# Udacity: Machine Learning Final Project

**by Rachel Foong**

## Project Goal

The goal of this project is to identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset.

## Dataset Exploration

Machine learning is useful to comb and refine the dataset that consists of **146 data points** with **21 features** with **18 Persons of Interest (POI)**.

This is especially true when there are a large number of missing values represented as "NaNs" in each feature. To analyse the data, I've replaced NaNs with 0.

#### Observations

- The loan advances, deferral_payments, director_fees and restricted_stock_deferred columns have the highest number of missing values
- Out of the set, total_payments and total_stock_value have the lowest amount of missing values

| Feature | No. of NaNs | Mean | Max. | Min. |
|---|---|---|---|---|
| bonus | 64 | 1,333,474.23 | 97,343,619 | 0 |
| deferral_payments | 107 | 438,796.52 | 32,083,396 | -102,500 |
| deferred_income | 97 | -382,762.21 | 0 | -27,992,891 |
| director_fees | 129 | 19,422.49 | 1,398,517 | 0 |
| email_address | 35 | N/A | N/A | N/A |
| exercised_stock_options | 44 | 4,182,736.2 | 311,764,000 | 0 |
| expenses | 51 | 70,748.27 | 5,235,198 | 0 |
| from_messages | 60 | 358.6 | 14,368 | 0 |
| from_poi_to_this_person | 60 | 38.23 | 528 | 0 |
| from_this_person_to_poi | 60 | 24.29 | 609 | 0 |
| loan_advances | 142 | 1,149,657.53 | 83,925,000 | 0 |
| long_term_incentive | 80 | 664,683.95 | 48,521,928 | 0 |
| other | 53 | 585,431.79 | 42,667,589 | 0 |
| poi | 0 | 0.12 | 1 | 0 |
| restricted_stock | 36 | 1,749,257.02 | 130,322,299 | -2,604,490 |
| restricted_stock_deferred | 128 | 20,516.37 | 15,456,290 | -7,576,788 |
| salary | 51 | 365,811.36 | 26,704,229 | 0 |
| shared_receipt_with_poi | 60 | 692.99 | 5,521 | 0 |
| to_messages | 60 | 1,221.59 | 15,149 | 0 |
| total_payments | 21 | 4,350,621.99 | 309,886,585 | 0 |
| total_stock_value | 20 | 5,846,018.08 | 434,509,511 | -44,093 |

Through this simple table, it's easy to hypothesise that the POIs are essentially skewing the Max values. When we isolate the 18 POIs, we see a different story.

POIs generally average higher in all values. It's strange to see that the Max values don't match the Max values earlier observed; which subsequently means that the outliers in the data are not all coming from POIs.

| Feature | No. of NaNs | Mean | Max. | Min. |
|---|---|---|---|---|
| poi | 0 | 1.0 | 1 | 1 |
| total_payments | 0 | 7,913,589.78 | 103,559,793 | 91,093 |
| total_stock_value | 0 | 9,165,670.94 | 49,110,078 | 126,027 |
| salary | 1 | 362,142.39 | 1,111,258 | 0 |
| deferral_payments | 13 | 144,415.06 | 2,144,013 | 0 |
| exercised_stock_options | 6 | 6,975,862.44 | 34,348,384 | 0 |
| bonus | 2 | 1,844,444.39 | 7,000,000 | 0 |
| restricted_stock | 1 | 2,189,808.5 | 14,761,694 | 0 |
| restricted_stock_deferred | 18 | 0.0 | 0 | 0 |
| expenses | 0 | 59,873.83 | 127,017 | 16,514 |
| loan_advances | 17 | 4,529,166.67 | 81,525,000 | 0 |
| other | 0 | 802,997.39 | 10,359,729 | 486 |
| director_fees | 18 | 0.0 | 0 | 0 |
| deferred_income | 7 | -632,691.56 | 0 | -3,504,386 |
| long_term_incentive | 6 | 803,241.61 | 3,600,000 | 0 |

## Outliers

When we dig deeper into the Max. values, we find that the email address which has the Max Value for total_payments is actually the "TOTAL" value for all columns. When we remove the outlier, the Max values now match the POI values and the total means are lower.

