# ENRON EMAIL FRAUD DETECTION WITH MACHINE LEARNING

# ENRON EMAIL FRAUD DETECTION

**BY CASEY IANNONE**

## Summary

*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 determine if by using the Enron email and financial dataset, can we create a model that can accurately identify a Person Of Interest (POI). A person of interest could be someone worth investigating for fraud charges. Machine learning is useful in trying to accomplish this goal because of it's ability to find trends, categorize data and learn and put this information to use in classifying possible cases of fraud before they are committed.

An example that applies well is that of credit card fraud. Credit card companies use data from past fraudulent events to recognize credit fraud either before it happens or before it becomes more devastating than it already has. For example, many credit card companies may disable a clients card if unusual spending patterns occur. These patterns have been derived using machine learning from data collected in the past. Identifying POIs would work the same way, potentially identifying possible company fraud before it takes place or is exacerbated.

### Dataset

Enron was one of the largest companies in the United States. By 2002, it collapsed into bankruptcy due to widespread corporate fraud, and during a federal investigation the Enron corpus, emails and financial information, which was retrieved/gathered together was released to the public. *The dataset used for this project was preprocessed and stitched together by Katie Malone from Udacity.*

### Data Exploration

Data exploration was the first step, determining size of data set, number of features, etc. When exploring the data, a number of outliers were discovered, some were not relevant while others were. In order to determine next steps, more data analysis was required.

Extreme outliers were removed from the dataset. One was a data point representing the total amount of values and not an actual observation. Another value that was removed was listed as "The Travel Agency In The Park," which seems to be a company account and not the value of a single employee and therefore it was removed. Additional outliers were discovered with bonuses of one and five million plus, these are listed below in the data overview. I decided to keep these outliers as I believe the data points represented the true data and were important to my overall goal for this particular project.

As with any dataset there is bound to be some missing data or gaps. It is important to provide a high-level aggregation of the dataset to as to ensure context. Below I have aggregated a few key points of the data, such as missing values and interesting data points.

## Data Overview

- Number of people in dataset: 146
- Number of features for each person: 21
- Number of Persons of Interests (POIs) in dataset: 18 out of 34 total POIs
- Number of non-POIs in dataset: 128
- POIs with zero or missing to/from email messages in dataset: 4
  - Kopper Michael J
  - Fastow Andrew S
  - Yeager F Scott
  - Hirko Joseph

Salary Bonus Crazy (Employees receiving a salary or bonus of 1M+ and 5M+, respectively): - Lavorato John J - Lay Kenneth L - Belden Timothy N - Skilling Jeffrey K - **Total** ('grand total' - data point removed) - Frevert Mark A

Incomplete data - NaN values in features:

---

| Feature | # of NaN | Feature | # of NaN |
|---|---|---|---|
| salary | 51 | to_messages | 60 |
| deferral_payments | 107 | total_payments | 21 |
| loan_advances | 142 | bonus | 64 |
| email_address | 0 | restricted_stock_deferred | 128 |
| total_stock_value | 20 | shared_receipt_with_poi | 60 |
| long_term_incentive | 80 | exercised_stock_options | 44 |
| from_messages | 60 | other | 53 |
| from_poi_to_this_person | 60 | from_this_person_to_poi | 60 |
| poi | 0 | deferred_income | 97 |
| expenses | 51 | restricted_stock | 36 |
| director_fees | 129 | | |

## Feature Selection & Engineering

*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 importances 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"]*

Both univariate feature selection and engineering were performed and used in tested when creating the final model. Feature scaling was also utilized as there were a number of outliers which could skew the results (be used as a primary predictor) but due to the validity of the data, these points could not be removed.

This had todo with the goal of this project. Using salary and bonus to predict POIs, removing those with extremely high bonuses will skew the results, but this is needed to an accurate representation of the data and the model.

Although performance was tested with and without feature scaling as a reassurance to the process, the final model I choose did not utilize feature scaling, as I believed it unnecessary as it relates to the goals of this project.

**Feature Selection**

Feature selection was performed through the use of SelectKBest to obtain the the best precision and recall scores during testing. It should be noted that selection of these features was *not* done manually. Train and test sets were created with the use of Stratified Shuffle Split cross validation due to small size of the data and that of the POI list.

The final features that were utilized in the model are:

- salary
- bonus
- deferred_income
- exercised_stock_options
- total_stock_value
- to_poi_fraction score
- shared_receipt_with_poi
- total_payments
- expenses score
- from_poi_fraction score

**Feature Engineering**

In reviewing this project as a whole and its goals, I decided to engineer three additional features to be utilized in the testing of the model.

- *to_poi_fraction*: a fraction of the total 'to' emails that were sent to a POI.
- *from_poi_fraction*: a fraction of the total 'from' emails that were received from a POI.
- *salary_bonus_fraction*: a fraction of salary to bonus money.

With the project goal of identifying POIs, I believed adding two additional features that may identified if person A sends (or receives) a large portion of their total emails to/from a POI, there may be a greater likelihood that person A may also be a POI. I believe such an interaction could shed some additional insight in providing another factor for the algorithm to take into account.

In addition, if such an interaction existed with those outliers with 1 and 5 million plus bonuses that were identified as outliers could help provide the algorithm with additional data to link other individuals who may have had high contact with these individuals.

