Identifying Fraud from Enron Email

Udacity Intro to Machine Learning – Final Project

Christopher Giler

January 7, 2017

1. Introduction

The "Enron Scandal," as it came to be called, was a well-known case of corporate fraud and corruption which eventually led to the bankruptcy of one of the largest energy service companies in the world. During court hearings of employees and top executives in the company, the financial records of these employees were made public. These records were later updated to reflect the results of the court hearings, labeling those found to be involved in the fraud as persons of interest (POIs).

The goal of this project is to select and tune a machine learning algorithm which can be applied to predict a POI based on a number of metrics found in these released financial records. The project applies the scikit-learn Python library to systematically test a number of machine learning classifiers, and partition the data and select the best features to train and validate the model.

2. The Data

Data Exploration

The selected Enron dataset contains 146 separate data points, each for a different employee and consisting of mostly senior management and top executives of enron. Each data point includes information across 21 features, including one which classifies the employee as a "Person of Interest" (POI) in the Enron fraud. The feature breakdown, split by data type, is as follows:

Feature Type	String	Integer	Boolean
Feature	Deferral Payments	Salary	POI
	Exercised Stock Options	'To' Messages	
	Restricted Stock Deferred	Total Payments	
	Loan Advances	Bonus	
	Director Fees	Restricted Stock	
	Deferred Income	Shared Receipt with POI	
	Long Term Incentives	Total Stock Value	
	Email Address	Expenses	
		'From' Messages	
		From This Person to POI	
		From POI to This Person	
		Other	

For all 146 data points in the dataset, 18 of them are classified as POIs:

1. HANNON KEVIN P
2. COLWELL WESLEY
3. RIEKER PAULA H
4. KOPPER MICHAEL J
5. SHELBY REX
6. DELAINEY DAVID W
7. LAY KENNETH L (Founder / Chairman)
8. BOWEN JR RAYMOND M
9. BELDEN TIMOTHY N
10. FASTOW ANDREW S (CFO)
11. CALGER CHRISTOPHER F
12. RICE KENNETH D (Chief Executive – High-Speed Internet Unit)
13. SKILLING JEFFREY K (CEO)
14. YEAGER F SCOTT
15. HIRKO JOSEPH
16. KOENIG MARK E (Director of Investor Relations)
17. CAUSEY RICHARD A
18. GLISAN JR BEN F (Treasurer)

There are a number of features with missing values (marked 'NaN' in the dataset):

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_messages	60
from_poi_to_this_person	60
from_this_person_to_poi	60
other	53
salary	51
expenses	51
exercised_stock_options	44
restricted_stock	36
email_address	35
total_payments	21
total_stock_value	20
poi	0

Outlier Investigation

To remove outliers from the data, the following data points were first removed:

- 'TOTAL' – Contains the sum of each feature over all data points
- 'THE TRAVEL AGENCY IN THE PARK' – Contains data for a travel agency, rather than a single employee
- 'LOCKHART EUGENE F' – Not identified as a POI, and does not contain data in other features besides email address

Because the feature 'email_address' holds a different and unique string value for each data point, it would not provide any use in setting up a predictive model for the data set. For this reason, the feature was removed.

Scikit-learn's estimators assume all values in an array are numerical, and that all have and hold meaning. Some features were found to contain string-type values of numeric information, so to allow use of these features in a machine learning algorithm, all features were converted the same 'float' type data. The high number of NaN values described earlier also had to be addressed. However, simply setting these values to zero could cause the data to be skewed low, especially for features with a high count of missing values ('loan_advances', 'director_fees', etc). To avoid this bias in the data, the missing values were replaced with the median value over all data points for that feature.

df = df.apply(lambda x: x.fillna(x.median()), axis=0)

