Machine Learning Analysis of Person Of Interest in the Enron Scandal

Ju Yang

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?*

Goal:

To identify person of interest (poi) involved in the Enron Scandal using financial and email data. Machine learning provides a powerful prediction tool based on features by building a classifier in the training dataset. The classifier can be then used in the test dataset to predict whether a person is poi or not.

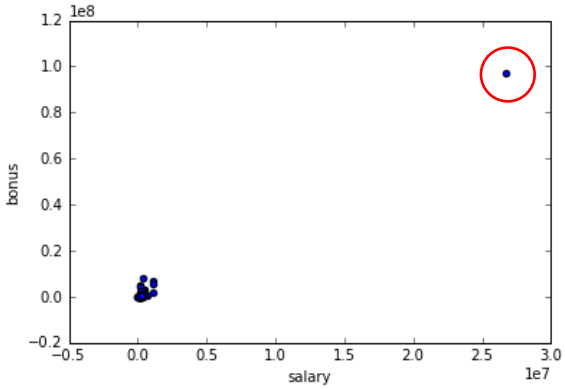
Dataset review:

The original Enron dataset final_project_dataset.pkl contains **146 data points, with 21 features** .The features in the data fall into three major types, namely financial features, email features and POI labels.

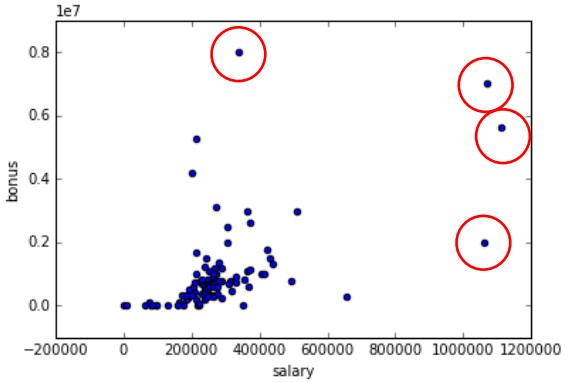
```
financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances',
'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses',
'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock',
'director_fees'] (all units are in US dollars)


email features: ['to_messages', 'email_address', 'from_poi_to_this_person',
'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are
generally number of emails messages; notable exception is ‘email_address’, which is a
text string)


POI label: [‘poi’] (boolean, represented as integer)
```

Outliers:

I plotted feature 'salary', 'bonus' in a scatter plot. It is clear there is one data point that is an outlier. After checking the original financial document, this data point represents the **total accumulative value**.



I removed the **'TOTAL'** key in the dataset. Now 4 data points stand out from most others with high salary or high bonus (circled out). These data points are likely to represent distinct and interesting information.



After removing 'TOTAL' outlier, **the number of people in the dataset is 145, with 18 poi, accounting for 12.4%**.

Because the total data point is relatively small and the percentage of poi is low (skewed), it is challenging to perform machine learning and build a good classifier on this small dataset. In the classifier selection and tuning process, I used stratified shuffle split cross validation to evaluate the performance of classifiers.

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.*

Feature selection:

From the pdf document, **'total_payments'** and **'total_stock_value'** are the sum of other variables. I decided not to include them in the features list since they can be linearly presented by individual components. 'email_address' is a text string and corresponds to the unique person's name, I decided not to include it either.

I created 2 features **'from_poi_to_this_person_ratio'** and **'from_this_person_to_poi_ratio'** to capture the percentage of poi email in to_ and from_ messages for each person. Higher percentage of poi message might suggest a higher interest, while the absolute value of messages can be misleading because depending on the job position, some people may receive more emails than others no matter what.

