# Project 5: Identify Fraud from Enron Email

Name: Peter Eisenschmidt

Email: peter.eisenschmidt@airbus.com

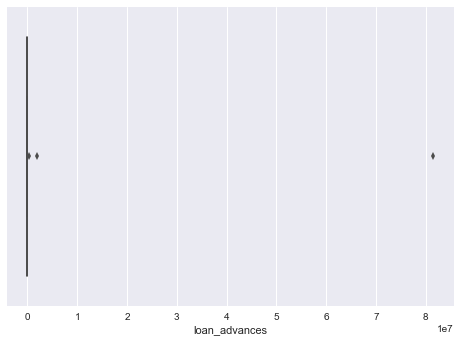
Date: February 17, 2017

# Data Exploration

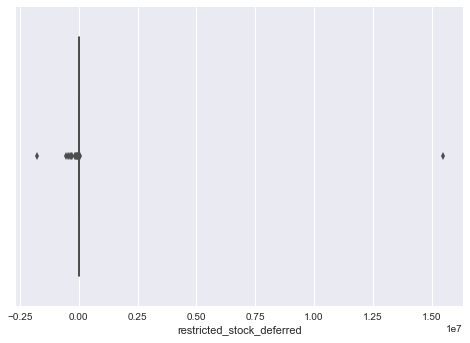
The first step is to detect any outliers and to see if they need to be removed from the dataset. The TOTAL entry is removed, as it details the sum over all persons included in the dataset. For the remaining entries, boxplots are created, as this allows detecting outliers easily.

This is done in the Python script “outliers\_investigation.py”.

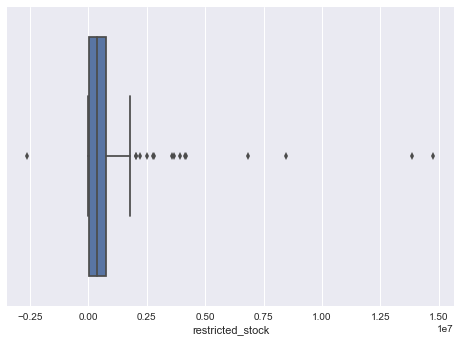
Three parameters show outliers that require further investigation:



loan\_advances: Only few data points are available for this parameter. One extreme point (81,525,000) sticks out. As this data point belongs to Ken Lay, it should be considered as valid.



restricted\_stock\_deferred: Only few data points available, most of them negative. The positive value requires further checking.



restricted\_stock: one negative value whereas the remaining data points are positive.

The parameters total\_payments and total\_stockvalue are the sum of all payments (salary, bonus, etc) and stock values (exercised stock option, restricted stock option, and restricted stock option deferred). So, either all other values are excluded and only the sums are included or vice versa.

# Algorithm Selection and Tuning

Before selecting an algorithm for the final analysis, 5 different algorithms are checked initially and compared:

1. Gaussian Naives Bayes Classifier
2. KNearestNeighbors Classifier
3. Decisision Tree Classifier
4. Random Forest Classifier
5. Support Vector Classifier

The first step is to run each classifier with its default parameters and all features, except total\_payments, total\_stockvalue, and email\_address.

# Validation Strategy

In the previous section it was seen that the results in terms of precision and recall differ when simply using train\_test\_split and when using StratifiedShuffleSplit. In almost all cases, e.g. for the KNearestNeighbor Classifier both recall and precision were sufficiently high (precision = .429 and recall = .5, resulting in f1 = .5). With the test\_classifier function, the value drop to precision = .227, recall = .202, and f1 = .214.

Only the Naives Bayes Classifier improved:

1. Precision: .286, recall: .4, f1: .3333 (train\_test\_split)
2. Precision: .340, recall: .381, f1: .359 (StratifiedShuffleSplit)

This shows the importance of the validation, as the performance may depend a lot on the selected test set.

The metrics used to evaluate the performance are precision and recall. What this means for this project is as follows:

* Low Recall: Number of False Negatives is too high, i.e. a lot of employees are predicted not to be a POI whereas in reality they were
* Low Precision: Number of False Positive is too high, i.e. a lot of employed are predicted to be a POI whereas in reality they were not.
* In both cases the number of True Positives may also be too low, i.e. the number of correctly identified POIs

It is arguably which one is more important but I would tend to achieve a better recall than precision, as it seems better to falsely identify someone as POI (and to exonerate them later) than to miss a potential POI. On the other hand, … So f1