Porject 5: Identify Fraud from Enron Email

Udacity Data Analyst Nonedegree

By Nan-Tsou Liu @ 2016-07-17

Introduction

Enron Corporation was an American energy, commodities, and services company based in Houston, Texas. Before its bankruptcy in 2001, there were about 20,000 employees and was one of the world's major electricity, natural gas, communications and pulp and paper companies.

The Enron Corpus is a large database of over 600,000 emails generated by 158 employees of the Enron Corporation and acquired by the Federal Energy Regulatory Commission during its investigation after the company's collapse.

Short Question

> *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"]*

Goal of the Project:

In this project, the prediction model is built by using the python module called scikit-learn to identify the person of interest (POI). The following skills are applied to carried out the project:

- feature selection and scaling
- algorithm selection and tuning
- validation and classic mistakes
- evaluation metrics and interpretation of algorithm's performance

Basic Info of the DataSet

First of all, I would like to take a look at the original dataset to realize what data structure and characters this dataset is. The original dataset contains **146** records with **1 labele (POI)**, **14 financial features** and **6 email features**. In the data of labels, there are **18** records have been marked as **POI**. Furthermore, we also concern that whether there are **missing data** or not. According to pdf file, **enron61702insiderpay.pdf**, we could simply find out lots of data are lost. Thus, I counted data loss of each feature to check how many data of each feature we loss. And in order to have a insight into the difference between the records of POI and Non-Poi. I counted the loss with respect **POI and Non-POI respectively**. Also, I presented the overall loss count together.

