Enron Submission - Alexandre Shimono

A critical part of machine learning is making sense of your analysis process and communicating it to others. The questions below will help us understand your decision-making process and allow us to give feedback on your project. Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response! When your evaluator looks at your responses, he or she will use a specific list of rubric items to assess your answers. Here is the link to that rubric: [**Link**] Each question has one or more specific rubric items associated with it, so before you submit an answer, take a look at that part of the rubric. If your response does not meet expectations for all rubric points, you will be asked to revise and resubmit your project. Make sure that your responses are detailed enough that the evaluator will be able to understand the steps you took and your thought processes as you went through the data analysis.

Once you've submitted your responses, your coach will take a look and may ask a few more focused follow-up questions on one or more of your answers.

We can't wait to see what you've put together for this project!

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 objective of the project is to predict if a given person is part of a group of interest of suspects of commiting fraud in a famous case of a company called Enron. We are given many possible features to use such as financial (salary, bonuses, etc) and social interactions (such email exchanges and shared receipts), and are expected to predict in a group of individuals which ones could be involved in the scheme.

The first steps when analyzing the data was to take a look at the dataset overall. There are 145 entries, from which 18 are positive POIs. I selected most of the features and checked which are non zeroes and what percentage of the 145 entries that represent:

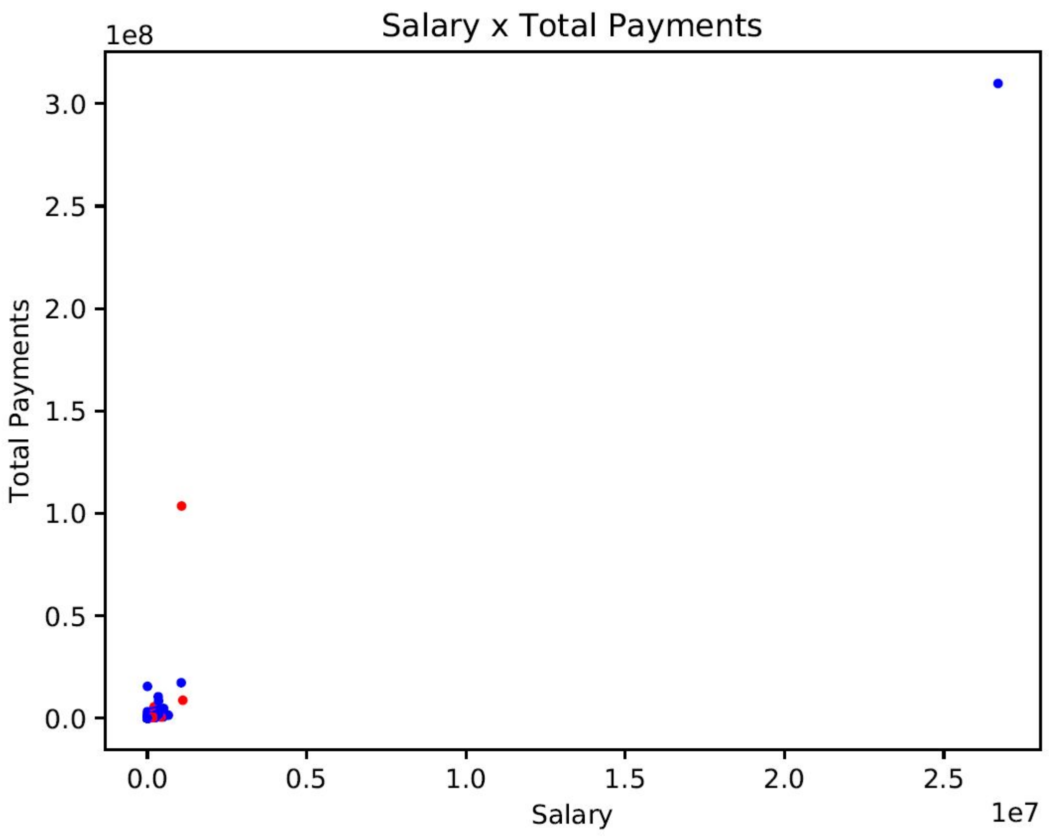
Name	*Count non zeroes*	*Percentage of non zeroes*
salary	*95*	*65.5172413793%*
total_payments	*125*	*86.2068965517%*
bonus	*82*	*56.5517241379%*
exercised_stock_options	*102*	*70.3448275862%*
long_term_incentive	*66*	*45.5172413793%*
restricted_stock	*110*	*75.8620689655%*
director_fees	*17*	*11.724137931%*
from_poi_to_this_person	*74*	*51.0344827586%*
from_this_person_to_poi	*66*	*45.5172413793%*
shared_receipt_with_poi	*86*	*59.3103448276%*
to_messages	*86*	*59.3103448276%*

from_messages	*86*	*59.3103448276%*
expenses	*95*	*65.5172413793%*
other	*93*	*64.1379310345%*

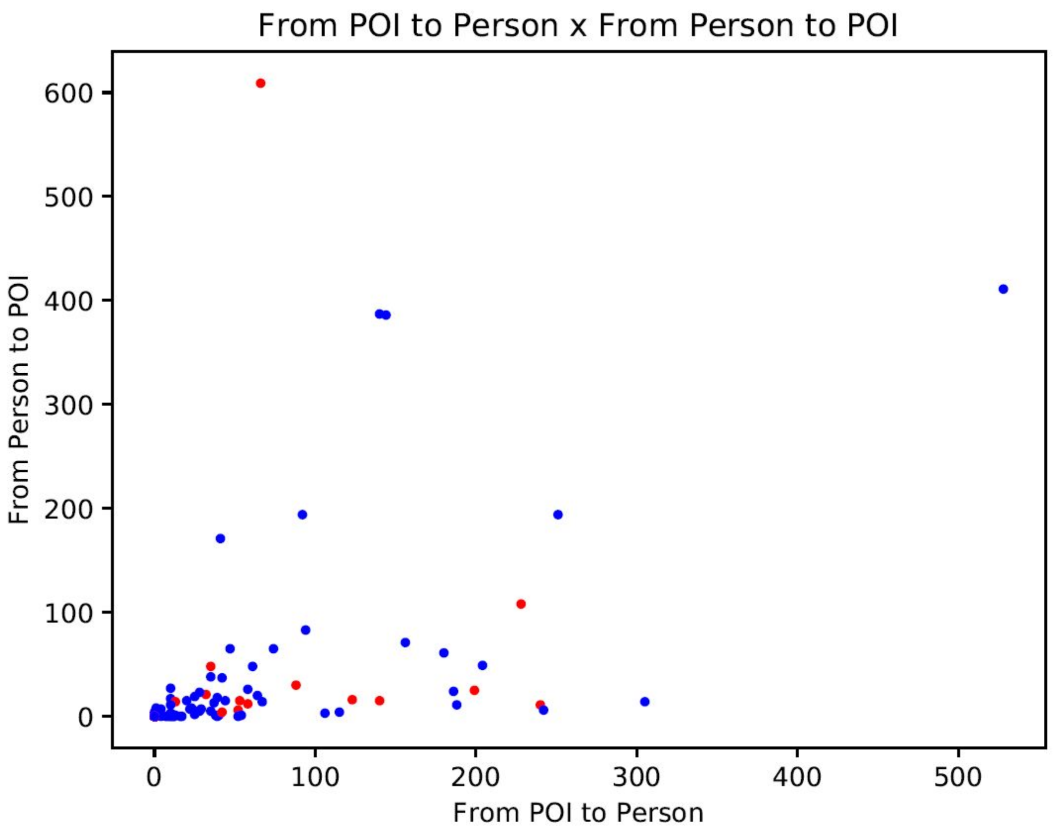
After that, I rant he Lasso tool from sklearn to determine how much influence each feature has on the labels:

```
[ -1.45127819e-07   6.47593124e-09  -2.38642361e-08   2.25498067e-08
   2.03737161e-08   5.06388695e-09  -1.28738886e-06  -1.00028528e-04
   5.87189229e-04   2.13469305e-04  -8.11863318e-05  -8.17908593e-06
   1.05943512e-08  -6.47615246e-08]
```

