Intro to Machine Learning
Final Project Summary

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 of this project is to find a machine learning algorithm that achieves better than .3 precision and recall when predicting whether or not a person can be identified as a “person of interest” (aka they are likey to commit fraud). The dataset provided for training is a combination of financial features and email features for previous Enron employees, along with labels as to whether or not that person is considered a person of interest (POI). The algorithms I chose to test were supervised learning algorithms since the data provided known labels (poi vs non-poi) to train the algorithms with. An employee was labeled as a person of interest if they met the following criteria: "individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity."

During my exploration of the dataset, I found 3 outliers that I chose to remove. The first was due to a spreadsheet quirk in importing the "TOTAL" row that was an aggregate of each of the financial features for all rows. I also found in my exploration that there was a name in that dataset called "THE TRAVEL AGENCY IN THE PARK". This did not seem like an individual employee, so I removed this one from the dataset. "LOCKHART EUGENE E" had "NaN" for all features, so I deleted him from the dataset as well as his data provided no useful information. Additionally, "BELFER ROBERT" and "BHATNAGAR SANJAY" had incorrect financial data due to values being transposed between columns. I manually corrected the data according to the values found in "enron61702insiderpay.pdf". Finally, I changed all “NaN” values of the financial features to 0 for those employees listed in “enron61702insiderpay.pdf” since we did have their financial data, they just did not receive those types of payments/stock options and the spreadsheet used ‘-‘ instead of 0 to show this.

Starting out there were 146 rows of data, however once the data was cleaned, there were 143 rows of data, 1 set of labels and 20 features (21 with the new feature I created). Of the 143 employees with data available, 18 were labeled as POIs and 125 were labeled as Non-POIs. Regarding email features vs financial features, 57 employees had missing email feature data. I left these values as “NaN” since there was no information available. All employees in this dataset had financial feature data.

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”, “intelligently select features”, “properly scale features”]

I started with using SelectKBest to give me all of the features in the order of their importance, and then I fed each feature in one by one to the pipeline and chose the number that resulted in the highest precision and recall scores(In other words, I started with k=1, k=2, k=3 all the way up to k=’all’). With k=1, precision was 0.78, recall was 0.71, and f1 was 0.74. When I increased K to K=2, all 3 scores were 0.84, when

I increased to k=3 all 3 scores were 0.9, when I went to k=4 precision was .88, recall was .82 and f1 was .85. Simply using the default of k=10 resulted in .84 precision, .79 recall and .81 f1. I chose to use k=3 since that gave me the best result for all 3 scores combined(K=6 was also an option as it gave a precision score that was higher(0.92), however it also had a lower recall score(.81)). The 3 features I decided to use were 'total_stock_value', 'exercised_stock_options' and 'bonus' for my features list since these were the 3 highest scoring features k=3 gave the optimal results (by "highest scoring" I mean that when I used k="all" and ordered the scores from highest to lowest, these were the top 3). My pipeline included a MinMaxScaler, Primary Component Analysis, GridSearchCV and a Decision Tree. I scaled the features using MinMaxScaler since Decision Trees calculate the decision boundary by maximizing the distance between each category, so if one feature has a wider range of values, it would be weighted more in the decision process than any other feature, so scaling helps each feature get weighted accurately. I also used f1 scoring as a parameter to the GridSearchCV so it would choose the combination of parameters that would give the highest f1 score. Since there are so few POI's compared to Non-POIs, I used f1 scoring as a parameter for the GridSearchCV instead of accuracy. High accuracy scores with this dataset can be deceiving since 87% of the dataset is Non-POIs, you would automatically get a high accuracy score by simply assigning "Non-POI" to every data point. F1 scoring is much more robust since it evaluates precision and recall.

When asked to create a new feature, my intuition told me that those most likely to commit fraud might be inclined to engage in other risk taking, like investing more of their money in the stock market compared to safer forms of payments. The feature I created was a proportion of the total stock value to the sum of the total payments and total stock values combined. This feature ending up not having a very high score (0.0002) when using SelectKBest, so it was not used in the final analysis. Below are the SelectKBest scores. Since I used PCA in my pipeline along with a Decision Tree, features were converted to "components" so feature importances no longer correspond to features, but to the newly unnamed components.

Feature	SelectKBest scores
'total_stock_value'	22.511
'exercised_stock_options'	22.349
'bonus'	20.792

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 ended up using the Decision Tree Classifier. I tested the Gaussian Naïve Bayes Classifier as well as the Decision Tree. Below are the results of each.

Gaussian Naïve Bayes	**Accuracy**	**Precision**	**Recall**	**F1**
poi_id.py	0.907	0.890	0.910	0.900
tester.py	0.764	0.246	0.374	0.296

Decision Tree	**Accuracy**	**Precision**	**Recall**	**F1**
poi_id.py	0.897	0.900	0.900	0.900
tester.py	0.827	0.431	0.385	0.047

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). [relevant rubric items: “discuss parameter tuning”, “tune the algorithm”]

Tuning the parameters of an algorithm means testing out different parameter values to find the combination that optimizes the result of the algorithm. If you don’t do this well, you might miss finding patterns in the data (i.e. data might not be linearly separable, however changing the “kernel” parameter to “rbf” in a SVM allows you to create a boundary that is linearly seperable) . For my algorithm, my strategy was to use PCA to project the features to the primary components, then add one parameter/value set to the GridSearchCV at a time, using the scoring = ‘f1’ parameter, until I could get a > 0.3 precision and recall score. I started with the “criterion” parameter with the value options of “gini” or “entropy”. Also, to ensure all test results are consistent, I removed the randomization of the Decision Tree and set “random_state” to 10 to start out and see what resulted. I tested this out for just 1 feature all the way to using all features to see what gave the best result. In the end, the optimized parameters included:

- (SelectKBest) **K =** 3
- (Decision Tree) **Criterion :** ‘entropy’
- (Decision Tree) **Random_state**: 10
- (GridSearchCV) **Scoring:** ‘f1’

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 items: “discuss validation”, “validation strategy”]

Validation is a method of testing whether or not your algorithm is doing what you want it to do by splitting the data into a training and testing set so you have an independent dataset to use to get an estimate of your algorithms performance. If you do this wrong, you can overfit your data. I used train_test_split cross validation with 30% of the data in the test set and 70% in the training set, fit the training set to the algorithm, created predictions with the algorithm and then evaluated how the predictions compared to the true labels in the test set.

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”]

The evaluation metrics that I used were recall, precision, and f1 score. The recall score was .385, which means out of all actual POIs, the algorithm correctly identified 38.5% as being a POI (or 61.5% were incorrectly labeled as a Non-POI when they were actually a POI). The precision score was .431, which means 43.1% of all people labeled as POI’s were indeed a POI (56.9% were labeled as a POI, but were actually a Non-POI). The F1 score is a combination of precision and recall, the higher the F1 score, the better the algorithm is doing at classifying.