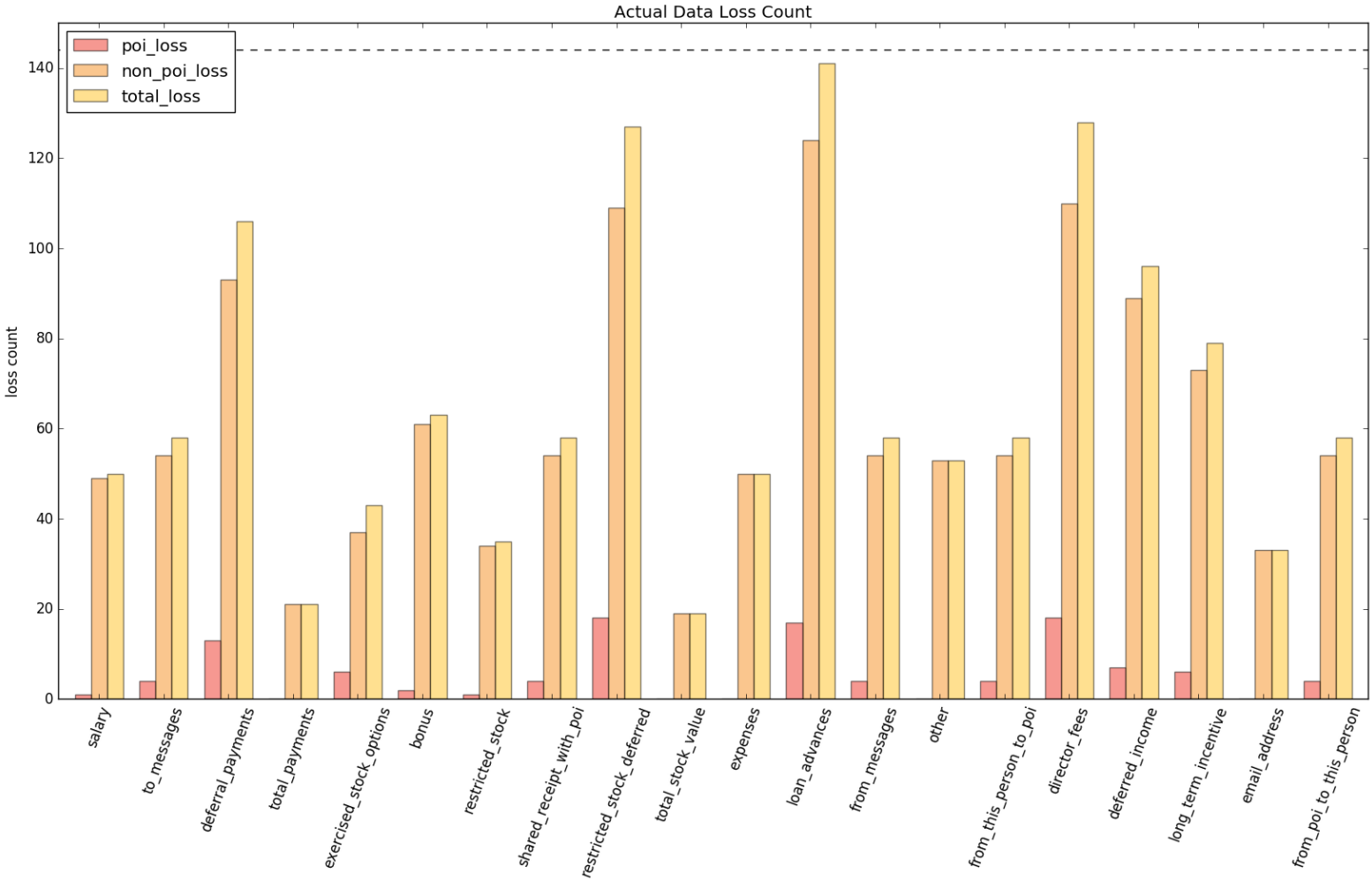
Data Loss Count of Each Feature

Feature	POI	NON POI	TOTAL
salary	1	49	50
to_messages	4	54	58
deferral_payments	13	93	106
total_payments	0	21	21
exercised_stock_options	6	37	43
bonus	2	61	63
restricted_stock	1	34	35
shared_receipt_with_poi	4	54	58
restricted_stock_deferred	18	109	127
total_stock_value	0	19	19
expenses	0	50	50
loan_advances	17	124	141
from_messages	4	54	58
other	0	53	53
from_this_person_to_poi	4	54	58
director_fees	18	110	128
deferred_income	7	89	96
long_term_incentive	6	73	79
email_address	0	33	33
from_poi_to_this_person	4	54	58

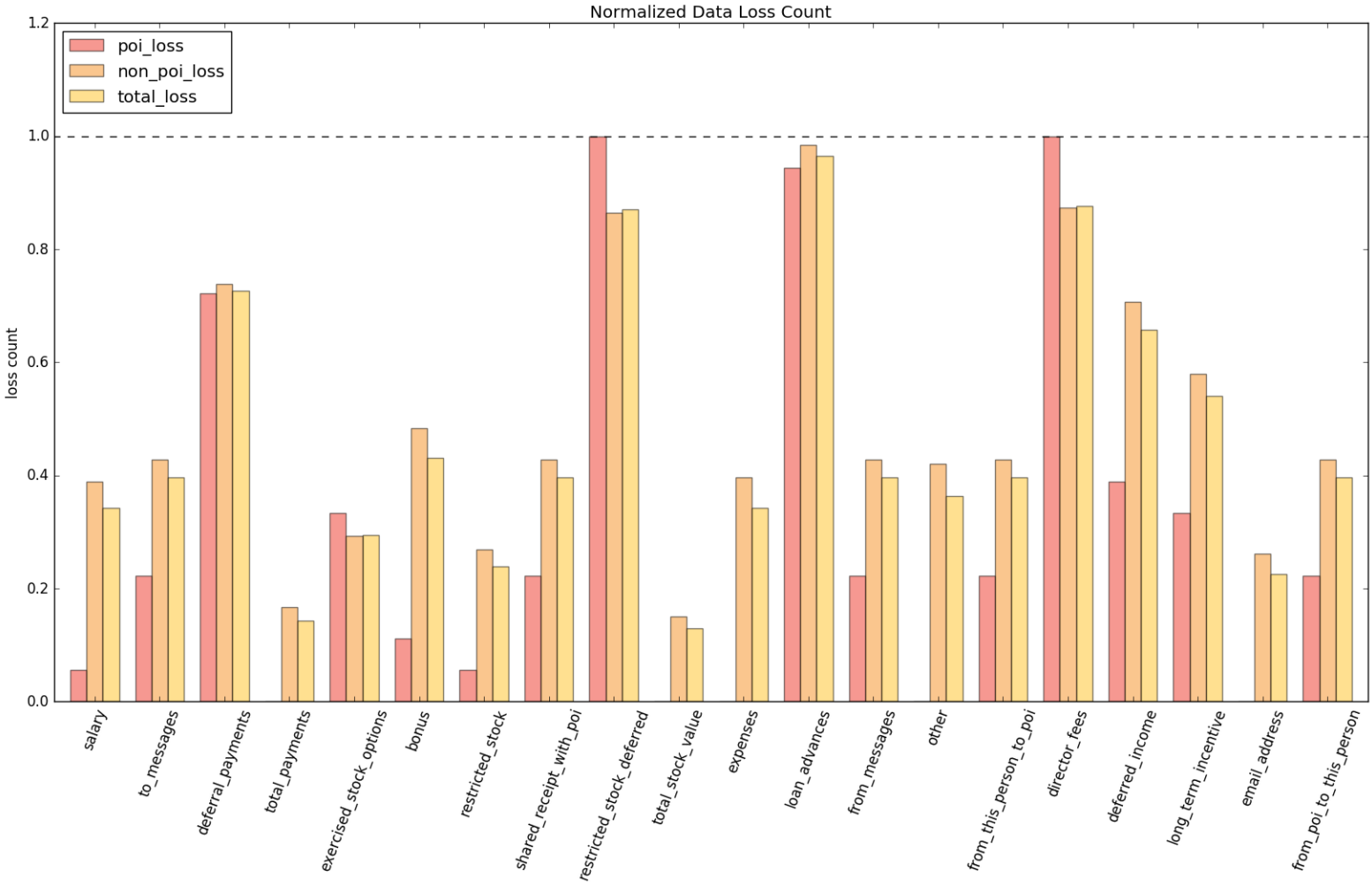
Outlier and label POI are not counted
POI: 18 records Non-POI: 126 records TOTAL: 144 records

