Enron Submission Free-Response 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? [relevant rubric items: "data exploration", "outlier investigation"**

   The goal was to predict whether a person is a POI based on financial and email data. Machine learning elucidates trends in the financial and email data and their relationship to POI labels. A properly trained and robust classifier can be used to identify potential POIs in a new data set for which only email and financial data are available.

   The data set contains 145 data points with 21 features (including 'poi' and 'email_address'). The POI label can be used as an outcome variable and the remaining features potential predictors to train a classifier. The data set is skewed, consisting of 127 non-POIs and 18 POIs (and 1 "TOTAL" data point).

   Scatterplots for each feature were first generated to visualize the spread of data and the prevalence of NaN values. POI points and non-POI points were differentiated on the plots as outliers corresponding to POI may be important for the analysis and should not be removed.

   The plots are saved in the folder *"plots"* and can be produced using the *outliers.py* file. Based on the plots, the following outliers were removed:

   1) "TOTAL"
   2) "FREVERT MARK A" [a non-POI outlier for 'deferral_payments', with a value twice that of the next highest value]
   3) "BHATNAGAR SANJAY" [a non-POI outlier for 'restricted_stock_deferred', with a value of 15 million compared to all other points at less than 1 million]
   4) "HUMPHREY GENE E" [a non-POI anomaly for 'from_messages_POI' with a value of 1.0. This value is the fraction of 'from' messages sent to POI. I found it strange that he was sending emails exclusively to POIs]

   Some of the features identified from the plot as not being useful:

   1) 'directors_fees' & 'restricted_stock_deferred'-> few non-NaN data points, all of which belong to non-POIs
   2) 'loan_advances' -> only 3 data points

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. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

New Features:

I created 3 new variables:

1) Fraction of 'to' messages from POI, 'to_messages_POI'
   formula: 'to_messages_POI' = 'from_poi_to_this_person'/ 'to_messages'
2) Fraction of 'from' messages to POI, 'from_messages_POI'
   formula: 'from_messages_POI' = 'from_this_person_to_poi'/'from_messages'
3) Fraction of messages received that were shared with POI recipient, 'shared_messages_POI'
   formula: 'shared_messages_POI' = 'shared_receipt_with_poi'/'to_messages'

These variables were created as they would be more informative on the distribution of correspondence with POIs when normalized by the total volume of correspondence that each person has.

Features selected for POI Identifier:

8 features were eventually used for the POI identifier, identified using SelectKBest. The choice of k=8 for SelectKBest was optimized and selected through tuning with GridSearch with SVM, using the following range of values for k:

params = dict(feat_select__k = range(4, len(features_list)-1)

SelectKBest was run again with k='all' to take a closer look at how the top 8 features and their scores compared with the remaining features, in order to verify that the choice of k=8 made sense.

The top 8 features and their scores are as follows:

bonus	22.1565110201
salary	19.673723248
exercised_stock_options	19.2032749527
total_stock_value	17.8378473776
from_messages_POI	16.0390837017
long_term_incentive	15.4633242445
deferred_income	13.213336839
total_payments	10.8991751192

The remaining features and their scores are as follows:

other	8.82411139764
loan_advances	7.64794560597
shared_messages_POI	7.13190743254
restricted_stock	6.07595633617
expenses	4.54156387444
to_messages_POI	4.15930147382
director_fees	1.57495282377
restricted_stock_deferred	0.975499393516
deferral_payments	0.75016732503

Looking at the scores, the choice of k=8 appears to be reasonable, with a score cut-off at 10 to exclude all other features. While it could be argued that 'other', 'loan_advances' and 'shared_messages_POI' could be included in the POI predictor as their scores were not too much lower than the lowest – scoring feature in the top 8 [i.e. 'total payments' at 10.899], I decided to stick with k=8 for the following reasons:

1) k=8 was already giving me acceptable performance. Therefore, I did not feel that it was justified making the model more complex by adding more features.

2) The next highest scoring feature outside of the top 8 was 'others'. I did not want to include this as this feature was not specific and could encompass very different data for different subjects. I would be skeptical of its reliability as a predictor for new data.

Prior to passing the data into SVM, the features were scaled using MinMaxScaler to ensure that all data was on the same order of magnitude so that SVM could work.

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

I used support vector machines (SVM) with a sigmoid kernel. I also tried decision trees('DT'), NaiveBayes('NB'), and randomforest('RF'), for which detailed results are shown in the appendix. A summary of the performances for each algorithm is illustrated in the table below:

Metric	SVM	DT	NB	RF
Single 70/30 split:				
Accuracy	0.860	0.814	0.814	0.884
Precision	0.500	0.333	0.333	1.000
Recall	0.330	0.333	0.333	0.167
100-fold stratified shuffle split:				
Accuracy	0.844	0.840	0.871	0.877
Precision	0.398	0.410	0.524	0.597
Recall	0.330	0.455	0.375	0.245

With the exception of RF, all algorithms gave precision and recall of greater than 0.3. Comparing the precision using stratified shuffle split, DT and NB had values of 0.410 and 0.524 respectively, higher than that of SVM at 0.398.

