Enron Final Project

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?

The goal of the project is to create a model for accurately determining whether an individual is a person of interest in the Enron case. Machine learning helps determine which features are most important in distinguishing as POI from a non-POI. The dataset is made up of 27 features with 146 observations. There are 18 POIs and 127 non-POIs (after removal of "TOTAL"). This is important to consider because an algorithm can get 87% accuracy just by always predicting non-POI.

One removed outlier is the “TOTAL” row which just the sum of other rows. There are some individuals who had much higher salaries, bonuses, etc., but they should not be removed because that information could be important for distinguishing them as POIs.

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.

The features were initially selected by the percentage of samples that were NaN. The default threshold was set to 80%. This threshold was tested against 100% (no filtering) and 70%. The most stable model was found with an 80% threshold.

Features selected by this method:

['poi', 'salary', 'deferral_payments', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'to_messages', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi']

The features were scaled by sklearn's StandardScaler. Many of the features do not have the same scale, and normalizing them will prevent a feature with a large range (e.g. salary) from biasing the model.

During model testing, I created 2 new features that calculated for each person the fraction of emails that were received from and sent to a POI. I thought these features could better represent the frequency of communication with POIs rather than just having the absolute number of correspondences with POIs. These new features seems to have no effect on the accuracy of the algorithms, but removing the features used to calculate the new features has a negative impact.

Without new features:

- Accuracy: 0.85100 Precision: 0.43343 Recall: 0.38250

With new features and original features:

- Accuracy: 0.85100 Precision: 0.43343 Recall: 0.38250

With new features and without original features:

- Accuracy: 0.84080 Precision: 0.36103 Recall: 0.25200

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

I ended up using the k-nearest neighbors algorithm. 5 different algorithms were tested: k-nearest neighbors, random forest, decision tree, logistic regression, and MLP. I compared the algorithms using a GridSearchCV for parameter tuning for each one. For these runs I also adjusted the features using the threshold filtering (decribed above), with and without the created features, and with and without scaling. The most balanced result (between recall and precision) was found with an 80% threshold and with scaling.

Comparing the results from several thresholds on the original set of features.

100%

- Accuracy: 0.83547 Precision: 0.37310 Recall: 0.34400

90%

- Accuracy: 0.84107 Precision: 0.40041 Recall: 0.38600

80%

- Accuracy: 0.85100 Precision: 0.43343 Recall: 0.38250

70%

- Accuracy: 0.84247 Precision: 0.40361 Recall: 0.38000

60%

- Accuracy: 0.84227 Precision: 0.39248 Recall: 0.33400

50%

- Accuracy: 0.82827 Precision: 0.35657 Recall: 0.35800

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? What parameters did you tune? (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).

Parameter tuning means adjusting parameter values until some optimal set of parameter values are found. Optimality can depend on the requirements of the task. One researcher might want to find the most accurate model, while another researcher might want to find a model that is right most of the time but does not take too long to train. As mentioned above, I tuned the parameters using GridSearchCV.

The tuned parameters were:

- n_neighbors(number of neighbors)
- weights (weight function)
- algorithm (algorithm used to compute nearest neighbors)
- leaf_size (leaf size for the computational algorithm)
- p (the distance calculation method)

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 testing an algorthm's accuracy. The classic mistake is using the same dataset for training the model and testing the model. Doing this will cause overfitting of the model, and the model will not be generalizable to any new data points. Since I used GridSearchCV for parameter tuning, cross validation was automatically used. The default 3-fold validation was used. This constraint was to reduce the training and parameter tuning time for 5 classifiers. The final model was tested with the given tester function which uses StratifiedShuffleSplit. Stratification is good to use for this dataset because there is a major disproportion between the number of POIs and non-POIs. With stratification the proportion of POIs to non-POIs is preserved which reduces the variation between estimator for each split.

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.

Metrics:

- accuracy: 0.85
- precision: 0.43343
- recall: 0.38250

Accuracy tells us that the model was able to accurately label 85% of the individuals as a POI or a non-POI in the test dataset. Precision tells us that out of all of the individuals that were labeled by the model as a POI 43% were actually POIs. Recall tells us that out of all of the individuals who were actually POIs only 38% were labeled by the model as a POI.