Then, I ran a graphical analysis of many different dimensions in order to visually address this issue. Here is a sample of what I observed, full file can be found on the zip file as first_image_analysis.pdf:



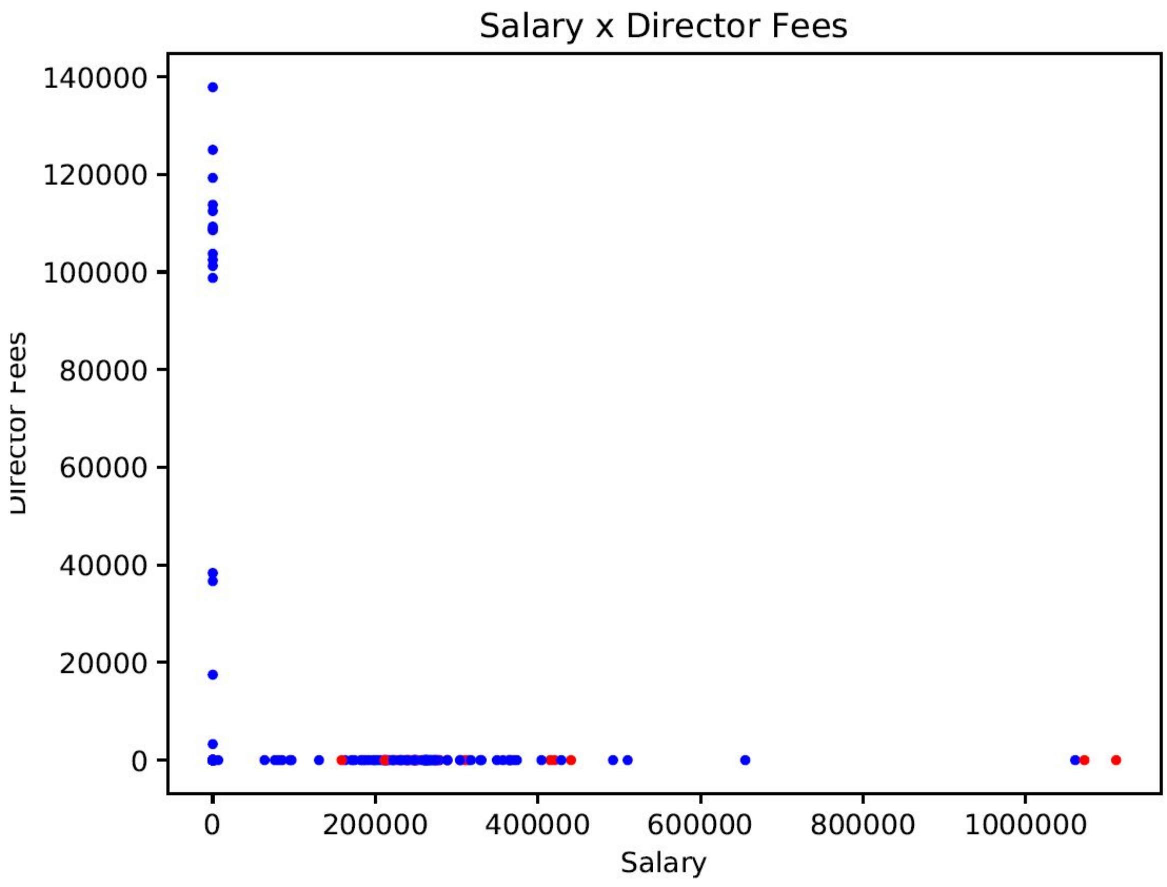
We can see clearly an outlier on the top-right corner. Also there was another one on the from_poi_to_person emails:



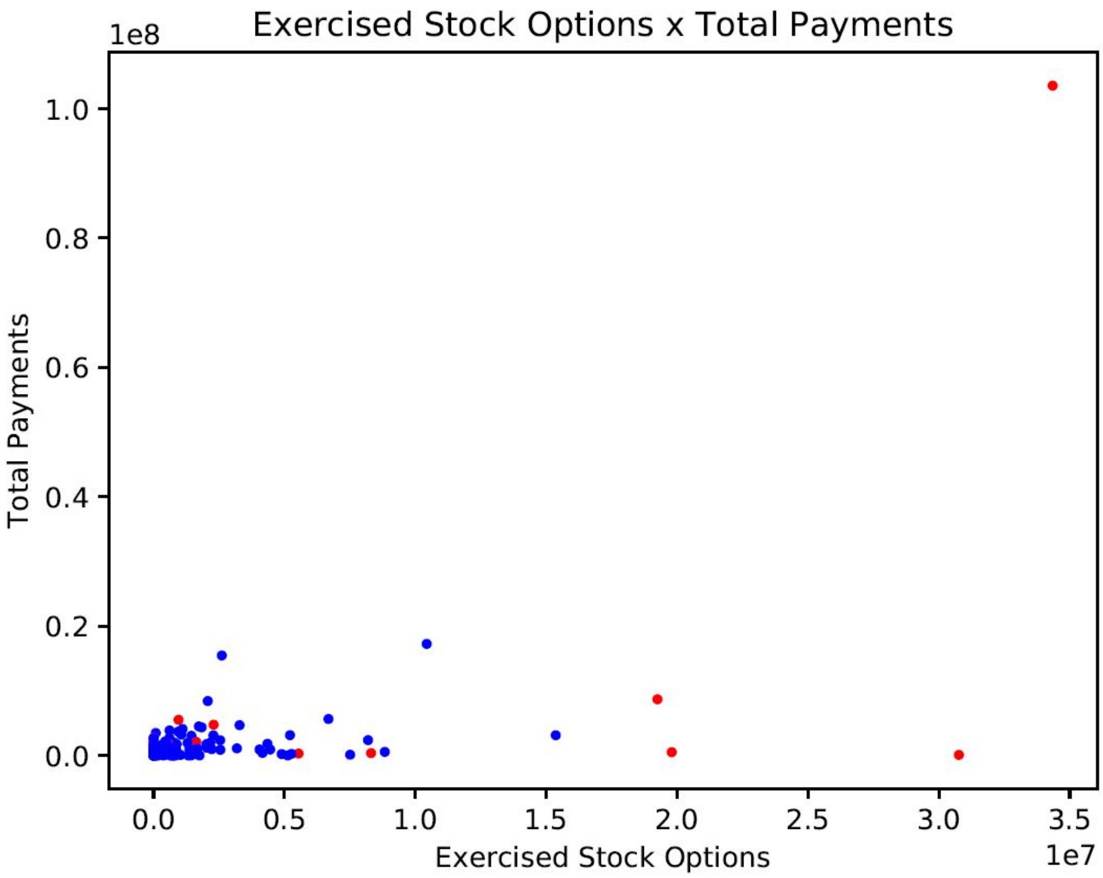
The right-most item was removed and considered an outlier as well. So, 2 entries were removed: one named TOTAL, which had very large values for financial features, and another for the emails from JOHN J LAVORATO.

