Enron Submission Free-Response Questions Â¶

Enron corporation was an American energy commodities and servies company, and was one of the world's major electricity, natural gas, communications and pulp and paper companies. At the end of 2001, it was revealed that its reported financial condition was sustained by institutionalized, systematic and creatively planned accouting fraud.

With the development of machine learning technique, we are going to try a new method to assist the inspection institute and court to invest the case of the inner corruptions of Enron. We will use the features from the financial data and email data to label the person of interest, who has high chance to be related with the fraud.

Question 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?

The goal of our project is to predict the person of interest *poi* from the records of the features from the dataset. We will use a few machine learing technique to train our data and select the best fit algorithm and parameters to predict the potential poi person from the other features avaiable in dataset.

In our datasets, we have total 146 records. We also have total 21 features, including 14 features related to finance records, 6 features related to email communication, and 1 feature named *poi* for prediction. So there are total 20 features that could be used to train and predict for *poi* person.

The number of marked *poi* records is only 18, and there are 128 records marked as *non-poi* persons. So the distribution of *poi* in all records in dataset is quite rare.

In our dataset, we have a few outliers. *TOTAL* appears in the key of the dictionary of dataset and should be removed since it does not record a person's information. We also remove record with key *LOCKHART EUGENE E* since all the features have NaN values except poi equal to false. This record will not provide us any useful information to predict the poi from the other features. The total number of records after removing outliner is 144. We also remove the feature named email_address since it is used to identify a person not related to *poi* feature. Since *poi* is also a feature but prediction, we only use the 19 featuers to predict the *poi* feature.

We also check the number of *NaN* record each feature contains.

Feature	Number of NaN
loan_advances	142
director_fees	129
restricted_stock_deferred	128
deferral_payments	107
deferred_income	97
long_term_incentive	80
bonus	64
to_messages	60
shared_receipt_with_poi	60
from_this_person_to_poi	60
from_poi_to_this_person	60
from_messages	60
other	53
salary	51
expenses	51
exercised_stock_options	44
restricted_stock	36
email_address	35
total_payments	21
total_stock_value	20

Question 2: 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

In order to select the best features for our poi identifer, we use the *SelectKBest* function from sklearn module. We first rank all the 19 features as follow:

Feature	Score
'exercised_stock_options'	25.097

'total_stock_value'	24.468
'bonus'	21.059
'salary'	18.576
'deferred_income'	11.596
'long_term_incentive'	10.071
'restricted_stock'	9.346
'total_payments'	8.867
'shared_receipt_with_poi'	8.746
'loan_advances'	7.243
'expenses'	6.234
'from_poi_to_this_person'	5.344
'other'	4.205
'from_this_person_to_poi'	2.427
'director_fees'	2.108
'to_messages'	1.699
'deferral_payments'	0.217
'from_messages'	0.164
'restricted_stock_deferred	0.065

What features did you end up using in your POI identifier, and what selection process did you use to pick them?

We select the best score features depend on the feature score. But we are not sure how many features that we should use. In order to find the optimal parameter k of SelectKBest function and the optimal number of features that will be used in machine learning algorithm, we tried the number of features k = 2, 4, 6, 8, 10, 12, 14. For example, if we select k = 2, we will pick the feature *exercised_stock_options* and *total_stock_value*, and so on. Later, we will compare the performance of our selected machine learning algorithm (using evaluation algorithm) on the number of selected features, to find the optimal number of features that should be used for machine learning.

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.

We made two new features in our datasets. The first new feature is called *from_poi_message_rate*. It is the ratio between the number of emails from poi email to this person and total number of emails to this person, that is,

$$\text{from\_poi\_message\_rate} = \frac{\text{from\_poi\_to\_this\_person}}{\text{to\_messages}}$$

The second new feature is called to_poi_message_rate, the rate between the number of emails from this person to poi and total number of emails from this person, that is,

$$\text{to\_poi\_message\_rate} = \frac{\text{from\_this\_person\_to\_poi}}{\text{from\_messages}}$$

The reason why we create the two new features is that each person may send and receive a lot of emails each day. There are a few emails in the sending emails and receving emails are from or to *poi*. If we only use the original features *from_poi_to_this_person* and *from_this_person_to_poi* to measure how many emails this person communicate with poi, it may not accurate since this person may have a lot of email communication as well, for instance, the secretary of department. So we create two new features measuring the ratio of each person communicating with *poi*. The more percent of email this person communicating with *poi*, the closer relation this person has with *poi*. As we expected, the *SelectKBest* function returns the higher score on these two new features than the original features.