GridSearchCV was initially used but being unable to easily identify k best features, I did a number of separate SelectKBest testing and viewed their precision and recall scores. With SelectKBest, the number 10 for k was decided upon after a number of performance and tuning, which determined this to deliver the best mix of performance (timing), precision and recall.

Below is a table of features and their scores:

| **Feature** | **Score** |
|---|---|
| bonus | 38.290312 |
| deferred_income | 19.454073 |
| salary | 18.012912 |
| exercised_stock_options | 14.621981 |
| total_stock_value | 14.544973 |
| to_poi_fraction score | 14.271175 |
| shared_receipt_with_poi | 12.597418 |
| total_payments | 11.007505 |
| expenses score | 2.995212 |
| from_poi_fraction score | 1.437558 |

## Machine Learning Model Review

*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"]*

The POI identifier model uses the AdaBoost and DecisionTree algorithm as it provided the best validation results. The other algorithms I used were Decision Tree (without AdaBoost), Random Forest and GaussianNB, and Logistic Regression, all of which performed adequately in one aspect or another.

After deciding to keep those outliers mentioned above I made an assumption that using something like an ensemble method like AdaBoost with a DecisionTree would more than likely provide some of the best validation results. I believe this happened given some of the feature scores that were noted above. Using an ensemble method increases the weight of misclassified data points, which I thought would be common given the fact I choose to keep those 1 and 5 million plus outliers. And while Random Forest trains disturbed data across the various trees, which could have help with those outliers, it did help provide the best accuracy score but came with low precision and recall scores.

**Tuning**

*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 a machine learning algorithm is typically the last piece of the pie. Each problem we use machine learning to solve involves different data and outcomes. This means that each model and the algorithm used needs to be adjusted given a number of parameters and context. Each algorithm can be tuned based on a variety of parameters that are decided based on your dataset, hypothesis, research and a number of other variables that need to be taken into account. As such, finding the best combination of parameters can be treated as a bit of a search problem.

Many machine learning algorithms have parameters that are set with a default value, so sometimes it's not always necessary to adjust these parameters and you can just relay on the defaults of the given algorithm. If you do tune your algorithm, but not correctly, you may end up with a model that seems correct but is actually providing false data and overfitting, which can be very easy with certain algorithms.

For example, let's say we have 20 factors overall and we decided to start with a simple decision tree. With decision trees you can tune the algorithm with a number of parameters, one such parameter is max_depth. If you put a value of 20 for this parameter the tree will create a node for every feature, which will probably give you an outstanding accuracy, but nonetheless will be useless in applying to new data because you ran the tree and overfit the data. Thus giving you the illusion of unbelievable accuracy using unrealistic parameters.

In the case of my final model, I used the GridSearchCV function to determine the optimized parameters. Given a set of parameters, this function evaluates (fit, transforms) all of the possible combinations, then returns a classifier, that provides the best score.

| **Algorithm** | **Precision** | **Recall** | **# of Training (StratifiedShuffleSplit folds)** |
|---|---|---|---|
| Random Forest | 0.41747 | 0.21750 | 100 |
| Logistic Regression | 0.31064 | 0.60900 | 100 |
| AdaBoost (DecisionTree) | 0.43123 | 0.40600 | 100 |

## Validation

*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 the process of checking your model's prediction against your test dataset. The test dataset wasn't used to train the model. In the example provided above regarding overfitting with the decision tree, we would find out quickly that our model was tuned incorrectly when we ran out test dataset using that same algorithm with its set of parameters.

Overfitting can occur typically when you do one of two things, either train the algorithm with all of your data or adjust parameters to fit all data points perfectly during the training. While there isn't a function or failsafe to stop these things from happening, which makes overfitting easy todo, understanding the appropriate steps in data science and analysis and model parameters can help sure you understand how data interacts with the model and the results you receive.

The final model used in this project used the stratified shuffle split method. This method creates stratified randomized folds of the given dataset by preserving the same percentage for each target class as in the complete set. What this actually means is that individuals end up in different testing folds and the same individual *is never* in both the testing and training data. This method can lead to an imbalance in regards to the amount of data in both testing and training.

This was an ideal approach given the small dataset and even smaller number of POIs within the dataset.

**Model Results**

*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 final model uses Precision, Recall and F1 scores to evaluate how good the model is in predicting POIs. The raw data can be seen below. Each observation is a test, and each test made 15,000 predictions.

| **Precision** | **Recall** | **F1** | **# of Training (StratifiedShuffleSplit folds)** |
|---|---|---|---|
| 0.50000 | 0.45000 | 0.47368 | 10 |
| 0.47458 | 0.42000 | 0.44562 | 100 |
| 0.43123 | 0.40600 | 0.41823 | 1000 |

- *Precision is the measurement of how many selected items were identified as relevant*
- *Recall is the measurement of how many relevant were selected*

The model's precision is approx. 45% i.e from the people classified as POIs by the model, 45% of them are actual POIs. However, the model's recall is approx. 40% i.e from the number of actual POIs in the total dataset, the model correctly identifies 40% them. It can be concluded that although the model "spreads a wide net" it will capture over 40% of actual POIs.