Udacity - Introduction to Machine Learning

Final Project (P4) for Data Analyst Nanodegree by Susan Streisand

[ss7214@att.com](mailto:ss7214@att.com) and also cc [s.streisand@ieee.org](mailto:s.streisand@ieee.org)

# Introduction

I did most of my work using iPython notebook (copying code from poi\_id.py and importing tester.py). I tried to copy it back to poi\_id.py, but am not sure if you want all the exploratory code that is not part of poi\_id.py???

# Questions

1. ***Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it.  As part of your answer, give some background on the dataset and how it can be used to answer the project question.  Were there any outliers in the data when you got it, and how did you handle those?  [relevant rubric items: “data exploration”, “outlier investigation”]***

The goal of this project is to identify “persons of interest” (POI’s) from data provided for 145 people. We are given the identities of the 18 POI’s so we can train the machine learning algorithm that should identify POI’s from the values of a set of 21 given features plus any more we create or derive. These features were financial (how much the person earned, got as stock, etc. from Enron) and communication statistics (how many emails to/from known POI’s, etc.). Machine learning techniques could determine, for example, whether POI’s typically communicated with other POI’s and thus someone with a higher count of emails to/from known POI’s would be more likely to be a POI.

There was one outlier in the dataset that was identified in Lesson 7 – Outliers. This was the line for the user “TOTAL” from the pdf. The other outliers with large salaries/bonuses were actual values and should stay there. Lesson 7 only looked at salary and bonus data. I looked at all of the fields for anomalies by turning the data dictionary into a pandas dataframe. I only investigated when data values didn’t seem right and did not verify that all values matched what was in the PDF file for financial info. I found there was a deferral\_payment < 0 and searching for it I found it belonged to BELFER ROBERT and that should have been his deferred\_income. It seems all of his data was skewed one to the left. The reason for this was not obvious, but somehow an extra NaN was inserted as a value. Also, the NAME BHATNAGAR SANJAY had an incorrect value for restricted\_stock\_deferred of 15456290, which should have been total\_stock\_value. There must have been an inconsistency in the data input. I see in the pdf file there is no ‘-‘ in the ‘Other’ category, so the data must have been read in inconsistently for him .

Fixed the two bad data lines (all on financial data features) with this code:

## Fix the two bad data lines

ordered\_feature\_list = ['salary', 'bonus', 'long\_term\_incentive', 'deferred\_income', 'deferral\_payments', 'loan\_advances', 'other', 'expenses', 'director\_fees', 'total\_payments', 'exercised\_stock\_options', 'restricted\_stock', 'restricted\_stock\_deferred', 'total\_stock\_value']

for i in range(3,13):

data\_dict["BELFER ROBERT"][ordered\_feature\_list[i]] = data\_dict["BELFER ROBERT"][ordered\_feature\_list[i+1]]

for i in range(len(ordered\_feature\_list)-1, 4, -1):

data\_dict["BHATNAGAR SANJAY"][ordered\_feature\_list[i]] = data\_dict["BHATNAGAR SANJAY"][ordered\_feature\_list[i-1]]

1. ***What features did you end up using in your POI identifier, and what selection process did you use to pick them?  Did you have to do any scaling?  Why or why not?  As part of the assignment, you should attempt to engineer your own feature that doesn’t come ready-made in the dataset--explain what feature you tried to make, and the rationale behind it.  (You do not necessarily have to use it in the final analysis, only engineer and test it.)  If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.  [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]***

I used these features in my POI identifier: 'expenses', 'exercised\_stock\_options', 'other', 'from\_this\_person\_to\_poi' –

I picked these using the DecisionTreeClassifier feature\_importances attribute. I fed into it all the features and these were the ones with positive importance . I did not use any feature scaling because decision tree doesn’t need scaling. These were the feature importances I got when I had initially fed all of the features into generic decision tree:

8 expenses 0.297740835333

9 exercised\_stock\_options 0.196888833766

10 other 0.248585701838

12 restricted\_stock 0.177620053468

19 from\_this\_person\_to\_poi 0.0791645755956

I created two new features, neither of which were selected by this algorithm:

* **female** - created manually based on first name. It was 1 for names that were obviously female and 0 otherwise (male or ambiguous). My assumption is that there are few female POI’s, so if a person is female it is less likely that they are a POI. (I only identified two female POI’s)
* **has\_email\_address** – 1 if the person had an email address and 0 if it was ‘NaN’. This was to use instead of the actual email address because it was not numeric or really categorical and didn’t really tell us anything (all email addresses were @enron.com)

When I added these features, the scoring values changed slightly, but I don’t know if that was just due to the random seed chosen and how it interacted with the features. Importances of the other variables changed, but the importance of the two variables I added was zero (both separately and together).

