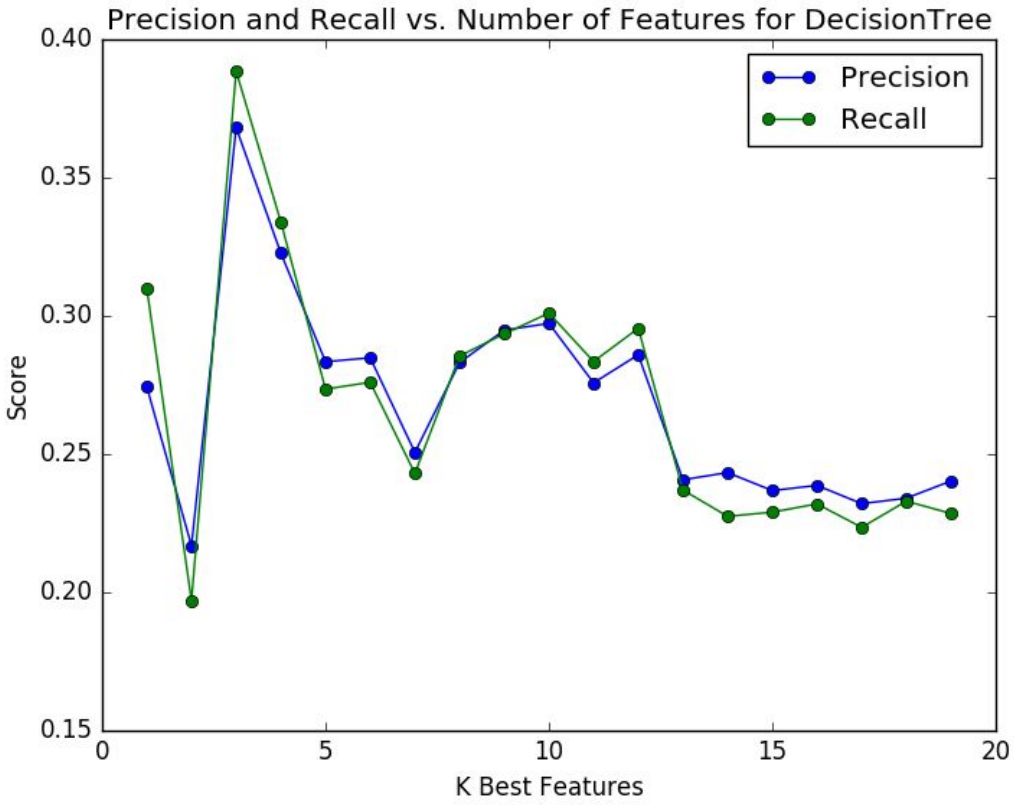
Answers are corresponding to the questions in this link

