Project 5: Identify Fraud from Enron Email

By: Sarah Alabdulhadi

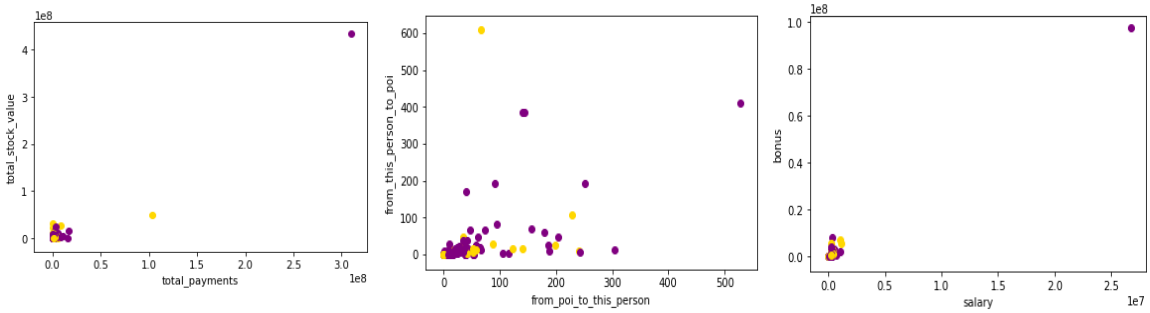
Introduction:

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.

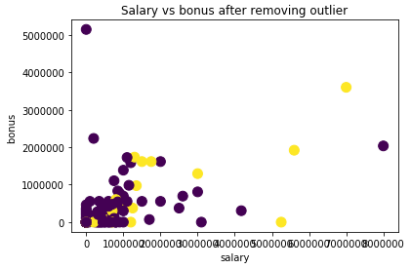
Questions:

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?**

The goal of the project is to use machine learning to build a predictive model to identify employees from Enron who may have committed fraud based on the public Enron financial and email dataset. Machine learning is a powerful prediction tool since it can process datasets faster than humans and it can see relevant trends that humans would have a hard time realizing manually.
The dataset contains 146 data points and 21 features. There are three different types of features 'financial features', 'email features' and 'POI labels'.
In the scatter plots below we can see that there are outliers that should be removed.

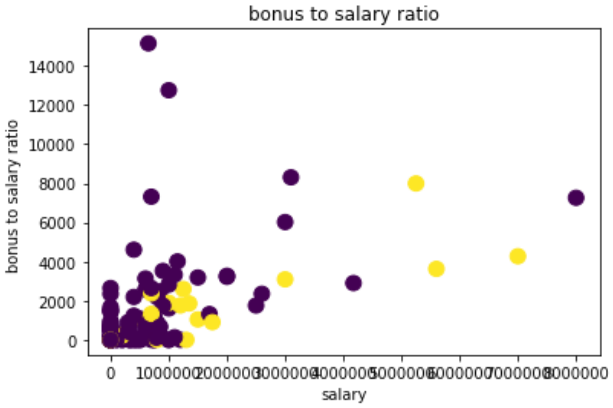


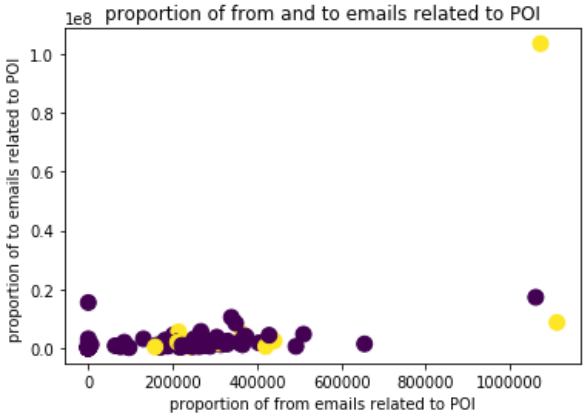
The three outliers I removed are 'TOTAL', 'THE TRAVEL AGENCY IN THE PARK' and 'LOCKHART EUGENE E'.

