**Outlier removal:**

The outlier removal was a iterative process. Since we are not dealing with normally distributed data, is pretty impossible to define what is or is not an outlier. We need an assumed distribution in order to be able to classify something as “lying outside the range of expected values”. Because of that, techniques like finding the top 10% bigger regression error or “Interquartile range” cannot be blindly applied. So I used a mix of the latest (“Interquartile range”) to suggest possible outliers and them analysed the data visually to verify if they should or should not be considered outliers.

To make that suggestion, I wrote a function that, for each column of a data frame, checks the values that are considered outliers based on the IQR rule.

An [IQR](http://www.statisticshowto.com/what-is-an-iqr/), or interquartile range, is the difference between the first and third quartiles of data. It’s one measure of spread — how far data is spread around the mean.

According to this rule, observations that fall below Q1 – “g” x (IQR), or above Q3 + “g” x (IQR) are identified as potential outliers. The value of "g" is normally 1.5. I chose to use 2.2 due to the explanation found on [this](https://www.youtube.com/watch?v=2HmopqF6V6w) YouTube video.

Very soon, I realized that the IQR rule on full dataset the wasn’t such a good idea due to the high number of zeros on the columns (which were originally NaNs) so I decided to exclude them from the analysis (in another words, the IQR rule was applied disconsidering the zeros from the calculation – I understand that is not the best technique but again, I was looking for suggestions, not a deterministic way of removing the outliers).

All the analysis is done with the “get\_outliers\_list” function documented on the poy\_id.py file that outputs a dictionary where the key is the column and the value is a list of suggestions. Alternatively the “get\_unique\_outliers” function can be used to output a list on Unique suggestions across all features. The output is the following:

Column: salary - Number of suggested outliers (ignoring 0s): 5 Indices: [47, 79, 104, 121, 129]

Column: to\_messages - Number of suggested outliers (ignoring 0s): 6 Indices: [6, 7, 73, 75, 78, 116]

Column: deferral\_payments - Number of suggested outliers (ignoring 0s): 2 Indices: [47, 129]

Column: total\_payments - Number of suggested outliers (ignoring 0s): 7 Indices: [11, 47, 78, 79, 85, 121, 129]

Column: long\_term\_incentive - Number of suggested outliers (ignoring 0s): 3 Indices: [79, 85, 129]

Column: loan\_advances - Number of suggested outliers (ignoring 0s): 0 Indices: []

Column: bonus - Number of suggested outliers (ignoring 0s): 9 Indices: [0, 7, 31, 75, 78, 79, 121, 129, 138]

Column: restricted\_stock\_deferred - Number of suggested outliers (ignoring 0s): 0 Indices: []

Column: total\_stock\_value - Number of suggested outliers (ignoring 0s): 11 Indices: [3, 32, 47, 65, 79, 102, 111, 121, 129, 139, 143]

Column: expenses - Number of suggested outliers (ignoring 0s): 3 Indices: [87, 129, 131]

Column: exercised\_stock\_options - Number of suggested outliers (ignoring 0s): 11 Indices: [32, 35, 47, 65, 79, 102, 109, 111, 121, 129, 143]

Column: from\_messages - Number of suggested outliers (ignoring 0s): 17 Indices: [0, 6, 7, 19, 31, 32, 58, 62, 66, 72, 73, 75, 78, 88, 115, 116, 138]

Column: other - Number of suggested outliers (ignoring 0s): 9 Indices: [3, 47, 69, 79, 85, 102, 118, 120, 129]

Column: deferred\_income - Number of suggested outliers (ignoring 0s): 0 Indices: []

Column: restricted\_stock - Number of suggested outliers (ignoring 0s): 12 Indices: [3, 47, 69, 73, 79, 102, 111, 121, 129, 138, 139, 143]

Column: director\_fees - Number of suggested outliers (ignoring 0s): 5 Indices: [8, 93, 107, 129, 131]

