Enron Report

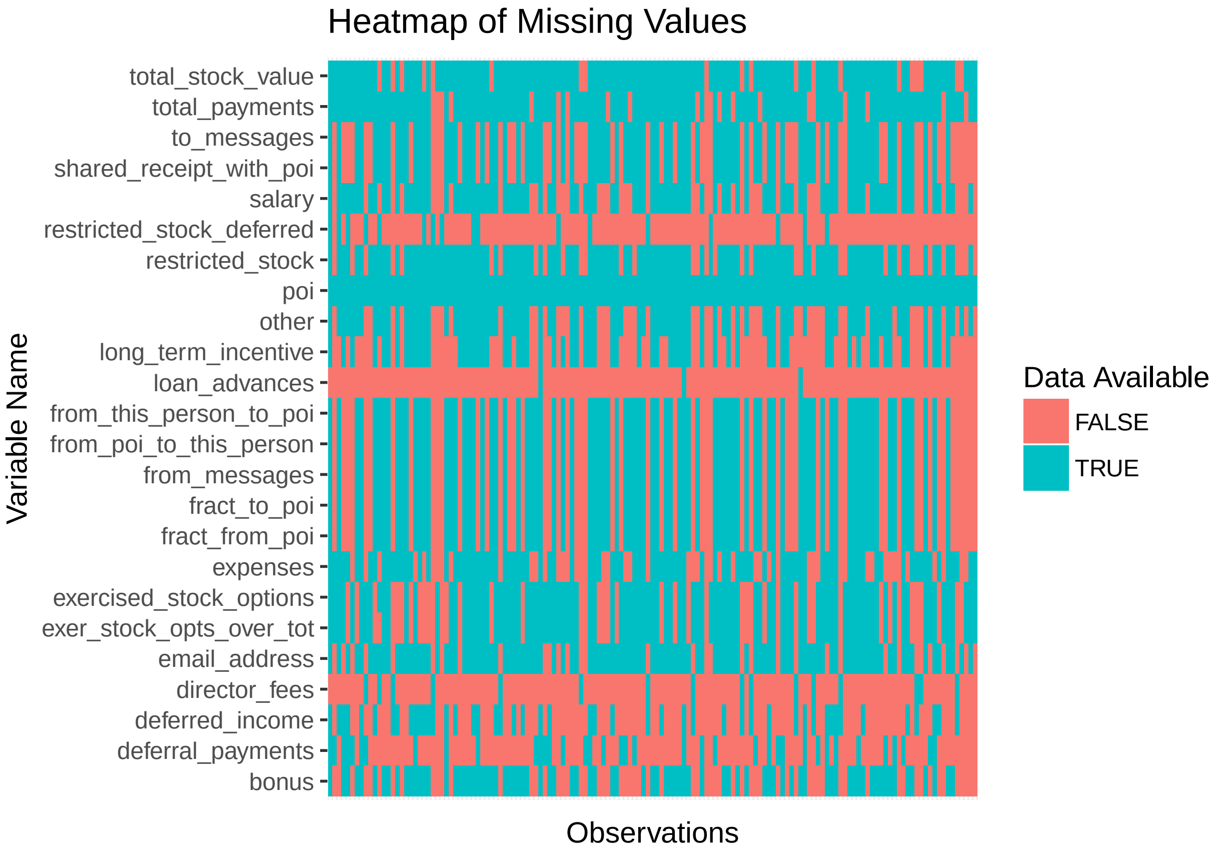
By: Courtney Ferguson Lee

June 29, 2017

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"]