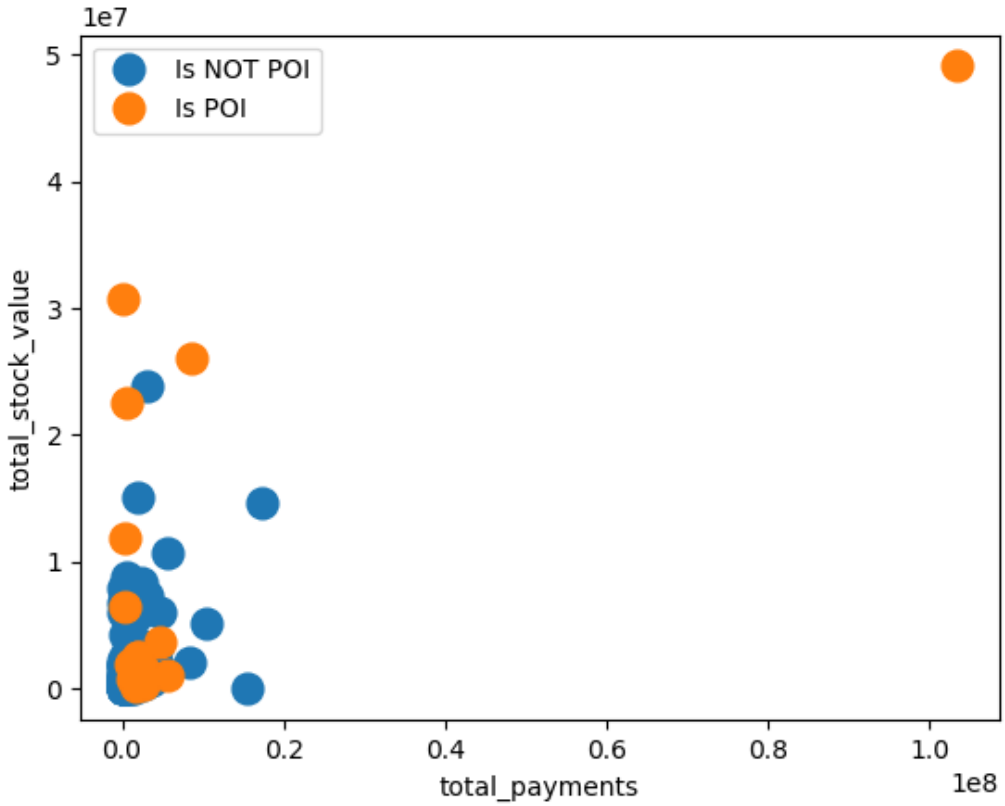
| Feature | No. of NaNs | Mean | Max. | Min. |
|---|---|---|---|---|
| poi | 0 | 0.12 | 1 | 0 |
| total_payments | 21 | 2,243,477.42 | 103,559,793 | 0 |
| total_stock_value | 20 | 2,889,718.12 | 49,110,078 | -44,093 |
| salary | 51 | 184,167.1 | 1,111,258 | 0 |
| deferral_payments | 107 | 220,557.9 | 6,426,990 | -102,500 |
| exercised_stock_options | 44 | 2,061,486.1 | 34,348,384 | 0 |
| bonus | 64 | 671,335.3 | 8,000,000 | 0 |
| restricted_stock | 36 | 862,546.39 | 14,761,694 | -2,604,490 |
| restricted_stock_deferred | 128 | 72,911.57 | 15,456,290 | -1,787,380 |
| expenses | 51 | 35,131.37 | 228,763 | 0 |
| loan_advances | 142 | 578,793.1 | 81,525,000 | 0 |
| other | 53 | 295,210.02 | 10,359,729 | 0 |
| director_fees | 129 | 9,911.49 | 137,864 | 0 |
| deferred_income | 97 | -192,347.52 | 0 | -3,504,386 |
| long_term_incentive | 80 | 334,633.99 | 5,145,434 | 0 |

While removing "TOTAL" helped in reconciling the differences between the POI and both POI and non-POI values, when we plot the two features for totals together, we find that there are still a couple of Outliers, especially one POI; namely **"LAY KENNETH L"**.