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"]

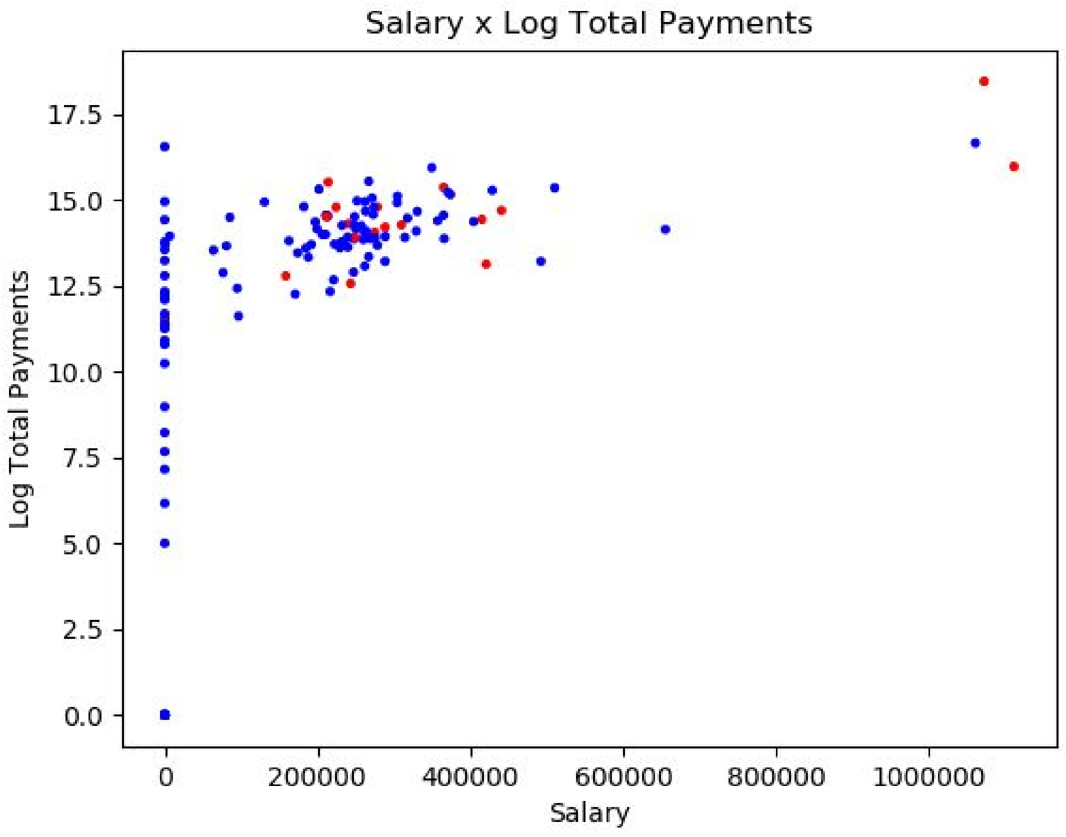
After running the image analysis for the second time(file attached as second_image_analysis.pdf), there was an improvement on the features. The next step was to run the Lasso tool from sklearn and verify the coef of each feature to have an estimate how much is it´s influence on the labels. After this step, I noticed some features had a coef as bad as e-9 and the best ones were of the order of magnitude of e-4. Other step was to plot graphs of money and emails to visualize how they are correlated. One interesting *example that was not intuitive: director_fees was one of the highest coef, but has a very weird graph. Turns out no POI has director_fee greater then zero:*



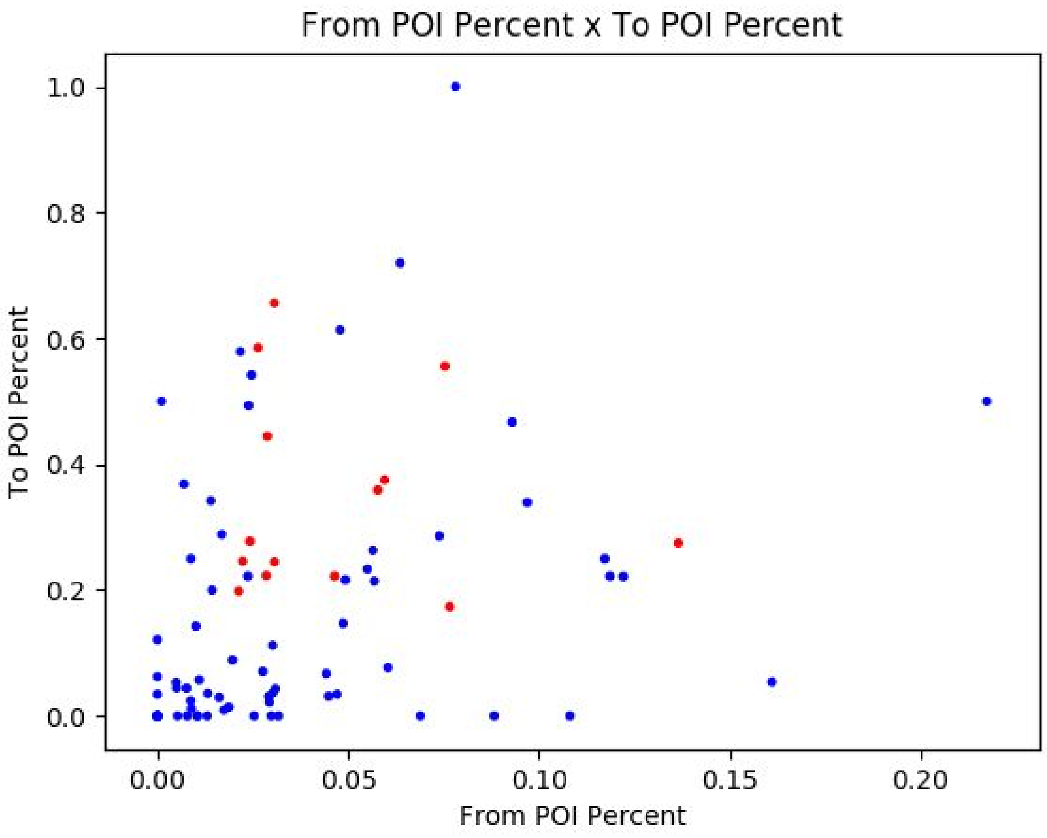
One dimension that needed scaling and was handled as a new feature was total_payments: one POI has a very high value and was not interesting to be removed.



