Goal of this project

Goal of this project is to come up with algorithm that automatically detect Person of Interest (POI) from Enron data set. Luckily we already have information on who was POI in reality, their financial information and email history. With these information we can use machine learning techniques to train algorithm.

Data Analysis

Originally there were 146 people in the data set, and 18 of them were flagged as POI. We had these informations on every people and these information will be used as features. Except for ‘poi’ which was used as label.

'salary', 'to_messages', 'deferral_payments', 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'restricted_stock_deferred', 'total_stock_value', 'expenses', 'loan_advances', 'from_messages', 'other', 'from_this_person_to_poi', 'poi, 'director_fees', 'deferred_income', 'long_term_incentive', 'email_address', 'from_poi_to_this_person'

Outliers

['TOTAL'] and ['THE TRAVEL AGENCY IN THE PARK'] were outliers and in same time obviously not person, so I deleted these two from data set.

New Feature

Employees with suspicious behaviour should spend more time communicating to POI than other non POI. So decided to set new 2 features as ratio of their email to POI and from POI. Those 2 features were named ‘from_poi_ratio’ and ‘to_poi_ratio’, and added into data_dict.

Feature Selection

Principal Components Analysis(PCA) dimension reduction were run to reduce features first. ‘exercised_stock_options’ and ‘restricted_stock’ were similar so I used PCA to put them into one new feature ‘financial_PCA’, and ‘to_poi_ratio’ and ‘from_poi_ratio’ were combined into ‘poi_ratio_PCA’.

After that I run SelectKBest with f-class-if to find out ANOVA F-value for all of possible features. I didn’t use chi2, because some financial_PCA values had negative values and it made chi2 not usable. And I deleted features which had substantially lower value, which was lower than 3.0 in this case. These were the best features with best ANOVA F-value against ‘poi’.

('expenses', 5.236) , ('deferred_income', 12.937), ('long_term_incentive', 3.048), ('shared_receipt_with_poi', 5.772), ('to_poi_ratio', 12.876), ('bonus', 8.241), ('from_poi_to_this_person', 3.898), ('from_poi_ratio', 3.251) ('salary', 6.409), ('exercised_stock_options', 10.634)

I noticed ‘from_poi_to_this_person’ and ‘from_poi_ratio’ are too similar, actually ‘from_poi_to_this_person’ was used to calculate ‘from_poi_ratio’, so decided to delete ‘from_poi_to_this_person’. ‘So I went with these 9 features.

I still wasn’t sure if choosing 9 features was enough or too much, so checked optimal number of features to choose by using RFECV of sklearn. The result was 8. Out of 9 features I already had, ‘long_term_incentive’ had weakest F-value so decided to test algorithm’s accuracy score with it and without it.

Gaussian Naive Bayes Accuracy Score
8 features without “long_term_incentive” : 0.840579710145
9 features with “long_term_incentive”: 0.8714285714

So, even though RFECV recommended 8 features, having 9 features showed better score, so decided to go with 9 features below.

```
features_list = [
   "poi", 'expenses', 'from_poi_ratio', 'to_poi_ratio', 'deferred_income', \
   ‘long_term_incentive’, 'shared_receipt_with_poi', 'bonus', 'exercised_stock_options']
```

Feature Scaling
After choosing features and deleting values with ‘NaN’, I used scikit-learn’s Standardization algorithm to standardize data. This is because Machine Learning algorithms, such as Support Vector Machines assume that all features are centered around zero and have variance in the same order (Scikit-learn documentation). Scikit-learn’s Standardization methods, preprocessing.scale(features) will transform data to meet that assumption.

Choose Algorithm
Next step is to validate algorithm. Each algorithms have now being trained, so we can run those with actual data to see how it performs. By this we can know how accurate those algorithm are and have idea if we can trust their result or not.
To test and validate which algorithm was the best one, I first separated data into training data and testing datal. Scikit Learn has easy function to do this, which is stratified shuffle split. I chose stratified shuffle split over train_test_split, because of the large imbalance in the distribution of the targeted classes.(scikit-learn documentation) After running it, and fitting training data into algorithm, used testing data to check its quality. Decision Tree, Naive Bayes and SVM were tested and here are the result.

Algorithm	Accuracy	Precision	Recall	f1-score
Decision Tree	0.8	0.75	0.8	0.77
SVC	0.8714285714	0.76	0.87	0.81
Gaussian Naive Bayes	0.8714285714	0.83	0.87	0.83

What this result showed that SVC & Gaussian Naive Bayes have exactly same accuracy score. But when Gaussian Naive Bayes made prediction, 83% of the times it was correct, while SVC got only 76% of the time correct. (Precision) On the other hand SVC successfully found out 87% of POI from the data, which was same result as Gaussian Naive Bayes.

(Recall). f1-score of Gaussian Naive Bayes was slightly better than SVC, but since f1-score is simply calculated by using precision and recall value as below, so this was not surprising result.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$ (Wikipedia)

But since Gaussain Naive Bayes is more trustworthy than SVC when it made prediction, which is the "Precision" score, I decided to go with it.

Parameter Tuning

Some parameters are determined manually before fitting actual data, so in order to squeeze out best result from Machine Learning, we have to tune parameters. Since SVC and Gaussian Naive Bayes showed similar score, I decided to tune both of algorithm's parameters.

SVC was tuned by GridSearchCV automatically and found out default setting of SVC was the best setting, so it didn't improve result.

For Gaussian Naive Bayes, only parameters we can change is "Prior possibilities", but we don't have this, so only option is to keep the default setting of "None". Not much of tuning could be done for this, so it's better to focus on pre-processing of data and feature selection. (S Ray, 2015) So again, tuning didn't improve result of algorithm and we got same result.

Reference

http://scikit-learn.org/stable/modules/preprocessing.html
https://en.wikipedia.org/wiki/Precision_and_recall
https://www.quora.com/What-are-hyperparameters-in-machine-learning
S Ray, Analytics Vidhya, 2015 September 13th: Available here.
https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/
http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
https://en.wikipedia.org/wiki/F1_score