Feature	**Score**
'to_poi_message_rate'	16.640
'from_poi_to_this_person'	5.344

Feature	**Score**
'from_poi_message_rate'	3.21
'from_this_person_to_poi'	2.427

We find the new features has better performance on our machine learning algorithm in our final evaluation section.

Did you have to do any scaling? Why or why not?

For all the selected features, we use the *MinMaxScaler* to scale the values of each records in dataset. Since each features are measured on different unit, for instance the payment is in USD and email is in number, the features on different measurement unit may have different weight on machine learning algorithm. In order to balance the features measured on different unit, we scale all the values in range [0, 1) using *MinMaxScaler* .

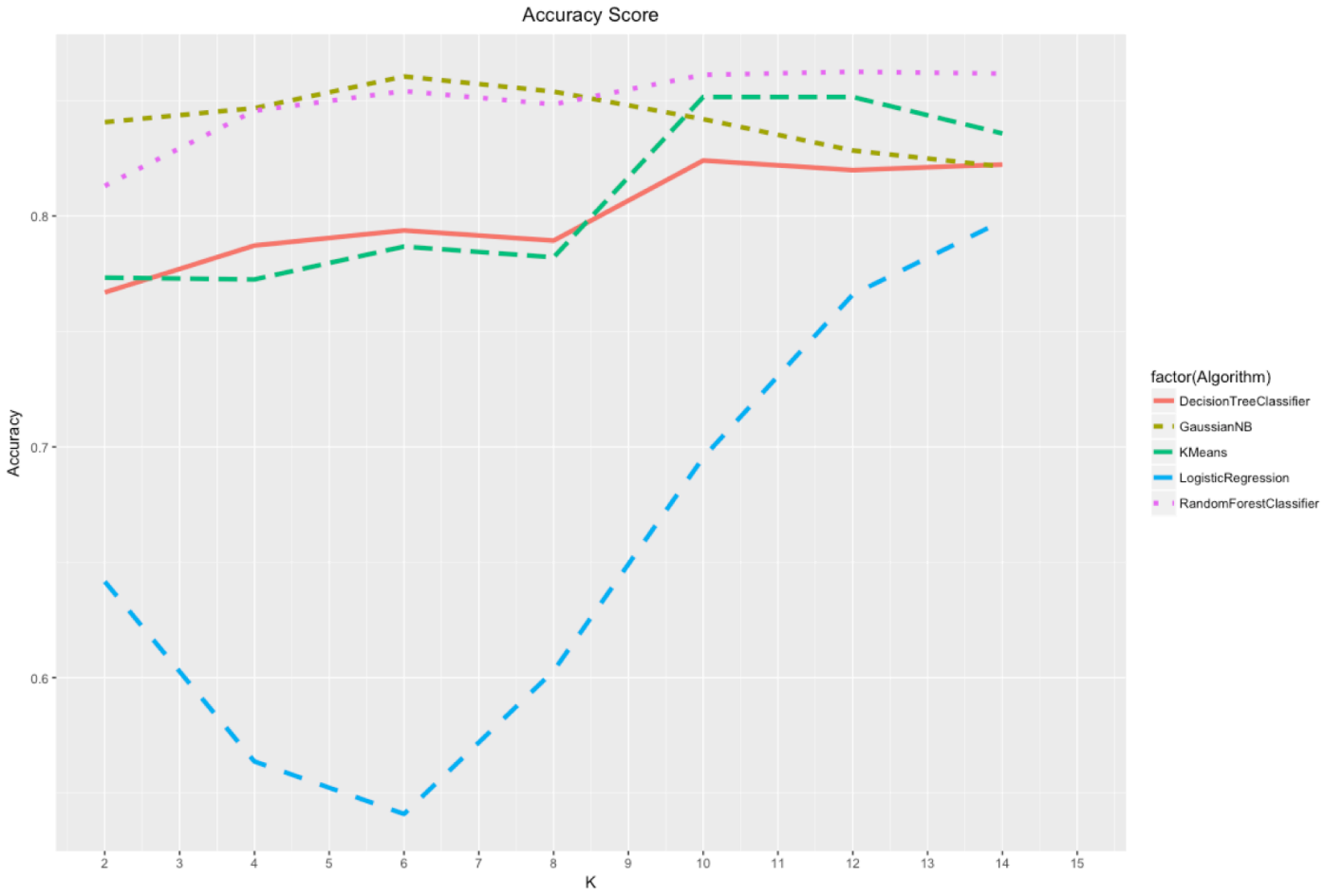
Question 3: What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

