Machine Learning – Identify Fraud from Enron Emails

By Susan Li

Introduction

Enron was an American energy and utility trading company that perpetuated one of the biggest accounting frauds in history. Many executives at Enron were indicted for a variety of charges and some were sentenced to prison. After the company's collapse, a large database of over 600,000 emails generated by 158 former employees were made public by the Federal Energy Regulatory Commission during its investigation.

1. Summarize 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

The goal of the project is to use machine learning to build a predictive model to identity persons of interest (POI) who may be involved in the fraud, based on the financial and email data. Machine learning is a powerful prediction tool that builds classifier on the training dataset, and then the classifier can be used to made predictions on the testing dataset.

Dataset overview

The original dataset contains 146 data points and 21 features. 18 of them are identified as POI, and the rest 128 are not POIs. The features fall into three major types, namely financial features such as "salary", "bonus" email features such as "to messages", "email address" and POI labels.

Outliers

When I visualized the dataset as a scatter plot, I discovered an outlier named "TOTAL", It was a spreadsheet artifact, it was removed. I also found a strange name called "THE TRAVEL AGENCY IN THE PARK" which I believe was a mistake. It was removed as well. In addition, I removed an observation called "LOCKHART EUGENE E" who has no non NaN value. After removing outliers, there are 143 people in the dataset.

Missing values

There are many missing values in the data I summarize as following:

Feature	Number of Missing Values
poi	0

salary	49
deferral payments	105
total_payments	20
loan_advances	140
bonus	62
restricted_stock_deferred	126
deferred_income	95
total_stock_value	18
expenses	49
exercised_stock_options	42
long_term_incentive	78
restricted_stock	34
director_fees	127
to_messages	57
from_poi_to_this_person	57
from_messages	57
from_this_person_to_poi	57
shared_receipt_with_poi	57

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.

Feature engineering

I started with all the features except “email_address” and “other”, because email address is a text string corresponding to a person’s name, “other” does not add any meaning to the analysis. I created two new features “from_poi_to_this_person_ratio” which measures how frequently a poi sends email to this person and “from_this_person_to_poi_ratio” which measures how frequently this person sends email to poi. After performing feature selection

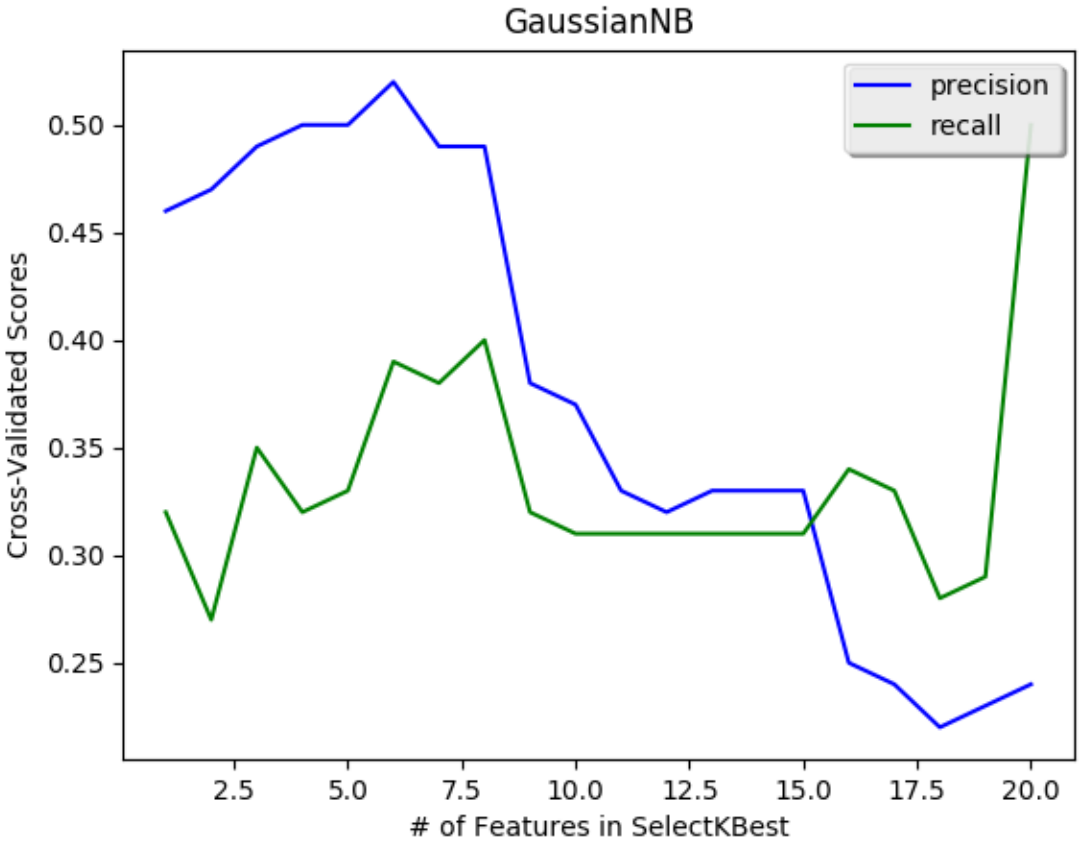
later, one of these two new features – “from_this_person_to_poi_ratio” remained in my analysis.