3. Feature Selection/Engineering

New Features

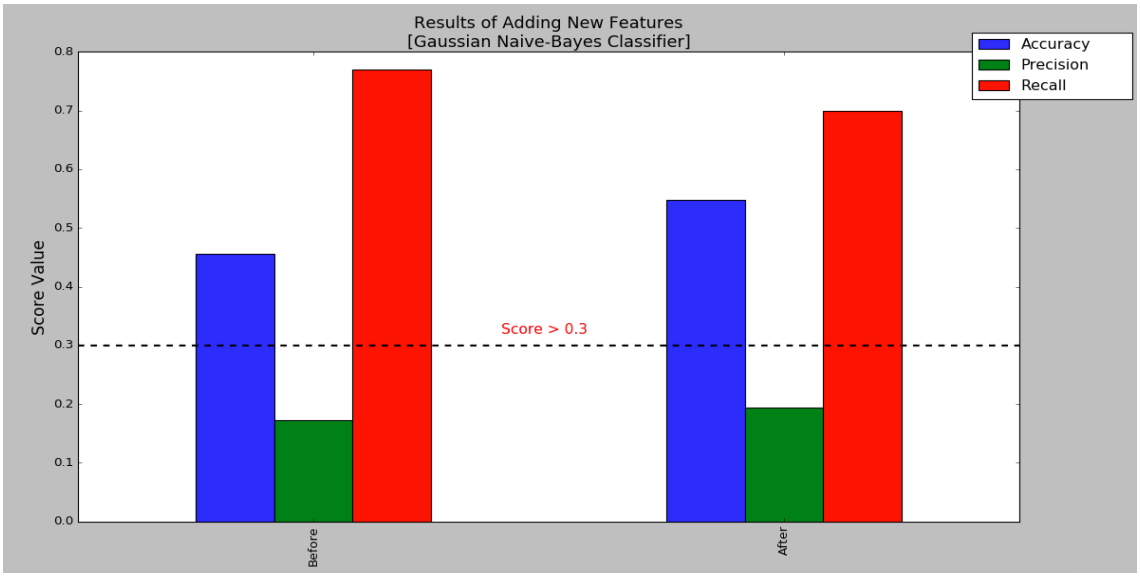
In an effort to improve the feature set to be used in a predictive model, two new features were added. These features provide a new perspective on the data by converting the number of emails sent and received to normalized ratios, and are defined as follows:

'fraction_from_poi' – the ratio of messages received by this person that are from a POI, compared to the total number of inbox messages
'fraction_to_poi' – the ratio of messages sent by this person to a POI, compared to the total number of outbox messages

After adding these new features, 3 metrics were calculated to determine the effectiveness of adding additional features, and to establish a base point for feature and algorithm selection in later steps. The following three scores were calculated for a Naive-Bayes classifier with default parameters applied:

- *Accuracy*: Overall statistical bias of the model
- *Precision*: Fraction of retrieved instances that are relevant (robustness against false positives). Target: Precision > 0.3
- *Recall*: Fraction of relevant instances that are retrieved (robustness against false negatives). Target: Recall > 0.3

Each metric was calculated as the mean score found over 50 tests with pseudo-randomized training and testing sample splits on the dataset. All features were normalized using scikit-learn's built-in min-max scaler function.

