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”]

This project is to build a person of interest identifier based on financial and email data made public as a result of the Enron scandal. The features in the data fall into three major types, namely financial features, email features and POI labels. And POIs are individuals who were indicted, reached a settlement, or testified in exchange for prosecution immunity. Machine Learning algorithms could be useful in identifying POIs by determining underlying patterns or relationships in the financial and email information and then applying them to predict POIs. There are 146 data points (people) in the Enron dataset, each people with 21 features. However, not every feature for each people has a value. It is NaN when a feature doesn’t have a well-defined value. What’s more, in this dataset, there are 18 POIs and the rest are Non-POIs.

As to the outlier problem, using a strategy of removing the top 10% of values with the largest residual errors seems more appropriate for a regression problem where to predict quantitative value, not this problem which is a classification problem with only two results: Non-POI and POI. So I just make it visualized by the feature ‘salary’ and ‘bonus’, and there is a very big point in the picture. I looked into the document ‘enron61702insiderpay.pdf ’ and the key of the biggest point is ‘TOTAL’. For this outlier, we should take it out since it is a spreadsheet quirk, which is added up all the data points by spreadsheet, not tied to an individual. After delete this point, there were other outliers associated with the financial data for some of the POIs (i.e., Kenneth Lay, Jeffrey Skilling, etc.), but they are valid data points for the further study, which are just biggest bosses in Enron.

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”]

For feature selection, I firstly calculate the fraction of valid data points (exclude NaN values) was calculated for each feature across three classes: everyone in the dataset, non-POIs in the dataset, and POIs in the dataset. The result is listed below:

```
{'salary': [0.65, 0.61, 0.94], 'to_messages': [0.59, 0.57, 0.78], 'deferral_payments':
[0.26, 0.26, 0.28], 'total_payments': [0.86, 0.83, 1.0], 'exercised_stock_options':
[0.7, 0.7, 0.67], 'bonus': [0.56, 0.51, 0.89], 'restricted_stock': [0.75, 0.72, 0.94],
'restricted_stock_deferred': [0.12, 0.13], 'deferred_income': [0.33, 0.29, 0.61],
'total_stock_value': [0.86, 0.84, 1.0], 'shared_receipt_with_poi': [0.59, 0.57, 0.78],
'loan_advances': [0.02, 0.02, 0.06], 'from_messages': [0.59, 0.57, 0.78], 'other':
[0.63, 0.58, 1.0], 'from_this_person_to_poi': [0.59, 0.57, 0.78], 'poi': [1.0, 1.0, 1.0],
'director_fees': [0.11, 0.13], 'long_term_incentive': [0.45, 0.42, 0.67], 'expenses':
```

```
[0.65, 0.6, 1.0], 'email_address': [0.77, 0.73, 1.0], 'from_poi_to_this_person': [0.59,
0.57, 0.78]}
```

For the financial features, the ‘total_payments’, ‘total_stock_value’, ‘expenses’ and ‘other’ have 100% coverage for POIs and non 100% coverage for non-POIs. This could lead to the classifier simply using the existence (or non-existence) of those features on a testing point to predict if it was a POI, instead of using the values. So those features were discarded. In addition, ‘restricted_stock_deferred’, and ‘director_fees’ both had no POI data coverage, so they were discarded as the same reason as above. For the email features, the ‘email_address’ feature had 100% POI coverage, so it was also discarded. Then, I use SelectKBest to choose the most influential 10 features, and all the features and their scores are listed below.

Because ‘POI’s are probably have very high bonus and frequent interaction with each other, so I create two new features 'fraction_bonus' and 'fraction_message_with_poi', which show the fraction of bonus of total payments and the interaction to ‘POI’ for individual person, respectively. I run the original feature_list for SelectKBest , and feature_list_new with the two new features for SelectKBest calculation, respectively.

The two outputs are sorted and listed below:

1. Original feature_list and their scores:

```
[('exercised_stock_options', 23.968332134035151), ('bonus',
19.989875526651559), ('salary', 17.432087382499653), ('deferred_income',
11.047759080679361), ('long_term_incentive', 9.47251713657446),
('restricted_stock', 8.8120939338279367), ('shared_receipt_with_poi',
8.1199380824959668), ('loan_advances', 7.0080408695438905),
('from_poi_to_this_person', 4.940571962715488), ('from_this_person_to_poi',
2.2516565052501232), ('to_messages', 1.4913280996536644),
```

('deferral_payments', 0.24851659019005992), ('from_messages', 0.1872002308412164)]

2. New feature_list and their scores:

[('exercised_stock_options', 23.968332134035151), ('bonus', 19.989875526651559), ('fraction_bonus', 19.947513782033049), ('salary', 17.432087382499653), ('deferred_income', 11.047759080679361), ('long_term_incentive', 9.47251713657446), ('restricted_stock', 8.8120939338279367), ('shared_receipt_with_poi', 8.1199380824959668), ('loan_advances', 7.0080408695438905), ('fraction_message_with_poi', 5.0946702203046721), ('from_poi_to_this_person', 4.940571962715488), ('from_this_person_to_poi', 2.2516565052501232), ('to_messages', 1.4913280996536644), ('deferral_payments', 0.24851659019005992), ('from_messages', 0.1872002308412164)]

After comparing these two results, we could see the two new features added are relatively important in this dataset. So I will add them in my dataset.