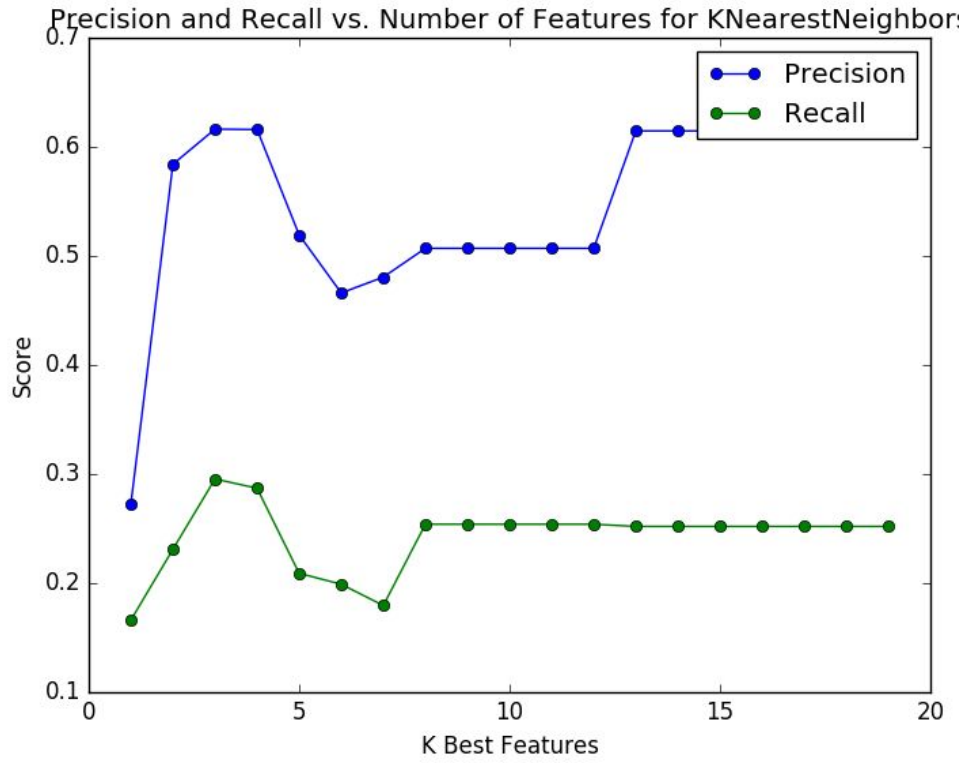
1. The goal of this project is to identify who are the persons of interest of Enron scandal. The dataset contains 146 data points which represents staff of Enron. There are 18 POIs and 127 non-POIs. Each data point has some financial features and email features, such as salary, bonus, stock options and the number of mails with person of interest. However, there are some missing values which are represented by 'NaN'. I made table which shows the number of 'NaN' for each feature. There are some features that almost all of staff has no data for it, such as loan_advances, director_fees and restricted_stock_deferred. From the viewpoint of the objective of this project, I thought the number of mails involved with POI is the key point of my analysis. Before I tune up my classifier, I made some scatter plots for each feature and found an outlier which represents the total amount of other data points. So I removed this data point from training dataset.

