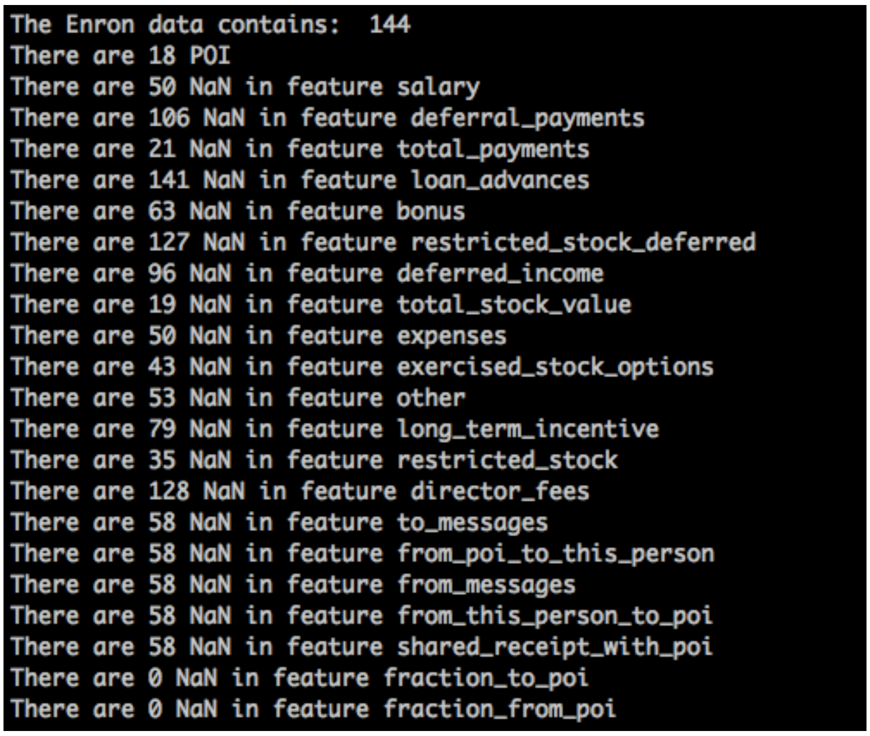
Enron Submission Free-Response Questions

Wei Zhang

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"]**

The goal for this project is to find the POI by machine learning.

The dataset is explored as shown in the picture:



```
The Enron data contains:  144
There are 18 POI
There are 50 NaN in feature salary
There are 106 NaN in feature deferral_payments
There are 21 NaN in feature total_payments
There are 141 NaN in feature loan_advances
There are 63 NaN in feature bonus
There are 127 NaN in feature restricted_stock_deferred
There are 96 NaN in feature deferred_income
There are 19 NaN in feature total_stock_value
There are 50 NaN in feature expenses
There are 43 NaN in feature exercised_stock_options
There are 53 NaN in feature other
There are 79 NaN in feature long_term_incentive
There are 35 NaN in feature restricted_stock
There are 128 NaN in feature director_fees
There are 58 NaN in feature to_messages
There are 58 NaN in feature from_poi_to_this_person
There are 58 NaN in feature from_messages
There are 58 NaN in feature from_this_person_to_poi
There are 58 NaN in feature shared_receipt_with_poi
There are 0 NaN in feature fraction_to_poi
There are 0 NaN in feature fraction_from_poi
```

The outliers are found in two ways:

1. By visualization: We can found the TOTAL outliers easily
2. Finding an outlier that is actually not a name, like THE TRAVEL AGENCY IN THE PARK
3. "LOCKHART EUGENE E" is all "NaN", has no valid information.

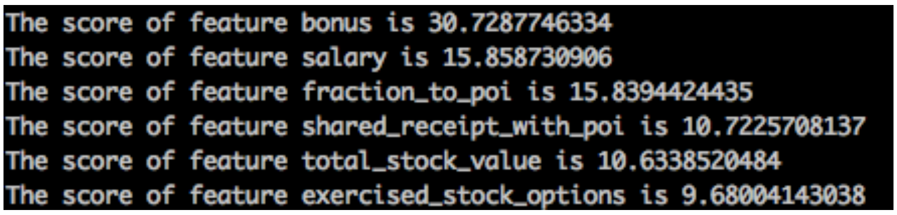
These two should be removed from the dataset.

There are extra data points looks like outliers, but they are POIs, so we should keep the data.