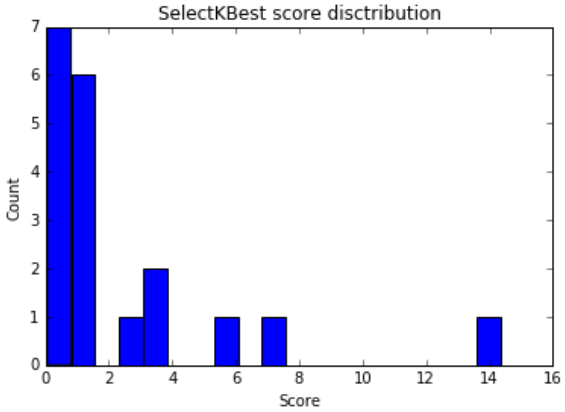
```
features_list  = ['poi','salary', 'deferral_payments', 'loan_advances', 'bonus',
'restricted_stock_deferred', 'deferred_income', 'expenses', 'exercised_stock_options',
'other', 'long_term_incentive', 'restricted_stock', 'director_fees', 'to_messages',
'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi',
'shared_receipt_with_poi','from_poi_to_this_person_ratio','from_this_person_to_poi_ratio'
]
```

I performed feature scaling using **MinMaxScaler**, because I used support vector machine (SVM) in the feature selection process. SVM is largely affected by the scaling process. I used **SelectKBest** in feature selection, and the best estimators are:

```
Pipeline(steps=[('features', FeatureUnion(n_jobs=1,

       transformer_list=[('univ_select', SelectKBest(k=1, score_func=<function f_classif
at 0x000000000C352908>))],

       transformer_weights=None)), ('svm', SVC(C=1, cache_size=200, class_weight=None,
coef0=0.0,

  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',

  max_iter=-1, probability=False, random_state=None, shrinking=True,

  tol=0.001, verbose=False))])
```

k = 1 in SelectKBest, suggesting that one feature is likely to be highly relevant. However, when I went through the GridSearchCV output, different k values generate same score 0.900000.

I wonder what the **importances** of features are. As we can see here, there are 5 features with score bigger than 3. These 5 features are **'salary', 'deferred_income', 'expenses', 'from_poi_to_this_person_ratio', 'from_this_person_to_poi_ratio'**.



From the financial document, ‘deferred income’ : Reflects voluntary executive deferrals of salary, annual cash incentives, and long-term cash incentives as well as cash fees deferred by non-employee directors. I think it is directly related to other income data rather than an independent variable, so I decided not to use it despite its high score.

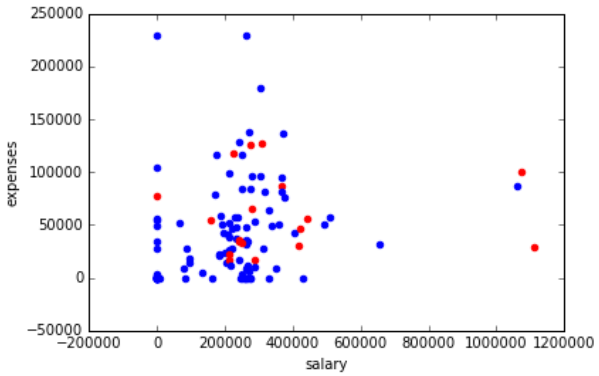
My final feature selection contains **2 financial features** and **2 email features**.

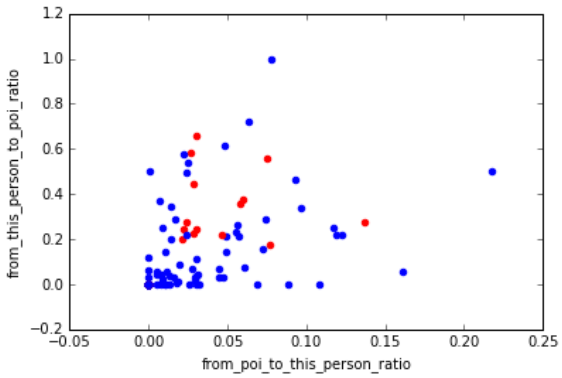
```
features_list = ['poi','salary', 'expenses', 'from_poi_to_this_person_ratio',
'from_this_person_to_poi_ratio']
```

I found there are **NaN** values for these features. Given that there are only 145 data points and a significant number of data points contain NaN value, this may greatly affect the machine learning performance.

```
number of NaN value for poi : 0

number of NaN value for salary : 51

number of NaN value for expenses : 51

number of NaN value for from_poi_to_this_person_ratio : 59

number of NaN value for from_this_person_to_poi_ratio : 59
```

I plotted financial and email features separately and colored poi in **red**, it seems that with each type of features alone, it is difficult to visualize the separation of poi and non-poi.