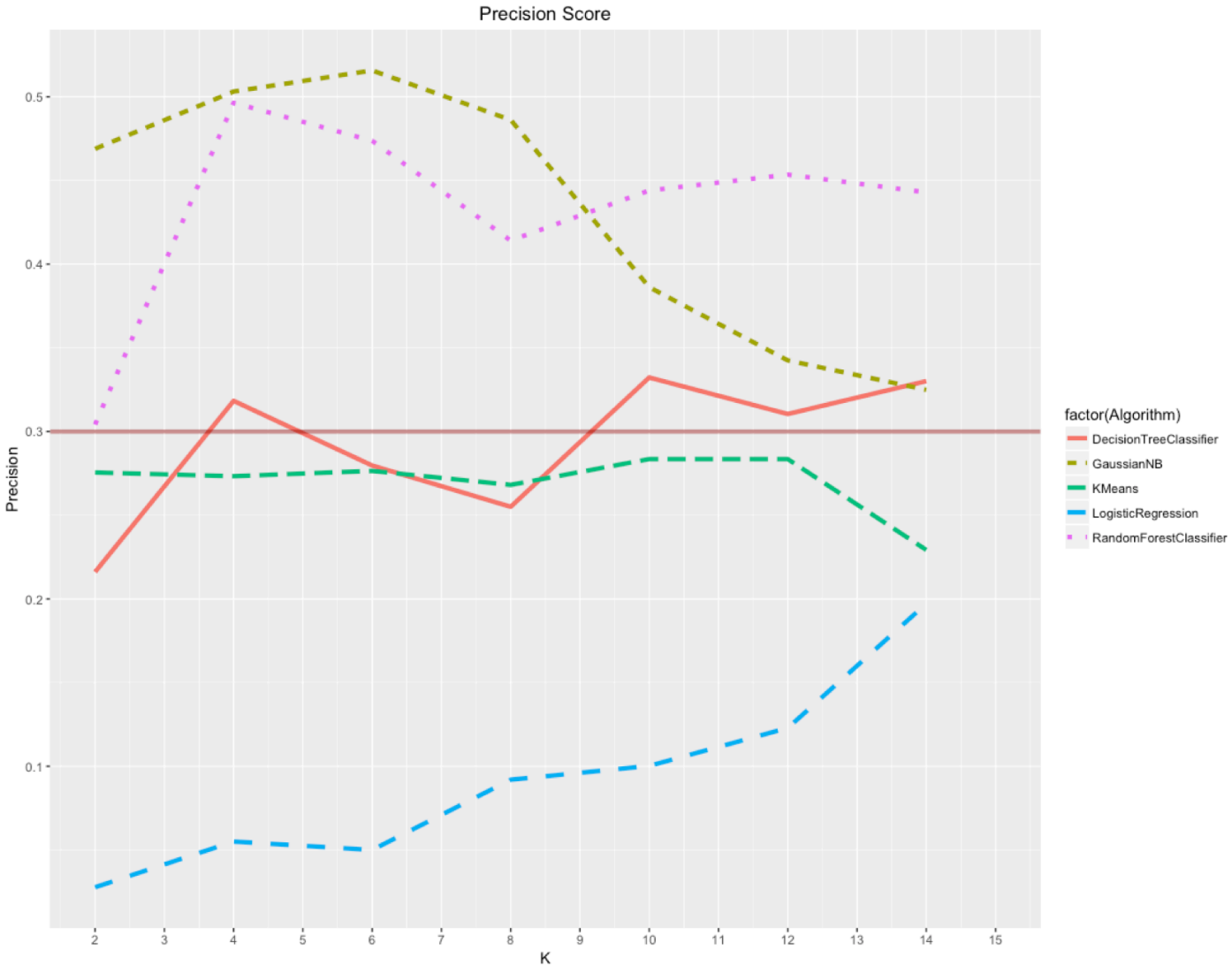
In this section, we tried five machine learning algorithm, *Naive Bayes*, *LogisticRegression, KMeans Cluster, DecisionTreeClassifier, RandomForestClassifier*, which are introduced in our course or mentioned in reading documents. In the inital phase of algorithm selecting, we use the default parameters of

