**Predicting persons of interest from enron’s financial and email data**

**Vivek Yadav, PhD**

**Question 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”]

**Response**:

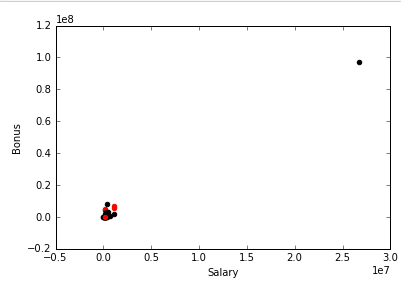
In this project I investigate the Enron email data. Enron email data set is a large database of about 0.5 Mn emails from about 150 employees at Enron. The Federal Energy Regulatory Commission (FERC) for investigation acquired the data after the company filed for bankruptcy in 2001. Several board members and management employees were involved in illegal business practices, and were eventually charged. Some of these people were found guilty of fraudulent practices, and others settled out of court. These people are tagged as Persons of Interest (POI) in the data set. This project aims to develop a formal scheme to identify POIs. The goal of this project is to develop an algorithm that has recall and precision above 0.3. Machine learning techniques are well suited to obtain data-driven solutions to such questions. Machine learning techniques are used for developing models that can be used to learn from and make predictions on data. In this project, I apply machine learning techniques to investigate if a POI can be identified using email and financial data.

**Data Cleaning**

The first step was exploratory analysis of the data. In the dataset, there were 18 POIs and 146 total entries. Each data point contained the 21 features, 2 of these are a variable called POI that identifies if a person is POI and other is the email address of the user.

*Outlier:*

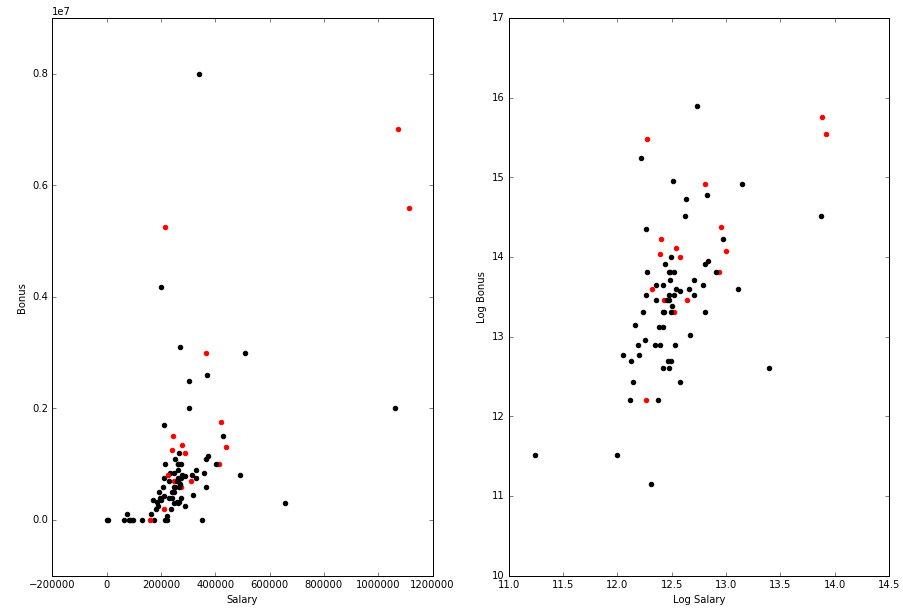
The first step was to identify any outliers in the data and remove them. To identify the outliers, I plotted salary vs bonus data. There was one clear outlier; further investigation revealed this outlier was 'TOTAL'. After removing this outlier, the other values followed the expected distribution where there were a few numbers in millions and many around 100,000.