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

   Because of the small size of the data and the low percentage of poi in the data, I used **StratifiedShuffleSplit** to repeat the performance calculation for 1000 times and calculate the total performance of different algorithms.

   - Naïve Bayes shows the **highest recall** 0.63050, but precision is not as high as other algorithms. It performs well on identifying a potential target, although the target may be a false positive.
   - Linear SVM shows the **highest accuracy** 0.82158 but very low precision and recall. This is because the label is highly skewed and only 12% of all data are poi. This algorithm has high accuracy by well predicting true negative non-poi data.
   - RBF SVM has lower accuracy than linear SVM, but higher precision and recall. Precision 0.30088 is even higher than Naïve Bayes, suggesting that when it predicts a positive poi, it is more likely to be a true positive than Naïve Bayes.
   - Decision Tree shows the **highest precision** 0.38365 and second highest recall 0.42700, and the **highest F1 score** 0.40416, together with the second highest accuracy 0.79017. It performs well on recalling a poi, and the precision of identifying a poi in the highest. It also performs well on correcty predicting poi and non-poi.

   I decided to choose Decision Tree as my algorithm.

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).*

   Machine learning algorithms have many parameters and their behavior can be tuned for a given **problem**. Combination of parameters can be modified to best achieve a certain goal.

   My goal is to identify a poi in the dataset, and it is the best if we can achieve **both high precision and high recall** so that we can identify as many poi as possible with as higher precision as possible. If we simply use accuracy, due to the dominant size of non-poi negative results, we may end up getting high accuracy with low recall. If we use precision alone, we may end up being too careful that we miss some poi and regard them as false negative, letting criminals escape. If we use recall alone, we may end up identifying false positive poi and creating unnecessary tension and stigma for individuals.

   Therefore, I decided to tune the following 3 parameters of Decision Tree to achieve a high F1 score: criterion, min_samples_leaf, min_samples_split. The highest F1 score is **0.41545 with criterion='gini', min_samples_leaf=4, min_samples_split=16**

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 is a method to assess how the results of a statistical analysis will generalize to an independent data set. In machine learning and prediction, we need to evaluate the performance of the model using different performance metrics. It is crucial to cross validate. If we use all data as the training set and validate on the training set alone, we may end up overfitting the data.

   I performed cross validation by splitting my data into training and test dataset in **feature selection, algorithm selection, and algorithm tuning**. In particular, because of the small size of the data, I used stratified shuffle split cross validation to evaluate the performance of classifiers.

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.*

   I calculated **accuracy, precision, recall,** and **F1-score** in the performance function. And using the tuned parameters, the results are Accuracy: 0.81592, Precision: 0.44126, Recall: 0.39250 F1: 0.41545

   Conclusion:

   My algorithm uses salary, expenses, the percentage of emails from poi in all emails received, and the percentage of emails to poi in all emails sent, and predicts whether or not a person is a person of interest. When the algorithm predicts a person is a poi, 44% of the time this person is indeed a poi, and 56% of the time the person is innocent, suggesting that further investigation is required to confirm the result. This is much better than guessing given that poi accounts for 12% in the dataset. 40% of the time, the algorithm can identify a poi, 60% of the time will miss it, suggesting that more than half poi may escape from this algorithm. In general, 4 out of 5 predictions are accurate.

   Next step:

   To increase the precision and recall of the algorithm, we may need more features. A potential useful information is the email content itself. Using text learning, we may gain insights into the distinct features of poi.