Unique Suggestions: [47, 79, 104, 121, 129, 6, 7, 73, 75, 78, 116, 66, 11, 85, 0, 31, 138, 3, 32, 65, 102, 111, 139, 143, 87, 131, 35, 109, 19, 58, 62, 72, 88, 115, 69, 118, 120, 8, 93, 107]

Unfortunately the outputs 40 unique indexes, meaning that, 40 out of 135 would be considered outliers based on the IQR rule. That’s not good so I went to the second part of the analysis:

First, by just looking at the names on the dictionary:

for employee in sorted(data\_dict):

print employee

I could exclude 2 key form it: “TOTAL” (which contains the total of the reports – not a valid data point) and “THE TRAVEL AGENCY IN THE PARK” (which doesn’t seem to be a person – and contain mainly zeroes)

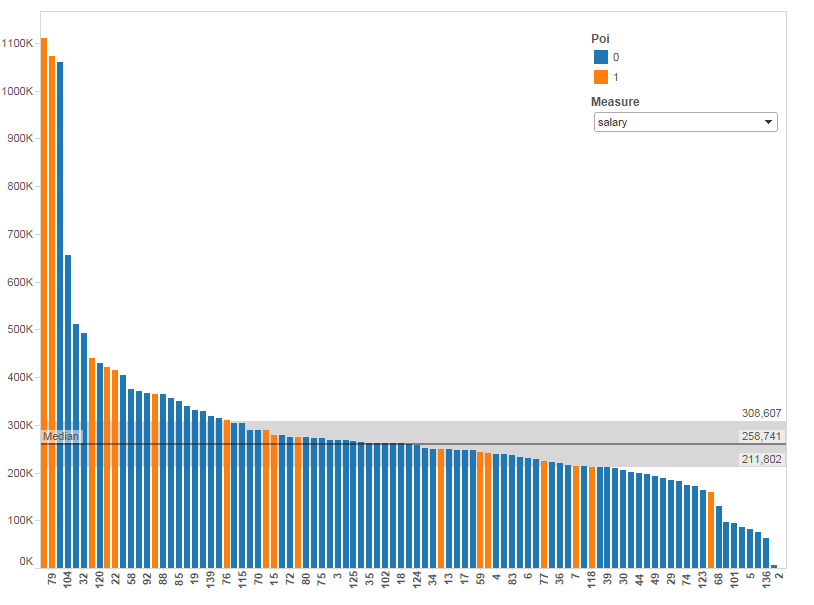
Second, looking at the dictionary, I found ‘LOCKHART EUGENE E'’, which doesn’t have any information (all values are NaN) but the “featureFormat” function (that will be applied in the future) seem to be removing it, so no action required.

Third, by plotting the data, I could very easily see 2 things:

* whether the indicated values were outliers or not and;
* some sort of indication on where I should or not use the feature in question on my analysis;

Important note: Since this is not a visualization or data analysis project, I used the public version of an external tool called Tableau to help on the visual analysis (rather than using Python). Here are my conclusions:

* On the fields Salary, to\_messages, deferral\_payments, long\_term\_incentive, bonus, expenses, other, deferred\_income, restricted\_stock, the distribution is skewed but the values identified as outliers aren’t necessary outliers. For example, if we look at a bar plot of salaries, the top 5 salaries aren’t necessarily outliers, they simply reflect people with much higher salaries.



* total\_payments: Index number 79 has a value of 103,559,793 (the biggest of the dataset, much bigger than the second biggest - 17,252,530) but since it is a POI (LAY KENNETH L), I decided to keep it;
* loan\_advances: only has 3 values different than zero so I decided not to do anything. This is also a good candidate for removal from the prediction;
* restricted\_stock\_deferred: no outliers suggested;
* total\_stock\_value: I believe this is a very good measure for the prediction algorithm because from the top 5 values, 4 are POIS, so it will probably have quite a significant weight;
* exercised\_stock\_options:also a good measure for the prediction. The top 4 are Pois;
* from\_messages: very skewed, probably a bad candidate for prediction. From the top 16, only one poi
* director\_fees: only 16 values greater different than zero, all non-poi

**New features:**

Given the nature of the dataset, I didn’t think I could benefit from any new features, nor that there were features to be derived from the feature I chose to keep so far. Nevertheless, I decided to give it a try by creating a “ratio” between Bonus and Salary.