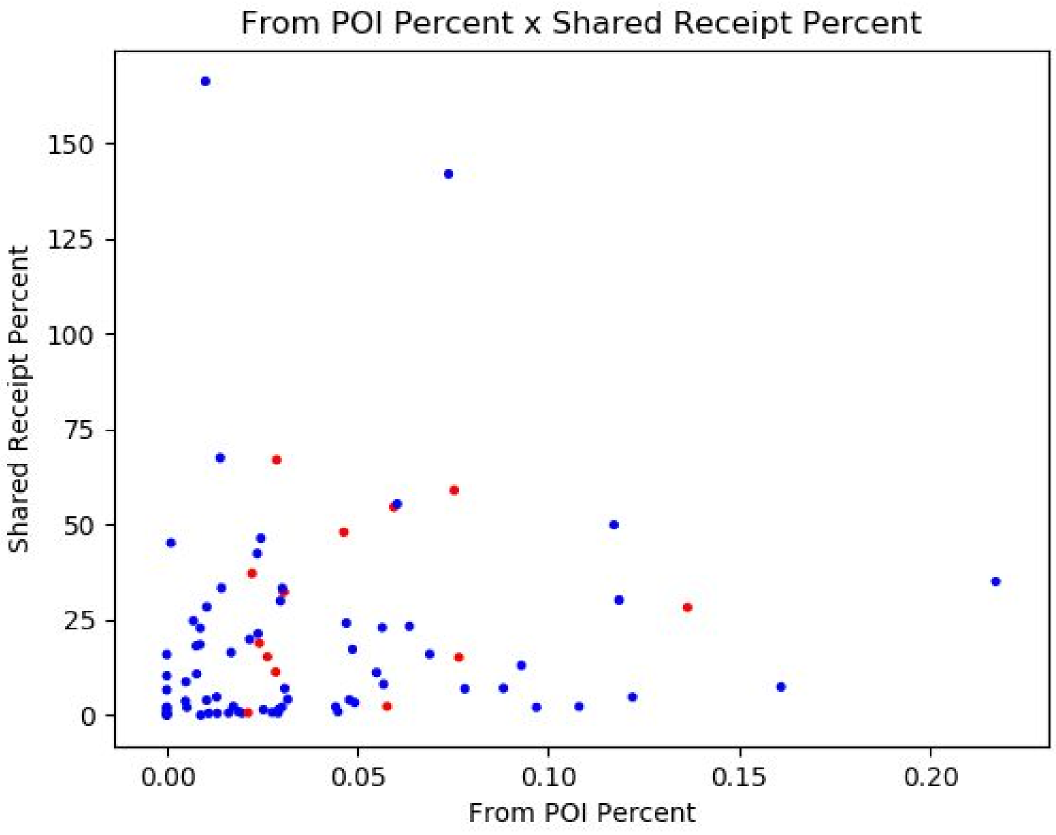
To do the scaling, the log function was applied to total_payments:

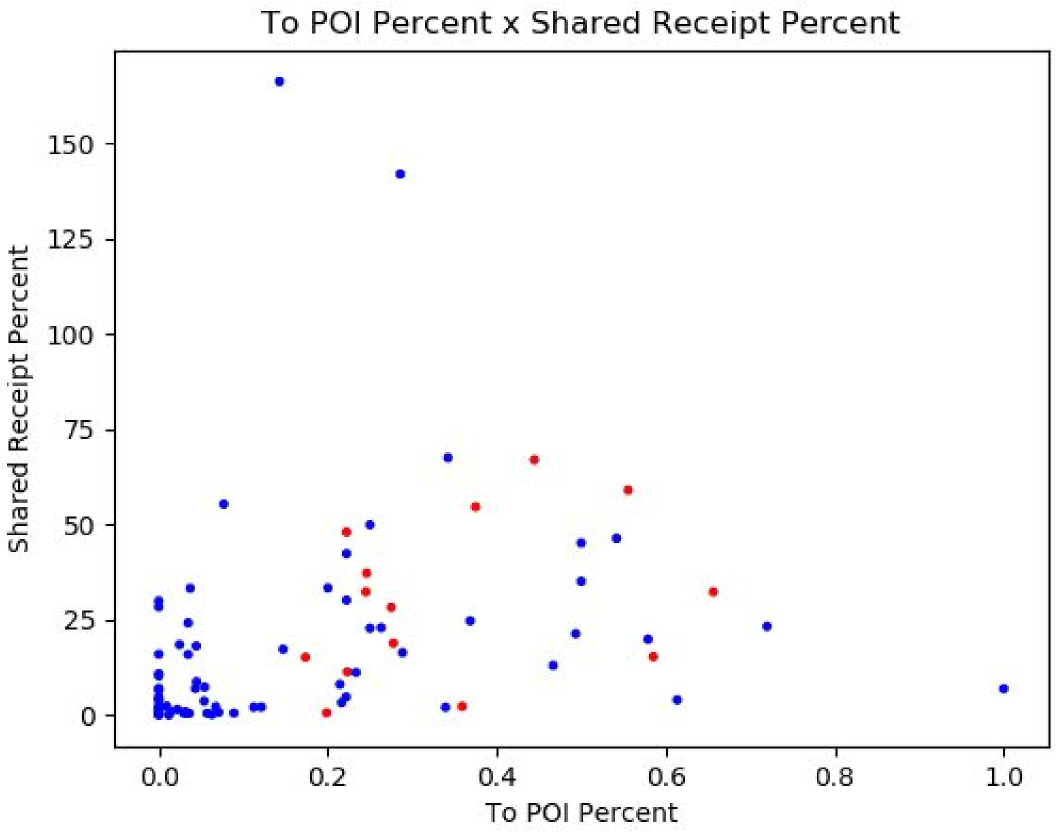


The first features created were those suggested on the video classes: the fraction of emails from and to POIs from all emails for each subject.

