratio, shared\_poi\_ratio,
* **New financial features**: bonus\_to\_salary\_ratio, bonus\_to\_total\_ratio, non\_salary

The motivation of using these email features is to identify if there is relation between the count of email exchanges from the person to the poi person and vice versa.

The financial features are created by creating co-relation between 2 or more strong features.

* + The to\_poi\_ratio is calculated as ratio between messages count for from\_poi\_to\_this\_person against to\_messages.
  + The from\_poi\_ratio is calculated as ratio between messages count for from\_this\_person\_to\_poi against from\_messages.
  + The from\_this\_person\_to\_poi is calculated as ratio between message count for shared\_receipt\_with\_poi against to\_messages
  + The bonus\_to\_total\_ratio is calculated as ratio between bonus and total\_payments
  + The bonus\_to\_salary\_ratio is calculated as ratio between bonus and total\_payments.

A new financial feature called non\_salary was created which involved summing all financial features excluding total\_payments and salary. The rationale behind creating this feature is that different poi’s may have exploited different non\_salary compensation mechanisms to increase their total compensation. The total compensation feature might not bring this feature out since some non-poi employees with a high salary with limited non\_salary compensation could have similar total compensation as the pois with high non\_salary compensation.

The evaluation metrics with and without this new feature are summarized in the answer to question 6 below.

**Properly scale features:**

Financial features were scaled using the MinMaxScaler function. Feature scaling is used to standardize the range of values in each of the features so that the classification algorithm does not put more weight on financial features that have a large range of values compared to financial features with a smaller range of values. This is because most classification algorithms use the Euclidian distance between data points in their algorithm.

**Intelligently select features:**

Financial features from the dataset were manually selected and stored in the financial\_features list. After scaling the features using the MinMaxScaler().

The feature score was calculated using SelectKBest(k='all') with results as below:

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Feature Name** | **Score** |
| 1 | **non\_salary** | 45.9301 |
| 2 | **bonus** | 41.5093 |
| 3 | **total\_stock\_value** | 33.2930 |
| 4 | **exercised\_stock\_options** | 26.7640 |
| 5 | **bonus\_to\_total\_ratio** | 25.8117 |
| 6 | **deferred\_income** | 23.3538 |
| 7 | **bonus\_to\_salary\_ratio** | 23.1929 |
| 8 | **restricted\_stock** | 18.8303 |
| 9 | **shared\_receipt\_with\_poi** | 16.3697 |
| 10 | **from\_poi\_to\_this\_person** | 14.8969 |
| 11 | **salary** | 13.7532 |
| 12 | **total\_payments** | 12.5170 |
| 13 | **long\_term\_incentive** | 11.5864 |
| 14 | **from\_poi\_ratio** | 11.5117 |
| 15 | **from\_this\_person\_to\_poi** | 9.8252 |
| 16 | **shared\_poi\_ratio** | 5.9511 |
| 17 | **to\_messages** | 3.4882 |
| 18 | **to\_poi\_ratio** | 2.6237 |
| 19 | **expenses** | 0.9655 |
| 20 | **restricted\_stock\_deferred** | 0.4768 |
| 21 | **loan\_advances** | 0.1454 |
| 22 | **from\_messages** | 0.0100 |
| 23 | **deferral\_payments** | 0.0046 |
| 24 | **other** | 0.0010 |

The final features were selected from the financial features subset using SelectKBest(k=22) after running multiple iterations of k values

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **KBest** | **Accuracy** | **Precision** | **Recall** | **F1** | **F2** | **Total Predictions** | **TP** | **FP** | **FN** | **TN** |
| all | 0.88733 | 0.58124 | 0.55450 | 0.56755 | 0.55965 | 15000 | 1109 | 799 | 891 | 12201 |
| 23 | 0.88633 | 0.578 | 0.5465 | 0.5618 | 0.5525 | 15000 | 1093 | 798 | 907 | 12202 |
| 22 | 0.89127 | 0.59914 | 0.5575 | 0.5776 | 0.5654 | 15000 | 1115 | 746 | 885 | 12254 |
| 21 | 0.8904 | 0.5956 | 0.5545 | 0.5743 | 0.5623 | 15000 | 1109 | 753 | 891 | 12247 |
| 20 | 0.89033 | 0.59538 | 0.554 | 0.5739 | 0.5618 | 15000 | 1108 | 753 | 892 | 12247 |
| 19 | 0.89073 | 0.59699 | 0.55550 | 0.57550 | 0.5633 | 15000 | 1111 | 750 | 889 | 12250 |
| 15 | 0.86953 | 0.51258 | 0.438 | 0.4724 | 0.4511 | 15000 | 876 | 833 | 1124 | 12167 |
| 14 | 0.8734 | 0.53029 | 0.442 | 0.4821 | 0.4572 | 15000 | 884 | 783 | 1116 | 12217 |
| 13 | 0.812 | 0.30039 | 0.3085 | 0.3044 | 0.3068 | 15000 | 617 | 1437 | 1383 | 11563 |
| 10 | 0.8264 | 0.35633 | 0.3745 | 0.3652 | 0.3707 | 15000 | 749 | 1353 | 1251 | 11647 |
| 5 | 0.79536 | 0.30956 | 0.3515 | 0.3292 | 0.3422 | 14000 | 703 | 1568 | 1297 | 10432 |