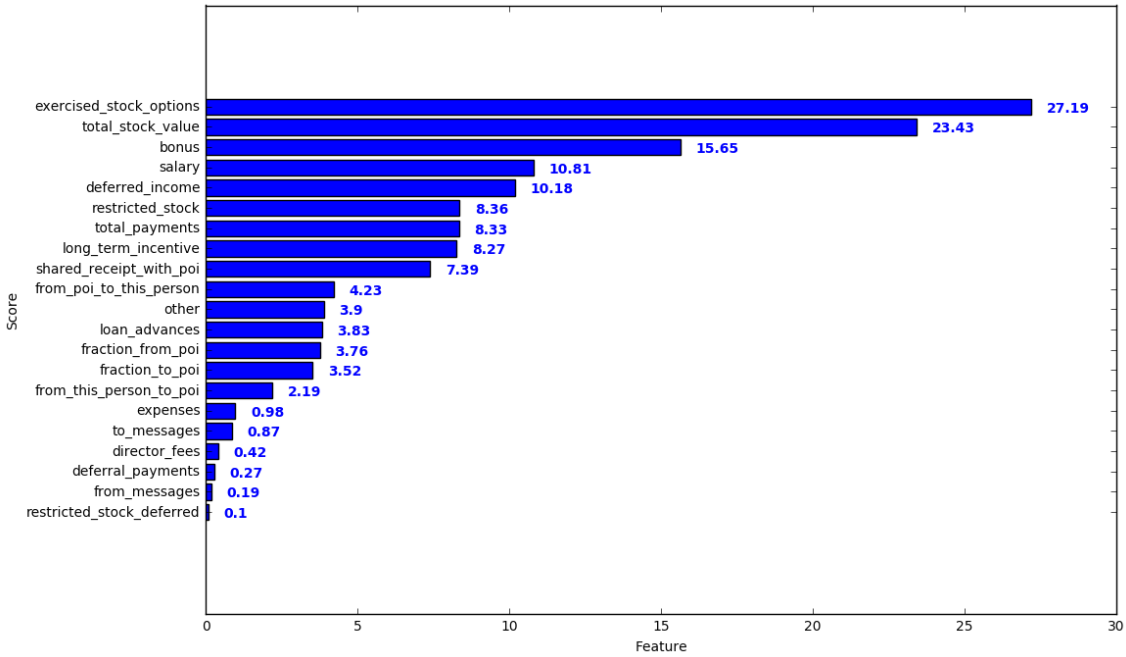


Metric	Accuracy	Precision	Recall
Before Adding Features	0.456	0.173	0.770
After Adding Features	0.548	0.194	0.7

The addition of the two new features resulted in an increase in classifier accuracy and precision, with the trade-off of a slight reduction in recall score. However, although the recall score obtained from this testing met our target, further steps were needed to meet our target for model precision. Also, although no target was set for an accuracy score, the low accuracy found in these results suggests there may be more room for improvement by ruling out some features not relevant to identifying a POI and/or selecting a different classifier for our model. These features were added to the dataset and list of features to be included in the feature selection stage of developing the model.

Feature Selection

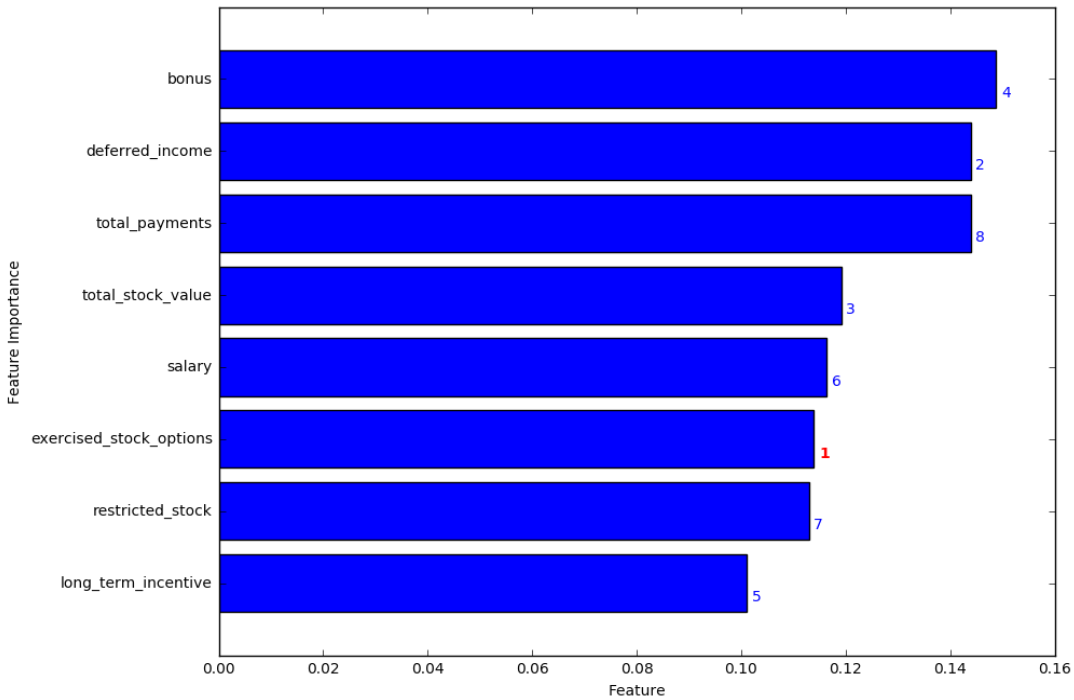
A univariate feature selection approach was applied by using scikit-learn's SelectKBest algorithm The number of features were reduced by selecting the top 8 features based on their score. Feature scaling was applied using a Min-Max Scaler to normalize the weighting across all features. The following feature scores were obtained in using this strategy.



