ENRON Corp. – identifying POI from available financial and email data

Background & Exploratory Analyses

Enron Corp., an American company that was in the energy sector and expanded in to commodities and services. It was based in Houston, TX. The storyline of the company can be seen below:

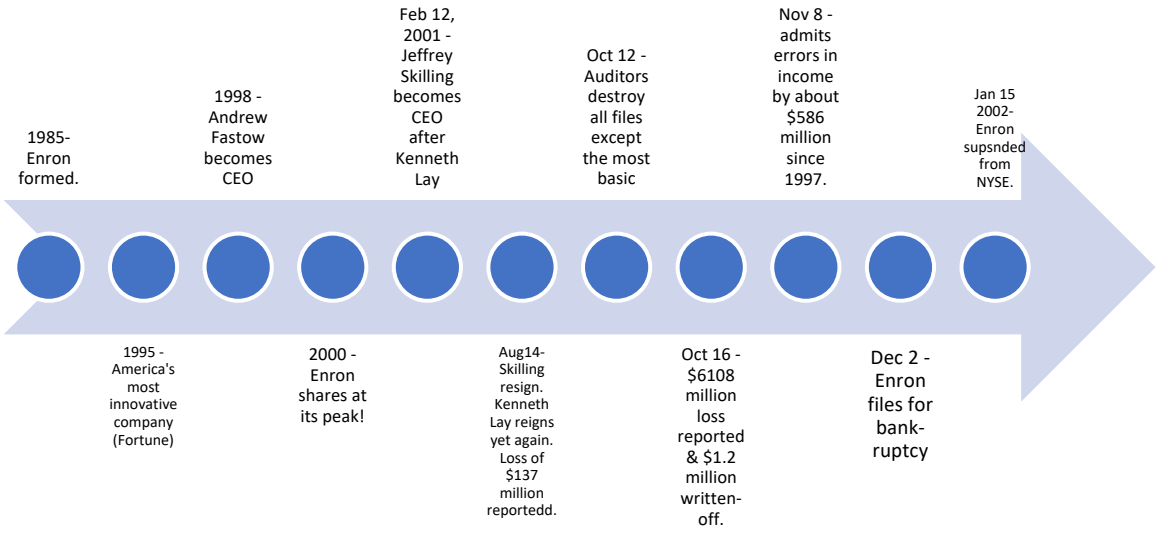

Figure 1 Adapted from https://www.investopedia.com/updates/enron-scandal-summary/

The goal is to identify the Persons-of-interest (POI) from the manually labelled Enron Email Corpus dataset along with the related financial data.

Machine Learning along with some feature transformations can be useful in wading through huge chunk of data with no obvious patterns. It can be automated for various parameters and give results in a shorter span of time than intuition and manual inspection.

To begin the process, I explored the dataset by looking at summary data and using graphs. The data consisted of 146 keys in the data dictionary. My exploration let me to the fact that one of the main outliers was “TOTAL”. Since this was not really a person, I removed that row of information from the dataset. I also noticed that several columns had “NaN” in the data which was handled by replacing them by 0.

Previous to the above analyses, I had spot checked some data and noticed that the entry of data for two individuals, “BHATNAGAR SANJAY” and “BELFER ROBERT” were erroneous. This was corrected using the data from “enron61702insiderpay.pdf” file. I also removed 5 individuals with more than or equal to 15% missing data. The 5 individuals removed were “GRAMM WENDY L”, “LOCKHART EUGENE E”, “THE TRAVEL AGENCY IN THE PARK”, “WHALEY DAVID A” and “WROBEL BRUCE”.
Below are two graphs that I had produced during exploration.