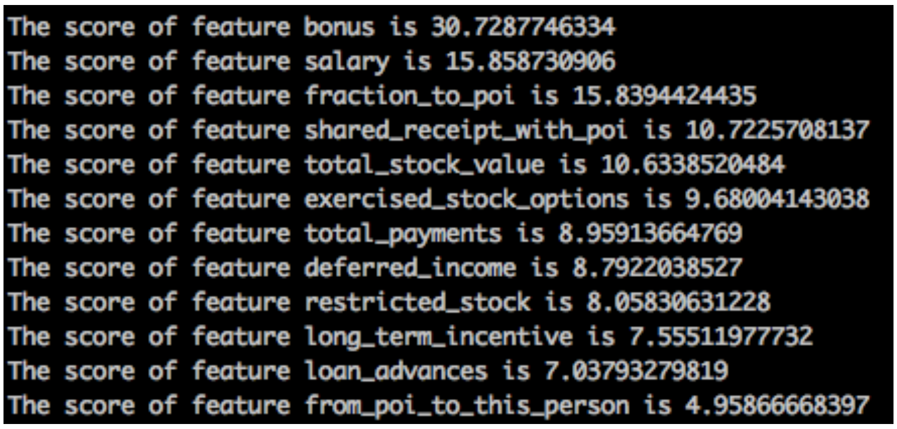
2. **What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what 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. [**

Before analyzing using model, I created two extra features, 'fraction_to_poi' and 'fraction_from_poi'. I have picked 21 features except 'email_address'.
During the exploration, I found out that there are negative values in some features like "exercised_stock_options"," restricted_stock_deferred". So MaxMinScaler is used to scale them into good format.

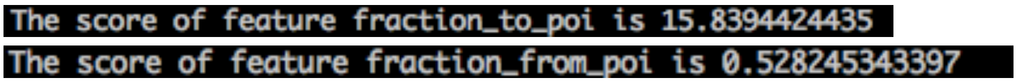
Different algorithms the SelectKBest has picked different combination of features, like during NB classification, I picked 6 features including:


```
The score of feature bonus is 30.7287746334
The score of feature salary is 15.858730906
The score of feature fraction_to_poi is 15.8394424435
The score of feature shared_receipt_with_poi is 10.7225708137
The score of feature total_stock_value is 10.6338520484
The score of feature exercised_stock_options is 9.68004143038
```

While in adaboost, I picked 12 features of them:


```
The score of feature bonus is 30.7287746334
The score of feature salary is 15.858730906
The score of feature fraction_to_poi is 15.8394424435
The score of feature shared_receipt_with_poi is 10.7225708137
The score of feature total_stock_value is 10.6338520484
The score of feature exercised_stock_options is 9.68004143038
The score of feature total_payments is 8.95913664769
The score of feature deferred_income is 8.7922038527
The score of feature restricted_stock is 8.05830631228
The score of feature long_term_incentive is 7.55511977732
The score of feature loan_advances is 7.03793279819
The score of feature from_poi_to_this_person is 4.95866668397
```

Using SelectKBest(), the new features' score is tested.


```
The score of feature fraction_to_poi is 15.8394424435
The score of feature fraction_from_poi is 0.528245343397
```

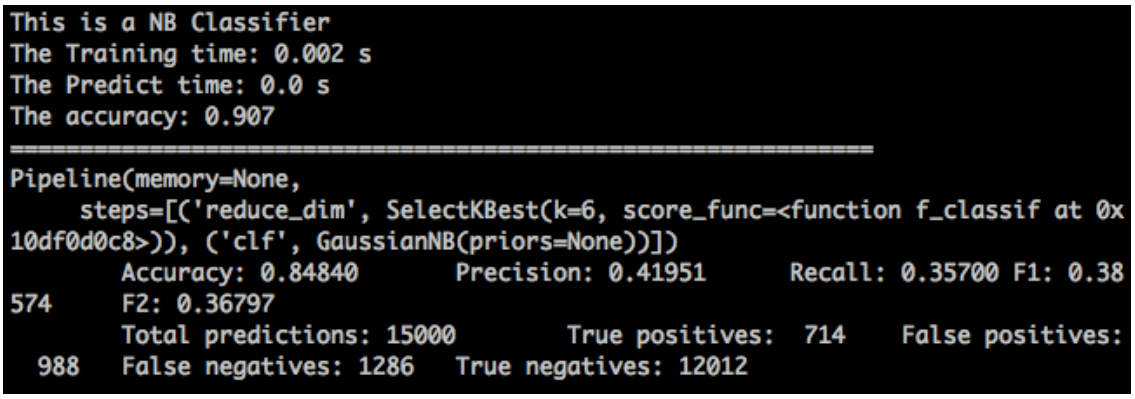
With he feature of fraction_to_poi, the result is using Gaussian_NB():
Accuracy: 0.84840 Precision: 0.41951 Recall: 0.35700 F1: 0.38574 F2: 0.36797

Without the new features, the result is :
Accuracy: 0.84213	Precision: 0.38889	Recall: 0.32200		F1: 0.35230	F2: 0.33347

So the new feature actually helped improve the models' both precision and recall rate. New feature 'fraction_to_poi' is selected to the final features.