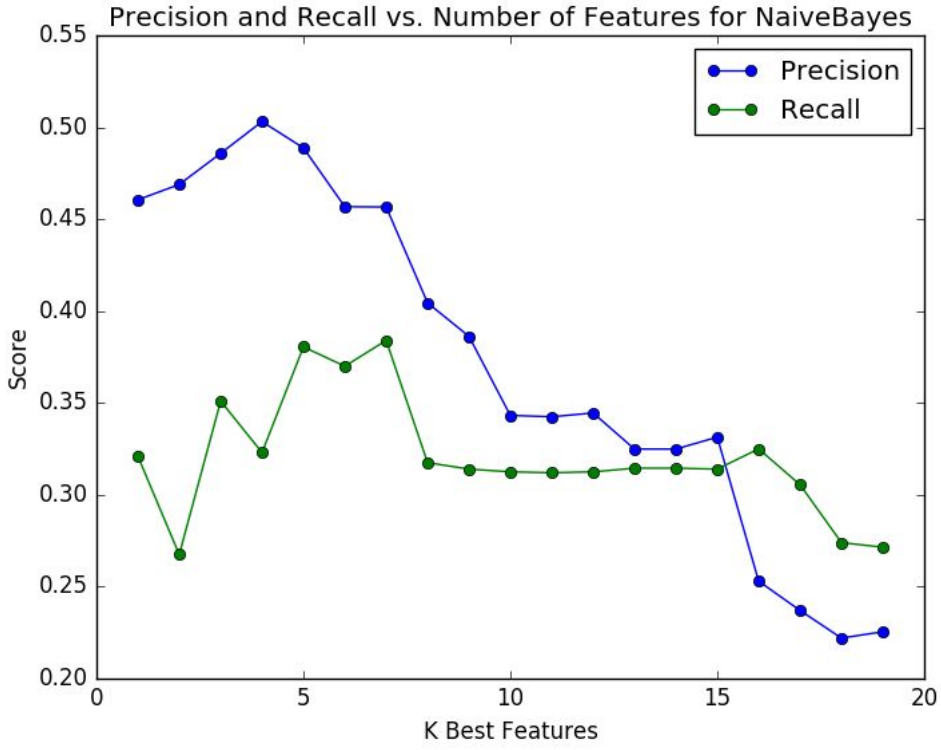
The number of 'NaN' for each feature

bonus	64	from_messages	60	restricted_stock	36
deferral_payments	107	from_poi_to_this_person	60	restricted_stock_deferred	128
deferred_income	97	from_this_person_to_poi	60	salary	51
director_fees	129	loan_advances	142	shared_receipt_with_poi	60
email_address	35	long_term_incentive	80	to_messages	60
exercised_stock_options	44	other	53	total_payments	21
expenses	51	poi	0	total_stock_value	20

2. For my final classifier, I used five features. I applied SelectKBest to find out how many feature will maximize parameters for each algorithm. The result was below. As you can see, K Nearest Neighbor doesn't seems to be good algorithm for this dataset. However, for Decision Tree classifier and Gaussian Naive Bayes classifier show good result when parameter k takes 3, 4 and 5. They exceed required benchmark for Precision and Recall. So I continued to tune parameters for k=3, 4, 5.







By applying SelectKBest, I got feature scores shown below.

The feature scores (Blue ones are used for final classifier)

bonus	21.06	from_messages	0.16	restricted_stock	9.35
deferral_payments	0.22	from_poi_to_this_person	5.34	restricted_stock_deferred	0.06
deferred_income	11.60	from_this_person_to_poi	2.43	salary	18.58
director_fees	2.11	loan_advances	7.24	shared_receipt_with_poi	8.75
email_address	-	long_term_incentive	10.07	to_messages	1.70
exercised_stock_options	25.10	other	4.20	total_payments	8.87
expenses	6.23	fraction_to_poi	4.17	total_stock_value	24.47

3. As described above, I tried three algorithms, Decision Tree, K Nearest Neighbor and Naive Bayes. Then, I finally chose Naive Bayes classifier. The results between algorithms with the features I used was below. GaussianNB and Decision Tree exceeded benchmark of Precision and Recall when the number of features was well selected(k=3 or k=5).

	Accuracy	Precision	Recall	F1
Decision Tree	0.80146	0.36291	0.38450	0.37339
KNearestNeighbor	0.86331	0.61627	0.29550	0.39946
GaussianNB	0.85464	0.48876	0.3805	0.42789

4. With tuning parameters of algorithms, we can optimize classifier. If we don't tune parameters, we might be influenced by outliers too much and our decision boundary doesn't work well. For Decision Tree Classifier I used, I tried GridSearchCV to tune parameters.
   Tested parameters are below.

```
param_grid = {"criterion": ["gini", "entropy"],
              "min_samples_split": [2, 5],
              "max_depth": [None, 5, 10],
              "min_samples_leaf": [1, 5],
```

```
            "max_leaf_nodes": [None, 10]
        }
```

I got best performance with criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2.
It means Decision Tree classifier performs best when its parameters take default value. So, compared with other algorithms, GaussianNB classifier gives us best result.

5. Validation is the way how I assess the performance of classifier. The common mistake in validation process is unconsidering overfitting. If I use same dataset between constructing classifier and assessing accuracy, the classifier might be optimized too much for that dataset. This would result poor performance in other dataset. To avoid this problem, we should split the dataset into training and testing sets, and try to optimize indicators with testing dataset. I used train_test_split() function to split original dataset into training and testing by 70% and 30%.

6. I used three evaluation metrics, precision, recall and F1 score. The table above shows evaluation metrics for each algorithm. Precision describes the accuracy of the number of algorithms predicted to POI. Recall describes the rate of the number of predicted to POI over the number of actual POI. So it could be say that if we have high precision, people who were predicted to POI is highly suspicious, and if we have high recall, it means we could predict most of POIs in the dataset. In addition, F1 score means a weighted average of the precision and recall, so it can be overall rating for algorithms.