\*The above metrics are calculated without feature scaling. With feature scaling, the algorithm perform a slightly better (Precision/Recall/Accuracy increased by 1%) was mentioned below.

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

I finalized 3 classifiers based on GridSearchCV parameters tuning and results.

Feature scaling was done using MinMaxScaler() and feature selection was done with SelectKBest (k=22) was the tuned k value with the features and their scores highlighted above).

Results for the evaluation metrics of top 3 algorithms chosen with SelectKBest (k=22):

|  |  |
| --- | --- |
| Classifier | Metrics |
| svm.SVC(kernel = 'rbf',  C = 10,  gamma =1) |  |
| KNeighborsClassifier  (n\_neighbors=3 ,  weights = 'uniform',  algorithm='auto',  p=1,  metric ='minkowski',  leaf\_size=10) |  |
| AdaBoostClassifier  (DecisionTreeClassifier  (criterion='gini',  max\_depth=1,  min\_samples\_leaf=2,  class\_weight='balanced'),  n\_estimators=50,  learning\_rate=.8) |  |

Ensemble algorithm AdaBoostClassifier provided **accuracy** of .89167 with **Precision/Recall** of 0.60086 / 0.55850.

SVC algorithm provided **accuracy** of .89620 with **Precision/Recall** of 0.65855 / 0.46000.

Since the overall purpose of this model is to identify more - **true** persons of interest **even if some non-persons of interest are falsely identified as person of interest by the model i.e. False Negative > False Positive, which means Recall weighs more, while choosing between 2 almost similar comparable algorithms.**

**Hence, the final algorithm I chose is Ensemble algorithm AdaBoostClassifier**

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

Other classifiers explored with parameter tuning included RandomForestClassifier, DecisionTreeClassifier, LogisticRegression and GaussianNB,

Random forest and Adaboost classifiers were found to be much slower for this classification project.

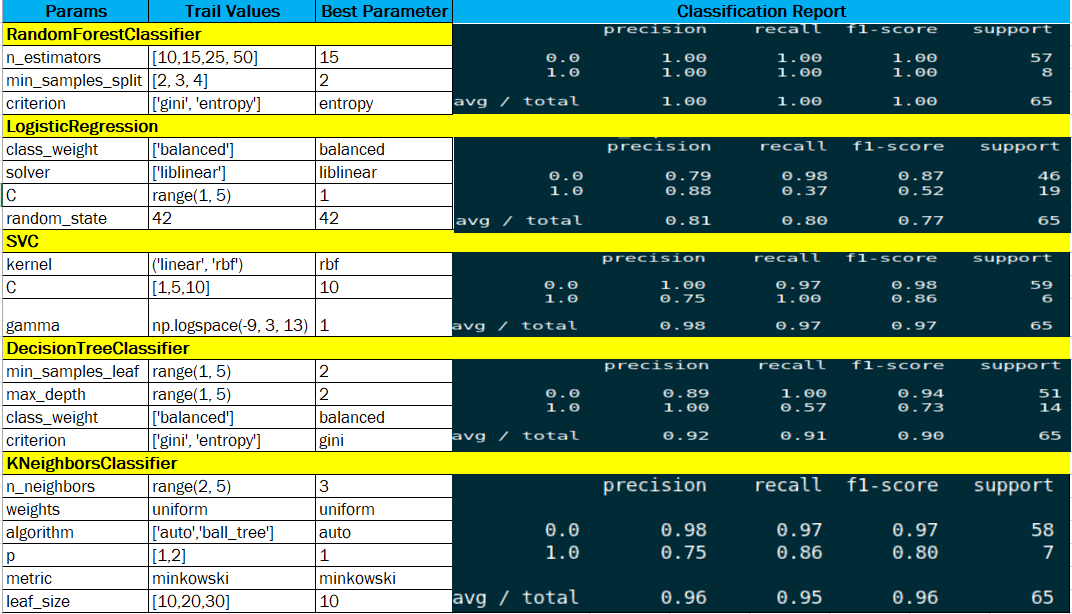
**4.What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well? How did you tune the parameters of your particular algorithm? (Some algorithms don’t have parameters that you need to tune--if this is the case for the one you picked, identify and briefly explain how you would have done it if you used, say, a decision tree classifier). [relevant rubric item: “tune the algorithm”]**

