Q1)

The goal of the project is to construct a Person of Interest (PoI) identifier based on data from the Enron fraud scandal.

Data is separated into:

- Financial Features: Financial data of Enron personnel
- Email Features: Email data of Enron personnel
- PoI Labels: Personnel labelled as a PoI

A super outlier "TOTAL" was observed which is the aggregate of values in the dataset and it was removed. Remaining outliers were retained as they might help to identify PoIs for example exceptional high salary.

```
Total Number of Data Points: 145
Number of PoIs: 18
Number of Non-PoIs: 127

Feature: poi ,  Has Nan: False
Feature: salary ,  Has Nan: True
Feature: deferral_payments ,  Has Nan: True
Feature: total_payments ,  Has Nan: True
Feature: loan_advances ,  Has Nan: True
Feature: bonus ,  Has Nan: True
Feature: restricted_stock_deferred ,  Has Nan: True
Feature: deferred_income ,  Has Nan: True
Feature: total_stock_value ,  Has Nan: True
Feature: expenses ,  Has Nan: True
Feature: exercised_stock_options ,  Has Nan: True
Feature: long_term_incentive ,  Has Nan: True
Feature: restricted_stock ,  Has Nan: True
Feature: director_fees ,  Has Nan: True
Feature: to_messages ,  Has Nan: True
Feature: from_poi_to_this_person ,  Has Nan: True
Feature: from_messages ,  Has Nan: True
Feature: from_this_person_to_poi ,  Has Nan: True
Feature: shared_receipt_with_poi ,  Has Nan: True
```

Dataset Details (Original Features with Two Removed)

Dataset Details:

- Total Number of Data Points: 145
- Allocation Across Classes: 18 PoIs, 127 Non-PoIs
- Number of Features Used: 19 for feature selection, 6 for final features
- Features with Missing (NaN) Values: All features except poi has NaN values

Removed feature "email_address" as it is just a representation of personnel email addresses and should not have any effects on features rankings.

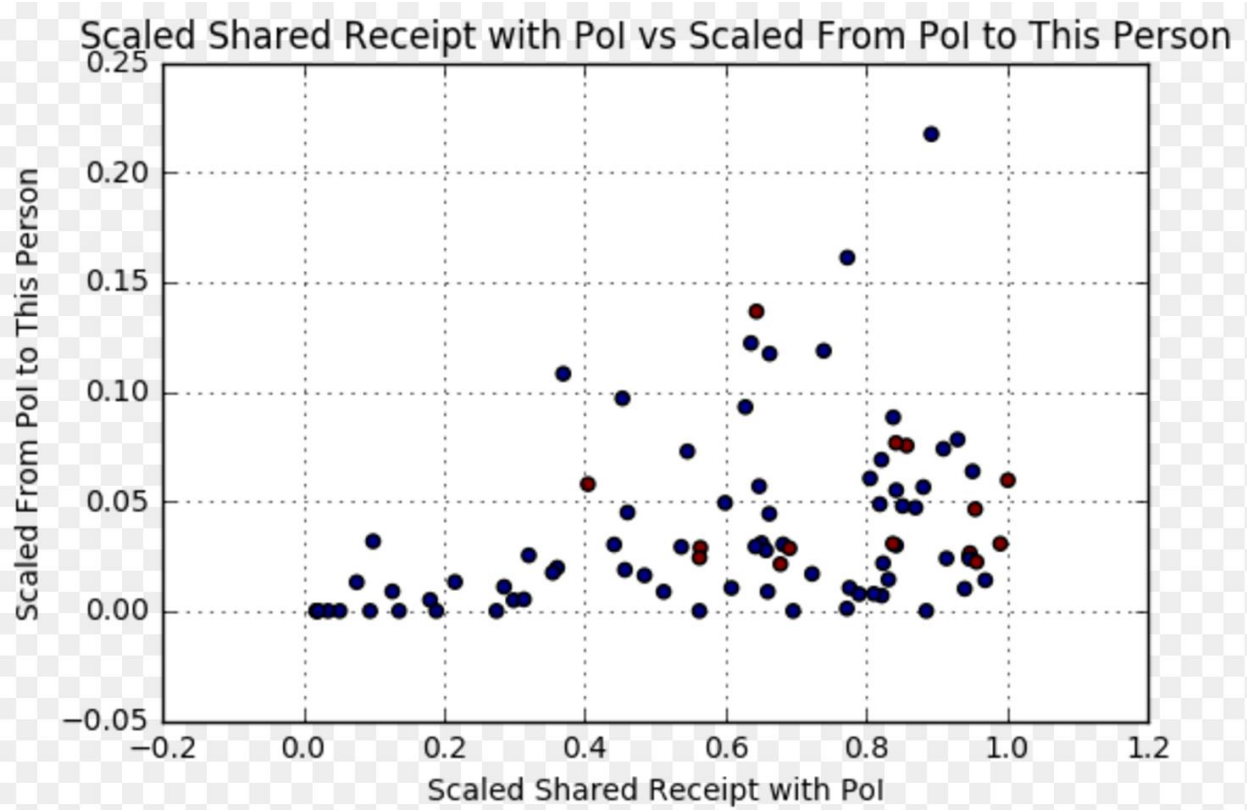
Unsure what does "other" refer to which maybe miscellaneous payments. Removed this feature as there was no clear indication of its function/meaning in the dataset.

