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 of this project is to explore machine learning techniques (capabilities) on Enron financial and email dataset.The data was collected after Enron Corporation bankruptcy and contains information from about 150 Enron employees and some people doing business with Enron. It was made public during its investigation by the Federal Energy Regulation Commission. Giving us a unique opportunity to research publicly available collection of authentic emails, as such data is typically bound by privacy and legal restrictions.The task is to build an algorithm to identify Employees who may have committed fraud while working in (or with) Enron. In the dataset, these people are labelled as POI (person of interest).

POI defines someone who has been convicted, testified in exchange for immunity or settled without admitting guilt.Summary tables, scatter and density plots were used to research the data. Data points furthest from the main cluster were investigated and manually chosen to be removed or not from the dataset. No automated outlier removal is done due to relatively small and unbalanced data sample. Our dataset includes only 18 POI's, as removing any of them to improve the accuracy of the identifier seemed too costly.

Outliers chosen to remove:

```
my_dataset.pop('DERRICK JR. JAMES V') #'exercised_stock_options' > Q3+1.5*IQR
my_dataset.pop('TOTAL')
my_dataset.pop('THE TRAVEL AGENCY IN THE PARK')
```

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"]

100 new features(*'x0','x1',...'x99'*) were created to represent text data from the emails. Emails sent to POI were grouped by the sender. TfidfVectorizer used to assign numerical values to the words and linear dimensionality reduction performed with sklearn TruncatedSVD to include only first 100 components. Another two features fraction_to_poi and fraction_from_poi created are number of messages to POI (or from POI) scaled by the total amount of emails sent.

Due to sparse nature of the data (a lot of zeros), feature selection was done first on *'x0,x1...x99'* features. From sklearn feature selection documentation, RandomizedLogisticRegression was recommended as suitable feature selection model for this classification task. Features with score

higher than zero([*(0.23, 'x6'), (0.055, 'x12'), (0.02, 'x71'), (0.02, 'x57'), (0.015, 'x44'), (0.01, 'x8'), (0.01, 'x64'), (0.01, 'x50'), (0.005, 'x89'), (0.005, 'x84'), (0.005, 'x58'), (0.005, 'x34')*...]) were added to feature_list.

Further feature selection was done by checking the assigned weights from multiple selectors (SelectKBest, feature_importance_ with RandomForestClassifier) and manually pruning features with smallest values. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Final choice of features: [*'bonus', 'exercised_stock_options', 'expenses','fraction_to_poi',*

'restricted_stock','salary', 'shared_receipt_with_poi', 'total_payments', 'x6','x8','x12','x34','x58']

SelectKBest score: [*(16.575, 'exercised_stock_options'), (15.471, 'x34'), (12.458, 'fraction_to_poi'), (7.749, 'shared_receipt_with_poi'), (6.118, 'bonus'), (4.905, 'expenses'), (1.985, 'x6'), (1.775, 'x58'), (1.124, 'salary'), (0.468, 'x12')*]

Selected features were normalized with *sklearn.preprocessing.StandardScaler* by removing the mean and scaling to unit variance.

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"]

Chosen algorithm: SVC(*kernel='sigmoid'*)
To choose the best identifier for this project, loop was made to check the performance of multiple classifiers simultaneously. Classifiers chosen: Kneighbors,NearestCentroid,Random Forest,Decision Tree,Naive Bayes, AdaBoost and SVC with multiple kernels.

Kneighbors and NearestCentroid classifiers needed fewest features to start classifying positive samples. SVC performed noticeably better than all of the rest algorithms, when only features from text data were trained. Kneighbors and Decision Tree based classifiers seem to have lower optimal number of features compared to SVC or NearestCentroid.

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"]

Parameter tuning is essentially selecting the best parameters for an algorithm to optimize its performance. It enables your classifier to be more flexible by adapting to specific characteristics of given task. For example, considering Enron classification task, setting gamma to be 0.1 or more would lead to overfitting having high bias and low variance.

Parameters were chosen in automated form with a help of GridsearchCV function. An exhaustive search algorithm over specified parameter values. best_params_ attribute shows parameters giving the highest accuracy score by the estimator.

Brief explanation of SVC parameters:

C is a penalty parameter for misclassifying labels. Raising C , makes svm to aim for better precision by penalizing more strongly for every false prediction. Small C makes the decision surface smoother (might be used to reduce overfitting)

Gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.

Kernel is a mathematical function to measure similarity between samples. Most widely used kernel (default parameter with sklearn.svm.svc) is 'rbf'. Setting 'linear' kernel for SVC, means not using any kernel.

class_weight can be set as scaling constant on parameter C for a label(i). Might be useful for unbalanced data as our Enron dataset (e.g. many negative and few positive).

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”]

The idea behind cross validation is not using the whole data set when training your estimator. A part of the data set (common case: 30% of the dataset) is left out. Then when training is completed, the data that was removed can be used to test the performance of the learned model.

My chosen classifier is optimized by cross-validation, which is done using the sklearn.grid_search.GridSearchCV object on a training set that comprises only 67% of the data. The performance of the selected parameters and trained model is then measured on a test set that was not used during the training.

Wrong way to do cross-validation is to perform feature selection on a full data set. If some simple classifier chosen to pick the best features will be trained before splitting the data into training and test samples, the features selected will be biased by the samples that going to be stored in test data, giving an unfair advantage to a classifier chosen to identify labels of that test_set.

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”]

Resulsts from tester.py:

Accuracy: 0.92093 Precision: 0.80974 Recall: 0.53200 F1: 0.64212 F2: 0.57118

Accuracy(proportion of correct guesses) score on its own does not give us an informative answer. Unbalanced nature of Enron dataset(18 POI's out of 150 samples) makes accuracy not a reliable metric for the performance of a classifier. For example, failing to find any true positive sample and classifying all samples as not POI's would give us relatively high 0.88 score. 0.53 Recall indicates that from all positive POI's in the test dataset our trained classifier finds approximately half of them. From the precision score(0.8), we can assume that the classifier makes a careful prediction accusing someone being involved in fraud(with 80% probability of getting it right).