Visualization of Data Loss

In order to realize the difference in data between POI and Non-POI, I visualized the loss count with **BAR chart** as below.



With actual data loss count figure, it is a little bit hard to compare the differnce in records between POI and Non-POI, because the number of records are quite different (POI: 18 records, Non-POI: 126 records). However, we can simply understand that some features like **total_payments**, **total_stock_value**, **expense** and **other** should be the important features to identify POI because there is no data loss in POI records. Surely, this one can also be observed with the talbe above.



On the other hand, with normalized data loss count figure, we can simply indicate that which feature has weak effect on the prediction. In this case, the features like **deferal_payments**, **exercised_stock_options**, **restricted_stock_deferred**, **loan_advances** and **director_fees** could be weak features because there are almost no data in both POI and Non-POI. This fact could be observed with the table and figreu above, but normalized figure could provide quick insigt into it. Besides, normalized figure also tells what the percentage the data lost in both POI and Non-POI record. It can help us to choose those which have data over 50% to do the prediction.

As the simple conclusion in this section, we have over all **146 records** which has two outlier, **20** features and **1** label called **POI**. And there are four features, **deferal_payments**, **restricted_stock_deferred**, **loan_advances** and **director_fees**, which loss data over 50%.

Outliers

By observing the data and pdf file, **enron61702insiderpay.pdf**, the obvious outliers are **TOTAL** and **THE TRAVEL AGENCY IN THE PARK** which are simply removed by `pop()` method of dictionary in Python. Besides, the record of **LOCKHART EUGENE E** is empty, thus it was also removed.

I found out that the records of **BELFER ROBERT** and **BHATNAGAR SANJAY** are not consistent with the data in pdf file by accident. Therefore, I fixed them before removing outliers and replacing NaN with 0. I did not check every records so that I am not sure whether there are others not consistent or not.

> *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.

At the beginning, I used all the original features except **mail_address** in the dataset. However, after I had a look at the data and considered data loss with the aspect of POI and Non-POi. I decided to exclude four features which loss data over 50%. Thsu, there are 16 featrues used in this project shown as below.

Financial Features

`'salary'`, `'bonus'`, `'expenses'`, `'other'`, `'deferred_income'`, `'long_term_incentive'`, `'exercised_stock_options'`, `'restricted_stock'`

Email Features

`'from_messages'`, `'from_poi_to_this_person'`, `'from_this_person_to_poi'`, `'shared_receipt_with_poi'`, `'to_messages'`

Besides, I added **financial relatived features** like the ratios of each payment feature or stock feature to the total amount of financial and **message related feature**.