Q2)

Two new features were created, “scaled_share_with” which indicates the scaled amount of email receipts shared with PoIs over the total number of email messages.

“scaled_from_poi” indicates scaled the number for emails sent by PoIs to this person over the total number of emails.

Rationale for creating them is that PoIs should be receiving and sharing more emails between them as compared to non-PoIs.



Scatter-plot of Scaled Shared Receipt with PoI vs Scaled From PoI to This Person Colored by PoI

From the scatter-plot, if a person has around between 0 to 2% and 16 to 22% of their email sent by PoIs to them and, 0 to around 39% of their emails consisting of shared receipts with PoIs, then the person is likely not a PoI.

The 20 features (Including the newly created ones) were then fed into the SVC and DecisionTree pipelines with MinMaxScaler to select the best parameters and number of features.

Decision to scale the features were due to that some had massive scales such as salary. Features compared with salary would be skewed towards it due to its values.

Using MinMaxScaler to scale all the features to proportions and then compare them will be the best option.

From the SVC pipeline results, the best features percentile is 60% and parameters are:

- Gamma: 0.0
- C: 100000.
- Kernel: RBF

```
SVC Best Parameters: {'clf__gamma': 0.0, 'clf__C': 100000.0, 'spercentile__percentile': 60, 'clf__kernel': 'rbf'}
Best Recall Score: 0.32
Best SVC Estimator: Pipeline(steps=[('spercentile', SelectPercentile(percentile=60,
         score_func=<function f_classif at 0x0000000009ACE668>)), ('minmaxer', MinMaxScaler(copy=True, feature_range=(0, 1))),
 ('clf', SVC(C=100000.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.0, kernel='rbf',
  max_iter=-1, probability=False, random_state=42, shrinking=True,
  tol=0.001, verbose=False))])
```

Results from SVC Pipeline

For DecisionTree, the best features percentile is 50% and parameters are:

- min_samples_split: 1
- random_state: 34

Originally, the random_state passed to the pipeline was in increments of 10s; 10, 20, 30, etc but since tester.py results were still below 0.3 for recall and precision, I decided to fine tune this parameter further by incrementing 1 resulting in SelectPercentile 50% , random_state 34 and min_samples_split 1.

```
Decision Tree Best Parameters: {'spercentile__percentile': 50, 'clf__random_state': 34, 'clf__min_samples_split': 1}
Best Recall Score: 0.34
Best DecisionTree Estimator: Pipeline(steps=[('spercentile', SelectPercentile(percentile=50,
         score_func=<function f_classif at 0x00000000098053C8>)), ('minmaxer', MinMaxScaler(copy=True, feature_range=(0, 1))),
 ('clf', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=1, min_weight_fraction_leaf=0.0,
            presort=False, random_state=34, splitter='best'))])
```