**Model evaluation:**

Evaluation method:

The model evaluation is being done done using the “sklearn.cross\_validation.StratifiedShuffleSplit” function with a 1000 iterations.

For each of these iterations of the splitting, The function provides train/test indices to split data into training and test sets.

The model is then built on the training set and tested on the test set and the amount of true\_positives , true\_negatives, false\_positives and false\_negatives are summed over all 1000 iterations, which allows the computation of 5 evaluation metrics: Accuracy, precision, recall, f1 and f2.

The definition of all these metrics can be found [here](https://en.wikipedia.org/wiki/Precision_and_recall#F-measure).

Choosing the evaluation metric:

Since this is quite an unbalanced data set, some care was exercised while selecting the best evaluation metric

Initially I started evaluating my models using "accuracy" but I realized that it isn't a good measure due to the low number of POIs we have .

Since Accuracy is defined by number of TP + TN divided by the total number of predictions, if we guess "non-poi" for everyone, we'll get an accuracy of 87% (which seems pretty good but clearly represents a very poor model).

Accuracy by itself, especially for an unbalanced dataset, does not provide adequate information about the classifier's performance and can also be deceiving (refer to the [Accuracy Paradox](https://en.wikipedia.org/wiki/Accuracy_paradox)).

Since we are building a model that, given a person, decides whether we should investigate him\her for fraud or not, we'd want to penalize "false negatives" more than "false positives".

In other words, its worst if our model tells us to investigate a person who in the end is not POI (false positive), than not investigate a person that's actually a POI (false negative).

Based on that, I thought I’d use the “recall” measure that represent the ration between TP and (TP + FN). So in this case, since the number of false negatives is on the denominator, a high number of false negatives would bring the recall metric down.

But very quickly I found that there’s also ways in which this metrics can be deficient. If we just say that everyone is a POI, then our recall will be 1 since we've successfully marked all of the POIs. Of course, our false positive rate will also be extremely high, and our precision (and accuracy, as you've observed) will suffer.

So, in the end, I decided to focus a ta metric that takes the FP and FN balance into account, such as the F-measures. The F1 score is an equal weighting between precision and recall performance, while other F-measures can emphasize precision or recall performance as desired by the task at hand.

**Feature Selection:**

From all the features available, these initial steps were performed:

* ignored the 'email\_address' feature: unique individual email, no predictive power whatsoever;
* ignored the following features: from\_poi\_to\_this\_person, from\_this\_person\_to\_poi, shared\_receipt\_with\_poi: these features represent the number of emails sent from and to a “poi” and the number of receipts shared with a poi. There is an extensive [discussion](https://discussions.udacity.com/t/mistake-in-the-way-email-poi-features-are-engineered-in-the-course/4841) on the Udacity forum regarding the validity of these feature, mainly related to the fact that we are trying to predict “poi” so we should have available features that were calculated based on already labels “pois”. There are arguments pro and against these feature that were extensively discussed on the forums so I won’t get into much detail, I tend to agree that they may constitute of [Data Leakage on the model so I chose not to use them.](https://www.kaggle.com/wiki/Leakage)
* Ignored the loan\_advances feature: because it contains only 3 non zero values;

The next step was to run the “sklearn.feature\_selection.SelectKBest” function to output the best features from our list and, from that list, select the top 5 (The reason I chose 5 was after analysing the algorithms on further steps. Basically, any feature on this list with a score less than 18 won’t have enough predictive power – that includes the ratio I created on the previous step). That selection is done on the “SelectKBestFeatures”.