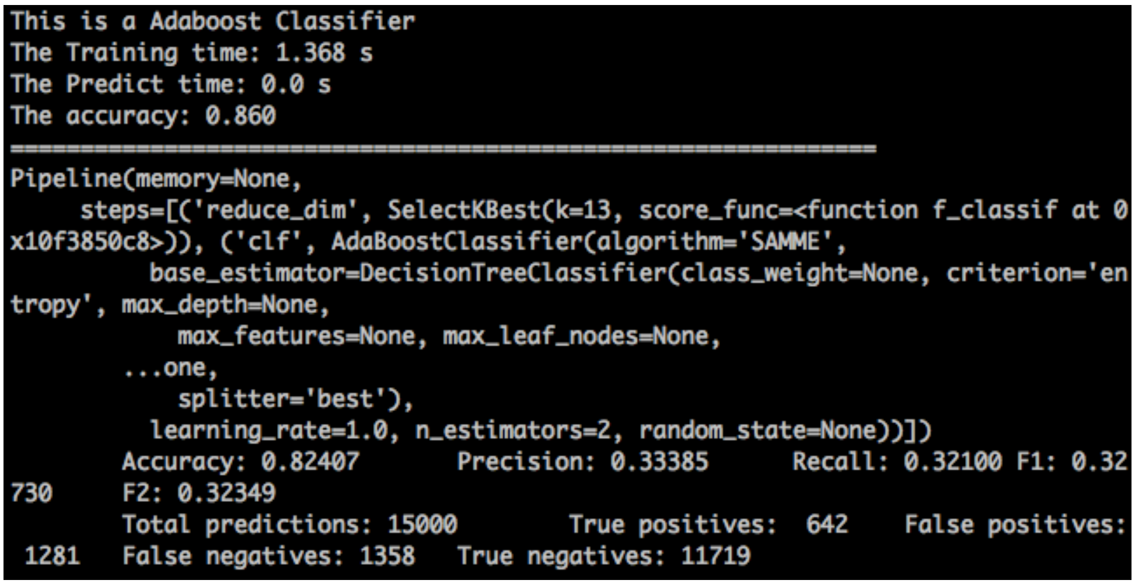
3. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]**

I have tried three algorithm: GaussianNB(), AdaBoostClassifier(), KNeighbors_classifier()
The algorithm with best performance is GaussianNB().



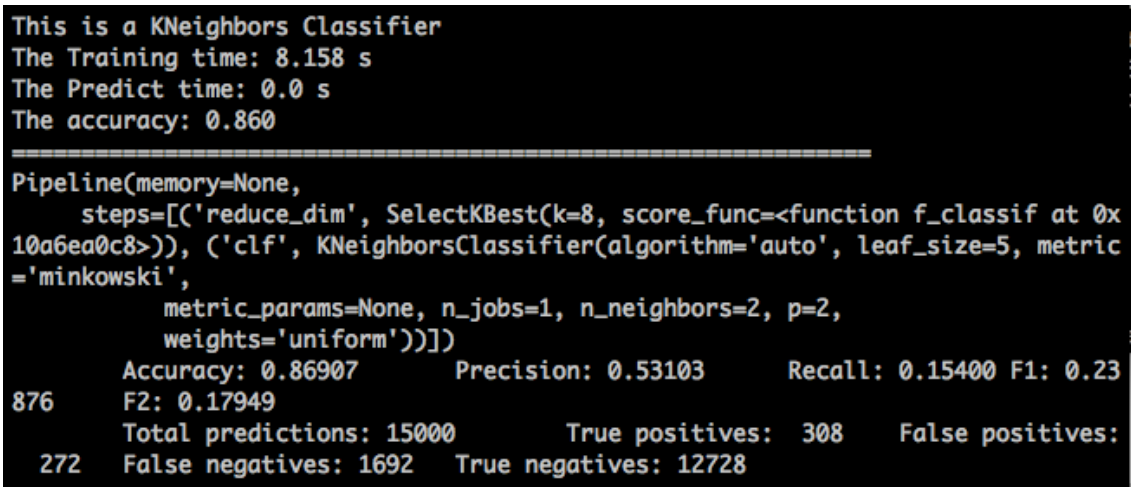
```
This is a NB Classifier
The Training time: 0.002 s
The Predict time: 0.0 s
The accuracy: 0.907
===============================================================
Pipeline(memory=None,
     steps=[('reduce_dim', SelectKBest(k=6, score_func=<function f_classif at 0x
10df0d0c8>)), ('clf', GaussianNB(priors=None))])
        Accuracy: 0.84840       Precision: 0.41951      Recall: 0.35700 F1: 0.38
574     F2: 0.36797
        Total predictions: 15000        True positives:  714    False positives:
  988   False negatives: 1286   True negatives: 12012
```