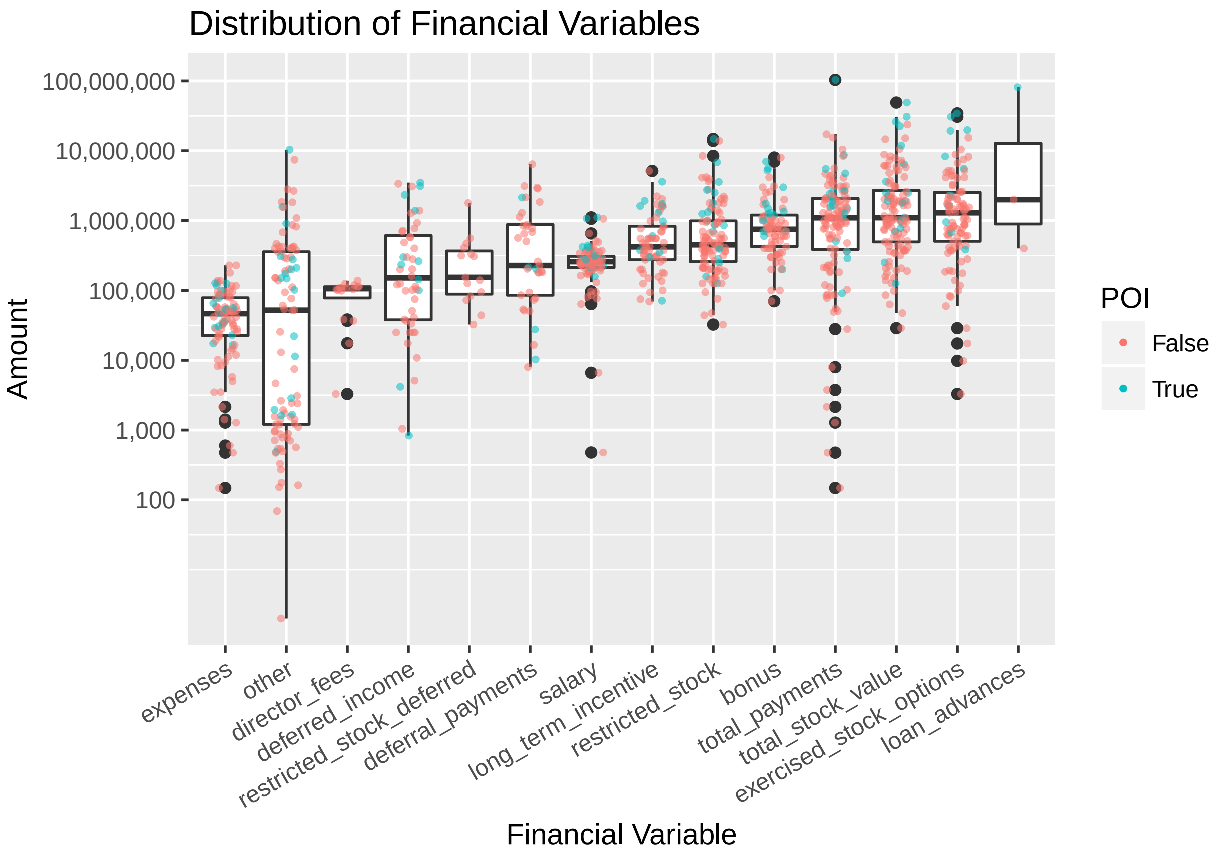
The goal for this project was to identify persons of interest (POIs) from a dataset of enron employees. The dataset contained both financial records and email records, as well as their POI status label. The financial information included salary, bonuses, stock options, expenses and payments. The email data included the total number of outgoing and incoming messages as well as the proportion of email interactions with known POIs. I used a supervised machine learning algorithm to identify patterns in the email and financial data that separated POIs from non-POIs.