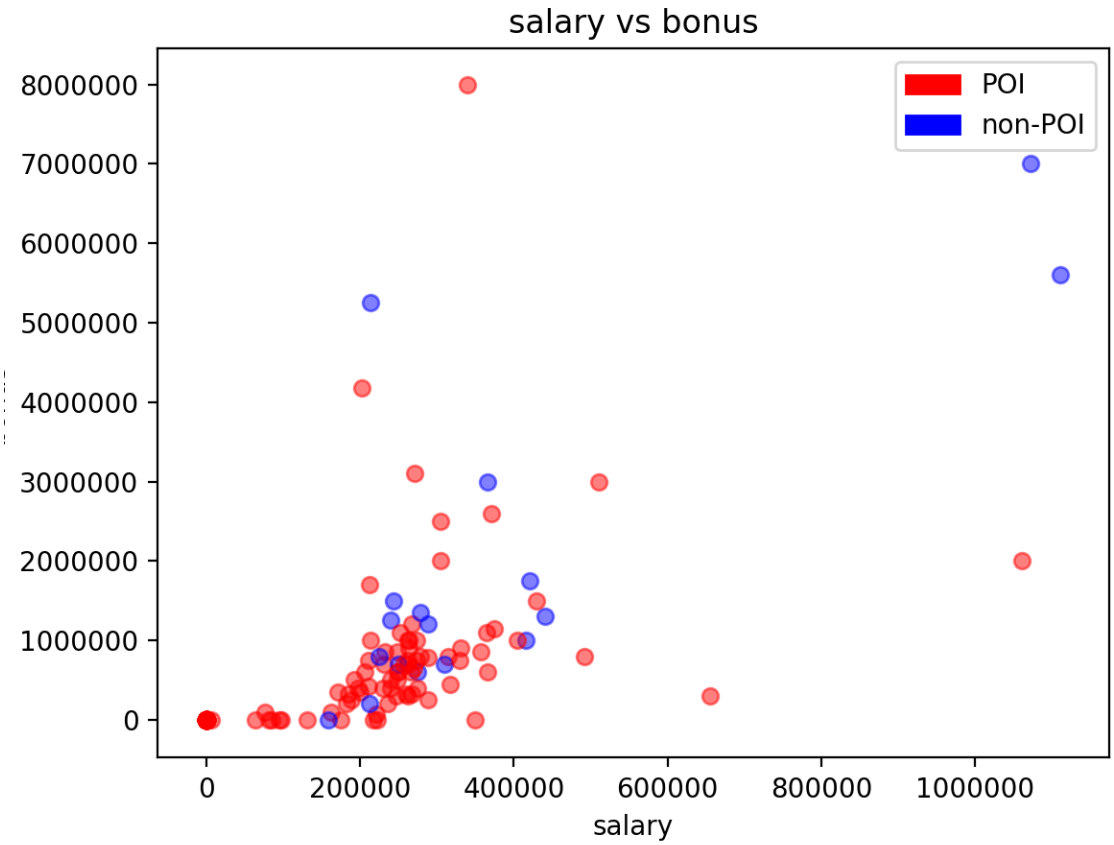

Figure 2 Salary versus Bonus categorised as POI or non-POI.