In [1]:

```
from IPython.display import Image
Image("Outlier1.png")
```

Out[1]:



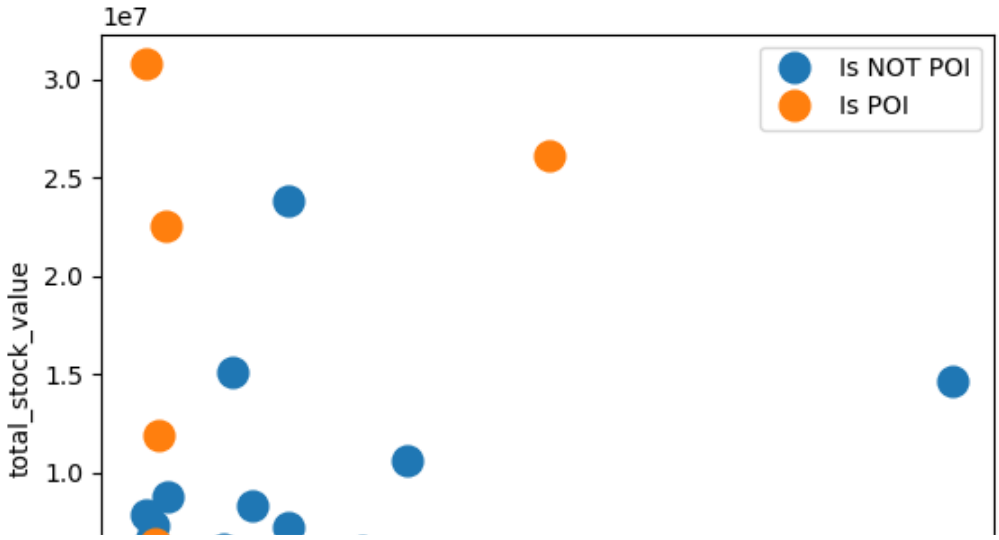

By removing "LAY KENNETH L", we get a really different look at the scatter plot and Mean/Max values.

While there seems to be a positive correlation between the two features, we can clearly see it is weak even without calculating the correlation coeff.

In [2]:

```
Image("Outlier2.png")
```

Out[2]:




| Feature | No. of NaNs | Mean | Max. | Min. |
|---|---|---|---|---|
| poi | 0 | 0.12 | 1 | 0 |
| total_payments | 21 | 1,539,891.9 | 17,252,530 | 0 |
| total_stock_value | 20 | 2,568,743.4 | 30,766,064 | -44,093 |
| salary | 51 | 177,999.36 | 1,111,258 | 0 |
| deferral_payments | 107 | 220,680.45 | 6,426,990 | -102,500 |
| exercised_stock_options | 44 | 1,837,271.53 | 30,766,064 | 0 |
| bonus | 64 | 627,386.24 | 8,000,000 | 0 |
| restricted_stock | 36 | 766,024.53 | 13,847,074 | -2,604,490 |
| restricted_stock_deferred | 127 | 73,417.9 | 15,456,290 | -1,787,380 |
| expenses | 51 | 34,682.06 | 228,763 | 0 |
| loan_advances | 142 | 16,666.67 | 2,000,000 | 0 |
| other | 53 | 225,317.53 | 7,427,621 | 0 |
| director_fees | 128 | 9,980.32 | 137,864 | 0 |
| deferred_income | 97 | -191,599.94 | 0 | -3,504,386 |
| long_term_incentive | 80 | 311,957.83 | 5,145,434 | 0 |

POI numbers without Kenneth Lay are also lower and Max. values also match the non-POI numbers.

Now that we have cleaned up our data, we can finally move on to feature engineering.