Salary vs bonus before removing outlier

*Missing values:*

The financial and email data were collected from different sources. Financial data came from the insider pay data, and email data came from Enron’s email set. Therefore, for several people, financial data was missing. For example, of the 146 entries, only 95 had information about salary (excluding TOTAL). As 0 was imputed for missing values, a machine learning algorithm may use the missing values to predict if an individual is POI or not. For example, of the 145 individuals in the data set, 50 individuals did not have email information. Of these 50 individuals, 4 (out of 18 total) were POIs. Therefore, a machine learning algorithm can utilize 0 value for salary to tag non-POIs. Further, there were only 94 individuals that had information regarding salary, of these 17 were POIs. Therefore, to avoid the chance of machine learning algorithm using lack of information as a feature, I included only individuals who had both financial and email data.



Bonus vs Salary, and log of Bonus vs log Salary

I selected individuals that had data for salary, bonus, restricted\_stock and to\_messages entries. After this step, the number of individuals in the data set reduced to 94 and the number of POIs to 17. Table below presents individual features, and the numbers after removing the missing values.

Table 1: Financial and email data. Numbers in bracket are number of data points before and after removing missing values. Bold values indicate the features that were used for model building.

|  |  |
| --- | --- |
| Financial | Email |
| 1. **Salary (95,94)** 2. **Bonus (82,81)** 3. **Total payments (125,94)** 4. **Exercised stock options (102,71)** 5. Deferral payments (39,26) 6. **Restricted stock (110,86)** 7. Restricted stock deferred (18,9) 8. **Total stock value (126,90)** 9. **Expenses (95,81)** 10. Loan advance (4,3) 11. **Long term incentive (66,64)** 12. Deferred income (49,36) 13. **Other (93,88)** 14. Director Fees (17,0) | 1. **To Messages (86, 86)** 2. **From Messages (86, 86)** 3. **Shared receipt with POI (86, 86)** 4. **To poi from this person (86, 86)** 5. **From poi to this person (86, 86)** |

**Question 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”, “properly scale features”, “intelligently select feature”]

