Identify Fraud from Enron Email

Note: From steps below it might seem that each step was performed once and after the previous one. However, there were many repetitions and many steps were revisited after changes were made to algorithm or validation approach.

Answer to questions:

1. **Summarize for us 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?**

   This goal of this project is to identify the people involved in the Enron fraud from their financial and communication information. The financial and communication information about Enron employees are provided along with indication whether they are a person on interest (poi). The objective is to utilize this data and develop a classification tool to differentiate between poi people and non-poi people.
   There was one sample in the dataset that was the sum of all samples. This sample was removed as it is likely unintentionally included in the dataset. Additionally, not every person had all the features available. From the data, it is not directly clear if the unavailable features are missing or not applicable to the person. I decided to move forward with replacing the unavailable features with zero. Assuming that some features are unavailable because the do not apply to that person, it is reasonable to consider zero for them.

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.**

   Since the number of features were fairly limited, I decided to choose them manually. First, I identified the features that were unavailable for most of the ‘poi’s and excluded them from feature list. Excluding these features, we would have 16 features left. However, we have only 18 ‘poi’ samples and ideally the features should be less than that to avoid overfitting. I was using a DecisionTreeClassifier (DTC), and removed the features with low ‘feature_importances_’. Since I used DTC-based and linear classifiers, there was no need for scaling.
   Table below show the initial feature_importances observed.

salary	total_payments	bonus	deferred_income	total_stock_value

0.0375	0.05	0.0625	0.0875	0.025
expenses	exercised_stock_options	other	long_term_incentive	restricted_stock
0.175	0.0375	0.175	0.	0.125
to_messages	from_poi_to_this_person	from_messages	from_this_person_to_poi	shared_receipt_with_poi
0.0375	0.025	0.025	0.1375	0.

Most email correspondences features were not included in important features. It seemed to me that a potential good feature would be the ratio of emails communicated with POI relative to total emails. I created three new variables:

1. `to_messages_poi_ratio = from_poi_to_this_person / to_messages`
2. `from_messages_poi_ratio = from_this_person_to_poi / from_messages`
3. `shared_receipt_with_poi_ratio = shared_receipt_with_poi / to_messages`

I included these new features; however, they had minimal improvement on the classification performance. Nevertheless, I decided to keep them in the classifier. I did manually remove and add features based on my intuition and feature_importances value and the final list of features and their importance are listed in table blow.

deferred_income	expenses	exercised_stock_options	other
0.05	0.1	0.2	0.3
shared_receipt_with_poi_ratio	from_messages_poi_ratio	to_messages_poi_ratio	
0.075	0.2	0.075	

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

   I started with DecisionTreeClassifier as it relatively fast and provide insight into features. However, the performance of the algorithm was not satisfactory. Then, I tried SupportVectorMachine (SVM). The linear SVM would train quickly, but did not result in good performance etiher. When I tried the rbf SVM, the performance improved slightly but the training time was very long and would not allow for trial and error approach for tuning the parameters and features. Therefore, I decided against it.
   Finally, I tried the AdaBoost and achieved much better performance than all the classifiers above. Given the performance obtained, I decided to use AdaBoost as the algorithm of choice.

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).**

Each algorithm usually has some parameters that need to specified before the training starts. These parameters are fixed during the training and are affected by training process. Therefore, they can impact the performance of algorithm if not chosen properly. I employed the GridSearchCV function in Sklearn to exhaustively search for various combination of parameters and choose the parameter set that result in best performance.
For the AdaBoost algorithm, I considered the following choices for tunable parameters: n_estimators : [5, 10, 20, 40, 80] and learning_rate : [.25, .5, 1, 2]. Resulting in 20 different combination. The algorithm will be then trained with each combination and the one with highest performance is chosen. The tuned parameters were n_estimators = 40, and learning_rate = 1.
I also tried to tune the parameters of the underlying DecisionTreeClassifier, but I was not successful in finding any parameters for DecisionTreeClassifier that would result in better performance than AdaBoost with default base_estimator.

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

Validation is the process of breaking the data samples into training and test, and utilizing the training data for training the machine learning algorithm and using test data to evaluate its performance. The goal is to ensure the machine learning algorithm can generalize to new and unseen samples and does to blindly memorize the training samples.
Since the number of POI samples were limited, I opted for a kfold cross validation. I employed 5 folds, resulting with 80% training and 20% test samples in each training and testing iteration. To ensure each fold has reasonable number of POI and non-POI samples, I employed the StratifiedKFold function in Sklearn. The average performance of the 5 iteration was then considered as the overall performance.

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.**

To tune the parameters, I started with 'accuracy' metric. But, given that disparity between number of POI and non-POI samples, accuracy would not paint a good picture of the algorithm performance for POIs. I tried both 'precision' and 'recall' then, but focusing on either would sacrifice the other performance of the other one. The recall score represents the probability of catching a POI, while precision score represents the probability of a person being a POI if that individual is flagged y the algorithm as POI. Eventually, I decided to use 'f1' score, to balance between 'precision' and 'recall' while also focusing on performance of POI samples.
The final performance numbers using the tester.py function is presented below. These performance metrics are for the final algorithm after deciding the features and tuning the parameters.
Accuracy: 0.87167 Precision: 0.52462 Recall: 0.39950 F1: 0.45359 F2: 0.41951

References

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.

I relied on class material as well as Sklearn and Python documentation to carry on the project. I have also relied on my previous knowledge on machine learning classification gained from my university education.