<http://www.graphpad.com/quickcalcs/pValue1/> - P value calculator

<http://data-blog.udacity.com/posts/2016/10/latex-primer/> - LaTex Primer

<http://blog.yhat.com/posts/r-lm-summary.html> - R linear model summary

Notes:

Note that the type of standard deviation calculated by default is different between numpy's .std() and pandas' .std() functions. By default, numpy calculates a population standard deviation, with "ddof = 0". On the other hand, pandas calculates a sample standard deviation, with "ddof = 1". If we know all of the scores, then we have a population - so to standardize using pandas, we need to set "ddof = 0".

df.add(s, fill\_value=0)

%pylab inline – plot appears within a notebook

# Intro To XLRD

You can also install the xlrd library locally on your computer via python pip and the following command:

**pip** install xlrd

or

**conda** install xlrd

The example code:

**import** xlrd

datafile = "2013\_ERCOT\_Hourly\_Load\_Data.xls"

**def** **parse\_file**(datafile):

workbook = xlrd.open\_workbook(datafile)

sheet = workbook.sheet\_by\_index(0)

data = [[sheet.cell\_value(r, col)

**for** col **in** range(sheet.ncols)]

**for** r **in** range(sheet.nrows)]

**print** "\nList Comprehension"

**print** "data[3][2]:",

**print** data[3][2]

**print** "\nCells in a nested loop:"

**for** row **in** range(sheet.nrows):

**for** col **in** range(sheet.ncols):

**if** row == 50:

**print** sheet.cell\_value(row, col),

*### other useful methods:*

**print** "\nROWS, COLUMNS, and CELLS:"

**print** "Number of rows in the sheet:",

**print** sheet.nrows

**print** "Type of data in cell (row 3, col 2):",

**print** sheet.cell\_type(3, 2)

**print** "Value in cell (row 3, col 2):",

**print** sheet.cell\_value(3, 2)

**print** "Get a slice of values in column 3, from rows 1-3:"

**print** sheet.col\_values(3, start\_rowx=1, end\_rowx=4)

**print** "\nDATES:"

**print** "Type of data in cell (row 1, col 0):",

**print** sheet.cell\_type(1, 0)

exceltime = sheet.cell\_value(1, 0)

**print** "Time in Excel format:",

**print** exceltime

**print** "Convert time to a Python datetime tuple, from the Excel float:",

**print** xlrd.xldate\_as\_tuple(exceltime, 0)

**return** data

data = parse\_file(datafile)

Naive Bayesian Classification (NBC) isreferred to as naive since it makes the assumption that each of its inputs are independent of each other, an assumption which rarely holds true, and hence the word naive. Research has however shown that even

In **overfitting**, a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) describes [random error](https://en.wikipedia.org/wiki/Random_error) or noise instead of the underlying relationship. Overfitting occurs when a model is excessively complex, such as having too many [parameters](https://en.wikipedia.org/wiki/Parameter) relative to the number of observations. A model that has been overfit has poor [predictive](https://en.wikipedia.org/wiki/Predictive_inference) performance, as it overreacts to minor fluctuations in the training data.

**Underfitting** occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. Underfitting would occur, for example, when fitting a linear model to non-linear data. Such a model would have poor predictive performance.

**Recall**: True Positive / (True Positive + False Negative). Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

**Precision**: True Positive / (True Positive + False Positive). Out of all the items labeled as positive, how many truly belong to the positive class.

My true positive rate is high which means that when a POI is present in the \

# test data I am good at flagging him or her

My identifier doesn’t have great precision, but it does have good recall.

#That means that, nearly every time a POI shows up in my test set, I am able

# to identify him or her. The cost of this is that I sometimes get some false

# positives, where non-POIs get flagged.

My identifier doesn’t have great recall, but it does have good precision\_.

#That means that whenever a POI gets flagged in my test set, I know with a

# lot of confidence that it’s very likely to be a real POI and not a false

#alarm. On the other hand, the price I pay for this is that I sometimes miss

# real POIs, since I’m effectively reluctant to pull the trigger on edge cases.

“My identifier has a really great **F1 score**

This is the best of both worlds. Both my false positive and false negative rates are **low**, which means that I can identify POI’s reliably and accurately. If my identifier finds a POI then the person is almost certainly a POI, and if the identifier does not flag someone, then they are almost certainly not a POI.”