| Feature | No. of NaNs | Mean | Max. | Min. |
|---|---|---|---|---|
| poi | 0 | 1.0 | 1 | 1 |
| total_payments | 0 | 2,287,342.53 | 8,682,716 | 91,093 |
| total_stock_value | 0 | 6,815,999.94 | 30,766,064 | 126,027 |
| salary | 1 | 320,367.18 | 1,111,258 | 0 |
| deferral_payments | 13 | 140,974.12 | 2,144,013 | 0 |
| exercised_stock_options | 6 | 5,365,714.12 | 30,766,064 | 0 |
| bonus | 2 | 1,541,176.41 | 5,600,000 | 0 |
| restricted_stock | 1 | 1,450,285.82 | 6,843,672 | 0 |

```
restricted_stock_deferred             17  0.0           0           0
expenses                               0  57,523.35     127,017     16,514
loan_advances                         17  0.0           0           0
other                                  0  240,836.71    1,573,324   486
director_fees                         17  0.0           0           0
deferred_income                        7  -652,261.65   0           -3,504,386
long_term_incentive                    6  638,726.41    1,920,000   0
========================  =============  ============  ==========  ==========
```

## Feature Engineering

#### New Feature: % of Stock Value over Payments

At this point, while total_payments and total_stock_value have a low level of missing values and therefore one would assume a better fit for the data, it's all because these features happen to be totals of all the other features.

However this pair makes for a good feature. When comparing averages of total stock value over payments, POI seem to have a higher total stock value per total payment ratio. We can use feature selection tools to evaluate this hypothesis. I've named the new feature "stock_value_ratio". I will test this once we've determined the best feature algorithm and parameters for the original dataset.

#### Feature Selection

In order to better evaluate prediction of the POI by their monetary activity and attributes, I have removed all features that look at the email activity from and between poi and sender (email_address, from_messages, from_poi_to_this_person, from_this_person_to_poi, shared_receipt_with_poi). This is to also ensure the variance in the data makes sense before applying feature reduction techniques.

After removing these features, the final feature list (without the new, untested feature) is:

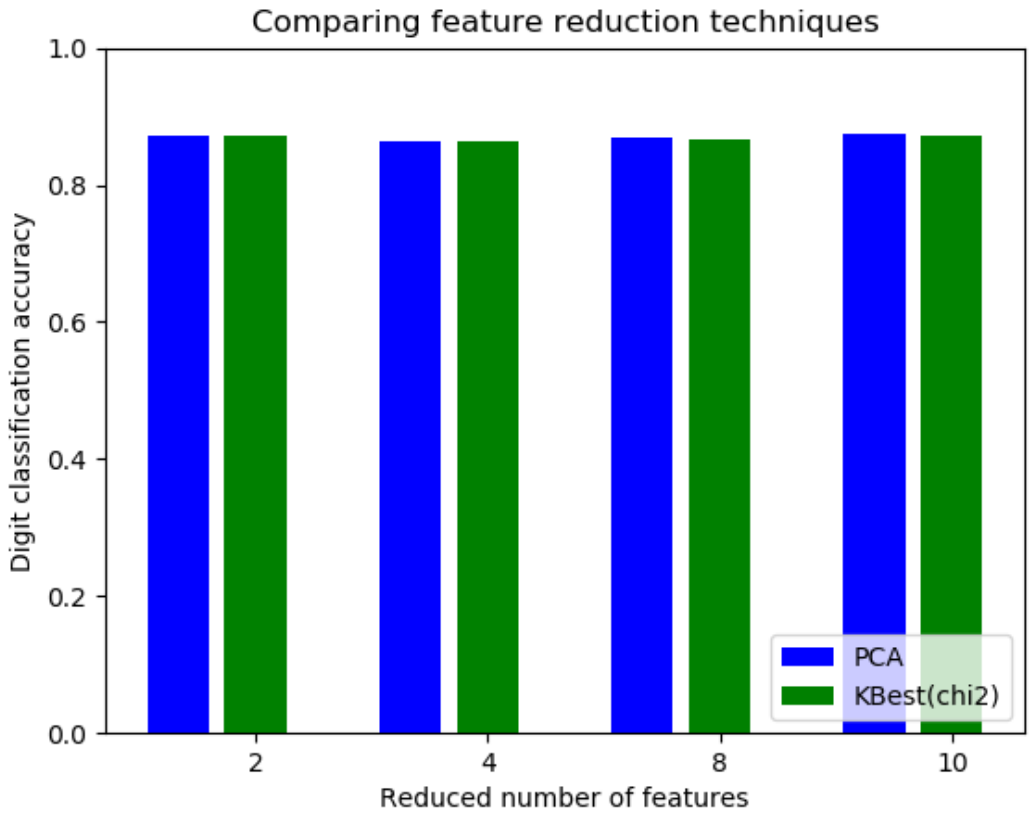
- 'total_payments'
- 'total_stock_value'
- 'salary'
- 'deferral_payments'
- 'exercised_stock_options'
- 'bonus'
- 'restricted_stock'
- 'restricted_stock_deferred'
- 'expenses'
- 'loan_advances'
- 'other'
- 'director_fees'
- 'deferred_income'
- 'long_term_incentive'