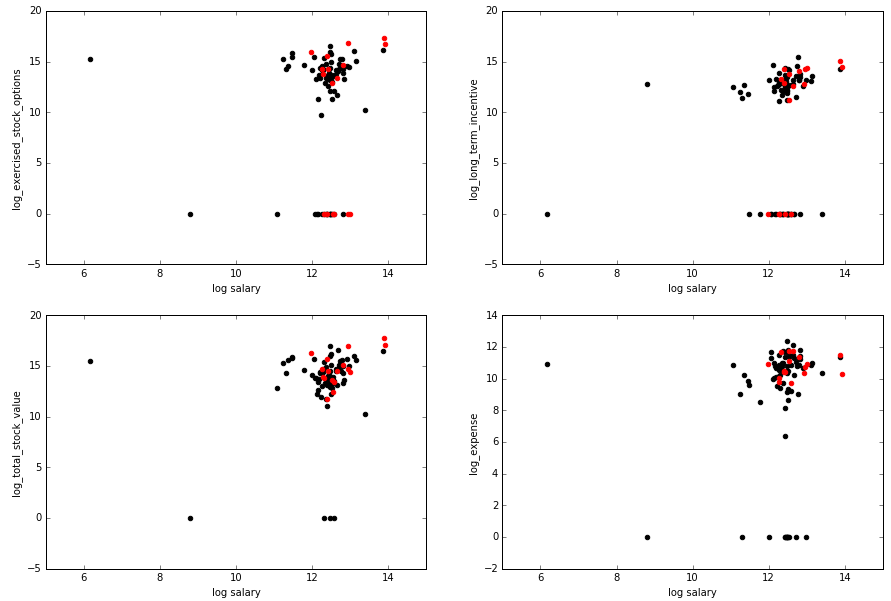
**Response:**

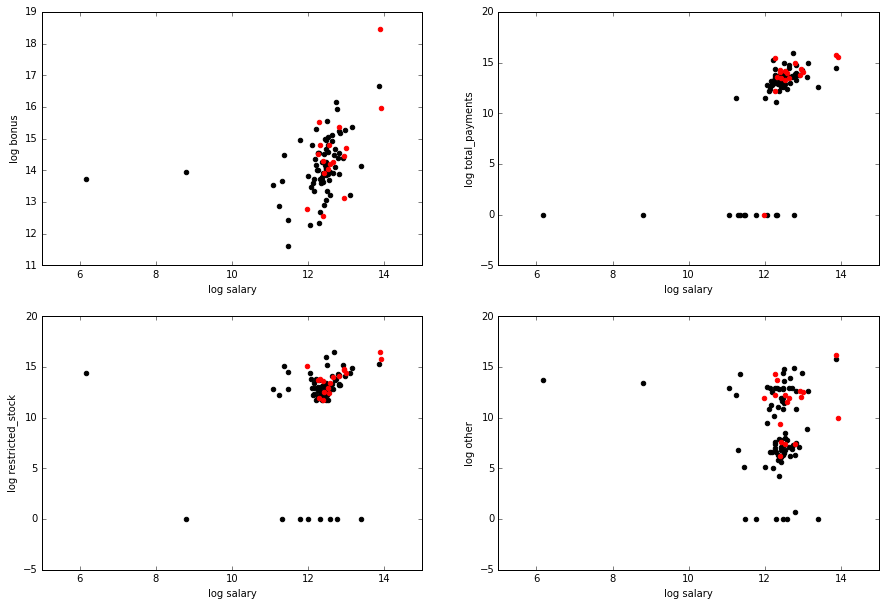
After removing the incomplete salary data, I had 94 data points. I used 9 financial features (listed below) these features occurred in atleast 2/3 of the data. I took log of these value to make the numbers have similar dimensions.

These are listed below,

1. Log of salary
2. Log of bonus
3. Log of total payments
4. Log of restricted stock
5. Log of other payments
6. Log of total stock value
7. Log of exercised stock option
8. Log of expenses
9. Log of long term incentive.

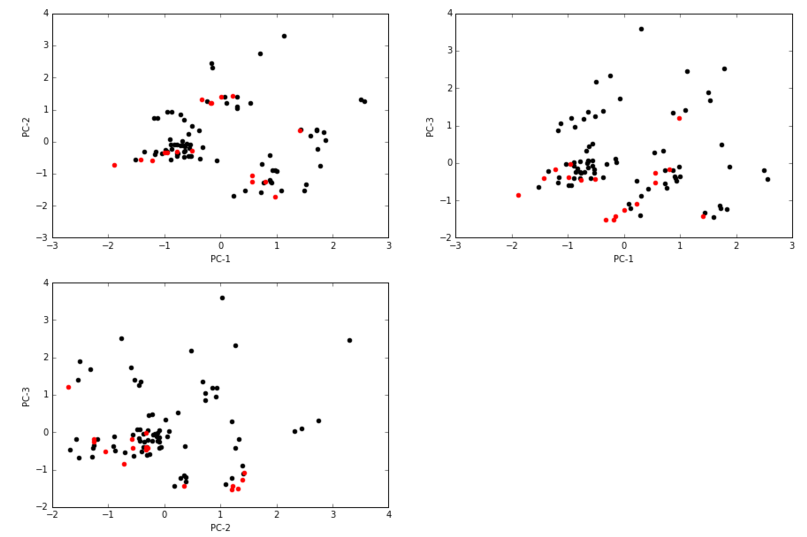
Below I plot financial features against one another,





Plot of financial features against log(salary),

It is clear that financial data is highly correlated. This is not surprising because an individual with higher salary is expected to have higher compensation via stocks, bonus and additional expesnes. I therefore calculated principal components and took 3 components. Therefore, I scaled the data using MinMaxScaler in SKlearn, and then applied principal components analysis (PCA) to obtain a low-dimensional representation of the financial data. After model tuning, I ended up using 3 principal components. After calculating PCA, the POI data became clustered as shown below,