The reasons I added these features are that first, I assumed that POI has somehow great relationship with the financial status. First, I calculated the ratio of each financial features to **total_financial** (summation of **total_payments** and **total_stock_value**) because I thought that the person of POI might had large percentage of restricted_stock or salary of the total financial status. It also means that **the composition of the financial status** might be the good features for model training.

However, after I considered more, the total amount of payments and stock values are much greater than each single payment and stock value, so that the ratios calculated by the equations above might elminate the effect of the new features to the prediction. Thus, I decided to redesign the new features. First, I **separated** the payments and stock values and calculated respectively. Second, I added **ratio of total payments to overall financial amount** and **raton of total stock value** to overall financial amount, where overall financail amount is the summation of total payments and total stock values.

On the other hand, the messages of each person should be a strong feature to identify POI. Therefore, I culaculated the ratio of **poi_relative_message** (summation of **from_poi_to_this_person**, **from_this_person_to_poi** and shared **receipt_with_poi**) to **total_messages** (summation of **from_message** and **to_message**).

Added Features

Feature	Description
total_financial	total_payments + total_stock_value
{each payment feature}_payment_ratio	{each payment feature} / total_payments
{each stock feature}_stock_ratio	{each stock feature} / total_stock_values
poi_ratio_messages	poi_related_messages / total_messages
poi_related_messages	from_poi_to_this_person + from_this_person_to_poi + shared_receipt_with_poi (FOR CALCULATION ONLY)
total_messages	from_messages + to_messages (FOR CALCULATION ONLY)

Engineering Data

The end-up using features were selected during GridSearchCV pipeline search with following steps:

1. scale all features to be between 0 and 1 with MinMaxScaler
2. dimension reduction with SelectKBest and Principal Components Analysis
3. tune parameters of each models

Features scaling was carried out since PCA and various models such as Logistic Regression perform optimally with scaled features. Feature scaling is also necessary since they were on very different scales, ranging from hundreds of e-mails to millions of dollars.

SelectKBest and **Principal Components Analysis (PCA)** dimension reduction were run during each of the cross-validation loops during the grid search. The K-best features were selected using **Anova F-value classification** scoring function. The K-best features were then used in reducing dimension with PCA. Finally,the N principal components were fed into a classification algorithm.

The results of the feature selection would be discussed in the section **Final Results of each Algorithm** below. And the K-Best result would be shown in section **Parameter Tuning**

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

The ended up using algorithm was determined by the results of **GridSearchCV** respected **recall score** and **precision score** with k-best feature selection, PCA reduction and parameters tuning. **Support Vector Classifier (SVC)** showed the best results, which are **recall score 0.70900 and precision score 0.34434**, and therefore it was the ended up using algorithm in this project. Besides SVC, I also tried **Logistic Regression (LogReg)**, **Linear Support Vector Machine (LSVC)**, **Decision Tree (DTree)** and **K-Means Classifier (KMeans)**.

Actually, LogReg and LSVC also showed the competive results compared with that of SVC. And the results of both algorithms were similar. The recall scores are about **0.70** and **0.68** of LogReg and LSVC respectively. And the precision scores are about **0.3** of both algorithms, which were calculated by the validation program I wrote. According to the obversation by manual parameter tuning, recall score and precision score were changed against with each other. And the parameter **C** affected the results mostly. And I found out an interesting phenomenon that the value ended with 5 like `[0.05, 0.5, 0.15]` could keep recall score at good value and promote precision score well. And, precision score kelp **around 0.3**. However, the results tested by the program provided by Udacity were around **0.27 to 0.28**, which did not meet the requirements of the assignments.

The results of DTree were fair although the results matched the requirements of the assignment. However, the tuning parmeters are quite different from the algorithms above. I have tried to tune parameter **max_depth**, **min_samples_leaf**, and **min_samples_split**. And the final result in this case was **Precision: 0.33299** and **Recall: 0.64200**.

On the other hand, the precision score of KMeans (which was told by Udacity reviewer that it is intended for unsupervised learning) kept at the value **about 0.15**, which is far from the requirement of the assignment. No matter how I tuned the parameters, although precision score was about **0.5**. In my opinion, KMeans might not suitable for this prediction after I added the new features. Thus, I decided to exclude it from the discussion of this report.