By using LinearSVC as an initial classifier for comparing the two feature reduction techniques, PCA seems to works best when the number of features are 10 when we get the best parameters. However, both KBest and PCA are neck to neck in terms of scores. We shall use PCA then to for feature reduction.

In [3]:

```
Image("Feature_Selection.png")
```

Out[3]:




**Feature Scaling**

To see if scaling will get a better score, I have tested scaled features to get the PCA scores and also with the classifier scores.

After scaling though, the PCA amount of variance between the variables have dropped significantly which isn't surprising given that the features are now standardized.

## Algorithm comparison and Validation

With PCA as a means of feature selection, I chose GaussianNB algorithm and the Decision Tree algorithm as a means of validation. I validate by seeing what the **accuracy scores** are between the two models.

Comparing and validating models is important because it will then help me achieve a higher accuracy score in terms of identifying POIs.

GaussianNB seems to work best with unscaled features but Decision Tree works best with scaled features. I have decided to continue with Decision Tree given the low variance in accuracy scores between Scaled and Unscaled features.

```
BEFORE SCALING
----------------
Explained Variance:
[  7.93417839e-01   1.12588783e-01   5.45143796e-02   1.87372640e-02
```

```
   1.09751938e-02   5.34335020e-03   2.60893422e-03   1.15848667e-03
   4.77398369e-04   1.03453271e-04]
GNB Score with Unscaled Features:  0.860465116279
DecisionTree Score with Unscaled features:  0.813953488372


AFTER SCALING
----------------
Explained Variance (with scaled features):
[ 0.36551414  0.17295331  0.12682769  0.1060265   0.06395091  0.0498712
  0.03826859  0.02896928  0.02296713  0.01548911]
GNB Score with Scaled Features:  0.162790697674
DecisionTree Score with Scaled features:  0.860465116279
```

## Parameter Tuning and Validation

In order to achieve the possibility of a higher score, it's best to tune the parameters of the algorithm. If I don't do this well, I'll miss out on that opportunity and could possibly overfit the data.

Under the DecisionTreeClassifier documentation (http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) there are several parameters that can be tuned. I've chosen to tune the criterion, max_depth and the splitter out of curiosity rather than focusing on the leaf nodes.

To sync with the tester.py random state, I have left the random state at 42.

With Grid Search CV, I found that the "entropy" criterion, mixed with the max_depth of 3 and a random splitter produced the best results.

However, with the current feature list, the score drops to 0.14.

## Evaluation

#### Evaluating the new feature

After gaining the low score, I returned to test the new feature (stock_value_ratio) to see if it helps in gaining a higher score. I found that the score return a favourable score of 0.86.

I also find that the average precision and recall score is higher with the new feature. With the new feature, the recall score is higher than the precision score.

The higher recall score means that whenever a POI shows up in the test set, I am able to identify him/her. The cost of this is that sometimes I get some false positives where non-POIs get flagged.

A higher precision score would mean that when a POI gets flagged in my test set, I know with confidence that it's a real POI and not a false alarm. However I would probably miss the real POIs since I'm effectively reluctant to pull the trigger on edge cases.

Looking at the classification reports below, while adding a feature does have a higher precision and recall score than not, the chances of identifying a POI also becomes low due to the smaller amount of POIs in the dataset.

```
BEFORE NEW FEATURE
-------------------
             precision    recall  f1-score   support
```

```
        0.0       0.50      0.03      0.05        37
        1.0       0.12      0.83      0.21         6

avg / total       0.45      0.14      0.07        43


AFTER NEW FEATURE
-------------------
             precision    recall  f1-score   support

        0.0       0.86      1.00      0.92        37
        1.0       0.00      0.00      0.00         6

avg / total       0.74      0.86      0.80        43
```