('exercised\_stock\_options', 24.815079733218194),

('total\_stock\_value', 24.182898678566879),

('bonus', 20.792252047181535),

('salary', 18.289684043404513),

('deferred\_income', 11.458476579280369),

('bonus\_salary\_ratio', 10.785573495441602),

('long\_term\_incentive', 9.9221860131898225),

('restricted\_stock', 9.2128106219771002),

('total\_payments', 8.7727777300916756),

('loan\_advances', 7.1840556582887247),

('expenses', 6.0941733106389453),

('other', 4.1874775069953749),

('director\_fees', 2.1263278020077054),

('to\_messages', 1.6463411294420076),

('deferral\_payments', 0.22461127473600989),

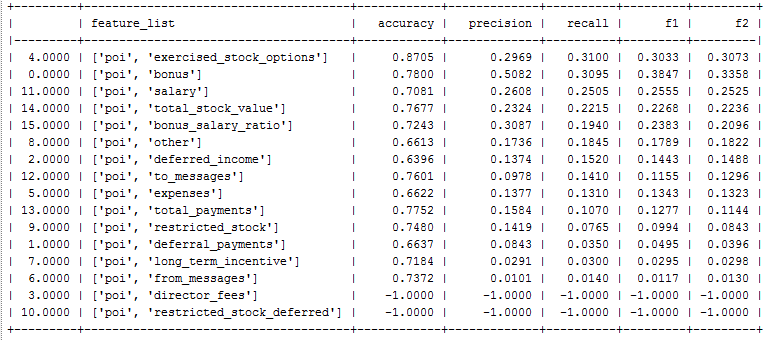
('from\_messages', 0.16970094762175533),

('restricted\_stock\_deferred', 0.065499652909942141)

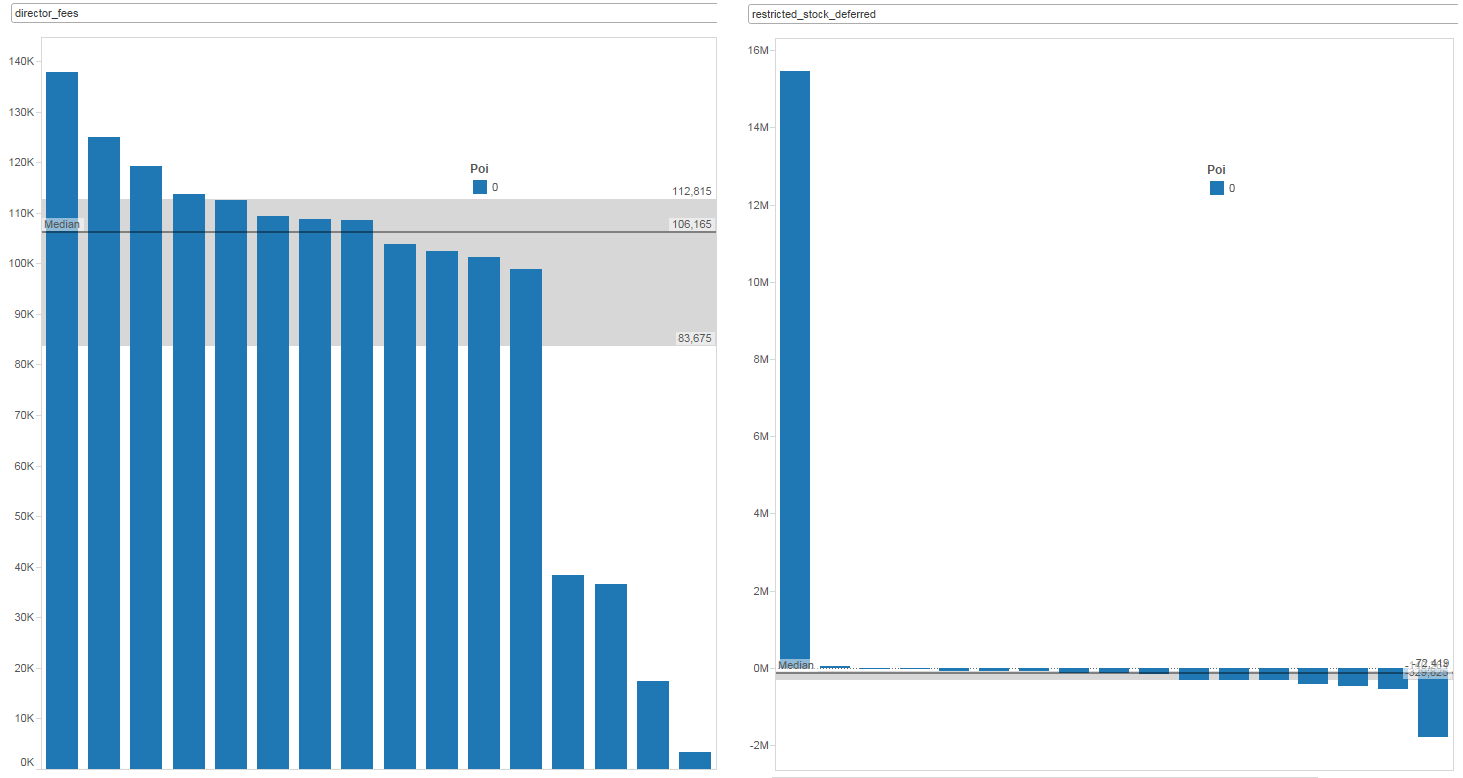
So the features selected at this point were: 'exercised\_stock\_options', 'total\_stock\_value', 'bonus', 'salary', 'deferred\_income'