> *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).*

Tuning the parameters of an algorithm is a process to promote the performance of the model. Depended on the structure and nature of the dataset, tuning the parameters would cost lots of time when doing model training. In this project, **Grid Search** was used to tune the parameters of the algorithms with 1000 randomized stratified cross-validation stratified splits. The parameters of each algorithms with highest average score were choosen for the models.

Final Results of each Algorithm

Parameter	Logistic Regression	Linear Support Vector Classifier	Support Vector Classifier	Decision Tree
C	1.3	0.146	1.335	-
class_wight	auto	auto	auto	balanced
tol	1e-64	1e-32	1e-8	-
n_components of PCA	0.5	0.5	0.5	0.5
whiten of PCA	True	False	False	False
selection of SelectKBest	16	15	15	11
gamma	-	-	11.55	-
kernel	-	linier	rbf	-
criterion	-	-	-	entropy
splitter	-	-	-	best
max_depth	-	-	-	2
min_sample_leaf	-	-	-	18
min_sample_split	-	-	-	2

> *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 the process to ensure that the results produced built model with other unknown data is reliable. In my case, A classic mistake is over-fitting, which makes the model too particular so that the results produced with the other data are poor and unreliable. One of major purpose of validation is to avoid over-fitting.

Over-Fitting on SVC with Parameter C

I found out an interesting results when I manually tuned parameters of SVC. As the table shown below, I obtained recall score which is **1.0000** when I set 0.05 to parameter **C**. At the beginning, I did not noticed that it was the result caused by over-fitting. But I noticed that it was caused by parameter C. So, I did the simple investigate on the internet. The simple description of C is that the value of the regularization constraint, which tells the SVM optimization **how much you want to avoid misclassifying** each training dataset. And a very small value of C will cause the optimizer to look for a larger-margin to separate hyperplane, which means that there might be many points which are hyperplane misclassifies.

CASE	C	recall score	precision score
1	0.01	0.7335	0.3317
2	0.05	1.0000	0.1333

Cross-Validation

Cross-Validation was applied on the validation of model. It is a process that randomly split the data into training and testing dataset. And then it trains the model with the training data and validates with the testing data. In this project, the whole dateset was splitted with 1000 randomized **stratified cross-validation splits**. And then the parameters with the best performance over 1000 splits were selected.

According to the structure and nature of this dataset. The records we have are quite few. There are only 18 POI and 144 records. And lots of data of some features are lost. Thus, **StratifiedShuffleSplit**, which combines **StratifiedKFold** and **ShuffleSplit** and returns

stratified **randomized folds**, is much more suitable for this dataset. And this is also the main reasion that I fianlly used this method to do cross-validation.

Parameter Tuning

As I mentioned above, **GridSearchCV** with over 1000 stratified shuffled cross-validation 90%-training/ 10%-testing splits was used to tune the parameters in this project. Besides, K-best selection and PCA reduction processes were embraced into the parameter tuning loop. Compared with outside selection, the selection in the loop might promote the consistence of parameter tuning and give a less biased estimate of performance on any new unseen data that this model might be used for.

As the results of selected the final model, 15 features were selected. PCA reduction gave 2 principal components. These parameters were used in the final **Support Vector Machine** classification model.

One should be notified is that these features might change slightly each time since the k-best selection was carried out inside of the pipeline. Below are the final 15 features chosen when the entire dataset was fit to the final chosen model pipeline:

K-Best Feature (Top 15)

Added Feature

feature	score ↑
total_stock_value	22.5105490902
exercised_stock_options	22.3489754073
bonus	20.7922520472
bonus_payment_ratio	20.7155962476
salary	18.2896840434
total_financial	17.3887977785
long_term_incentive_payment_ratio	13.8508684172
deferred_income	11.4248914854
poi_ratio_messages	10.0194150056
long_term_incentive	9.92218601319
total_payments	9.28387361843
restricted_stock	8.83185274222
shared_receipt_with_poi	8.58942073168
loan_advances	7.18405565829
expenses	5.41890018941

> *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.*

Final Evaluation Matric Results of each Algorithm

Recall and **Precision** were used as the primary evaluattion metrics. The definition are shown below:

```
recall = True_Positive / (True_Positive + False_Negative)
```

According to the difinition of recall, it is reliable that the target is not what we are interested if it is marked negative with high recall

score. Because high recall score also means low false negative.

