Project Goal

The goal of this project is to build a classifier that can identify a person of interest in the Enron fraud case. Machine learning is useful because the data set is very large for this project and we will use ML techniques to explore the data, condense the data, engineer new features, and test the accuracy of our model predictions.

I inspected the enron61702insiderpay pdf file and removed three outliers:

TOTAL: This should not be in the data set as it skews the analysis

LOCKHART, EUGENE E: This record contained no data

THE TRAVEL AGENCY IN THE PARK: This is not a person and subsequently can't be a POI.

I defined an outlier as a very extreme data point. Since we only have 146 records in this data set, I did not want to remove any additional data. Additionally, there are 18 POIs after the outlier's have been removed.

Features Used

I used correlation to select the top 6 features and this resulted in taking the data set from 21 features to 6 excluding poi. I did this based on correlation:

```
bonus                       0.358486
deferral_payments          -0.039880
deferred_income            -0.274150
director_fees              -0.121886
exercised_stock_options     0.386853
expenses                    0.203545
from_messages              -0.034671
from_poi_to_this_person     0.189352
from_this_person_to_poi     0.128908
loan_advances               0.220183
long_term_incentive         0.256405
other                       0.169829
restricted_stock            0.247652
restricted_stock_deferred  -0.021548
salary                      0.338851
shared_receipt_with_poi     0.239625
to_messages                 0.107431
total_payments              0.242020
total_stock_value           0.382623
```

I chose the following features:

```
features_list = ['poi', 'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'long_term_incentive', 'deferred_income']
```

Additionally, I created a new feature called cash_payments which was based on salary, bonus, exercised stock options and director fees. I wanted to test if there was an impact on the classifier of the take home pay for a person. Therefore, during my first pass I tested the classifier with 7 features, but my classifier did not meet the required precision and recall in tester.py, so I dropped the cash_payments field in attempts to improve these scores. After running the classifier with 6 features, I was able to generate a passing score.

There are many options for feature selection and I arbitrarily chose 7 for my first pass. I followed the advice from a udacity forum and did trial by error first with feature selection by choosing the features with the highest correlation to POI. If I had not been able to achieve satisfactory scores, then I would have used other methods of feature selection such as SelectKbest.

After feature selection, I used PCA to further condense the features. I scaled the data prior to PCA because I wanted to ensure that each feature was treated equally by PCA and by the supervised learning algorithms.

Algorithm Used

I tested the random forest (RF) classifier and the gaussian naïve bayes (NB) classifier. I chose the NB classifier because it had more consistent accuracy scores between the training and testing sets. Additionally, I was worried the RF classifier was overfitting because it scored almost perfect on the training set and had a significant drop in accuracy on the testing set.

Parameter Tuning

Parameter tuning allows you to adjust variables in an algorithm, so that you can improve the score. For example, I used grid search to find the optimal amount of pca components and to tune the random forest classifier to yield better accuracy than an untuned model. More specifically, I tuned min_samples_split of the decision tree between 3 samples and 50 to tune between precision and recall since the gridsearch was scored on F1.

Evaluation Metrics & Validation

I used a holdout evaluation by splitting the data into independent testing and training sets. I did this to validate for overfitting and to see how my classifier performed against an independent test data set. In this case, my test set was 30% of the overall data.

Validation is evaluating the classifier performance on the training set against the test set. The goal is to build a classifier that performs well on the independent testing and training data sets. Once a classifier has been validated, then it can be used to predict new data sets.

Additionally, the default train/validation subsets for Gridsearch is 3. Meaning that the average of precision and recall would be scored against all 3 of these splits. I noticed that tester.py was based on 1000 folds of `StratifiedShuffleSplit`. Therefore, I used SSS in the gridsearch of 100 folds to ensure that the classifier was robust and built similar to how tester.py is constructed.

The evaluation metrics that I used were precision and recall. Precision measures of the people I classified as a POI, how many that I classified correctly. Recall measures out of the POIs, how many that I classified correctly as a POI. For instance, if we have 100 POIs in the dataset, our classifier flags 50 of them, and only 25 of the flagged ones are correct then the metrics would be as follows:

Recall: 25/100 or 25%

Precision: 25/50 or 50%

References:

Most of my sources were from the udacity machine learning nano degree and the udacity forums. Additionally, I also researched scikit-learn.org heavily for pipeline examples, feature importance, and the definition of precision and recall

I also used this link for a great definition of precision and recall: https://www.quora.com/What-is-the-best-way-to-understand-the-terms-precision-and-recall