Data exploration for the most important characteristic of the dataset:

Here we have the 146 data points from the given dataset. The allocation across classes of POI and non-POI are 18 and 128. There are 21 features in the existing dataset and in the final feature selection I used 20 features. I just remove one feature for the existing dataset "email_address". I created 14 new features for training. Almost all the features have missing values. The missing value count is as following:

salary: 49
to_messages: 57
deferral_payments: 105
total_payments: 20
long_term_incentive: 78
loan_advances: 140
bonus: 62
restricted_stock: 34
restricted_stock_deferred: 126
total_stock_value: 18
shared_receipt_with_poi: 57
from_poi_to_this_person: 57
exercised_stock_options: 42
from_messages: 57
other: 52
from_this_person_to_poi: 57
deferred_income: 95
expenses: 49
email_address:32
director_fees: 127

All the missing values have been assign zero.

Outlier investment

After reading the document of Enron insider pay and check the raw dataset, we found there are three outlier data point, TOTAL, THE TRAVEL AGENCY IN THE PARK and LOCKHART EUGENE E with zero value.

Create new features

I created 14 new features and all the new features are ratio type. The justification for these features, such as "from_poi_to_this_person_to_from_message_ratio", is the people having a higher possibility if they have a higher "from_poi_to_this_person_to_from_message_ratio". I supposed it could involve so many features can provide a better performance. I was wrong and I removed the feature I created. The performance of the identifier is better than before.
Here are two comparisons of the feature selection.

1. Remove all the features that I created.

Accuracy: 0.82140 Precision: 0.38923 Recall: 0.59650 F1: 0.47108 F2: 0.53909

2. Add three new created feature.

Accuracy: 0.79587 Precision: 0.32193 Recall: 0.48000 F1: 0.38539 F2: 0.43708

Intelligently select features

I used two approaches to select the features.
The first approach is selected automatically by using SelectKBest in the first step of the Pipeline. But the predicted score is not quit well. The result of the test is as following for K = [10,15]:

K= 10: Score:0.7 Accuracy: 0.70160 Precision: 0.22155 Recall: 0.49250
K= 15: Score:0.7 Accuracy: 0.69173 Precision: 0.19389 Recall: 0.41550

For the second approach, I selected the feature manually. I paste all the feature and new feature. The score is as following:
33 features:Score:0.7 Accuracy: 0.73607 Precision: 0.23128 Recall: 0.42150 F1: 0.29867 F2: 0.36196
Finally, I removed all the new feature and the score, precision and recall increase dramatically.
19 features:score:0.8 Accuracy: 0.82893 Precision: 0.40407 Recall: 0.59600 F1: 0.48162 F2: 0.54429

Properly scale features

I used the MinMaxScaler to re-scale all the feature in the dataset and use Pipeline chain the step for the first place.

Pick an algorithm

I created 7 algorithms and execute each of them to compare the performance. As you mention in the previous review. K-Mean is an unsupervised learning algorithm that does not attend to the targets of my training point, just remove it. Most of the precision and recall metrics are above the 0.3 except tuned logic regression. Here are the results of metrics:

Algorithm	Accuracy	Precision	Recall	F1	F2
GaussianNB	0.8142	0.28718	0.2655	0.27592	0.26957
Tree	0.8142	0.28718	0.2655	0.27592	0.26957
SVM	0.79473	0.25775	0.287	0.27159	0.28063
Random Forest	0.85153	0.37913	0.17800	0.24226	0.1993
Ada Boost	Too long training time	Too long training time	Too long training time	Too long training time	Too long training time
SGD	0.84727	0.30626	0.11500	0.16721	0.13141
Logic Regression	0.82893	0.40407	0.59600	0.48162	0.54429

Discuss parameter

The different parameter can address to different predict result. I only use C parameter of logic regression as an example to explain the different parameter performance:

C value	Accuracy	Precision	Recall	F1	F2
1e7	0.82893	0.40407	0.59600	0.48162	0.54429
1e6	0.82920	0.40494	0.59850	0.48305	0.54628
1e4	0.82227	0.39230	0.60650	0.47643	0.54679
1e2	0.77513	0.29366	0.48850	0.36681	0.43127
10	0.7598	0.27391	0.48550	0.35023	0.42053
Default value	0.74287	0.26344	0.51700	0.34903	0.43354

Tune the algorithm

GirdSearchCV used for parameter tuning (Done)
Several parameters tuned(Done)
Parameter tuning incorporated into algorithm selection(Done).
The parameters is as following:
lgr_parameters =
{"C":[1,5,10,1000,10000,10000000],'class_weight':['balanced'],'dual':[False],"fit_intercept":[True],'intercept_scaling':[1],'max_iter':[100],'multi_class':['ovr'],'penalty':['l2'],'random_state':[None],'solver':['libl inear'],'tol':[1e-05],'verbose':[0]}
The performance is as following, it is lower than directly using the logic regression. For the final version, I will directly using logic regression identifier.
Accuracy: 0.80800 Precision: 0.35915 Recall: 0.56100 F1: 0.43794 F2: 0.50431

Usage of Evaluation Metrics

There five metrics in the tester.py, accuracy, precision, recall, F1, F2
Accuracy: It measures how much the identifier accurately predicts the target whether POI or not. If the value is 0.8 which means 80% person predict as POI or Non-POI and these people are actually POI or Non-POI. The other 20% person predict as POI or Non-POI, but they are actually not a POI or Non-POI.
Precision: It measures the how much the accuracy for the identifier to predicted individual belongs to POI. If this value is high, the identifier has low chance to predict the Non-POI as a POI that also means it will not get too many false alarm.
Recall: It measures the how much the accuracy for identifier has good label of POI. If this value is high, the identifier has a low chance to predict the POI as a Non-POI.

Discuss validation and its importance

The goal of the validation is to measure the quality of identifier you created. It separate entire dataset into two parts, training data and testing data. But there are some trade off for splitting the data. If you maximize you training data, you can get a good training result. If maximizes the testing data. You can get a best validation. The importance of the validation is you can easily use validation to find the quality of the identifier improves or not.

Validation Strategy

Our validation strategy is using StratifiedShuffleSplit to split the data into 1000 parts. For each of 999 part to train the identifier and the 1 to predict the values. Then use the predicted value to compare test dataset and count how many true_negatives, false_negatives, true_positives and false_positives. After loop all the splitting part, the four metrics number have been come out. We can use these four metrics to make validation.

Algorithm Performance

The previous performance is as following:

Precision:0.30311
Recall:0.51650
Current performance is as following:
Precision: 0.40407
Recall: 0.59600