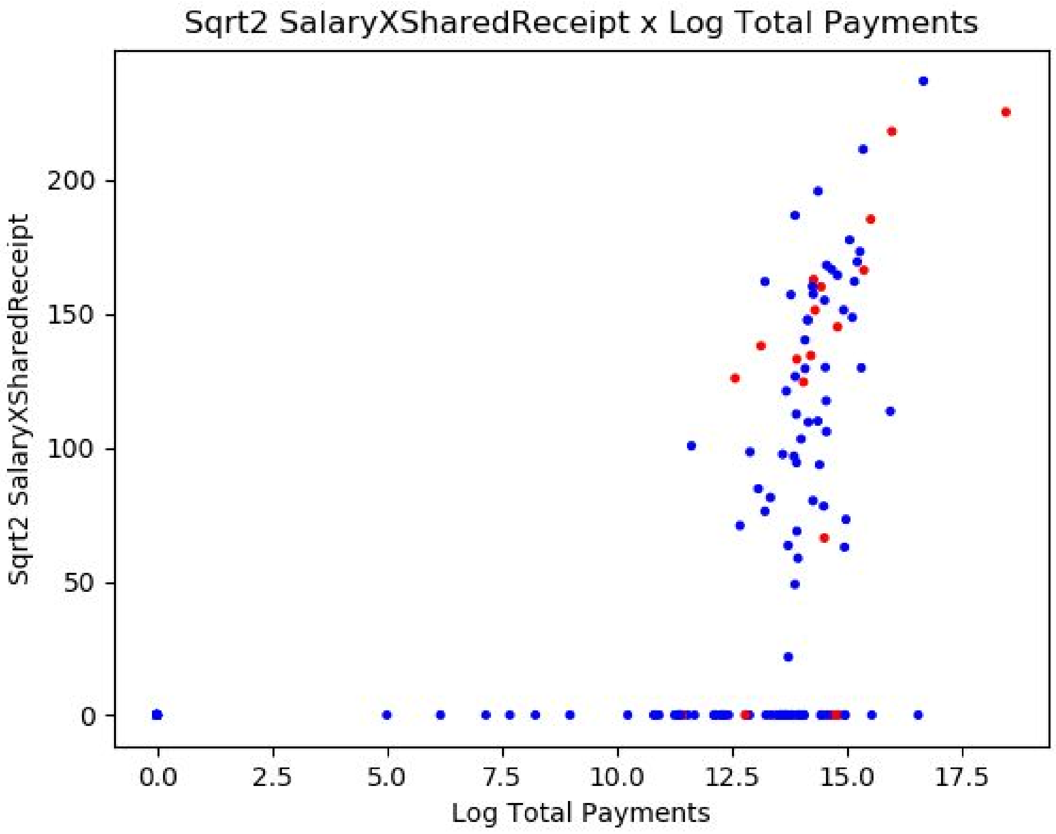


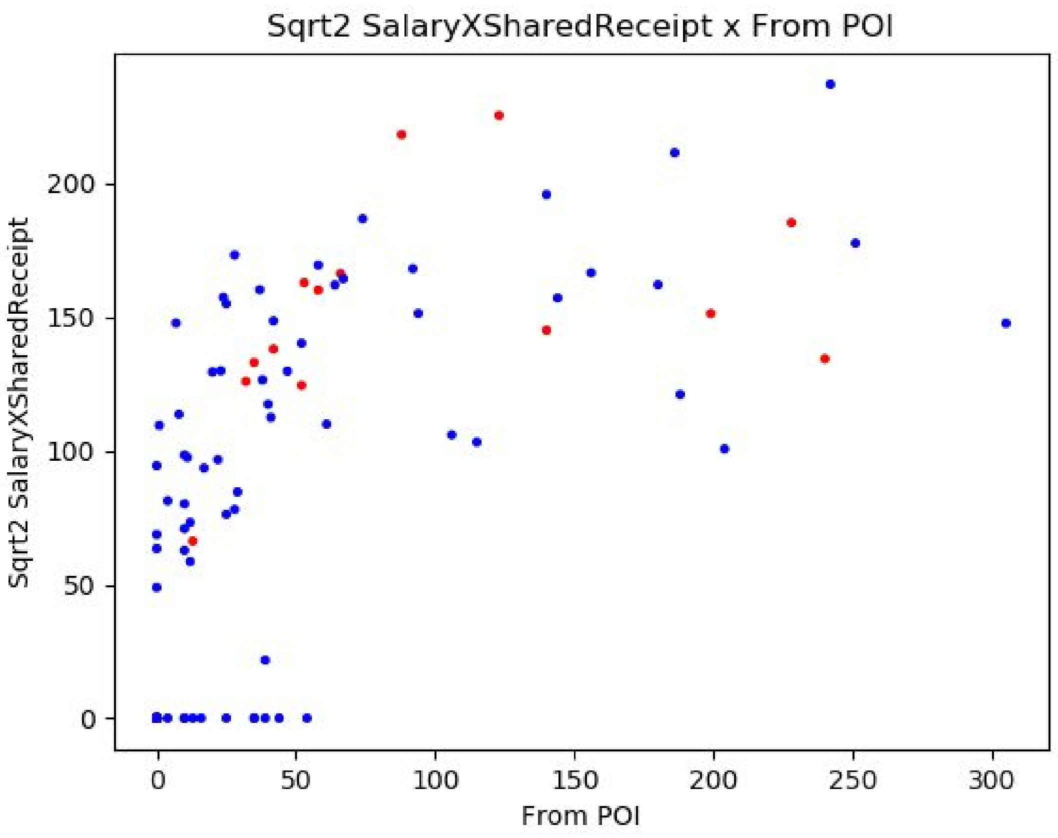
Following the same logic, I tried to produce another one dividing shared_receipt by to_messages, but results were not good.