Model build:

To have a baseline of measures, my first step was to try to predict “poi” by running a simple decision tree using each one of the feature individually. That was done on the “one\_feature\_predict” function.



The -1 represent a failure on the measure calculation, and this is due to the fact that for director\_fees and restricted\_stock\_deferred we only have non-poi values

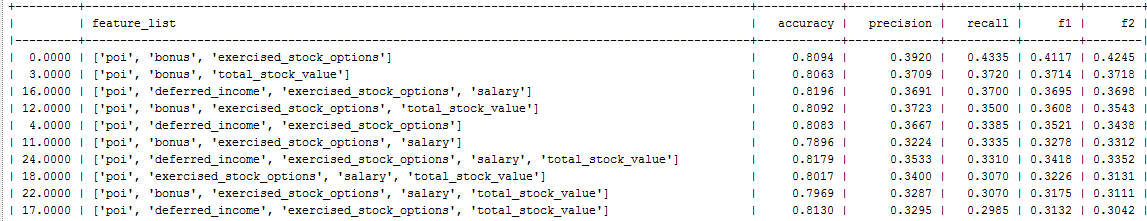


The next steps were very iterative. I ran a series of tests and algorithms trying to maximize the F1 score without scarifying the other measures too much. It was at this step I realized that 5 was the ideal number of features.

Since our dataset is quite small, I decided to take a brute force approach while fitting the models. I wrote a function called topk\_feature\_predict that given a list of features (output from SelectKBestFeatures) produces all possible combinations from those features from 2 to n (n being the number of elements on the SelectKBestFeatures array).

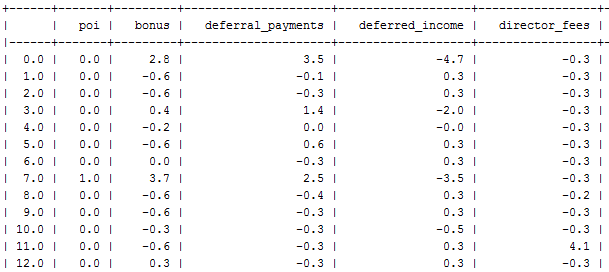
By using the function I was able to fit several different models (most of them are still commented on the code) until I got to a “reasonable” one. Here is an example of running he function with a simple tree model:

clf = tree.DecisionTreeClassifier(min\_samples\_split = 4)



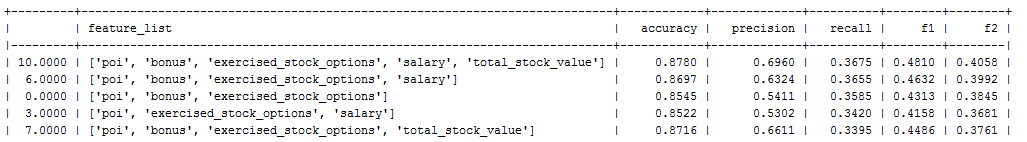
As we can see the result is a little better, but not by much.