For recall, DT and NB had values of 0.455 and 0.375 respectively, again higher than that of SVM at 0.33.

While SVM appeared to perform worse than DT and NB, its performance was acceptable, with precision and recall greater than 0.3. I decided to go with SVM because I felt it was a more robust model that could be tuned and would perform better for new data, as opposed to Naïve Bayes (which can't be tuned) and decision trees which tend to overfit (particularly in this case where stratified shuffle split is used and the data set is small)

The SVM classifier used is as follows:
Pipeline(steps=[('scaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('svm', SVC(C=70, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.5, kernel='sigmoid', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False))])

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). [relevant rubric item: "tune the algorithm"]

Tuning is to vary the parameters for a particular algorithm in order to optimize algorithm performance on the training data. If not done properly, the resulting model could be suboptimal.

I used GridSearchCV to tune the algorithm's parameters, using a range of reasonable values for gamma, C and kernel:

params = dict(svm__gamma=[0.01, 0.1, 0.5, 1.0],
svm__C= range(10,100,10), svm__kernel=['rbf', 'poly', 'sigmoid', 'linear'])

For C in particular, I was mindful not to have values that were too large as it would take a long time to train and would be likely to overfit to the data.

The best parameters were identified from the GridSearchCV tuning process and fit into the final SVM classifier for export.

5. **What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]**

Validation is evaluation of classifier performance on a test set that has been held out from the training data. Not holding out a test set and instead, training on all the data available would make validation results appear more rosy than they actually were.

The data set was split into training and test sets in a 70/30 split. All training and parameter tuning was done using only the training set. The test set was used for validation. Evaluation metrics were then calculated to assess the performance of the classifier on the test set. As the data set is skewed with a small proportion of POI, evaluation metrics used were accuracy, recall and precision using the sklearn.metrics module.

Due to the small data set and small proportion of positive labels, stratified shuffle split was also utilized to evaluate performance over 100 folds, adapted from the tester.py data.

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. [relevant rubric item: "usage of evaluation metrics"]**

<u>Evaluation metrics from single 70/30 training-test split:</u>
Recall: 0.33
Precision: 0.5
Accuracy: 0.860

<u>Evaluation metrics from 100-fold stratified shuffle split:</u>
Recall: 0.33
Precision: 0.39759
Accuracy: 0.844

Based on the second set of metrics:
The recall is 0.33 i.e. 33% of all true POIs will be correctly identified as POI by the classifier

The precision is 0.398, i.e. 39.8% of all samples identified as POI by the classifier are true POIs
The accuracy is 0.844, i.e. 84.4% of all samples will be labelled correctly by the classifier.

APPENDIX

OUTPUT FOR OTHER TRAINING ALGORITHMS

1) Gaussian NB

Features used in classifier: ['poi', 'salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'from_messages_POI']

Performance using 100-fold stratified shuffle split:
GaussianNB()
Accuracy: 0.87133 Precision: 0.52448 Recall: 0.37500 F1: 0.43732 F2: 0.39767
Total predictions: 1500 True positives: 75 False positives: 68 False negatives: 125 True negatives: 1232

Accuracy, Precision and Recall calculated using sklearn.metrics and 70/30 data split :
0.813953488372
0.333333333333
0.333333333333

2) RandomForest

Features used in classifier: ['poi', 'salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'from_messages_POI']

Performance using 100-fold stratified shuffle split:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', max_leaf_nodes=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=1,
oob_score=False, random_state=None, verbose=0,
warm_start=False)
Accuracy: 0.87733 Precision: 0.59756 Recall: 0.24500 F1: 0.34752 F2: 0.27778
Total predictions: 1500 True positives: 49 False positives: 33 False negatives: 151
True negatives: 1267

Accuracy, Precision and Recall calculated using sklearn.metrics and 70/30 data split :
0.883720930233
1.0
0.166666666667

3) DecisionTreeClassifier

Features used in classifier: ['poi', 'salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'from_messages_POI']

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
random_state=None, splitter='best')
Accuracy: 0.84000 Precision: 0.40991 Recall: 0.45500 F1: 0.43128 F2: 0.44521
Total predictions: 1500 True positives: 91 False positives: 131 False negatives: 109 True negatives: 1169

Accuracy, Precision and Recall calculated using sklearn.metrics and 70/30 data split :
0.813953488372
0.333333333333
0.333333333333

4) SVM

Features used in classifier: ['poi', 'salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'from_messages_POI']

Pipeline(steps=[('scaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('svm', SVC(C=70, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.5,
kernel='sigmoid', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False))])
Accuracy: 0.84400 Precision: 0.39759 Recall: 0.33000 F1: 0.36066 F2: 0.34161
Total predictions: 1500 True positives: 66 False positives: 100 False negatives: 134 True negatives: 1200

Accuracy, Precision and Recall calculated using sklearn.metrics and 70/30 data split :
0.860465116279
0.5
0.333333333333