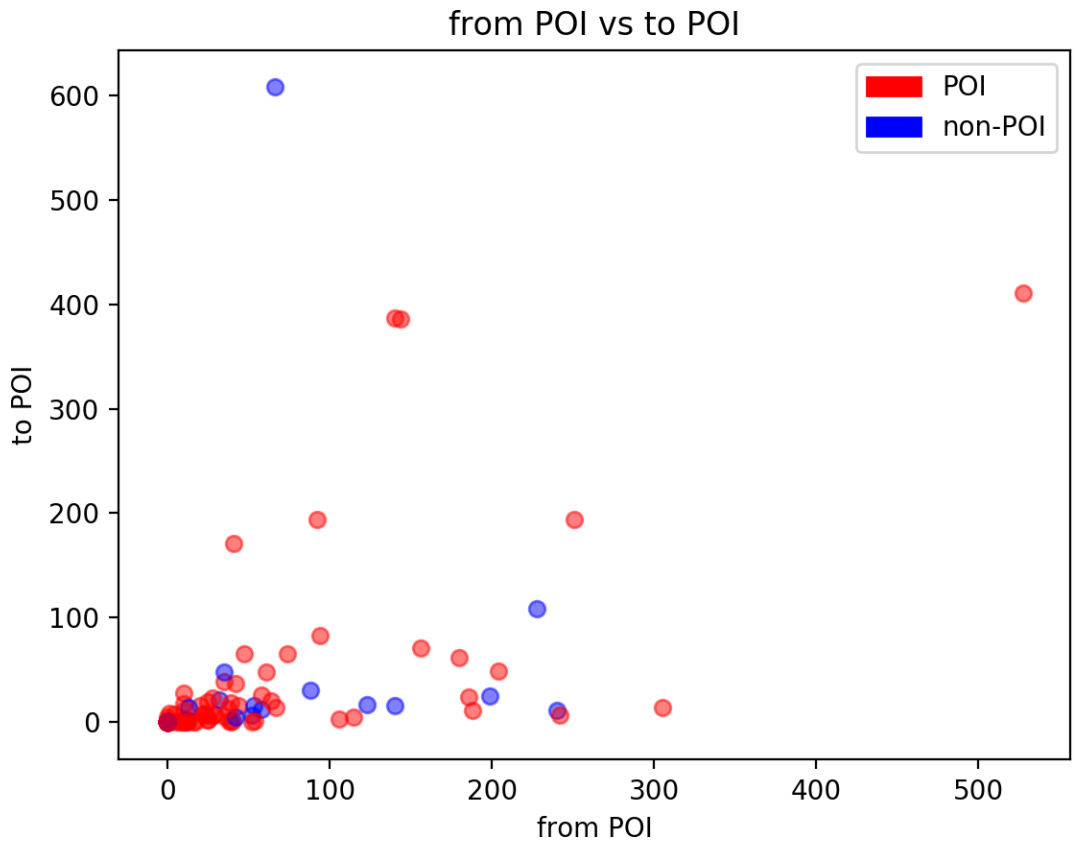

Figure 3 Number of e-mail messages from POI versus to POI

After removing outliers and those with more than 15% missing data, the dataset consisted of 140 individuals. 118 of the 140 were non-POI and 12 were POI.

Features – creation, transformations

I picked a combination of features (n=16) from the financial and email data. They were picked after plotting during the exploratory process. I also created three new features (bringing the total number of features to n=19) engineered from the email data – "propFromPoi", 'propToPoi' and 'propSharePoi'. These features are the proportions of emails from, to or shared receipt with persons-of-interest. Rationale in looking at proportions is the hypothesis that a POI probably had more email exchanges with other POIs than the non-POIs.

Since I was using both financial and email data which on different scales and also as there were some interesting outliers, I decided to use the RobustScaler.

I realized while exploring and looking at graphs like the one below that there were some latent features that I should extract and use for my model.