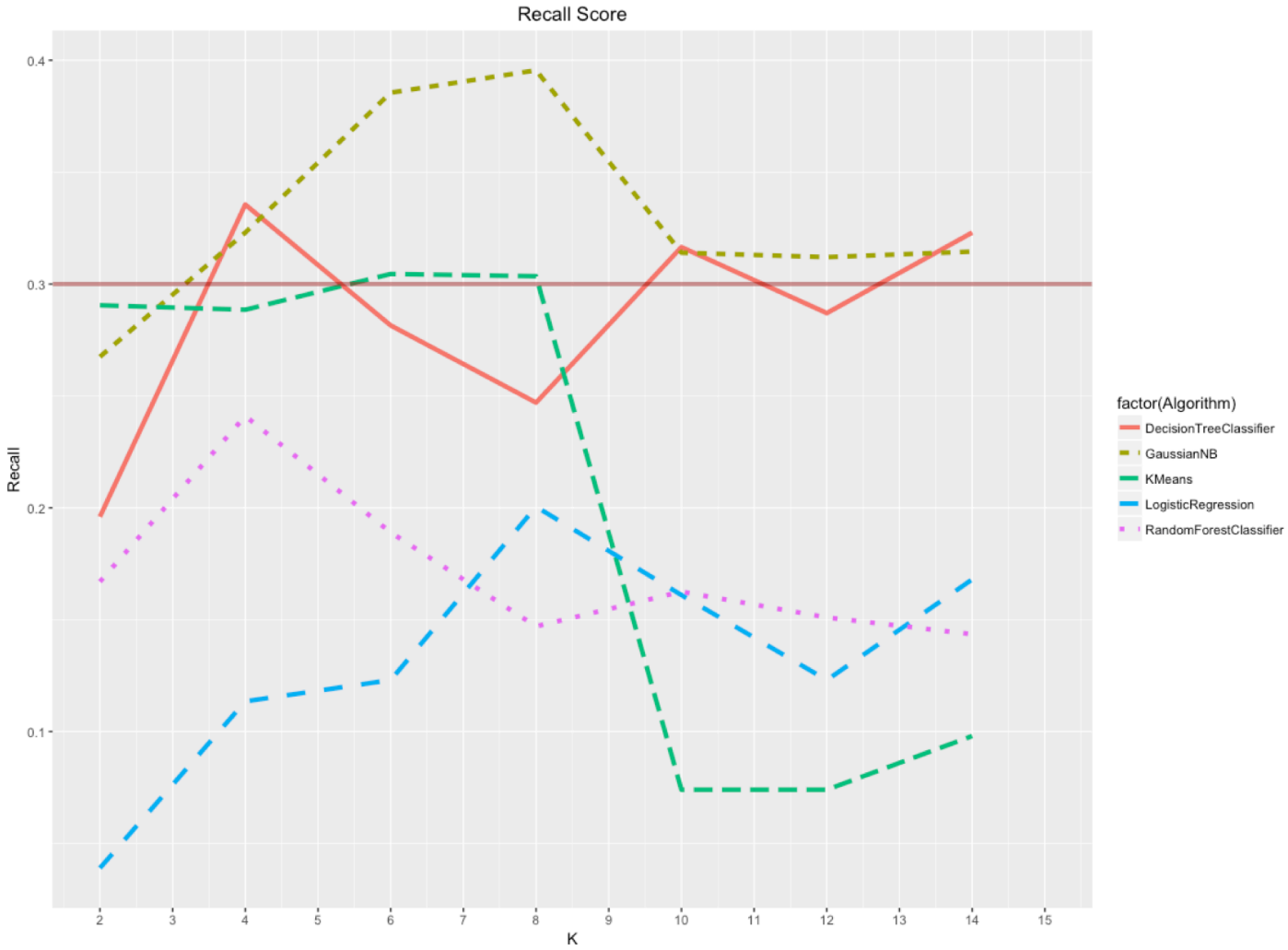
machine learning algorithm, except the *kmean* cluster we select the *n_clusters* equal to two, since there are only *non-poi* and *poi* in our predicted object. We also test each machine learning algorithm on select number of features $k = 2, 4, 6, 8, 10, 12, 14$.

