1. I will try to identify who are the people from Enron that were involved in illegal activities (persons of interest, or POI). I will use multiple machine learning techniques to preprocess the data, choose and train an ML algorithm, and validate it at the end.
   First, I have split the features into those related to money, and those about the count of e-mails. Next, I have plotted each feature against all others in a group, just get some overview of the data and see if there are some groupings and how the data is spread across the features (I have plotted POI data differently than regular data).
   The first outlier I noticed was TOTAL, and I have removed it from data and plotted all plots without it.

2. I tried using PCA just to get an overview of how many features could be important. Then I started manually removing features to see how it influences the performance.
   I did not do any scaling as I ended up using Decision Tree..
   I created two new features, but did not use them in my classifier as they did not improve the score. They were 'to_poi_percent' and 'from_poi_percent', two features that represent the percent of all emails that were sent to/from POI. Plotting these two against each other gave very nice plot, with all POI points inside a square.
   I ended up using only 3 features:
   a. 'other' - 0.676191012772
   b. 'total_stock_value' - 0.323808987228
   c. 'expenses' - 0.0
3. I ended up using DecisionTree, but I have tried GaussianNB and SVC with different parameter variations. With default parameters they were almost the same and almost all with recall and precision around 0.3, but none could make both of them go above 0.3.
4. But changing 'min_samples_split' put DecisionTree ahead of others. Further, I tried different variations of parameters ('criterion', 'max_features', 'random_state'), and finally made a breakthrough when I set "class_weight = 'balanced'". That significantly boosted the performance, and I ended up using "min_samples_split = 50". Everything else was left to default.
5. Validation is the key step to algorithm development. It basically gives you an answer if your algorithm is usable or not, and what for. One very important step is to always test your model on different data that was used for training. Also, keep a close watch for overfitting, if your algorithm is doing too good, it's probably some kind of overfitting, or there's an feature that holds all the information.
   I have used precision and recall to validate my algorithm. Recall as the measure of how good will the algorithm do if the test point is actually a POI. Precision measures what is the fraction of the POI's from all the points that algorithm predicts as POI.
6. Accuracy: 0.77000, Precision: 0.37087, Recall: 0.87600, F1: 0.52112, F2: 0.68846
   The goal of my algorithm was to capture all possible POI's, at the expense of maybe marking some non-POI as POI. This is reflected in the Recall, as it measures if I will be able to recognize a POI if it was given one. The algorithm does this pretty well 87.6%.
   The price of this is that Precision is low, which tells us that only 37.1% of all people marked

as POI are actually POI. The reason for choosing this strategy is explained in 1.
As you can see, both Precision and Recall are above the 0.3 margin.