Based on this evaluation, the top 8 features selected are:

'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'deferred_income', restricted_stock', 'total_payments', 'long_term_incentive'

Following this feature selection, the top 8 features were ranked using a method of Recursive Feature Elimination (RFE). RFE was applied on a Logistic Regression model and features were ranked from 1 to 8. Feature importance for each feature was also calculated using a Random Forest Classifier to give another set of data to consider in the final feature selection. Features were weighted independently using a min-max scaler.



The final 5 features for our model were selected based on the two metrics calculated.

- *exercised_stock_options*
- *deferred_income*
- *total_stock_value*
- *bonus*
- *salary*

The remaining three features were ruled out for the following reasons:

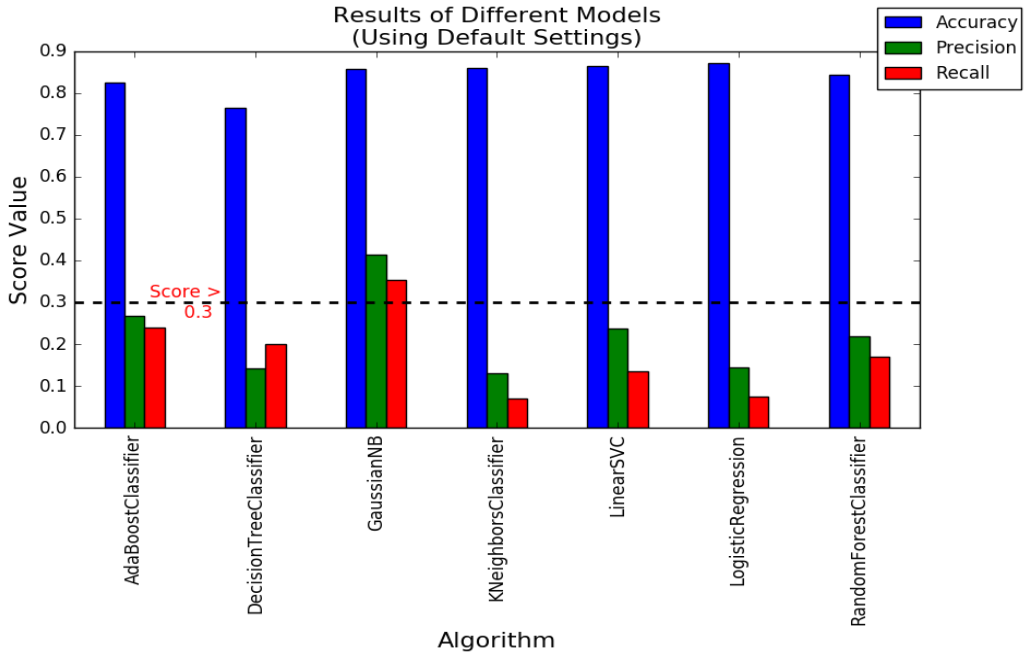
- *total_payments:* showed a high feature importance, but the feature was ranked lowest among the set tested
- *long_term_incentive:* showed a ranking in the top 5 of all features tested, but the calculated feature importance was lower than the rest
- *restricted_stock*: showed the second lowest ranking in the set, and was also found to have a relatively low feature importance compared to the other features tested.

4. Algorithm Selection and Tuning

Algorithm Selection

To start the search of an optimal algorithm for our predictive model, a number of potential classifiers were selected and tested. For initial testing, the default parameters for each classifier were used, and cross-validation was applied to test each algorithm on a number of randomly selected training sets and calculate the mean score for each.

The following results were obtained for the classifiers tested over 100 iterations of pseudo-randomized cross-validation.



The Naive-Bayes algorithm showed the highest values of precision and recall, with both falling well over the target. It also produced a high accuracy score compared to the other algorithms tested. The cross-validation test was performed a number of times to ensure consistency in these predictions.