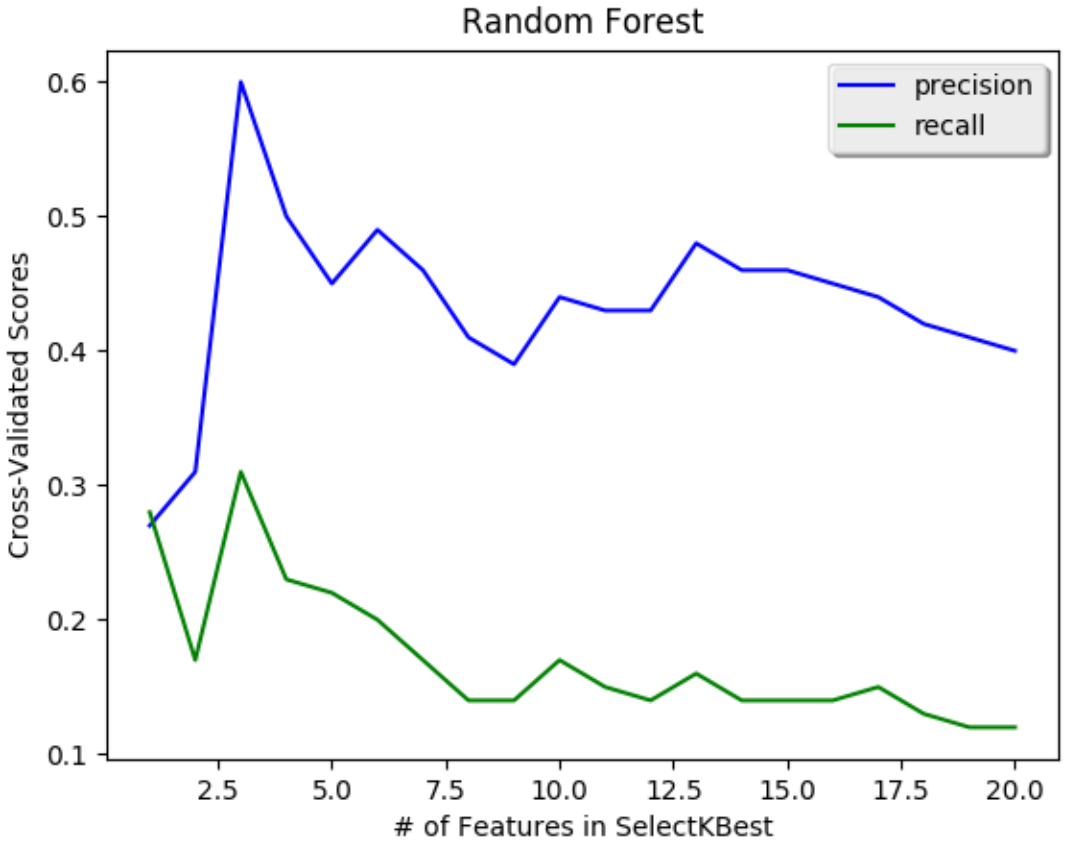
Feature selection

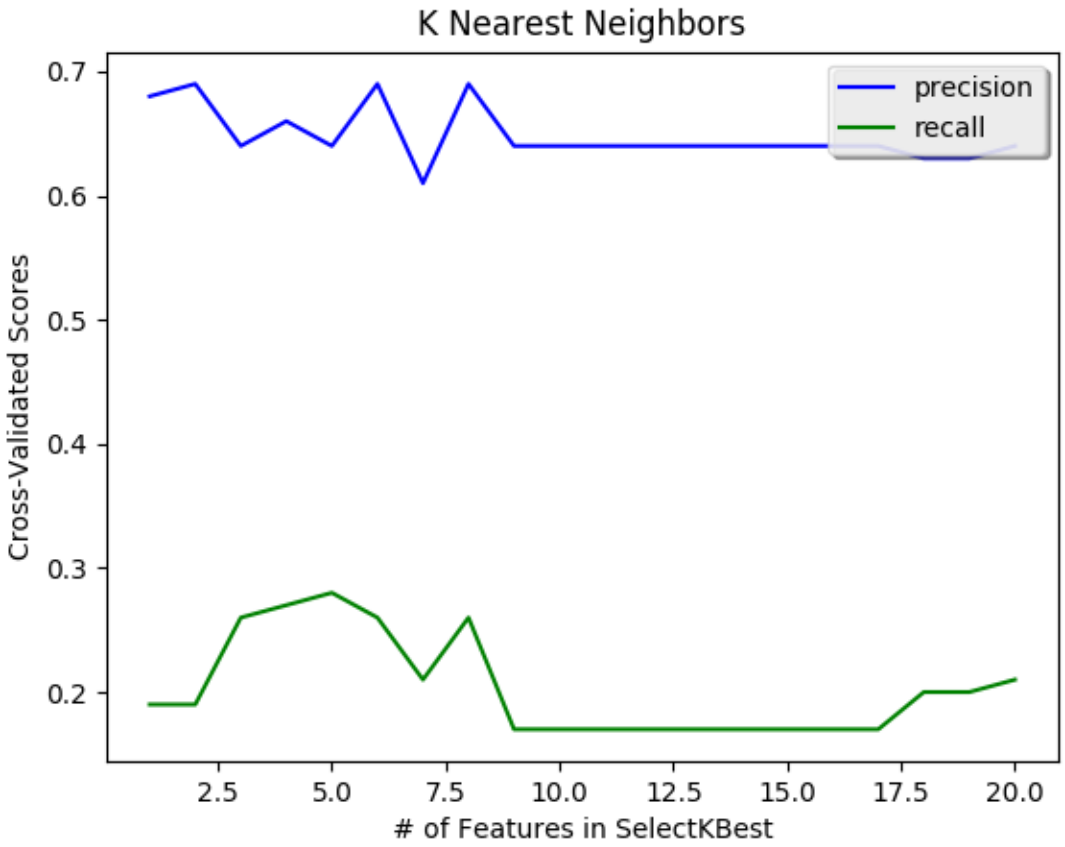
To select the most relevant and influential features, the manual selection and SelectKBest module from scikit-learn were utilized. I then performed feature scaling using MinMaxScaler model to makes sure that the features would be weighted equally before applying to machine learning algorithm classifier.