The function also allows for data normalization by setting the “normalize\_data” parameter to true, which I testes on every scenario, but didn’t help.



A few other examples – K Nearest neighbours:

clf = KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski', metric\_params=None, n\_neighbors=5, p=2, weights='distance')

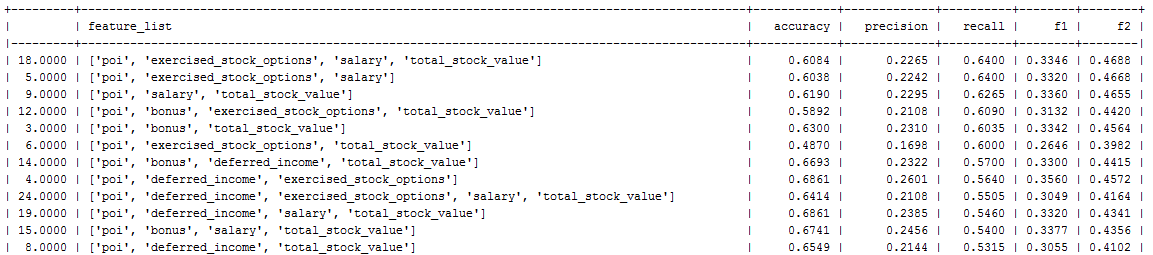


Logistic regression:

Great improvement on recall

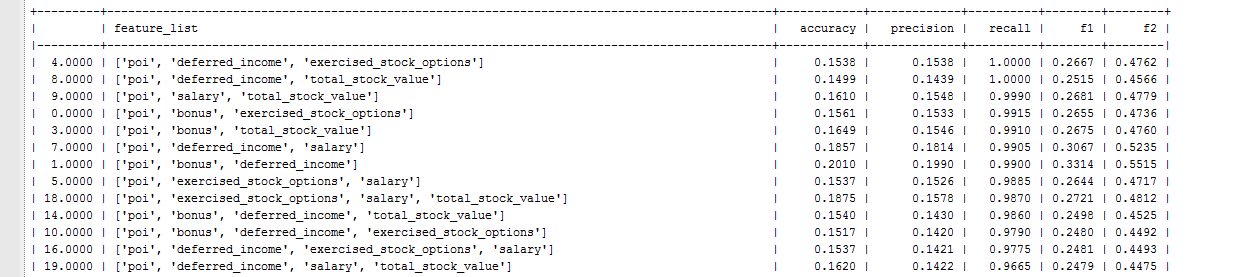
Worst accuracy and precision

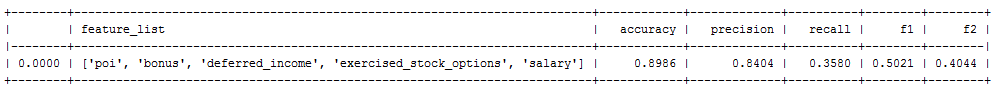
normalizing made it quite worst



5 logistic regression, low C

clf = LogisticRegression( C=0.1,penalty='l1',random\_state=42,tol=10\*\*-10,class\_weight='auto')





* Accuracy paradox: <https://en.wikipedia.org/wiki/Accuracy_paradox>
* IQR Rule: <http://www.statisticshowto.com/what-is-an-iqr/>
* “g” value on IQR: <https://www.youtube.com/watch?v=2HmopqF6V6w>
* Sklearn Documentation: <http://scikit-learn.org/stable/user_guide.html#user-guide>