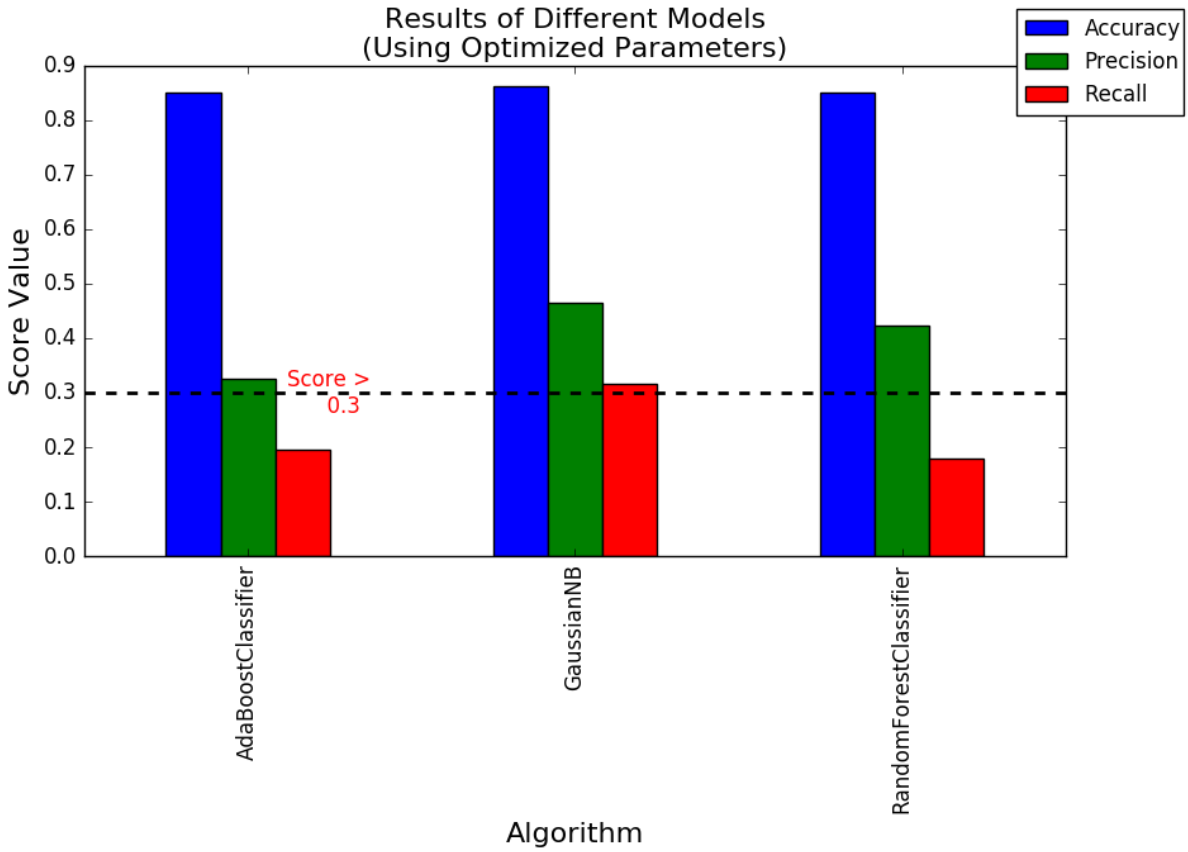
However, because the comparison used only default values for each algorithm's parameters, we cannot be sure that Naive-Bayes will produce the best results after the parameters of each classifier are more finely tuned. For this reason, the top 3 classifiers in terms of performance (Naive-Bayes, Random Forest, and AdaBoost) were selected for parameter tuning and optimization.

Parameter Tuning

Parameter tuning is a critical step in ensuring that an algorithm is configured appropriately for the type of data it is modeling. Sklearn's GridSearchCV function was used to perform a search over the following parameter values for the Random Forest and AdaBoost classifiers.

