Analysis of ENRON Dataset

Project Overview :

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives.

In this project I will build a person of interest identifier based on financial and email data made public as a result of the Enron scandal. I use email and financial data for 146 executives at Enron to identify persons of interest in the fraud case. A person of interest (POI) is someone who was indicted for fraud, settled with the government, or testified in exchange for immunity. This report documents the machine learning techniques used in building a POI identifier.

There are four major steps in my project:
1. Enron dataset
2. Feature processing
3. Algorithm
4. Validation

The goal of this project is to find employees in ENRON who might be involved in corporate fraud based on the publicly available datasets containing financial and email data. This dataset will be useful as it contains the most crucial information about the ENRON's employees which can be used as features in machine learning to predict whether a person might be involved in fraudulent activities or in other words 'is he/she a 'person of interest' (POI)?'.

The dataset contains samples of around 145 employees from ENRON. 18 people have been declared as POI and 127 as nonPOI in the dataset. There are 21 features available for each sample. Some of the features are salary, bonus, stock value, total payments and email related information like number of messages received and sent and number of interactions with POIs. Some of the features with large number of missing values are deferral_payments(107 values missing), loan_advances(142 values missing), restricted_stock_deferred(128 values missing), deferred_income(97 values missing) and director_fees(129 values missing). I found an outlier in

the dataset with key ‘TOTAL’ which looks to be summing the values of various features of each employe. I removed it by dropping the corresponding key and item from data_dict.

As part of the assignment, you should attempt to engineer your own feature that does not come readymade 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.

The new features that I created are ‘to_poi_fraction’ = ‘from_this_person_to_poi’/’from_messages’ and ‘from_poi_fraction’ = ‘from_poi_to_this_person’/’to_messages’. I created these fractions because I felt that it’s the fraction of the total messages sent or received which involved a POI that would help us in deciding whether a person is POI, rather that just an absolute number of messages to and from POI.

I did feature scaling on all the features that I used because I wanted to use them in SVC.
The feature scores for the features that I took into consideration were salary: 18.57570327, to_poi_fraction: 16.64170707, from_poi_fracton: 3.21076192, deferral_payments: 0.21705893, total_payments: 8.86672154, exercised_stock_options: 25.09754153, bonus: 21.06000171, restricted_stock: 9.34670079, shared_receipt_with_poi: 8.74648553, restricted_stock_deferred: 0.06498431, total_stock_value: 24.46765405, expenses: 6.23420114, loan_advances: 7.2427304, other: 4.20497086, director_fees: 2.10765594, deferred_income: 11.59554766, long_term_incentive: 10.07245453.

I used SelectKBest technique with ‘precision’ as scoring function to add one feature at a time in a loop. I used precision as scoring function because here we won’t optimize specificity of the algorithm i.e. if our algorithm should only declare a person as POI if it is pretty sure about it, otherwise we might end up scrutinizing or penalizing the innocent in this ENRON fraud. These scores turned out to be close to maximum for K=5 features in both the classifiers, so I decided to use five features in my model, namely 'salary’, 'to_poi_fraction', 'bonus', 'total_stock_value', ‘exercised_stock_options’. For choosing parameter values of decision tree and svc, I used GridSearchCV to iterate through various options for the parameter and to choose the best performing ones returned by the GridSearchCV.

DTC: Decision Tree Classifier, SVC: Support Vector Classifier
Below is the best precision score achieved after parameter tuning by Grid Search for both the algorithms for each value of K as number of best features.

I tried Decision Tree Classifier and SVM. To compare the performance of the model, I relied upon the recall and precision score of each model in the local data. With the five features that I

chose to use in my model, the Decision Tree classifier gave me precision of 0.275 and recall of 0.27. The SVC gave precision of 0.349 and recall of 0.75. Therefore I chose SVC as my final classifier.

The performance of each Machine Learning algorithm varies greatly based on the selection of the parameters and underlying dataset. Therefore, in addition to choosing the right algorithm for the use case, we also need to tune the parameters based on our training data by trying out variety of values for parameters and choosing the ones for which the classifier performs the best on our training and cross validation data. I tuned the parameters using GridSearchCV and in my final SVC I used parameters: gamma=0.0, tol=0.01, C=10, class_weight=auto, kernel=rbf.

Validation is check the performance of our algorithm on sample dataset. One way to validate our model is to hold out certain amount of data from the available training data to serve as evaluation purpose, using which we can select the algorithm, tune the parameters of our algorithm and measure performance. A classic mistake is to use the same set of training data to train and evaluate the model, which may lead to over optimistic score to our model. Such model may underperform in the new test data, as we might have overlooked the issue of overfitting while using wrong validation technique. I used KFold technique with 10 folds to separate given data in training and validation sets.

To Explain an interpretation of your metrics that says something human understandable about your algorithm's performance.
I used precision, recall and f1 score to evaluate my model during validation phase in each KFold iteration. Precision in our case means the fraction of people who are actually POI out of all the people predicted as POI by our algorithm. Recall means the fraction of people identified as POI out of all the people who are actually POI.

In poi_id.py, The average precision: 0.349, average recall: 0.75, average f1 score: 0.4523 In tester.py the scores are:

Accuracy: 0.79400
Precision: 0.37563
Recall: 0.66750
F1: 0.48073
F2: 0.57772
Total predictions: 14000 True positives: 1335 False positives: 221 False negatives: 665 True negatives: 9781