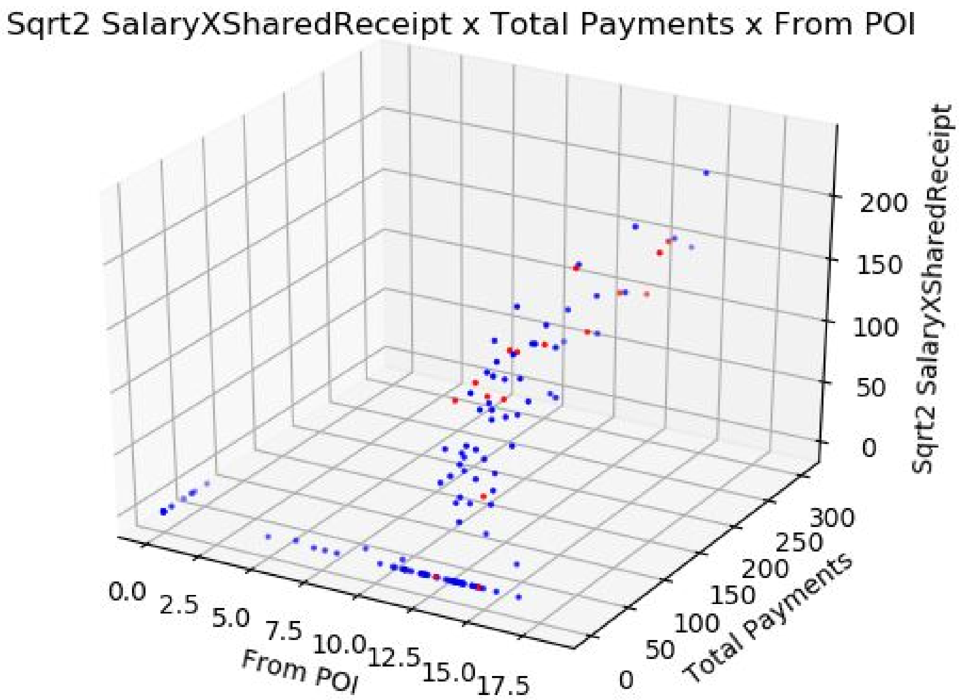

From POI Percent x Shared Receipt Percent
Shared Receipt Percent
From POI Percent
0
25
50
75
100
125
150
0.00
0.05
0.10
0.15
0.20


To POI Percent x Shared Receipt Percent
Shared Receipt Percent
To POI Percent
0
25
50
75
100
125
150
0.0
0.2
0.4
0.6
0.8
1.0

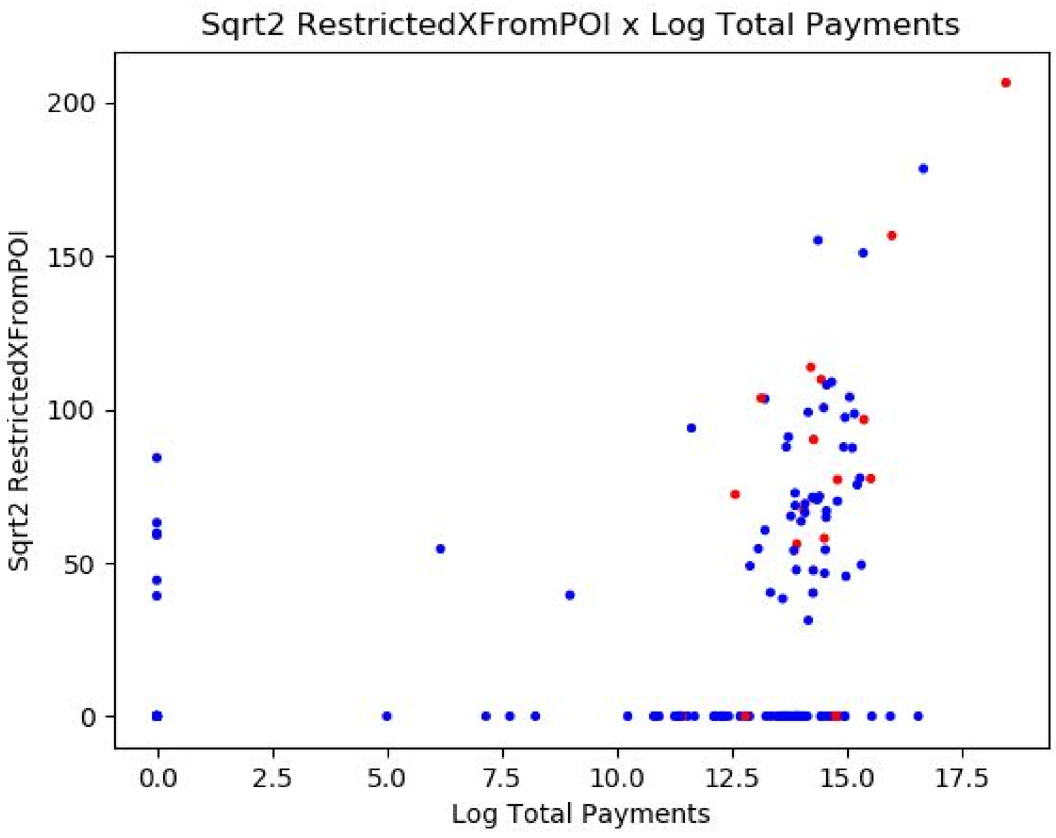
Next attempt was to pick one financial feature that already have a high coef and mix it with some email feature. I tried to multiply salary and shared_receipt and scale it applying a sqrt twice, and then check what the graph looks like when comparing to the log of total_payments and another email feature:



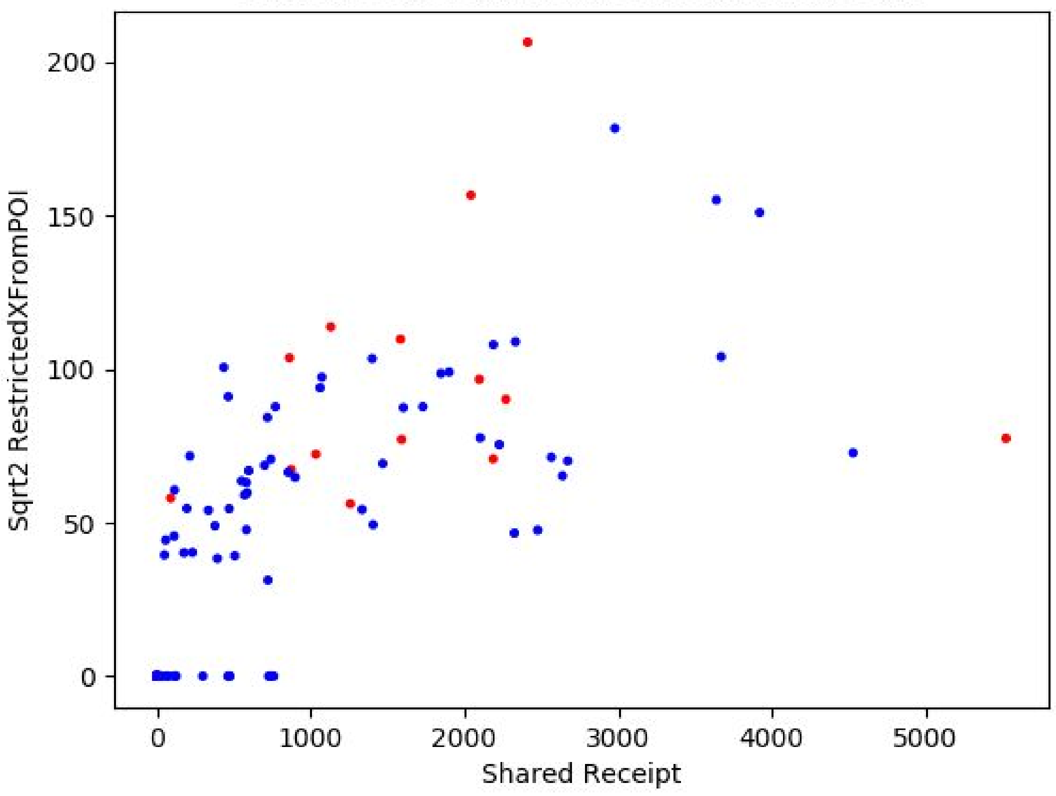

Sqrt2 SalaryXSharedReceipt x From POI
Sqrt2 SalaryXSharedReceipt
From POI
0
50
100
150
200
250
300