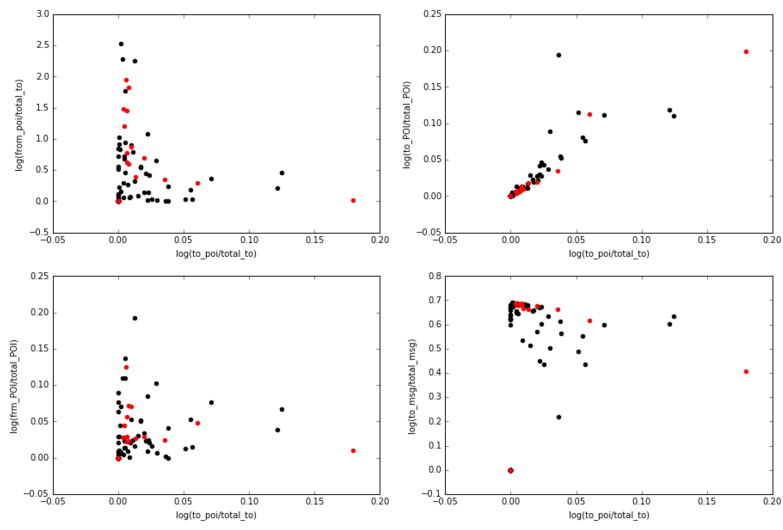
Plot of principal components for financial data.

Email features were computed by taking log of ratios. As 0 is a valid entry, I added one to the ratio before computing the log.

I added 1 to the ratio before taking log.

Email features

1. Log of ratio of emails to POI and total outgoing emails
2. Log of ratio of emails from POI and total incoming emails
3. Log of ratio of emails to POI and total interaction with POI (sum of to, from and shared with POI emails).
4. Log of ratio of emails from POI and total interaction with POI (sum of to, from and shared with POI emails).
5. Log of To-From mail ratio to quantify how active a user is in receiving and sending out emails.



Email features:

From plots above, its clear that second and 3rd email features provide little information regarding POIs. This is also conformed by selectKbest algorithm that for K=3 gave first, second and fifth feature as the main email features. I therefore used these 3 email features for further modeling,

**Question 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”]

**Response:**

**Predictive Modeling**

***Scoring function:***

Goal of this project was to design a machine learning algorithm that has precision and recall above 0.3. Therefore, instead of using precision or recall as scoring function, I used a custom scorer that returned minimum of precision and recall, and the goal of the algorithm was to maximize the minimum of precision and recall.

***Model Candidate selection:***

I tested 6 algorithms. These were naïve bayes, support vector machines, decision trees, random forest, adaptive boosting, and kmeans clustering. It is crucial to select model parameters appropriately, because bad choice of parameters can result in overfitting of data or an algorithm that is too sensitive to variations in data. I used GridsearchCV to compute optimal parameters for individual models. The parameters and the values tested for individual models are given below.

1. SVC:
   * 'C': [100,500,1e3, 5e3, 1e4, 5e4, 1e5],
   * 'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1],
2. DecisionTree:
   * 'min\_samples\_split': [1,2 ,3,4,5,6,7,8],
   * 'max\_depth':[1,2,3,4,5,6],
   * 'max\_features':[2,3,4,5],
   * 'criterion':['gini','entropy']
3. Adaptive Boosting:
   * 'learning\_rate': [.1,.2,.3,.4,.5,.6,.7,.8,.9,1.0],
   * 'n\_estimators': [5,10, 25,50,75,100],
4. K-Means:
   * 'n\_clusters': [4,5,6,7,8,9]
5. Random forest:
   * 'min\_samples\_split': [1,2,3,4,5,6,7,8]
   * 'max\_depth':[1,2,3,4,5,6]
   * 'max\_features':[2,3,4,5]
   * 'criterion':['gini','entropy']
6. Naïve Bayes

To get the best model candidate, I used GridsearchCV. It is crucial to test the performance of the model on a dataset/split that wasn’t used for training or parameter selection, because fine tuning model parameters can result in overfitting of the model. Another issue is in choosing the split data is that the training and test sets may not have the same proportion of entries for different classes. I therefore incorporated StratifiedShuffleSplit in GridsearchCV to perform crossvalidation while testing different parameters.