Each classifier explored in the assessment has one or more parameters related to the structure of the algorithm. 'Tuning' these parameters affects the score of our classifier based on our selected evaluation metrics so it is important to explore how changes to the default parameter assumptions influence relevant evaluation metrics. If you don’t tune the parameters, you are going to end up with a model that has poor performance relative to a parameter tuned model.

GridSearchCV with StratifiedShuffleSplit(n\_iters=500) as the cross validation method was used to tune parameters of the algorithms identified above. StratifiedShuffleSplit was found to be valuable as a cross-validation method because it preserves the ratio of targets to features in each cross-validation ‘fold’.

This is important when the frequency of a target class (e.g. poi’s) in the dataset is low because having randomly selecting few of a target class of interest (for us poi’s) in a random fold will limit the effectiveness of model training. GridSearchCV takes in as arguments that algorithm, a cross-validation method, a scoring and a dictionary of key:value pairs that include the parameters to be tuned, and an array of values for each of these parameters. For each algorithm, accuracy, precision and recall scores were passed as scoring metrics for GridSearchCV to optimize.

Below are the parameters and scores achieved after parameter tuning with GridSearchCV.



Results presented above for the final algorithm extracted the tuned parameters for SelectKBest and used these to select the best features that were passed to the final 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 item: “validation strategy”]**

Validation involves training our model and exploring how good this trained model is, in terms of how effective it is at providing the correct prediction, when we throw new data at it, as judged by specific evaluation metrics.

A classic validation mistake is to train a model using all of the data which leads to overfitting. To avoid overfitting, sklearn’s cross validation method StratifiedShuffleSplit was used to split the data into training and testing sets when using GridSearchCV. StratifiedShuffleSplit provides a set of randomized training/testing sets ensuring that the target class ratio (i.e. poi’s to non-poi’s) of the training and testing datasets are the same as in the original dataset.

The test\_classifier() function also applies StratifiedShuffleSplit when evaluating the performance of the model. As pointed out above, ensuring the ratio of each target class are equivalent in the training and testing sets are similar to those in the original dataset is important when the number of a particular class is low since this affects the validity of a model.

In the extreme case, if no pois are in a given fold, we are unable to develop a useful model to predict poi’s from this fold and this will influence the scoring metrics of the final algorithm.

**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”]**

Three performance metrics were explored:

* Accuracy – the percentage of poi's that were correctly identified (true positives / total people)
* Precision is measured over the total predictions of the model. It is the ratio between the correct predictions and the total **predictions**. In other words, precision indicates how good is the model at whatever it predicted. - (true positives/ (true positives + false positives)). In this case it’s the ratio between how many times the classifier **correctly** predicts as POI vs. how many time the **classifier predicts** as POI (correct and incorrect)
* Recall is the ratio of the correct predictions and the total number of correct items in the set. It is expressed as % of the total correct(positive) items correctly predicted by the model. In other words, recall indicates how good is the model at picking the correct items (true positives/ (true positives + false negatives)). In our case it’s the ratio between how many times the classifier correctly predicts as POI vs. how many times the **person actually is POI.**

|  |  |  |
| --- | --- | --- |
| **Occurrence count** | **Classifier** | **Actual** |
| **True positives (TP)** | POI | POI |
| **True negative (TN)** | NON-POI | NON-POI |
| **False positive (FP)** | POI | NON-POI |
| **False positive (FN)** | NON-POI | POI |

Since the overall purpose of this model is to identify more - **true** persons of interest **even if some non-persons of interest are falsely identified as person of interest by the model i.e. False Negative > False Positive, which means Recall weighs more, while choosing between 2 almost similar comparable algorithms. The classifier does a very good job in identifying POI with a recall of 0.55750. i.e. model is correctly able to identify poi when in reality the person is poi with very low occurrence where it is not able to identify the poi.**

**Confusion Matrix :**



