Qikify Documentation

Release 0.2.0

Nate Kupp

March 27, 2012

CONTENTS

PDF Version **Date**: March 27, 2012 **Version**: 0.2.0

Source Repository: http://github.com/trela/qikify

qikify is a Python package providing data structures and algorithms for semiconductor manufacturing data analysis.

Note: This documentation assumes general familiarity with NumPy. If you haven't used NumPy much or at all, do invest some time in learning about NumPy first.

See the package overview for more detail about what's in the library.

CHAPTER

ONE

API REFERENCE

1.1 General functions

1.1.1 Helper Functions

create_logger(logmodule)	
bool2symmetric(data)	Changes True/False data to +1/-1 symmetric.
standardize(X[, scaleDict, reverse])	Facilitates standardizing data by subtracting the mean and dividing by the standard deviatio
zeroMatrixDiagonal(X)	Set the diagonal of a matrix to all zeros.
getParetoFront(data)	Extracts the 2D Pareto-optimal front from a 2D numpy array.
is1D(data)	Determine if data is 1-dimensional.
partition(data[, threshold, verbose])	Partitions data into training and test sets.
nmse(yhat, y[, min_y, max_y])	Calculates the normalized mean-squared error.
computeR2(yhat, y)	Computes R-squared coefficient of determination.

qikify.helpers.create_logger

qikify.helpers.**create_logger = <function create_logger at 0x1037a1e60>**

qikify.helpers.bool2symmetric

qikify.helpers.**bool2symmetric = <function bool2symmetric at 0x1037a8488>**

Changes True/False data to +1/-1 symmetric.

qikify.helpers.standardize

qikify.helpers.**standardize = <function standardize at 0x1037a8500>**

Facilitates standardizing data by subtracting the mean and dividing by the standard deviation. Set reverse to True to perform the inverse operation.

Parameters **X** : numpy ndarray, or pandas.DataFrame

Data for which we want pareto-optimal front.

scaleDict: dict, default None :

Dictionary with elements mean/std to control standardization.

reverse: boolean, default False :

If this flag is set, the standardization will be reversed; e.g., we take a dataset with zero mean and unit variance and change to dataset with mean=scaleDict.mean and std=scaleDict.std.

Examples

TODO

qikify.helpers.zeroMatrixDiagonal

qikify.helpers.**zeroMatrixDiagonal = <function zeroMatrixDiagonal at 0x1037a8578>**
Set the diagonal of a matrix to all zeros.

Parameters **X** : numpy ndarray

Matrix on which to zero out the diagonal.

Examples

Xp = zeroMatrixDiagonal(X)

qikify.helpers.getParetoFront

qikify.helpers.**getParetoFront = <function getParetoFront at 0x1037a85f0>**
Extracts the 2D Pareto-optimal front from a 2D numpy array.

Parameters **data** : numpy ndarray, or pandas.DataFrame

Data for which we want pareto-optimal front.

Examples

p = getParetoFront(data)

qikify.helpers.is1D

qikify.helpers.**is1D = <function is1D at 0x1037a8668>**
Determine if data is 1-dimensional.

qikify.helpers.partition

qikify.helpers.**partition = <function partition at 0x1037a86e0>**
Partitions data into training and test sets. Assumes the last column of data is y.

Parameters **data** : numpy ndarray, or pandas.DataFrame

Data to partition into training and test sets.

threshold : float

Determines ratio of training : test.

Examples

TODO

qikify.helpers.nmse

qikify.helpers.**nmse = <function nmse at 0x1037a8758>**
Calculates the normalized mean-squared error.

Parameters **yhat** : 1d array or list of floats

estimated values of y

y : 1d array or list of floats

true values

min_y, max_y : float, float

roughly the min and max; they do not have to be the perfect values of min and max, because they're just here to scale the output into a roughly [0,1] range

Examples

nmse = nmse(yhat, y)

qikify.helpers.computeR2

qikify.helpers.**computeR2 = <function computeR2 at 0x1037a87d0>**
Computes R-squared coefficient of determination.