`precision = True_Positive / (True_Positive + False_Positive)`

On the other hand, the target marked positive with high precision score can be thought it is what we are interested with high confidence. Because high precision could be thought as that false positive is low.

General speaking, to explain the meaning of recall and precision with the term of this project. If the predicting model with **recall score 0.710 and precsion score 0.344** marks the target as **Non-POI (negative)**, then we are confident to assert this target is not POI because 71% is quite high accryacy. However, if the model marks the target as **POI (positive)**, we have to doubt the result and do more confirmation because 34.4% accruacy is not relaiable to identify the target is POI. Otherwise we mignt label the innocent person with quilty. So, the model with **high reall score and low precision scores** could be used to identify the target is **Non-POI**. As the result, we can decrease the number of suspects and then focus on those who have high possibilty to be the **REAL POI**.

In this porject, lots of records in the dataset are unknown and we have much more negative labels than positive ones. Thus, in my opinion, it is a little bit not practical to build a model which can identify the target is POI, which means the model with high precision score. On the other hand, it should be more practical to build the model which can identify the target is not POI, which means the model with high recall score. As the result, it is not surprised that recall socres are **much greater** than precision scores of all the results of all the algorithms in this project.

Model	Recall Score by GridSearch	Recall	Precision	KBest Features	PCA Components
Logistic Regression	0.7290	0.71750	0.28044	16	2
Linear SVC	0.70050	0.69800	0.27665	15	2
SVC	0.70650	0.70900	0.34434	15	2
Decision Tree	0.64650	0.67000	0.33230	11	2

Conclusion

During this project, I understand what machine learning more. In order to finished this project. I did many study on the skills and knowledge on the internet, especially for the usage of scitkit-learn module. Besides, I also noticed that the structure and nature of the feature dataset do affect the result deeply. To work on create useful feature dataset with original dataset should be a big knowledge and needed much experience. Of course, with this course and the project, I did learn lots of skills and knowledge about not only machine learning but also python. I was glad that I have learn how to do basic machine learning with python and then where and how to find the resource to train myself.

Besides, the previous reviewer did help me very much, especially for the advices he gave. I reviewed my project again. And then digged the data a little more. It is interesting to explore data more so that the model I built became more realiable. About the future work, I would like to train my model with different feature set to make the model more realible. This time, I just had a look at how the results change with changing the features. As what I expected, the feature selection and engineering data did have impact on the model training. This project made me learned lots about machine learning, which strengthen my will to take **Machine Learning of Udacity** as the next step of my plan.

Script

File	Description
poi_id.py	main script to train POI classification model
poi_data.py	fix non-consistent data, reform data structure and count the data loss
poi_add_feature.py	add new feature to dataset with original dataset
poi_pipeline.py	build pipeline for GridSearch of each algorithm
poi_validate.py	validate the model whoes parameters are tuned by GridSearch
poi_plot.py	create the figures about data loss count
tester.py	a tools provided by Udacity to test trained model
tools/feature_format.py	a tools provided by Udacity to reform dataset for machine learning

Reference

Udacity	https://www.udacity.com/
Enron Corporation	https://en.wikipedia.org/wiki/Enron
The Enron Corpus	https://en.wikipedia.org/wiki/Enron_Corpus
Support Vector Classifier	http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
Linear Support Vector Classifier	http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
Logistic Regression	http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
Decision Tree	http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
KMeans	http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
GridSearchCV	http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html
Pipeline	http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
StratifiedShuffleSplit	http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html
StratifiedShuffleSplit	http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html
LogReg vs LSVC 1	https://www.quora.com/What-is-the-difference-between-Linear-SVMs-and-Logistic-Regression
LogReg vs LSVC 2	http://stats.stackexchange.com/questions/95340/comparing-svm-and-logistic-regression
What is the influence of C in SVMs with linear kernel?	http://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel
Introduction to Machine Learning with Python and Scikit-Learn	http://kukuruku.co/hub/python/introduction-to-machine-learning-with-python-andscikit-learn
Using Pipeline and GridSearchCV for More Compact and Comprehensive Code	https://civisanalytics.com/blog/data-science/2016/01/06/workflows-python-using-pipeline-gridsearchcv-for-compact-code/