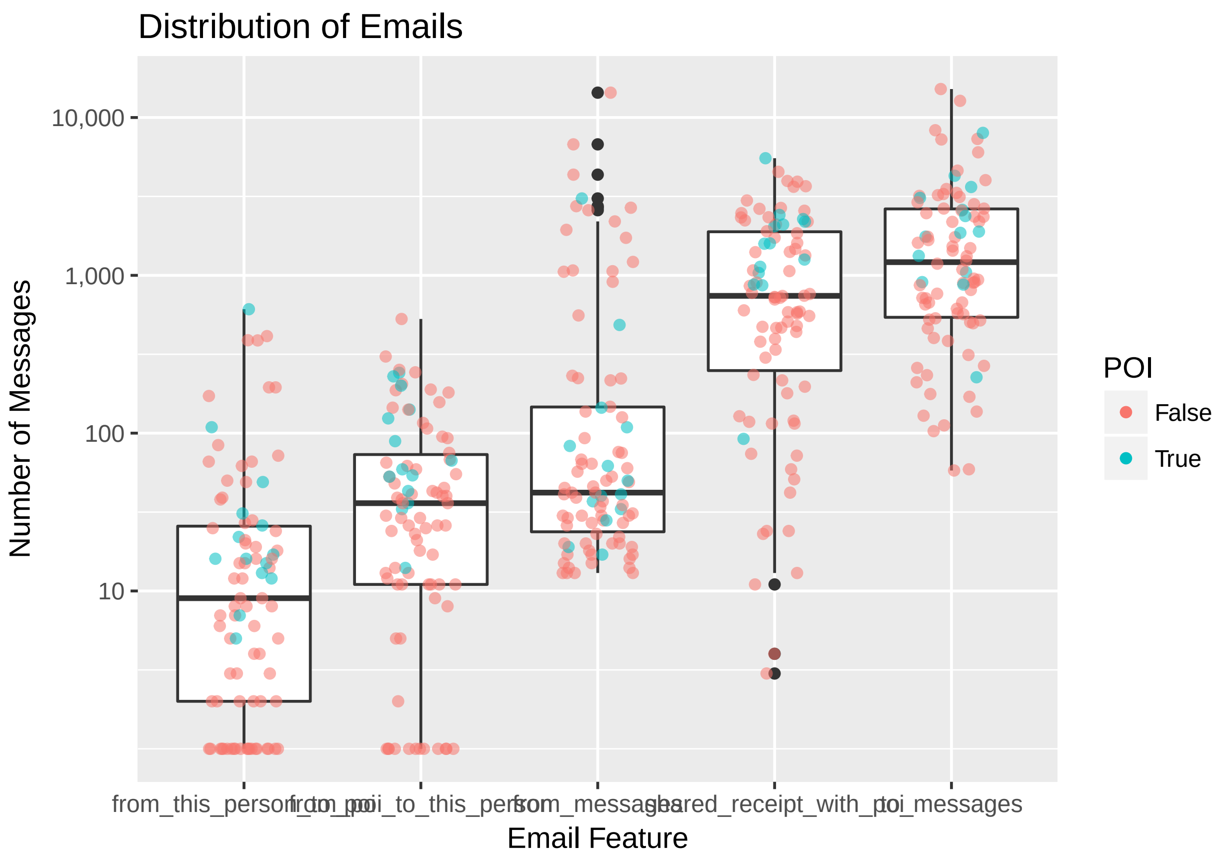
Two outliers in the dataset needed to be removed: "TOTAL" and "THE TRAVEL AGENCY IN THE PARK". Neither of these were actual people, so they did not contribute to the model. There were several financial and email outliers, but I ultimately included them because they contained useful information. Because of the nature of the dataset, it was difficult to remove any outliers that varied significantly from the rest of the population. For example, Ken Lay and Joseph Hirko had the largest exercised stock options, but they were both persons of interest. In addition, the dataset only consisted of less than 150 people and there was a lot of missing data, as shown in the following chart. With such a small number of observations, excluding any outliers would have had a significant impact on the model.