**When I was tuning my algorithm, I looked again at clf.feature\_importances\_ and saw they had changed and I removed restricted\_stock because it was now 0. The final values were:**

feature, clf.feature\_importances\_

* 'expenses', 0.27481622
* 'exercised\_stock\_options', 0.42680663
* 'other', 0.12796366
* 'from\_this\_person\_to\_poi', 0.17041348

1. ***What algorithm did you end up using?  What other one(s) did you try? [relevant rubric item: “pick an algorithm”]***

I ended up using DecisionTreeClassifier. I ran through a bunch of algorithms with mostly default settings and this was the one that did the best for me. These are the algorithms I tried:

for clf, name in ((GaussianNB(), "GaussianNB"),

(RandomForestClassifier(n\_estimators=100), "Random Forest"),

(linear\_model.Perceptron(), "Perceptron"), (tree.DecisionTreeClassifier(), "Decision Tree" )):

test\_classifier(clf, my\_dataset, features\_list)

1. ***What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well?  How did you tune the parameters of your particular algorithm?  (Some algorithms don’t have parameters that you need to tune--if this is the case for the one you picked, identify and briefly explain how you would have done it if you used, say, a decision tree classifier). [relevant rubric item: “tune the algorithm”]***

The various sklearn algorithms come with different parameters that help determine how the algorithm works – like how many iterations, different model parameter values, whether it normalizes features, etc. When you tune the parameters of the algorithm, you change the way the algorithm works by changing parameters to get an optimum output for the dataset input. If you don’t do it well, the algorithm might not converge or might take too long or not produce the optimum result.

I used a decision tree classifier, which has parameters. I looked at the parameters of DecisionTree and created a script in iPython notebook to go through some values of the parameters to try to find the best settings. I was able to increase my accuracy from Accuracy: 0.86357 to Accuracy: 0.89871. I went through a loop with some parameters, then picked the best combination out of those and then manually varied other parameters until I hit the sweet spot. (splitter=’best’, min\_samples\_split = 8, min\_samples\_leaf = 3, max\_features = ‘auto’ , criterion = ‘gini)

1. ***What is validation, and what’s a classic mistake you can make if you do it wrong?  How did you validate your analysis?  [relevant rubric item: “validation strategy”]***

Validation is when you test if the results you obtained are acceptable. The results need to make sense and the test should be applicable to other data sets besides the training set. If you do it wrong you can overfit where it does very good classification on your training data and it might not work on a different test set because it was too specific to the training set.

I ran the tester.py from the final\_project which does a stratifiedShuffleSplit cross-validation which should relieve overfitting. It fits on training set and tests on test set and shuffles up the breakdown of test/training data many times so you are not overfitting on a specific test/training split (with a random state of 42 so it is repeatable). I changed the random state of the stratifiedShuffleSplit a few times and it still got similar results, although the fine tuning I did may not be optimal for the different random states. Hmm, maybe I could be overfitting on the combined shuffled test/training split, but overall it seems to be OK.

1. ***Give at least 2 evaluation metrics, and your average performance for each of them.  Explain an interpretation of your metrics that says something human-understandable about your algorithm’s performance. [relevant rubric item: “usage of evaluation metrics”]***

This was my performance with random\_state = 100.

**DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=5,**

**max\_features='auto', max\_leaf\_nodes=None, min\_samples\_leaf=3,**

**min\_samples\_split=8, min\_weight\_fraction\_leaf=0.0,**

**random\_state=100, splitter='best')**

**Accuracy: 0.89871 Precision: 0.76648 Recall: 0.41850 F1: 0.54140 F2: 0.46029**

**Total predictions: 14000 True positives: 837 False positives: 255**

**False negatives: 1163 True negatives: 11745**

(There was StratifiedShuffleSplit cross-validation, which split into test and training sets differently among several iterations, which is why there are more predictions than initial samples.

* True positives is the number of people identified as POI who were POI
* True negatives is the number of people identified as non-POI who weren’t POI.
* False positives is the number of people identified as POI who were not POI.
* False negatives is the number of people identified as not POI who were POI.

Accuracy **(Fraction tested correctly as POI out of all tested)** = (True positives + True negatives)/Total Predictions = 0.89871

Precision **(Fraction that were really POI out of all those that tested as POI)**= True positives/(True positives + False positives) = 0.76648

Recall **(Fraction that tested as POI out of all those that really were POI (including those that were not identified by test as POI))** = True positives/(True positives + False negatives) = 0.41850

F1 and F2 (and F*n*) are weighted average of precision and recall (best at 1, worst at 0)

F1 **(harmonic mean of precision and sensitivity)** = 2\*True positives/(2\*True positives + False positives + False negatives) = 0.54140

F2 **(weighs recall higher than precision)** = 5 \* True positive/(5 \* True positive + 4\*False negative + False positive) = 0.46029

Since recall here is lower than precision, F2 is lower than F1.