Sqrt2 SalaryXSharedReceipt x Total Payments x From POI
Sqrt2 SalaryXSharedReceipt
Total Payments
From POI
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

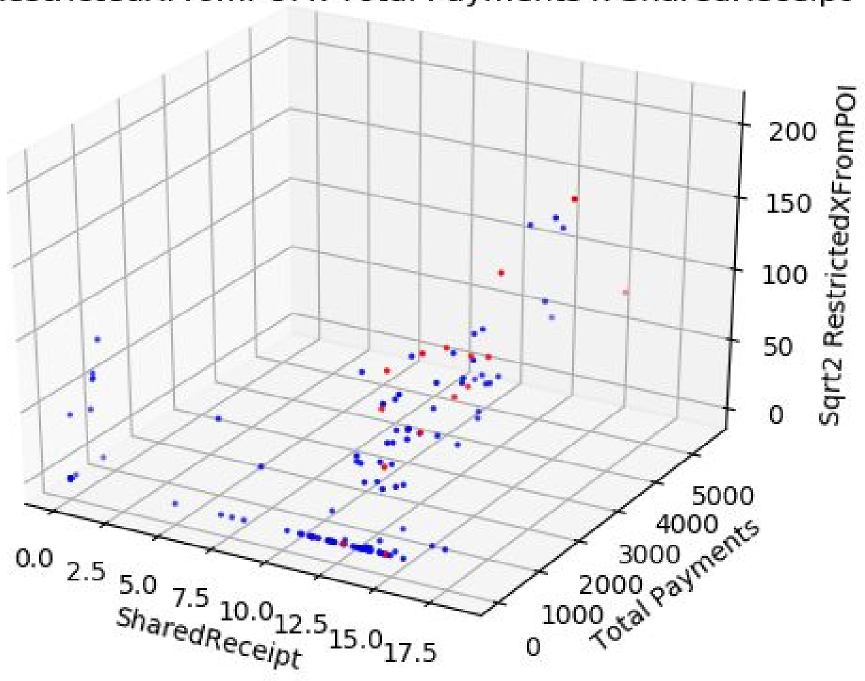
So, graphically, it seems a good new feature. Another try is to multiply restricted_stock by from_poi:



Sqrt2 RestrictedXFromPOI x Shared Receipt



Sqrt2 RestrictedXFromPOI x Total Payments x SharedReceipt



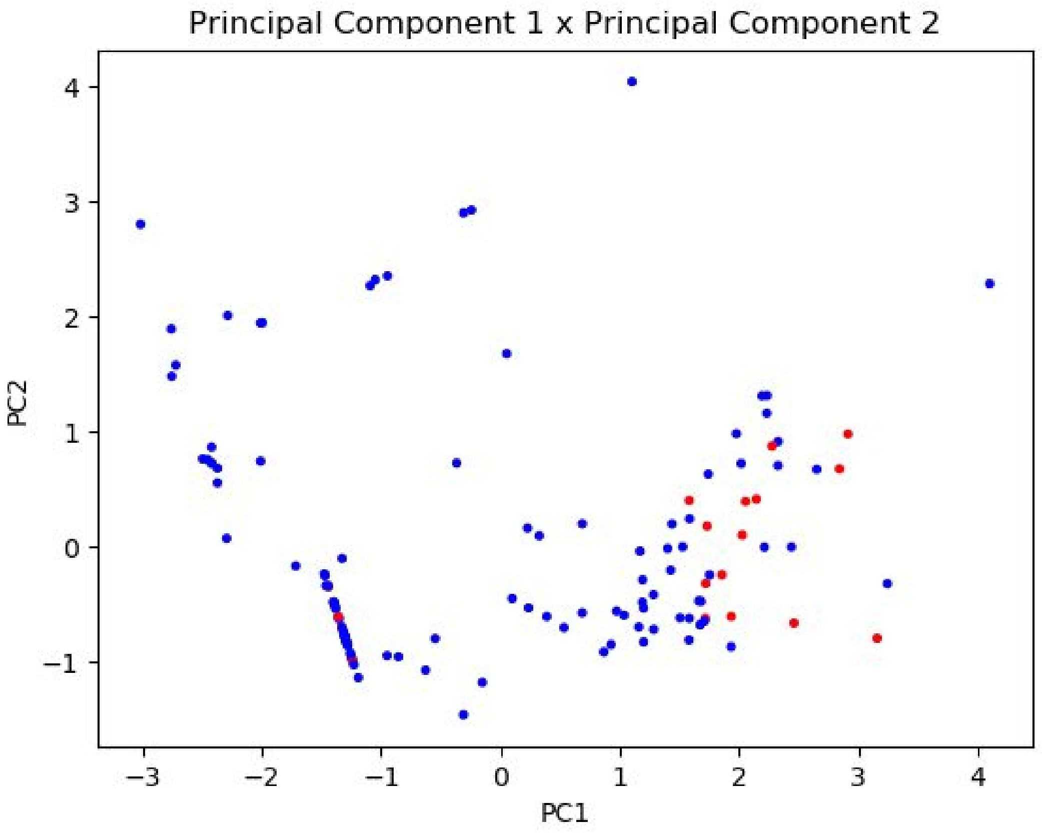
This one also appears to have a nice result graphically. I ran the same function to count number of non zeroes and check Lasso coefs for the new features and surprisingly, the percentages of to_poi and from_poi got 0, and the the other new features all had good coefs comparing to the original features. For this reason, I replaced many of the original features with the new features.