In the begining we only use the accuracy_score measurement to test the prediction accuracy of our dataset. But we found that all the machine learning algorithms have a good performance under number of features. This is because the dataset has unbalanced values in labels: *poi*. The majority of *poi* are marked as false or zero. If the machine learning algorithm simply predict all the *poi* as false, it can still get a high accuracy score. Therefore, in order to make a better test evaluation of performance of each machine learning algorithm, we measure the performance by accuracy score, precision score and recall score.

For each machine learning algorithm, the accuracy, precision and recall are as follow:




Precision Score
Precision
K
0.5
0.4
0.3
0.2
0.1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
factor(Algorithm)
DecisionTreeClassifier
GaussianNB
KMeans
LogisticRegression
RandomForestClassifier



Based on the plots of accuracy, precision and recall measurement, the *Decision tree* and *kmean cluster* have the best performance. The other machine learning algorithms such as *RandomForest* may has a good performance on one measurement, but poor performance overall.

Question: 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? What parameters did you tune?

In this section, we try to find the optimal parameter of the machine learning algorithms selected in last section: *KMeans* and *DecisionTree*, in order to get the best performances.

To find the optimal parameters, we use two steps: In the first step, find the optimal number of features of each machine learning algorithm. In previous section, we tried to find the optimal number of features for each machine learn algorithm, that is $k = 2, 4, 6, 8, 10, 12, 14$, to find the optimal number of features. For *KMeans*, we find that the *KMeans* machine learning would perform the best if the number of features $k = 6$. It means we select the features: exercised_stock_options, total_stock_value, bonus, salary, deferred_income, long_term_incentive for *KMeans* algorithm; For *DecisionTreeClassifier* we find it performs well if the number of features $k = 4$. It means we select the features: exercised_stock_options, total_stock_value, bonus, salary for *DecisionTreeClassifier* algorithm.

In the second steps, we tried to find the optimal parameters of each machine learning algorityms. We use the hyperparameter tuning to find these optimal parameters.

> **Definition**: *In machine learning, the hyperparameters are parameters whose values are set prior to the commencement of learning process. By contrast, the values of other parameters are derived via training. Hyperparameter optimization or tuning is to choose a set of optimal hyperparameters for a learning algorithm. The same kind of machine learning model could require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can best solve the machine learning problem. Usually a metric is chosen to measure the algorithm's performance on an independent data set and hyperparameters that maximize this measure are adopted.*