After adding two new features, the following are the features I am starting with:
['poi', 'salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'director_fees', 'to_messages', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi', 'from_poi_to_this_person_ratio', 'from_this_person_to_poi_ratio']

I then ran SelectKBest for all values from K1 to K20 manually, for GaussianNB, Random Forest and KNN, and recorded value on KBest, precision and recall, then generated the plots as follows.


GaussianNB
precision
recall
Cross-Validated Scores
0.50
0.45
0.40
0.35
0.30
0.25
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
# of Features in SelectKBest


Random Forest
precision
recall
Cross-Validated Scores
0.6
0.5
0.4
0.3
0.2
0.1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
# of Features in SelectKBest



After this process, I was able to determine the best value of k for each algorithm. My final algorithm is GaussianNB with the following feature set:

Feature	Score
exercised_stock_options	24.82
total_stock_value	24.18
bonus	20.79
salary	18.29
from_this_person_to_poi_ratio	16.41
deferred_income	11.46

The Precision is 0.5157 and the Recall is 0.38550.

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

I tested three algorithms manually selected from k1 to k20. Eventually I found k6 is the sweet spot for GaussianNB. I ended up selecting Gaussian NB as the best performer because it has the better recall than the alternatives. The best result for Random Forest is Precision 0.60, Recall 0.31, with the following feature set:
['poi', 'exercised_stock_options', 'total_stock_value', 'bonus'].
The best result for K Nearest Neighbors is Precision 0.64, Recall 0.28, with the following feature set:
['poi', 'exercised_stock_options', 'total_stock_value', 'bonus', 'salary',
'from_this_person_to_poi_ratio']

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)

My algorithm - GaussianNB does not have parameters that I need to tune.

Tuning the parameters of an algorithm is a process we go through to impact the model and enable the algorithm to perform the best. For example, we must specify the number of clusters in a dataset when we use K-Means, otherwise we either run into over-fitting or under-fitting.

This process can be difficult as I experienced below.

The algorithm I choose does not need to tune the parameter. To test how parameter tuning can make a difference, I tuned parameter for K Nearest Neighbors using GridSearchCV using the above best set of feature for KNN and following:

{"n_neighbors":[2, 5], "p":[2,3]}

The precision is 0.67, recall is 0.24, The precision is higher than before but the recall is lower. The results were not satisfying. I also tuned parameter for Random Forest using the above best feature for Random Forest and the following:

{"n_estimators":[2, 3, 5], "criterion": ('gini', 'entropy')}

The precision is 0.50 and the recall is 0.27. Again, the results were not satisfying compare with Gaussian NB.

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

Cross validation is a machine learning or statistical method of evaluating and comparing learning algorithm by dividing data into two segment, one segment used to train the model and another segment used to validate the model. Our purpose is to use trained data make reliable predictions on general untrained data. One classic mistake is overfitting, when a machine learning algorithm performed very well on training data but failed to predict on the validation (or testing) data, we call it overfitting. To prevent overfitting, it is essential to split data to training set and testing set, hold back the testing set from the algorithm until the end of the project (validation step).

The model used Stratified Shuffle Split cross validation in the tester.py to randomly create train and test dataset is an essential approach of validation. Because our dataset is small, it is possible that training or testing dataset does not represent the overall dataset. To avoid this problem, we use Stratified Shuffle Split cross 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.

For this project, I use precision and recall as the main evaluation metrics, Gaussian Naïve Bayes achieved the best results – precision: 0.5157 and recall: 0.38550. This is the final model as well.

Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples. In this case, if my algorithm predicts a person is a POI, about 51% of time this person is indeed a POI. And my algorithm can identify a POI correctly about 38% of the time.