Identify Fraud from Enron Email

By Rehab Fathi Ali

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. In this project, you will play detective, and put your new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal. To assist you in your detective work, we've combined this data with a hand-generated list of persons of interest in the fraud case, which means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.
We will use machine learning for building an algorithm to identify Enron Employees who may have
committed fraud based on the public Enron financial and email dataset.

● Data Exploration

- Number of persons: 146
- Number of features: 21
- Number of persons of interest: 18
- Number of persons non-POI: 128
- Features with many missing values: 'deferral_payments': 107, 'deferred_income': 97,
- 'director_fees': 129, 'loan_advances': 142, 'restricted_stock_deferred': 128

● Outlier Investigation

There were 3 outliers and all were removed
- ‘LOCKHART EUGENE E’ had all entries as NaNs
- ‘TOTAL’ had the total salary which was a huge outlier
- ‘THE TRAVEL AGENCY IN THE PARK’ was obviously not a person’s name

● Create New Features

- Two new features were created calculating the fraction of the emails data set that was sent to/from the POIs.
- My hypothesis was that the more emails were sent to/from a POI, the more data we could use to measure correlation and the more accurate our results will be.
- Using these two features did improve the accuracy of our used algorithm.

● Features Scaling

- The algorithm used (Naive Bayes) didn’t require feature scaling

● Feature Selection

- I used SelectKBest with Gaussian NB for features selection. The optimum number of features was 6 features (['salary', 'bonus', 'total_stock_value', 'exercised_stock_options', 'deferred_income', 'fraction_to_poi'])

Number of Features	**F-Score**
Salary	18.013
Bonus	38.29
total_stock_value	14.54
Exercised_stock_options	14.62
deferred_income	19.45
fraction_to_poi	14.27

● Algorithm Selection:

After trying 3 different algorithms I ended up selecting a tuned Gaussian NB and here is the summary of the algorithms

- Decision Tree:

Accuracy: 0.82127 Precision: 0.28075 Recall: 0.21800

- KNN:

Accuracy: 0.82940 Precision: 0.26050 Recall: 0.15200

- Gaussian NB with new features (SELECTED):

Accuracy: 0.85073 Precision: 0.42788 Recall: 0.35450

- Gaussian NB with original features:

Accuracy: 0.84600 Precision: 0.40264 Recall: 0.32050

● Paramater Tuning

- Paramater tuning is finding the best combination of parameters for our model usually to have the best bias/variance tradeoff and prevent underfitting/overfitting
- Tuned paramaters can greatly affect the accuracy of the model
- I used automated tuning in the selected algorithm using the SelectKBest() module from sklearn. It selects the features that output the best score

● Validation

- Validation is testing the model on a new dataset to VALIDATE our results and make sure the algorithm does not overfit the training data

● Validation Strategy

- For validation, I used cross validation by which we split our data into training and testing set. By splitting our data, we won't overfit and could test on independent data set. Since our dataset is of small size(~140), we use stratified splits for number of iteration

● Evaluation Metrics

- Precision is the measurement of how many selected POIs were identified as actual POIs (True Positive(TP)/[True Positive(TP) + False Positive(FP)])
- Recall is the measurement of how many real POIs were selected by the model (True Positive(TP)/[True Positive(TP) + False Negative(FN)])
- The used algorithm: Precision: 0.427 && Recall: 0.35450