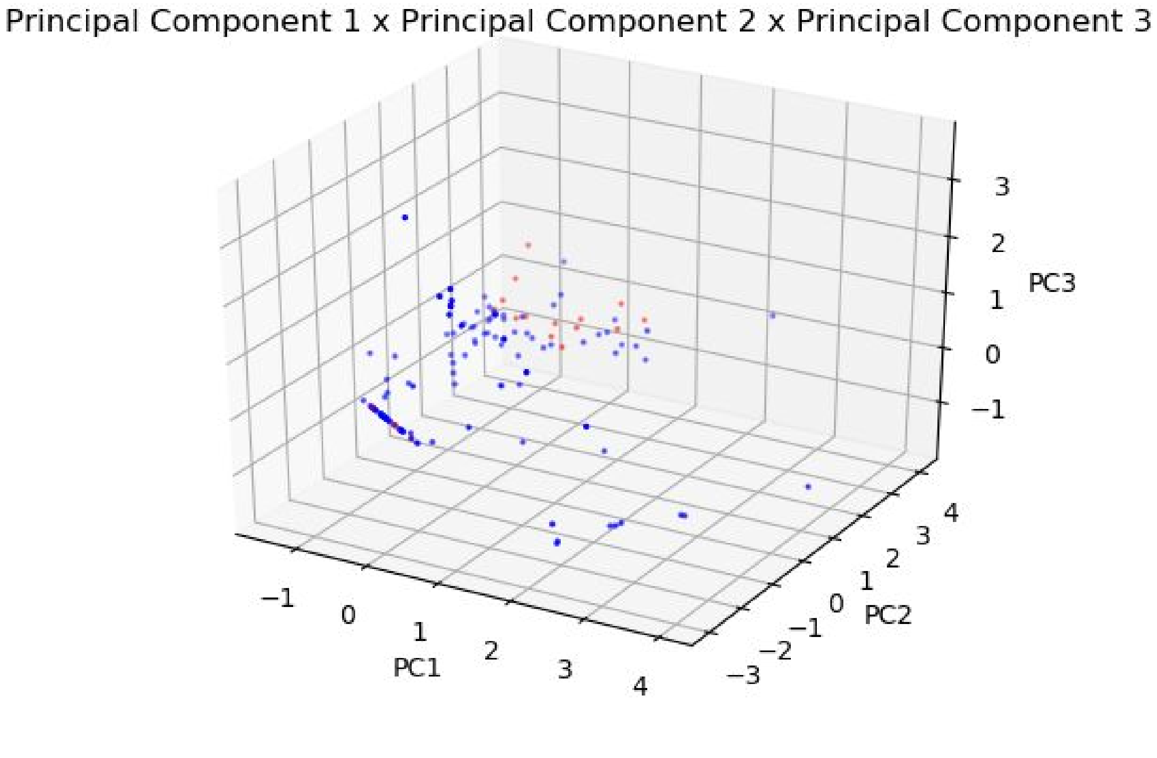
Name	*Total Count*	*Percentage*
from_poi_percent	*73*	*51.048951049%*
to_poi_percent	*65*	*45.4545454545%*
shared_receipt_percent	*85*	*59.4405594406%*
log_total_payments	*123*	*86.013986014%*
salaryXreceipt	*66*	*46.1538461538%*
restrictedXfrom_poi	*66*	*46.1538461538%*

```
Coef:
[-0.         0.         0.0002659   0.00202666  0.00052639  0.00152021]
```

To determine which features I should use, I picked those with highest coef values plus the to_poi_percent and from_poi_percent which graphically had a nice correlation with POIs. Also, if a feature was rescaled or combine, there was no sense in using the original feature. So the final list was: ['poi','director_fees','from_poi_percent','to_poi_percent', 'restrictedXfrom_poi','log_total_payments','salaryXreceipt']

As a last step, I tried to apply PCA to see what results I would get. I had to apply scaling before PCA since it is affected by scale, according to this source: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60 . They were pretty good, specially the one with 3 dimensions:


Principal Component 1 x Principal Component 2
PC2
PC1
4
3
2
1
0
−1
−3
−2
−1
0
1
2
3
4


Principal Component 1 x Principal Component 2 x Principal Component 3
PC3
3
2
1
0
−1
PC1
−1
0
1
2
3
4
PC2
−3
−2
−1
0
1
2
3
4

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 tried to use Gaussian Naive Bayes and SVM. In the end I used SVM, with RBF kernel. They had a similar performance on the tester function, but SVM performed slightly better on the tester function. I also tried using 2 dimension PCA and 3 dimension PCA, and the best result was 3 dimension.

***GaussianNB**: Accuracy: 0.79450 Precision: 0.30776 Recall: 0.35100 F1: 0.32796 F2: 0.34141*
***SVM**: Accuracy: 0.84214 Precision: 0.43675 Recall: 0.36250 F1: 0.39617 F2: 0.37526*

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). [relevant rubric items: “discuss parameter tuning”, “tune the algorithm”]

Parameter tuning is an important step in order to get the optimal results for a given algorithm and I did it using the GridSearchCV class for the SVM. If we don´t tune our algorithm properly performance decrease drastically: as an example, I am creating another instance of the SVC to be stored because the GridSearchCV object gets really slow if used directly on the tester.py class. Sometimes whenever I did some change on the code, like if I changed the features, the number of dimensions of the PCA, etc, the best parameter configuration for the SVM would change (gamma and C values) but if I forgot to update the values on the clf object, the results for the SVM with the old values of C and gamma would often be worse than the GaussianNB.

I also tried to use other kernels with the SVM, but did not notice a significant improvement. Just was unable to get a good result using the linear kernel, and had a problem with computing time using the polynomial kernel. The GridSearchCV was useful to get the best gamma and C parameters for the SVM, which were tested on the range of 0.0000001 to 10000 for the C and 0.000000001 to 1000 for the gamma.

As for the Gaussian Naive Bayes, there were no parameters to be tuned.

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”]

Cross-validation is the technique of splitting our data into a group used to train our model and another to test it. It is important because by doing so we avoid our model to *overfit our data, which means that our model is over specialized to the data presented to us but will perform poorly in a more general case, which is generally what we are looking for.*

The tester python script provided uses cross-validation to perform its analysis, and since this was our final evaluation I ended up using it directly. It uses the StratifiedShuffleSplit class with 1000 folds.

Stratified Cross-validation was used in this case because the classes (POI and non-POI) are clearly unbalanced: we have only 18 POIs in our dataset and 143 non-POIs after the removal of outliers. For this reason, when splitting the dataset into smaller sets for training and testing, there is a big chance many of those subsets will not keep this proportion and will likely have no POI at all. Stratified cross-validation assured this proportion of POI and non-POI will be kept during the process.

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"]

For the SVM we have: Precision: 0.43675 Recall: 0.36250.

The Precision measure is defined as the number of True Positives divided by the sum of True Positives and False Positives. This means that in order to get a higher number, we must have a greater number of true positives and a smaller number of false positives. The highest possible value is 1, which would happen if we could get all true positives and 0 false positives, and the lowest happens when we get 0 true positives and a greater than 0 number of false positives. The idea behind it is to know out of all the results the algorithm identifies as positive, which are correctly classified.

The Recall is defined as the number of True Positives divided by sum of True Positives and False Negatives. So, in order to get a high number we have to get a high number of True Positives and a low False Negatives. The idea of this measurement is that it represents the fraction of correct results it is returning. This is tricky because if the algorithm identified all subjects as positives always, we would have Recall 1 and Precision 0.

On this project, a True Positive means the classifier predicted an entry to be a POI and the person was indeed a POI. A False Positive would be when the classifier predicted an entry to be a POI but it was not in fact a POI. A True Negative means a person is not a POI and the classifier did not classified that person as POI. A False Negative means a person was classified as POI but in fact was not a POI.