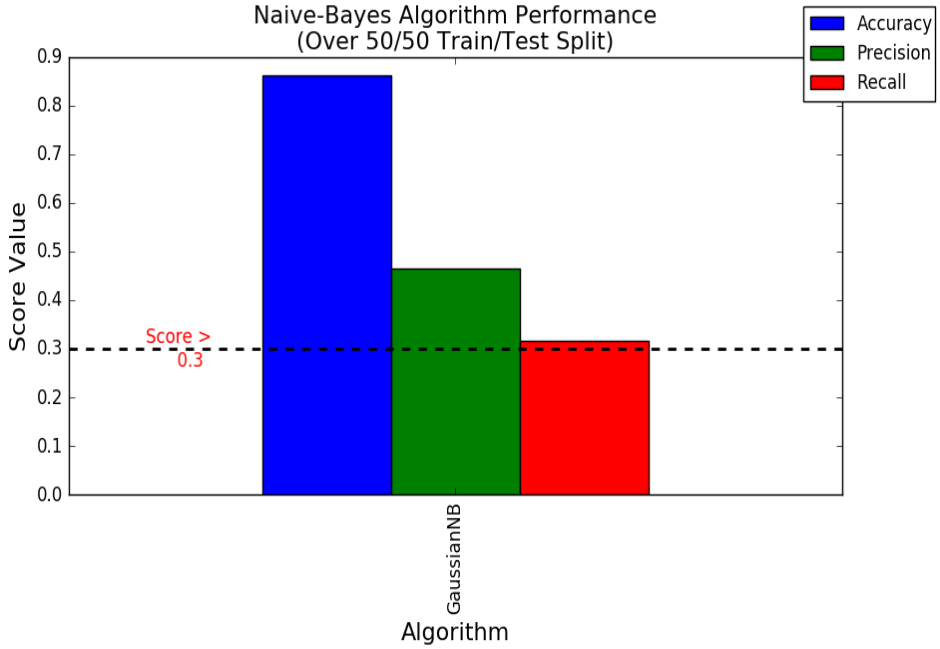
- *Random Forest*: n_estimators, min_samples_split, min_samples_leaf
- *AdaBoost*: n_estimators, algorithm, random_state

Unfortunately, the Naive-Bayes classifier does not have the same flexibility as other classifiers within Python's scikit-learn library, so the default values were still used for this algorithm. The following results were obtained after systematic parameter tuning of the Random Forest and AdaBoost classifiers.



Final Validation

After parameter tuning, the two alternative algorithms were still unable to meet our criteria and show lower recall and precision scores compared to the default Naive-Bayes algorithm. For final validation, the Naive-Bayes algorithm was tested on a number of randomly selected 50/50 train-test splits to determine the algorithm's precision and recall given a smaller training set.



Testing over this split yielded results that still met the criteria of both precision and recall scores above 0.3. The algorithm was also tested with the course-provided 'tester' script and yielded the following results.

Accuracy	Precision	Recall
0.87	0.50	0.40

5. Conclusion

In the end, the Naive-Bayes algorithm was found to be the best classifier for creating a predictive model for this dataset. However, tuning was also found to be a critical part of not only improving an algorithm after its selection, but in the selection process itself. Using a programmatic approach to parameter tuning, like the GridSearchCV method used in this project, is important to ensure the parameters for the classifier are appropriately set for the given type of dataset, and this is an important step to follow before making the final selection of an appropriate classification algorithm.

Cross-validation was essential in determining the accuracy, precision, and recall of the algorithm. Taking the mean score for each of these metrics over a number of randomized train-test splits helped provide robustness to the selection and tuning of an appropriate algorithm for our model.

Testing across a number of different classifiers during the start of the selection process was also important in showing that accuracy is not necessarily the best metric to see how well a model can classify data. For the initial selection using default parameter values, accuracy was, for the most part, between 0.8 and 0.9, while recall and precision scores varied greatly among the classifiers. The best method is to find a good balance among these three metrics to ensure the final model is robust to new sets of data

Of course, while the resulting scores of the final model meet the criteria set for this project, the model could be further improved with a larger dataset. Because only 18 of the data points in this dataset were labeled as POIs, having a dataset containing additional POI data points would help find a greater separation between the two classifications, resulting in even higher precision and recall scores.

References

- http://www.stackoverflow.com/
- http://www.nytimes.com/2006/01/29/business/businessspecial3/10-enron-players-where-they-landed-after-the-fall.html
- https://en.wikipedia.org/wiki/Precision_and_recall
- https://en.wikipedia.org/wiki/Enron_scandal
- https://en.wikipedia.org/wiki/Enron