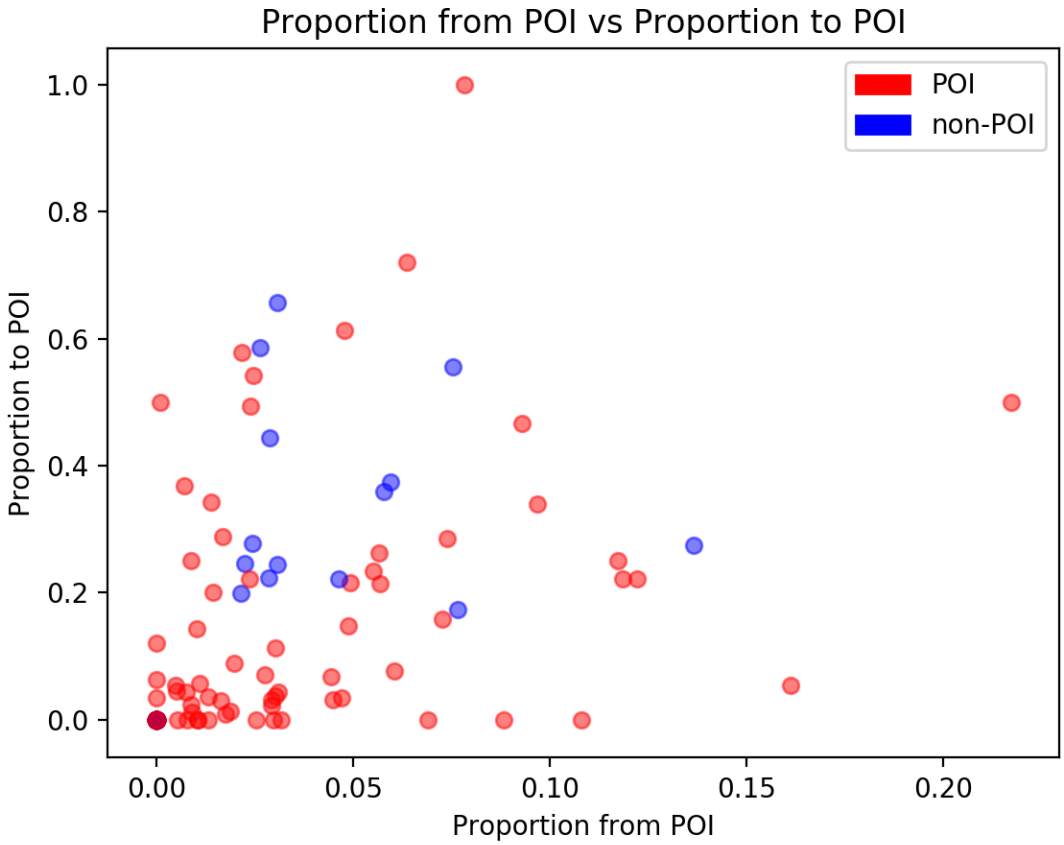

Figure 4 Proportion of messages from and to POI.

In an effort to reduce the number of features used, I had tried PCA. However, it didn't achieve the precision and recall that was required. Thus, I changed my code to select k-best features. The k was selected using a GridSearch. A table with the scores for the features of interest can be found on the next page. The top five features were "bonus", "total_stock_value", "exercised_stock_options", "propToPoi" and "propSharePoi".

Table 1 The scores of the features

Feature	Score
bonus	21.6365501
total_stock_value	16.3817584
exercised_stock_options	15.5592114
propToPoi	10.3428622
propSharePoi	10.2171679
expenses	9.24368744
restricted_stock	8.97772699
salary	8.68493399
from_poi_to_this_person	8.04063844
total_payments	7.04733788
shared_receipt_with_poi	6.74422004
deferred_income	6.452428
long_term_incentive	5.83379173
from_this_person_to_poi	5.28276107
propFromPoi	4.2126044
to_messages	1.15150303
deferral_payments	1.033863
restricted_stock_deferred	0.75016578
from_messages	0.01269847

Algorithm used

I explored a few algorithms – SVM, Decision Tree, Random Forest, KNeighbors Classifier, ExtraTree Classifier and AdaBoost Classifier. I ended up using the AdaBoost Classifier algorithm. The model performance was as follow for the various algorithms can be found below:

Table 2 Algorithm evaluation metrics

Algorithm	Class Label	Precision	Recall	F1 Score
SVM*	0	0.88	1	0.94
	1	0	0	0
Decision Tree	0	0.91	0.81	0.86
	1	0.22	0.4	0.29
Random Forest	0	0.9	0.97	0.94
	1	0.5	0.2	0.29
K-Neighbors	0	0.9	0.95	0.92
	1	0.33	0.2	0.25
ExtraTree Classifier	0	0.9	0.95	0.92
	1	0.33	0.2	0.25
AdaBoost Classifier	0	0.92	0.92	0.92
	1	0.4	0.4	0.4

*This model couldn't predict the POIs correctly. They resulted in warning that the metrics couldn't be estimated.

The recall was a higher for both POIs and non-POIs in AdaBoost Classifier when compared to the others and thus was selected.

Tuning parameters

The algorithm parameters play a critical role in the metrics of the resulting model as they can improve accuracy, but one also risks overfitting the data. I tuned the parameters for all the algorithms using the GridSearch.

For AdaBoost Classifier, I tuned the 'n_estimators' which passes the number of trees in the forest.

Validation Strategy

To reduce the risk of overfitting, one can validate the model using new unseen data. To validate my model, I split my data into training and testing sets using the StratifiedShuffleSplit. I used the stratified splitting as the classes of POI and non-POI are not equal in number. The model was built using only "training" subset (which was 70% of the data) and then applied to the "test" subset to ensure I do not overfit. I also ran the tester.py code provided to validate my results.

Evaluation Metrics

I looked at the precision and recall scores for evaluating my models. The precision of my model was 0.40 which mean 40% of the predicted POIs are POIs. While recall of my model was 0.4 which means 40% of the POIs have been predicted to be POIs.

References

https://www.investopedia.com/updates/enron-scandal-summary/
https://en.wikipedia.org/wiki/Enron
https://scikit-learn.org/stable/supervised_learning.html#supervised-learning