The stability of AdaBoostClassifier() is not stable, the performance is bellow:



```
This is a Adaboost Classifier
The Training time: 1.368 s
The Predict time: 0.0 s
The accuracy: 0.860
===============================================================
Pipeline(memory=None,
     steps=[('reduce_dim', SelectKBest(k=13, score_func=<function f_classif at 0
x10f3850c8>)), ('clf', AdaBoostClassifier(algorithm='SAMME',
          base_estimator=DecisionTreeClassifier(class_weight=None, criterion='en
tropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
        ...one,
            splitter='best'),
          learning_rate=1.0, n_estimators=2, random_state=None))])
        Accuracy: 0.82407       Precision: 0.33385      Recall: 0.32100 F1: 0.32
730     F2: 0.32349
        Total predictions: 15000        True positives:  642    False positives:
 1281   False negatives: 1358   True negatives: 11719
```

KNeighbors_classifier performed well in precision, but bad in recall:


```
This is a KNeighbors Classifier
The Training time: 8.158 s
The Predict time: 0.0 s
The accuracy: 0.860
==============================================================
Pipeline(memory=None,
     steps=[('reduce_dim', SelectKBest(k=8, score_func=<function f_classif at 0x
10a6ea0c8>)), ('clf', KNeighborsClassifier(algorithm='auto', leaf_size=5, metric
='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=2, p=2,
           weights='uniform'))])
        Accuracy: 0.86907       Precision: 0.53103      Recall: 0.15400 F1: 0.23
876     F2: 0.17949
        Total predictions: 15000        True positives:  308    False positives:
  272   False negatives: 1692   True negatives: 12728
```

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"]**

If I didn't use tune the parameters of an algorithm, the result of accuracy, precision, recall, F1, F2 will not be the best result.

In this project, I picked two good algorithms with default parameters based on the performance of precision and recall: GaussianNB() and adaboost_classifier.

GaussianNB() do not have many parameters, so I implemented a pipeline with a reduce_dim made by selectKBest(K = 6), while the K is 6, the precision and recall represented the best performance.

	Accuracy	Precision	Recall	F1	F2
GaussianNB(default)	0.73900	0.226	0.395	0.28	0.34
GaussianNB()_best	0.8484	0.42	0.36	0.36	0.37

So by tuning the parameters, the model has been modified and improved in 5 evaluation metrics.

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"]**

Using the small dataset for building models, splitting them as training set and testing set to build the model, I cannot guarantee the over fit problem. So validation the performance of the models is necessary. During validation, the project used stratified shuffle split method. The dataset is divided into K folds, which kept the features rate of every dataset. Using new training set and testing set, we can avoid overfiting problem and improved the generalization ability.

The main problem that I encountered is that I achieved a high accuracy in my dataset, but received a poor result in precision and recall while using validation.

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"]**

The main evaluation is Precision and Recall evaluation metrics,

As the sample dataset is small, the balance is not considered to be good, so using precision and recall to analyze the quality of the model is necessary.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	true positives (tp)	false positives (fp)
	Negative	false negatives (fn)	true negatives (tn)

Total = tp + fp + fn + tn
Accuracy = (tp + fp) / Total

// because of the imbalance of the small dataset, using precision to analyze the rate of true positive prediction among the actual positives
Precision = tp / (tp + fp)

// Using recall to analyze the rate of true positive prediction among predicted positives
Recall = tp / (tp + fn)

// We can also use F1,F2 to analyze the performance of the models

F1 = 2 * Precision * Recall / (Precision + Recall)
F2 = (1 + 2 * 2) * Precision * Recall / (4 * Precision + Recall)

Using GaussianNB(), TOTAL Predictions are 15000,

		Predicted	
		Positive	**Negative**
Actual	**Positive**	642	1358
	Negative	1281	11719

So the Prediction is 0.42, the Recall is 0.36.

In this case, using GaussianNB, we predicted 1923 POI, the actual POI is 642, the actual amont of POI is 2000.