R2 = 1 - sum((y_hat - y_test)**2) / sum((y_test - np.mean(y_test))**2)

Parameters **yhat** : 1d array or list of floats – estimated values of y

y : 1d array or list of floats – true values

Examples

r2 = computeR2(yhat, y)

1.2 Models

1.2.1 Chip

Chip(chip_dict[, LCT_prefix])	Encapsulates chip-level data.

qikify.models.chip.Chip

qikify.models.chip.**Chip = <class 'qikify.models.chip.Chip'>**
Encapsulates chip-level data.

1.3 Views

syntheticAndReal(sData, bData, d1, d2, filename)	
histogram(sData, bData, i[, filename])	
yp_vs_y(yp, y[, filename])	This method plots y predicted vs.
qq(x[, filename])	
coef_path(coefs)	Plot the coefficient paths generated by elastic net / lasso.
pairs(data[, labels, filename])	Generates something similar to R pairs()
te_and_yl(error, errorSyn, filename, description)	
laplacianScores(filename, Scores, Ranking)	
wafermap(x, y, val[, filename])	

1.3.1 qikify.views.charts.syntheticAndReal

qikify.views.charts.**syntheticAndReal = <function syntheticAndReal at 0x1059d65f0>**

1.3.2 qikify.views.charts.histogram

qikify.views.charts.**histogram = <function histogram at 0x1059d6668>**

1.3.3 qikify.views.charts.yp_vs_y

qikify.views.charts.**yp_vs_y = <function yp_vs_y at 0x1059d66e0>**

This method plots y predicted vs. y actual on a 45-degree chart.

1.3.4 qikify.views.charts.qq

qikify.views.charts.**qq = <function qq at 0x1059d6758>**

1.3.5 qikify.views.charts.coef_path

qikify.views.charts.**coef_path = <function coef_path at 0x1059d67d0>**

Plot the coefficient paths generated by elastic net / lasso.

1.3.6 qikify.views.charts.pairs

qikify.views.charts.**pairs = <function pairs at 0x1059d6848>**

Generates something similar to R pairs()

1.3.7 qikify.views.charts.te_and_yl

qikify.views.charts.**te_and_yl = <function te_and_yl at 0x1059d68c0>**

1.3.8 qikify.views.charts.laplacianScores

qikify.views.charts.**laplacianScores = <function laplacianScores at 0x1059d6938>**

1.3.9 qikify.views.charts.wafermap

qikify.views.charts.**wafermap = <function wafermap at 0x1059d69b0>**

1.4 Controllers

identifyOutliers.identifyOutliers(data[, k])	Compare a dataset against mu +/- k*sigma limits, and
identifyOutliers.identifyOutliersSpecs(data, ...)	Compare a dataset against expanded spec limits, and

1.4.1 qikify.controllers.identifyOutliers.identifyOutliers

qikify.controllers.identifyOutliers.**identifyOutliers = <function identifyOutliers at 0x102a792a8>**

Compare a dataset against mu +/- k*sigma limits, and return a boolean vector with False elements denoting outliers.

Parameters **data** : Contains data stored in a pandas DataFrame or Series.

1.4.2 qikify.controllers.identifyOutliers.identifyOutliersSpecs

qikify.controllers.identifyOutliers.**identifyOutliersSpecs = <function identifyOutliersSpecs at 0x102a79**

Compare a dataset against expanded spec limits, and return a boolean vector with False elements denoting outliers.

Parameters **data** : Contains data stored in a pandas DataFrame or Series.

1.4.3 KDE

KDE()	
KDE.__init__()	Performs non-parametric kernel density estimation.
KDE.run(X[, specs, nSamples, counts, a, bounds])	Primary execution point.

qikify.controllers.KDE.KDE

qikify.controllers.KDE.**KDE = <class 'qikify.controllers.KDE.KDE'>**

qikify.controllers.KDE.KDE.__init__

qikify.controllers.KDE.KDE.run

1.4.4 Recipes

atesim
basic_ML_testing
two_tier_test

qikify.recipes.atesim

qikify.recipes.basic_ML_testing

qikify.recipes.two_tier_test

PYTHON MODULE INDEX

q

INDEX