Enron Submission Free-Response Questions

By Timea Roik
October 24, 2017

1. The goal of this project is to build an identifier of the persons of interest in the Enron corporate fraud case, based on data available for public use as result of the scandal. The data contains 14 financial ('salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'), and 6 email features ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string) and 1 POI label.

   Outliers:

   Three outliers were detected and removed. Since one person, LOCKHART EUGENE E, does not have any available data and the "THE TRAVEL AGENCY IN THE PARK" is not a person of interest; they are removed from the analysis. Also, the "TOTAL" value for the columns, which was calculated by a spreadsheet program; is also removed.

   Main features of the dataset:

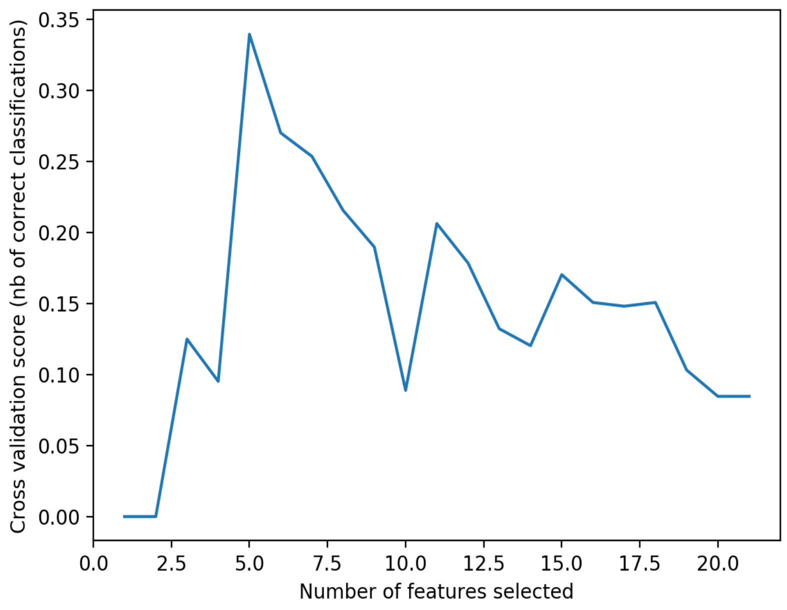
   - Overall, there are 21 features.
   - The total number of people in the dataset is 146.
   - There are 18 POI, and 128 non-POI.

   Patterns between the features are investigated by algorithms with the goal of achieving at least a 0.3 precision and recall score.

2. Two new features were created to identify people as POIs by looking at the fraction of emails they exchanged with other POIs. The assumption is that the people involved this fraud communicate more frequently with POI. The two features created and added are: **ratio_emails_from_poi** and **ratio_emails_to_poi**.

   Feature scaling was not implemented because Decision Trees, Random Forests, and Naive Bayes do not create distance based classifiers. Decision Tree splits are based on each feature and independent from one another, and will only split at points that make sense given the data. Naive Bayes generates probabilities for all of the features individually, assuming no correlation. For PCA, SVM, and other distance based variables, scaling matters since some features could dominate others due to their scaling.

The optimal number of features was chosen with recursive feature selection (RFECV). The results can be seen in the following graph:



Based on the results, I am going to use five features in this analysis based on their feature scores, which is calculated by SelectKBest.

Before adding the two new features, the feature scores returned by SelectKBest were the following: [('exercised_stock_options', 25.097541528735491),
('total_stock_value', 24.467654047526398),
('bonus', 21.060001707536571),
('salary', 18.575703268041785),
('deferred_income', 11.595547659730601)]

After adding the two new features the following 5 feature scores ranked the highest:
[('exercised_stock_options', 25.097541528735491),
('total_stock_value', 24.467654047526398),
('bonus', 21.060001707536571),
('salary', 18.575703268041785),
(***'ratio_emails_to_poi'***, 16.641707070468989),
('deferred_income', 11.595547659730601)]