The classic way of performing hyperparameter optimization is grid serach. In our case we need to find the hyperparameters for the two selected machine learning algorithms: *KMeans* and *DecisionTreeClassifier*. In detail, for each algorithm we use GridSearchCV of sklearn model to select the hyperparameter for each machine learning algorithm. For *KMeans* cluster algorithm we exhaustively search the hyperparameter *tol*, which is the relative tolerance with regards to inertia to declare convergence, That is `param_grid_kmean = { 'tol': [1e-11, 1e-10, 1e-9, 1e-8,  1e-7, 1e-6, 1e-5, 1e-4, 1e-3], }` to optimize *KMeans* parameter; For *DecisionTreeClassifier* we exhaustively search the hyperparameter *max_depth*, *min_samples_split* and *min_samples_leaf*, which are the maximum search depth, minimum number of samples required to split an internal node, and minimum number of samples required to be at a leaf node. That is `param_grid_dt = {'max_depth': [None, 2, 4, 6, 8, 10], 'min_samples_split' : [2, 4, 6, 10], 'min_samples_leaf' : [1, 2, 4, 8], }` to optimize *DecisionTreeClassifier* parameters.

> **Question: 5 What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?**

The validation of machine learning algorithm is to test if our machine learning algorithm has a good performance on new data. Therefore we need to use robust technique to train and evaluate our models on dataset. The more reliable of the validation we estimated on our model, the further we can push the performance and be confident on the operation of using our model.

There are a few potential mistakes we can make in our machine learning models. For example, we may over fitting our features so there are a good performance on training dataset but poor performance on the testing dataset and practical usage. In our dataset, we use the function *test_classifier* provided in the *tester.py* file to evaluate the performance of machine learning algorithm.

The *test_classifier* divide the dataset into training set and test set. The *test_classifier* use the *StratifiedShuffleSplit* to split the dataset into training set and test set. The benefit of using *StratifiedShuffleSplit* is to balanced split the dataset. The traing set and testing set will have the same percent of *poi*. It makes better splits than *train_test_split*, which I tried, without considering the balanced splitting. The *test_classifier* also set the parameters `folds = 1000` to loop the evaluation algorithm 1000 times. For each loop, *test_classifier* split the dataset into training set and testing set, calculate the number of samples of *true_negatives*, *false_negatives*, *false_positives* and *true_positives*. Finally the *test_classifier* sum the count of *true_negatives*, *false_negatives*, *false_positives* and *true_positives* in each experiments to generate the score of *accuracy*, *precision* and *recall*.

Question 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.

Our result of evaluation on the two selected algorithm: *KMeans* and *DecisionTreeClassifier* as follow:

Algorithm	Accuracy_score	Precision	Recall
KMeans	0.79507	0.28856	0.29650
DecisionTreeClassifier	0.79154	0.32391	0.32650

For *KMeans* we select the top scored 6 features by *SelectKBest* function, and select the optimized parameters as follow:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=600,
    n_clusters=2, n_init=6, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=1e-07, verbose=0)
```

For *DecisionTreeClassifier* we select the top scored 4 features by *SelectKBest* function, and select the optimized parameters as follow:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

The evaulation in *test_classifier* use the accuracy score, precision and recall to evaluate the machine learning model performance. Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. The definition of precision and recall are as follow:

Definition (Precision): *The precision equals to the ratio between the number of person classified as POI and are real POI, and the number of person classified as POI. That is,*

$$Precision = \frac{\text{The number of person classified as POI and are real POI}}{\text{The number of person classified as POI}}.$$

Definition (Recall): *The recall equals to the ratio between the number of person classifed as POI and are real POI, and the number of person real as POI. That is,*

$$Recall = \frac{\text{The number of person classifed as POI and are real POI}}{\text{The number of person real as POI}}.$$

Our optimized machine learning algorithm has better performance on the precision of predict the accuracy of *poi* from the other features of records. The high precision means that the prediction *poi* by machine learning algorithm has high chance to be the true *poi*; The high recall means that the real *poi* will has high chance to be predicted as *poi*.

Conclusion

We use machine learning algorithm to train and test the performance of 7 machine learning algorithms. We select three best performance algorithm as our final candidates. There are still some possible improvement in future. First the dataset has very limited size. There are total 146 records avaible. Since there are few data set, we can not get enough training set and test test for our algorithms. Second, there are very few poi person, that is 18, in our data set. If there would be more poi in our dataset, we could train our machine learning algorithm much better.