I fit the model on a training set, and tested its prediction another testing set. I generated training and testing sets using StratifiedShuffleSplit function in sklearn, and split data such that the training set is 30% of the total data. I repeated this process 25 times by setting n\_iterations to 25. I provided a custom scoring function that maximized the minimum of precision and recall. After performing grid search, I got the following parameters for the models.

* SVC: best score = 0.2803, 'C': 500, 'gamma': 0.0005
* DecisionTreeClassifier: best score =0.2789, 'max\_features': 5, 'min\_samples\_split': 8, 'criterion': 'gini', 'max\_depth': 6
* RandomForest: best score = 0.2387, 'max\_features': 4, 'min\_samples\_split': 4, 'criterion': 'gini', 'max\_depth': 6
* AdaBoostClassifier: best score = 0.222, 'n\_estimators': 100, 'learning\_rate': 0.4.
* KMeans: best score: 0.2202, 'n\_clusters': 9

After getting the candidate model, I checked performance on a larger set of training and test data. For this part, I used StratifiedShuffleSplit to generate 1000 sets of data with training set containing 90% of data and testing set containing 10%. This split was similar to the testing function in tester file.

***Validation of models***

It is crucial to test the performance of the model on a dataset/split that wasn’t used for training or parameter selection, because fine tuning model parameters can result in overfitting of the model. Another issue is in choosing the split data is that the training and test sets may not have same proportion of entries for different classes. I therefore used StratifiedShuffleSplit again but now used 1000 sets of data with test size restricted to 0.1. Below are performance of individual models,

1- SVC

- Mean Precision: 0.3313 , Mean Recall: 0.597 - STD Precision: 0.2145 , STD Recall: 0.3355 - CI Precision: (0.3179, 0.3445) - CI\_recall: (0.5761, 0.6178)

2- DecisionTree

- Mean Precision: 0.3465 , Mean Recall: 0.2875 - STD Precision: 0.4027 , STD Recall: 0.31 - CI Precision: (0.3215, 0.3715) - CI\_recall: (0.2682, 0.3067)

3- Adaptive Boosting

- Mean Precision: 0.3419 , Mean Recall: 0.2645 - STD Precision: 0.4127 , STD Recall: 0.3087 - CI Precision: (0.3163, 0.36755) - CI\_recall: (0.2453, 0.2836)

4- Random Forest

- Mean Precision: 0.2144 , Mean Recall: 0.1555 - STD Precision: 0.367 , STD Recall: 0.2521 - CI Precision: (0.1915, 0.2371) - CI\_recall: (0.1398, 0.1711)

5- Naive Bayes

- Mean Precision: 0.1311 , Mean Recall: 0.0995 - STD Precision: 0.2911 , STD Recall: 0.1996 - CI Precision: (0.1130, 0.1492) - CI\_recall: (0.0871, 0.1118)

6- K-Means

- Mean Precision: 0.1285 , Mean Recall: 0.211 - STD Precision: 0.2798 , STD Recall: 0.4 - CI Precision: (0.1111, 0.1458) - CI\_recall: (0.1861, 0.2358)

**Final candidate model selection**

From above, SVC provided the best precision and recall, further the 95% confidence interval for both precision and recall were above 0.3 for this algorithm only. I therefore will use this for further tuning.

**Additional Analysis on PCA**

I used PCA on all the data and calculated principal components for all the financial features. This method however computed PCA based on all the data, therefore, there could be some information leakage from test to training data. I therefore tested the effect of performing PCA separately on training set, and using those Principal components for dimensionality reduction. For this process, I used pipeline in sklearn. I varied the parameters of SVC (C and gamma) along with number of principal components of financial features. In addition I varied K in selectkbest to compute the best combination of email factors. After this process, the score changed as follows,

1- SVC with PCA on all data, with finer gridsearchCV, 'C': 150, 'gamma': 0.005.