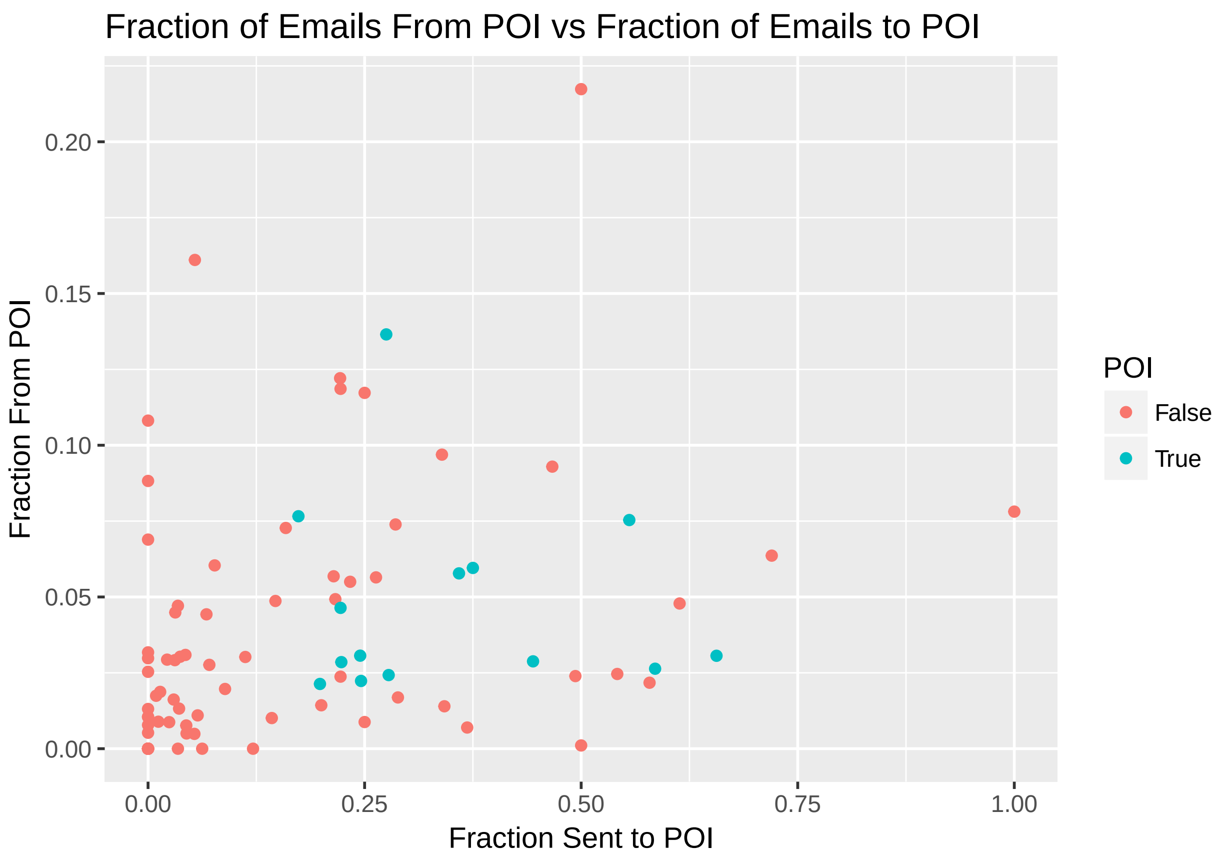


2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset – explain what feature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

I used the following features in my model: 'poi', 'fract_from_poi', 'fract_to_poi', 'from_messages', 'director_fees', 'restricted_stock_deferred', 'exercised_stock_options' and 'expenses'. I used Exploratory Data Analysis to identify features that would help create clear boundaries for my classifier. The plots below helped me understand the underlying nature of the financial and email variables:




Distribution of Emails
Number of Messages
10,000
1,000
100
10
from_this_person_to_poi
from_poi_to_this_person
from_messages
shared_receipt_with_poi
to_messages
Email Feature
POI
● False
● True



After identifying useful features, I scaled them before using PCA in my model. I scaled the features because the ranges of the variables were too large and would throw off my model. For instance, the range for fractional email variables ranged from 0.0 to 1.0 while the exercised stock options varied from 1,000 to around 30,000,000. With such a large range, PCA would not be able to properly determine which variable accounted for most of the variance in the data. I created my own MinMaxScaler and applied it right after creating new features. It ensured that all variables ranged from 0 to 1. PCA was a necessary feature selection step because it allowed the data to speak for itself in a way. It identified the variables that contributed the most to the classifier. The explained variance ratios from PCA and and features importances from DecisionTree were:

Explained Variance Ratios: [0.3306276, 0.21620042, 0.1924513, 0.10458867, 0.08131957, 0.07125716, 0.00355528]

Feature Importances: [0.06752799, 0., 0.14244186, 0.55146011, 0.20164066, 0.03692937, 0.]

I created two new features: the fraction of emails sent to POIs and the fraction of emails from POIs. I engineered these features because it did not make sense to use the raw number of emails to or from a POI. The total number of sent and received emails ranged from around 10 to over 10,000. To better compare human interactions, it made more sense to code this data as a proportion.

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”]

I used a DecisionTreeClassifier to categorize the observations. I started with a GaussianNaiveBayes to get a benchmark, then tried a few other algorithms before settling on the Decision Tree. I tested out the KNearest Neighbors Classifier, but it had the worst results overall at only .22 for its precision, recall and F1 scores. I then tried a Support Vector Machines model but it was slower than the others and didn’t improve the F1 scores much. I also tried out an ensemble method (AdaBoost) but it was both slower and did not improve evaluation scores much. At that point, I decided to try the DecisionTree Classifier and added PCA to transform the variables. This performed the best overall in terms of both evaluation metrics and speed.

4. What does it mean to tune the parameters of an algorithm, and what can happenif you don't do this well? How did you tune the parameters of your particularalgorithm? 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). [relevant rubric items: "discuss parameter tuning", "tune the algorithm"]

Tuning the parameters of an algorithm involves reading the sklearn documentation and trying out different combinations of input arguments. For instance, when I settled on a DecisionTree Classifier, I played around with the max_depth, minimal_samples_split, and criterion. We calculated entropy for various scenarios in the lessons so I tried using that as an alternative criterion, but the results didn't change much from baseline. I found the greatest variation in F1 scores when I changed the PCA parameters. When I did not specify a random_state, the scores kept bouncing all over the place. I discovered that this was an important step to ensure that others could reproduce my results.

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 items: "discuss validation", "validation strategy"]

Validation involves ensuring your model is effective on both training and testing data. One pitfall in validation is when you accidentally overfit your model to the training data. Overfitting is when your model performs well on training data but poorly on new testing data. This can happen if you don't partition a large enough chunk for your testing set or when you use an overly-convoluted boundary when training.

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"]

My average precision score was around .34 and my average recall was around .38- .39. Precision compares the number of false negatives to true positives in the results. In other words, it asks, "Out of all of the items that are truly positive, how many were correctly labeled as positive?" Recall measures the number of false positives to true positives. It asks, "Out of all of the items labeled as positive, how many truly belong to the positive class?" My results indicate that my algorith was better at lowering the number of false positives than it was at weeding out false negatives.