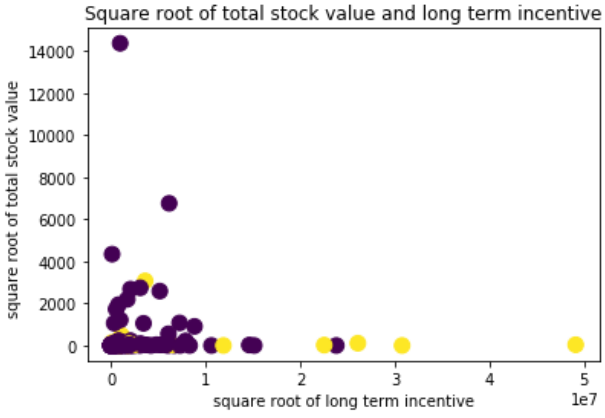


2. **What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what 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 importance 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.**

I started with the main features 'poi', 'from_messages', 'from_poi_to_this_person', 'from_this_person_to_poi', 'to_messages', 'bonus', 'long_term_incentive', 'salary', 'total_payments' and 'total_stock_value'. Then I added three other features "from_ratio", "to_ratio" and "bonus_salary_ratio" and these are the ratio of messages from POI to a person and form a person to POI and the ration of salary to bonus. After that I realized that the financial data might not work well with Naïve Bayes algorithm because it is largely skewed, and to solve this problem I created four other features with square root of 'salary', 'bonus', 'total_stock_value' and 'long_term_incentive'.







3. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?**

I tested three algorithms namely, Logistic Regression Classifier, K-means Clustering and Random Forest. I found that the best precision result is for Logistic Regression Classifier which is 0.36.

	precision	recall
LRC	0.36	0.28
K-means	0.23	0.37
RF	0.26	0.13

Therefore, I chose logistic regression classifier as the best algorithm for this data.

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? What parameters did you tune? (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).**

Parameters tuning refers to the adjustment of the algorithm when training, in order to improve the fit on the test set. Parameter can influence the outcome of the learning process, the more tuned the parameters, the more biased the algorithm will be to the training data & test harness. The strategy can be effective but it can also lead to more fragile models & overfit the test harness but don't perform well in practice. For my chosen algorithm I created the following parameters:

A. C → inverse regularization.
B. class weight → weights associated with classes.
C. max iteration → maximum number of iterations taken for the solvers to converge.
D. random_state → the seed of the pseudo random number generator to use when shuffling the data.
   solver → using 'liblinear' since we have very small dataset.

5. **What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?**

Validation comprises set of techniques to make sure our model generalizes with the remaining part of the dataset. A classic mistake, which was briefly mistaken by me, is over-fitting where the model performed well on training set but have substantial lower result on test set. In order to overcome such classic mistake, we can conduct cross-validation (provided by the evaluate function in poi_id.py where I start 1000 trials and divided the dataset into 3:1 training-to-test ratio. Main reason why we would use StratifiedSuffleSplit rather than other splitting techniques available is due to the nature of our dataset, which is extremely small with only 14 Persons of Interest. A single split into a training &

test set would not give a better estimate of error accuracy. Therefore, we need to randomly split the data into multiple trials while keeping the fraction of POIs in each trial relatively constant.

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.**

The two main evaluation metrics I used are precision and recall. I found that the best results were for logistic regression which was my final choice of model. As logistic regression is also widely used in text classification, we can actually extend this model for email classification if needed. Precision refer to the ratio of true positive (predicted as POI) to the records that are actually POI while recall described ratio of true positives to people flagged as POI. Essentially speaking, with a precision score of 0.386, it tells us if this model predicts 100 POIs, there would be 38 people are actually POIs and the rest 62 are innocent. With recall score of 0.4252, this model finds 42% of all real POIs in prediction. This model is amazingly perfect for finding bad guys without missing anyone, but with 42% probability for wrong.

With a precision score of 0.38, it tells us that if this model predicts 100 POIs, then the chance would be 38 people who are truly POIs and the rest 62 are innocent. On the other hand, with a recall score of 0.415, this model can find 42% of all real POIs in prediction. Due to the nature of the dataset, accuracy is not a good measurement as even if non-POI are all flagged, the accuracy score will yield that the model is a success.

References:

- Github page
- Udacity Introduction to machine learning
- Github page 2