- Mean Precision: 0.3745 , Mean Recall: 0.6715 - STD Precision: 0.2224 , STD Recall: 0.3367 - CI Precision: (0.3606, 0.3882) - CI\_recall: (0.6505, 0.6924)

2- SVC with PCA on training data in each validation/fitting cycle

- Mean Precision: 0.5043 , Mean Recall: 0.563 - STD Precision: 0.3688 , STD Recall: 0.35 - CI Precision: (0.4832, 0.5247) - CI\_recall: (0.5414, 0.5846)

**Final model**

After analysis above, I chose to use a model in which I fit PCA to each training data set, and use those principal components to transform test data. After using pipeline and gridsearchCV, I got the following modeling parameters,

* C = 5
* gamma = 0.01
* Number of principal components for financial features, 3
* Number of email features, 2

After using this values in the final model, I got precision and recall values above 0.3. These values are,

* Accuracy: 0.79380
* Precision: 0.48674
* Recall: 0.56900
* F1: 0.52467 F2: 0.55040
* Total predictions: 10000 True positives: 1138 False positives: 1200 False negatives: 862 True negatives: 6800

**Question 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? (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 item: “tune the algorithm”]

**Response:**

**Parameter selection:**

Parameters of a model are used to characterize the model. These parameters can be used to set the sensitivity of the model to data. If the model is made very sensitive, then it will fit the training data well, but will not generalize to newer dataset. On the contrary, if tuning parameters are set so the model is not sensitive to data, then the model may not capture certain variations that will result in overall poorer performance. Therefore, providing appropriate model parameters is crucial to trade-off between sensitivity to data and generalizability of the model.

**Model tuning:**

For my model, I used PCA on financial data and used select k-best for email features. I also varied C and gamma for the SVC classifer. I used pipeline and gridsearchCV in sklearn to compute optimal features. The goal of the optimizer was to select parameters that maximize the minimum of precision and recall.



After analysis above, I chose to use a model in which I fit PCA to each training data set, and use those principal components to transform test data. After using pipeline and gridsearchCV, I got the following modeling parameters,

* C = 5
* gamma = 0.01
* Number of principal components for financial features, 3
* Number of email features, 2

After using this values in the final model, I got precision and recall values above 0.3. These values are,

* Accuracy: 0.79380
* Precision: 0.48674
* Recall: 0.56900
* F1: 0.52467 F2: 0.55040
* Total predictions: 10000 True positives: 1138 False positives: 1200 False negatives: 862 True negatives: 6800

**Question 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 item: “validation strategy”]

**Response**:

Validation is a process to test if the predictions made by a model generalize to new data that the model had not seen in training. It is crucial to test the performance of the model on a dataset that wasn’t used for training, because fine-tuning model parameters with training data alone can result in over-fitting of the model. Another issue is in choosing the test data is that the training and test sets may not have the same proportion of entries for different classes. Therefore, there may be cases where the distributions of clusters are not appropriately represented in the test and train data. Finally, if the data is ordered, performing split where one takes first 70% of the data can result in the effects of order creeping in.

I generated training and testing sets using StratifiedShuffleSplit function in sklearn, and split data such that the training set is 30% of the total data. I repeated this process 25 times by setting n\_iterations to 25. I did this to select the candidate model.

**Question 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”]

**Response:**

After deciding upon the model, I used pipeline and gridesearchCV to compute optimal parameters of the model. , I got precision and recall values above 0.3. These values are,

* Accuracy: 0.79380
* Precision: 0.48674
* Recall: 0.56900
* F1: 0.52467 F2: 0.55040
* Total predictions: 10000 True positives: 1138 False positives: 1200 False negatives: 862 True negatives: 6800

Recall of 0.569 indicates that if a person is POI, the model can identify that person as POI 56.9% of the time.

Precision of 0.487 indicates that among all the people identified as POI by the model, only 48.7% are POIs.