Results from DecisionTree Pipeline

The features were then scaled and data passed into SelectPercentile to select the top 60 and 50% of the features

```
Feature: salary , Support: True , Score: 18.575703268
Feature: deferral_payments , Support: False , Score: 0.21705893034
Feature: total_payments , Support: True , Score: 8.86672153711
Feature: loan_advances , Support: True , Score: 7.24273039654
Feature: bonus , Support: True , Score: 21.0600017075
Feature: restricted_stock_deferred , Support: False , Score: 0.0649843117237
Feature: deferred_income , Support: True , Score: 11.5955476597
Feature: total_stock_value , Support: True , Score: 24.4676540475
Feature: expenses , Support: True , Score: 6.23420114051
Feature: exercised_stock_options , Support: True , Score: 25.0975415287
Feature: long_term_incentive , Support: True , Score: 10.0724545294
Feature: restricted_stock , Support: True , Score: 9.34670079105
Feature: director_fees , Support: False , Score: 2.10765594328
Feature: to_messages , Support: False , Score: 1.69882434858
Feature: from_poi_to_this_person , Support: False , Score: 5.34494152315
Feature: from_messages , Support: False , Score: 0.164164498234
Feature: from_this_person_to_poi , Support: False , Score: 2.42650812724
Feature: shared_receipt_with_poi , Support: True , Score: 8.74648553213
Feature: scaled_share_with , Support: True , Score: 8.74648553213
Feature: scaled_from_poi , Support: False , Score: 5.34494152315
```

Top 60% Features

Top 60% of features:

- exercised_stock_options , Support: True , Score: 25.0975415287
- total_stock_value , Support: True , Score: 24.4676540475
- bonus , Support: True , Score: 21.0600017075
- salary , Support: True , Score: 18.575703268
- deferred_income , Support: True , Score: 11.5955476597
- long_term_incentive , Support: True , Score: 10.0724545294
- restricted_stock , Support: True , Score: 9.34670079105
- total_payments , Support: True , Score: 8.86672153711
- shared_receipt_with_poi , Support: True , Score: 8.74648553213
- scaled_share_with , Support: True , Score: 8.74648553213
- loan_advances , Support: True , Score: 7.24273039654
- expenses , Support: True , Score: 6.23420114051

```
Feature: salary , Support: True , Score: 18.575703268
Feature: deferral_payments , Support: False , Score: 0.21705893034
Feature: total_payments , Support: True , Score: 8.86672153711
Feature: loan_advances , Support: False , Score: 7.24273039654
Feature: bonus , Support: True , Score: 21.0600017075
Feature: restricted_stock_deferred , Support: False , Score: 0.0649843117237
Feature: deferred_income , Support: True , Score: 11.5955476597
Feature: total_stock_value , Support: True , Score: 24.4676540475
Feature: expenses , Support: False , Score: 6.23420114051
Feature: exercised_stock_options , Support: True , Score: 25.0975415287
Feature: long_term_incentive , Support: True , Score: 10.0724545294
Feature: restricted_stock , Support: True , Score: 9.34670079105
Feature: director_fees , Support: False , Score: 2.10765594328
Feature: to_messages , Support: False , Score: 1.69882434858
Feature: from_poi_to_this_person , Support: False , Score: 5.34494152315
Feature: from_messages , Support: False , Score: 0.164164498234
Feature: from_this_person_to_poi , Support: False , Score: 2.42650812724
Feature: shared_receipt_with_poi , Support: True , Score: 8.74648553213
Feature: scaled_share_with , Support: True , Score: 8.74648553213
Feature: scaled_from_poi , Support: False , Score: 5.34494152315
```

Top 50% Features

Top 50% of features:

- exercised_stock_options , Support: True , Score: 25.0975415287
- total_stock_value , Support: True , Score: 24.4676540475
- bonus , Support: True , Score: 21.0600017075
- salary , Support: True , Score: 18.575703268
- deferred_income , Support: True , Score: 11.5955476597
- long_term_incentive , Support: True , Score: 10.0724545294
- restricted_stock , Support: True , Score: 9.34670079105
- total_payments , Support: True , Score: 8.86672153711
- scaled_share_with , Support: True , Score: 8.74648553213
- scaled_from_poi , Support: False , Score: 5.34494152315