Then, I tune the K parameter in GridSearchCV for Guassian NB and KNeighbors algorithms (DecisionTree has different way to calculate the importance of features). And my final algorithm is GaussianNB without PCA with the best k=4. So I choose the most important four features as my feature list:

['exercised_stock_options', 'bonus', 'fraction_bonus', 'salary']

While it isn’t necessary for all of the machine learning algorithms, I used the rescaling MinMaxScaler for Gaussian NB.

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”]

I choose Gaussian NB without PCA and rescaling as my final algorithm. I compare 4 algorithms: Gaussian NB, DecisionTree, KNeighbors and Adaboost, each with performing PCA and without PCA, respectively.

Table 1 Comparison the results of different algorithms

Classifier	Precision	Recall	F1 Score	PCA
Gaussian NB	0.416	0.274	0.330	Performing
	0.415	0.308	0.354	No
DecisionTree	0.344	0.316	0.329	Performing
	0.312	0.261	0.284	No
KNeighbors	0.598	0.239	0.342	Performing
	0.624	0.226	0.331	No
Adaboost	0.299	0.233	0.262	Performing
	0.350	0.257	0.296	No

From Table 1, we could see that the Gaussian NB without performing PCA has the best scores for precision, recall and F1 score among these four algorithms.

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"]

Tuning the parameters of an algorithm means changing the algorithm's input parameters and measuring the performance of each combination of them in order to determine the optimal combination of input parameters. If tuning the parameters is not done well, the result may be not good enough. Since the optimal combination would give the better result than what we get without tuning parameters, then we will miss the “best” parameters for the problem.

In this project, I will give the example how I tuned the parameters for DecisionTree classifier with performing PCA. Using ‘pipeline’ to operate PCA and DecisionTree classifier at one time firstly. Then applying ‘GridSearchCV’ in the operation in order to find the optimal combination parameters. The value set I give to different parameters as below:

- PCA(n_components): [2,4,6,8,10]
- DecisionTreeClassifier('clf__min_samples_split): [2,3,4,5,8]

After it runs all these parameter combinations, the best combination is:

'pca__n_components': 8, 'clf__min_samples_split': 3

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”]

Validation refers to the process of training and testing in a machine learning algorithm in order to assess its performance, and to prevent overfitting. The overfitting is also the classic mistake if you don’t do it. A model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. To avoid it, it is common practice when performing a supervised machine learning experiment to hold out part of the available data as a test set. In this Enron dataset project, I use StratifiedShuffleSplit cross validation since this dataset has a small size and unbalanced data. Moreover, it is split into 1000 folds in this project and get the mean of each different score as final output.

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"]

From my final algorithm with the optimal parameter combination, the result is:

Precision: 0.50220 Recall: 0.34300 F1: 0.40761

Precision is dividing the number of true positives by the total number of positive points, regardless of if they are predicted correctly. In this case, it means that when the algorithm identifies a point is POI, there is 50.22% possibility that it identification is correct. If this value is low, that means more non-POIs are flagged wrong as POIs.

Recall is dividing the number of true positives by the sum of true positives and false negatives, which means the ratio of the correct POI prediction to the all number of POIs in this dataset. In this case, when the algorithm predicts a POI point, there is 34.30% possibility to make a right prediction and flag it as POI. If this value is low, that means more POIs are flagged as non-POIs, when a POI shows up in the test set, I am not able to identify it.

F1 score is a weighted average of precision and recall. If this score is high, then both false positive and false negative rates are high. So I can identify POI's reliably and accurately. If my identifier finds a POI then the person is almost certainly a POI, and if the identifier does not flag someone, then they are almost certainly not a POI.

Reference

1. UdacityProjectDetails
   https://classroom.udacity.com/nanodegrees/nd002/parts/0021345409/modules/317428862475461/lessons/3174288624239847/concepts/31803986370923
2. Identifying Fraud From Enron Email
   http://broadwater.io/identifying-fraud-from-enron-email/
3. Scikit-Learn
   http://scikit-learn.org/stable/index.html
4. K-Fold Cross Validation and GridSearchCV in Scikit-Learn
   https://randomforests.wordpress.com/2014/02/02/basics-of-k-fold-cross-validation-and-gridsearchcv-in-scikit-learn/