During my evaluation of the model using the first five highest scoring features returned by SelectKBest, I have realized that exercised stock options and total stock value are likely to be correlated so I used deferred income as my fifth feature. Therefore, the final features are the following: features_list = ['poi', 'exercised_stock_options', 'bonus', 'salary', 'ratio_emails_to_poi', 'deferred_income']

3. The final algorithm chosen is Gaussian Naive Bayes, after trying Decision Trees and Random Forest. Although all three algorithms performed fairly similar in terms of accuracy, the decision to use Gaussian Naive Bayes was made after comparing the accuracy scores with the precision and recall scores with SSSCV validator in tester.py. Based on the metrics, Gaussian NB turned out to be the best performing algorithm.

<u>Best performance scores based on the *test set:*</u>

ALGORITHMS	ACCURACY SCORE	PRECISION	RECALL
GAUSSIAN NAIVE BAYES	0.881	0.5	0.6
RANDOM FOREST (MIN_SAMPLES_SPLIT=3, N_ESTIMATORS=15)	0.929	0.75	0.6
DECISION TREES (MIN_SAMPLES_SPLIT=15)	0.833	0.375	0.6

4. After the selection of the algorithms, we need to try optimizing the prediction capabilities of the chosen algorithms by adjusting their important parameters. If we do not optimize the parameters, the models might not return the best possible outcomes of our dataset. Optimizing can be done manually, or with algorithms such as GridSearchCV. For this assignment, I chose to tune the most important parameters manually and the results can be seen below. Parameter tuning was not necessary for Gaussian NB, however, I have tuned the Random Forest and Decision Trees algorithms on the test set, which are reported below.

	min_samples_split	**Accuracy**	**Precision**	**Recall**
Decision Trees	5	0.786	0.167	0.2
	10	0.833	0.333	0.4
	15	0.833	0.375	0.6
	20	0.833	0.375	0.6

	min_samples_split	n_estimators	Accuracy	Precision	Recall
Random Forest	2	10	0.881	0.5	0.2
	2	15	0.881	0.5	0.2
	2	20	0.881	0.5	0.4
	3	10	0.905	0.667	0.4
	3	15	0.929	0.75	0.6
	3	20	0.905	0.667	0.4
	4	10	0.881	0.5	0.2
	4	15	0.905	0.667	0.4
	4	20	0.905	0.667	0.4

5. Validation can be explained as a process in which we estimate how well our model has been trained by testing it on data points that are not in our dataset. A classic mistake is overfitting the data by training the model on the whole dataset and make predictions using the same data. This would give high accuracy, precision and recall scores; however, would perform poorly on other data points. In this project, validation is achieved by **cross-validation (train_test_split),** which means that the dataset is split to a training and test set, in this case, 70% and 30% respectively and the model is trained on the training set. Validation was also achieved by using **StratifiedShuffleSplit** is a combination of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. Stratification in our case is useful, since the dataset is small and skewed towards non-POIs. By using this method, we can ensure that the folds are made by preserving the percentage of samples for each class.

6. Based on tester.py, the final metrics by using StratifiedShuffleSplit are the following:

   - Accuracy: 0.86014
   - Precision: 0.51549
   - Recall: 0.34950
   - F1: 0.41657
   - F2: 0.37356

The model averaged 86.01% accuracy in predicting whether the person was a POI or not.
The precision score, or the ratio of true positives / (false positives + true positives) is 0.51549 which means that 51.55% of the people identified as POIs were, in fact, POIs. That value corresponds to about 9 in 18 people. The recall score of 0.3495, which can be calculated as true positives / (false negatives + true positives), however, indicates a 34.95% likelihood that the identifier will correctly flag the person as a POI. Therefore, about 3 in 9 POIs would have been correctly identified by this model.

These results might be improved by adjusting the features in features list, or tuning more parameters for the algorithms manually or in an automated process with GridSearchCV. However, in order to get a full picture of the possibilities or achieve a more distinct change in the results, we should try out other algorithms such as AdaBoost, or KMeans.