Q3)

After determining the number of features and classifier parameters, the information was passed onto the StratifiedShuffleSplit cross validation function to assess their performance

```
Precision StratifiedShuffleSplit: 0.0
Recall StratifiedShuffleSplit: 0.0
Accuracy StratifiedShuffleSplit: 0.866666666667
Predictions: [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

StratifiedShuffleSplit Results SVC With Top 60% Features

```
Precision StratifiedShuffleSplit: 0.283852380952
Recall StratifiedShuffleSplit: 0.31
Accuracy StratifiedShuffleSplit: 0.8154
Predictions: [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

StratifiedShuffleSplit Results DecisionTree with Top 50% Features

Based on the off and general comparison, SVC does not seem to be generating any predictions (All predictions are 0) while DecisionTree has more favorable results hence, I decided to go with the DecisionTree classifier with its 50% features.

Final Algorithm: Decision Tree Classifier

Final Features:

- exercised_stock_options
- total_stock_value
- bonus
- salary
- deferred_income
- long_term_incentive
- restricted_stock
- total_payments
- scaled_share_with
- scaled_from_poi

Q4)

Tuning of algorithm parameters is to adjust the settings of classifiers which in-turn affect and determine the results of its calculations.

To reduce the time needed to tune the SelectPercentile and classifier parameters, I created a list of the percentiles and parameters and fed them into a grid search pipeline to determine the best combination.

Classifier: DecisionTree, min_samples_split = 1, random_state = 34

SelectPercentile: 50%

Should I change the min_samples_split value to 5, the accuracy increases but both precision and recall drops. As the minimum number of samples increases, less splits are made and analyzed impacting precision and recall.

```
Precision StratifiedShuffleSplit: 0.283852380952
Recall StratifiedShuffleSplit: 0.31
Accuracy StratifiedShuffleSplit: 0.8154
Predictions: [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

min_samples_split = 1

```
Precision StratifiedShuffleSplit: 0.275225
Recall StratifiedShuffleSplit: 0.303
Accuracy StratifiedShuffleSplit: 0.817333333333
Predictions: [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

min_samples_split = 5

Q5)

Validation is used to evaluate the effectiveness of the training done on the model/classifier algorithm using data that has been hold-out or kept for testing purposes.

The testing data serves as unseen or un-trained data for the model. This allows us to have an estimated performance and decide if further tuning is required.

A mistake I could make is to blindly believe in the accuracy score for the SVM algorithm. If I were to use it without checking the prediction, precision and recall results, I would not have realized that the algorithm was not predicting anything and was resulting in 0.0 for precision and recall scores.

Hence using it would not have given me any results.

My classifier results were validated or checked by using the StratifiedShuffleSplit function. By iterating kfold number of times using randomized samples and averaging their precision, recall and accuracy scores.

I had also used tester.py to check on the results which are attached below.

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=1, min_weight_fraction_leaf=0.0,
            presort=False, random_state=34, splitter='best')
        Accuracy: 0.81540       Precision: 0.30861      Recall: 0.31000 F1: 0.30930     F2: 0.30972
        Total predictions: 15000        True positives:  620    False positives: 1389   False negatives: 1380   True negatives:
11611
```

Tester.py Results

Q6)

Using StratifiedShuffleSplit to determine the evaluation metrics; Precision and Recall, their averages are:

- Precision: ~0.284
- Recall: 0.31

Precision implies that out of the items labelled or predicted to be PoIs, how many are indeed Pois. For example in a dataset of 10 samples, where 6 were predicted to be PoIs, Precision would indicate out of the 6, how many are really PoIs.

Recall implies that out of the items that are indeed PoIs, how many were correctly classified. For example in a dataset of 10 samples where 8 are real PoIs, Recall would indicate out of the 8, how many have been predicted correctly as PoIs.