
qikify Documentation

Release

Author

March 29, 2012

CONTENTS

1	qikify Package	3
1.1	qikify Package	3
1.2	helpers Module	3
1.3	term_helpers Module	5
1.4	Subpackages	5
2	Indices and tables	15
	Bibliography	17
	Python Module Index	19
	Python Module Index	21
	Index	23

Contents:

QIKIFY PACKAGE

1.1 qikify Package

1.2 helpers Module

`qikify.helpers.bool2symmetric(data)`
Changes True/False data to +1/-1 symmetric.

`qikify.helpers.computeR2(yhat, y)`
Computes R-squared coefficient of determination.

$$R2 = 1 - \text{sum}((y_hat - y_test)**2) / \text{sum}((y_test - \text{np.mean}(y_test))**2)$$

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)
```

`qikify.helpers.create_logger(logmodule)`

`qikify.helpers.getParetoFront(data)`
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(data)`
Determine if data is 1-dimensional.

`qikify.helpers.nmse(yhat, y, min_y=None, max_y=None)`
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.partition(data, threshold=0.5, verbose=False)
```

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.standardize(X, scaleDict=None, reverse=False)
```

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(X)
```

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)
```

1.3 term_helpers Module

```
class qikify.term_helpers.colors
    Bases: object
```

Methods

```
disable()
```

```
disable()
```

```
qikify.term_helpers.outputPassFail(gnd)
```

1.4 Subpackages

1.4.1 controllers Package

KDE Module

```
class qikify.controllers.KDE.KDE
    Bases: object
```

This class implements non-parametric kernel density estimation.

Methods

```
run(X[, specs, nSamples, counts, a, bounds])
```

Primary execution point.

```
run(X, specs=None, nSamples=0, counts=None, a=0, bounds=None)
```

Primary execution point. Run either standard KDE or class-membership based KDE. If any of the class-membership based KDE arguments are set, it will be run instead of standard KDE.

Parameters **X** : array_like

Contains data stored in a pandas.DataFrame.

nSamples : int

The number of samples to generate.

specs : qikify.models.Specs, optional

If using partitioned sampling, boundaries defining pass/critical/fail subspaces must be provided.

counts : dict, optional

If using partitioned sampling, counts dictionary must be provided, with three keys: nGood, nCritical, nFail.

LSFS Module

class qikify.controllers.LSFS.**LSFS**
 Bases: object

Methods

<code>constructS(X, gnd[, k, t, bLDA, bSelfConnected])</code>	
<code>run(Xin, gnd)</code>	Run Laplacian Score Feature Selection.
<code>threshold(T_L)</code>	

constructS (*X, gnd, k=0, t=1, bLDA=False, bSelfConnected=True*)

run (*Xin, gnd*)
 Run Laplacian Score Feature Selection.

Note: Eventually, it'd be nice to maintain col names with Xin so that we can add a plot method to plot scores vs. column names.

Parameters **Xin** : array_like

A numpy.ndarray or pandas.DataFrame, with rows corresponding to observations and columns to features.

gnd : array_like

A numpy.ndarray or pandas.DataFrame pass/fail vector of the same dimension as Xin

Notes

This code is based on the definition from the paper [R1]:

threshold (*T_L*)

OLS Module

class qikify.controllers.OLS.**OLS**
 Bases: object

Ordinary least squares multivariate regression.

Methods

<code>JB()</code>	Calculate residual skewness, kurtosis, and do the JB test for normality
<code>computeStatistics()</code>	

Continued on next page

Table 1.4 – continued from previous page

<code>dw()</code>	Calculates the Durbin-Waston statistic
<code>ll()</code>	Calculate model log-likelihood and two information criteria
<code>omni()</code>	Omnibus test for normality
<code>train(X, y[, useQR, addConstant])</code>	Solve $y = Xb$.

JB ()

Calculate residual skewness, kurtosis, and do the JB test for normality

computeStatistics ()**dw ()**

Calculates the Durbin-Waston statistic

ll ()

Calculate model log-likelihood and two information criteria

omni ()

Omnibus test for normality

train (X, y, useQR=True, addConstant=True)Solve $y = Xb$.**Parameters** **x** : array, shape (M, N)**y** : array, shape (M,)**useQR** : boolean

Whether or not to use QR decomposition to fit regression line.

addConstant: boolean :

Whether or not to add a constant column to X

QFFS Module

class `qikify.controllers.QFFS.QFFS`Bases: `object`

Qikify feature selection library. Doesn't do much yet; right now only implements correlation coefficient-based feature selection.

Methods

<code>computeCorrCoefs(X, y)</code>	Returns the correlation coefficients between X and y,
<code>run(X, y[, n_features, intercept, method])</code>	Do feature selection on the basis of correlation coefficients.

computeCorrCoefs (X, y)

Returns the correlation coefficients between X and y, along with the arg-sorted indices of ranked most-correlated X-to-y vars.

run (X, y, n_features=10, intercept=True, method='corrcoef')

Do feature selection on the basis of correlation coefficients.

Parameters **X** : numpy array of shape [n_samples,n_features]

Training data

y : numpy array of shape [n_samples]

Target values

n_features : int, optional

Number of features to retain

intercept : bool, optional

Whether the first column is an all-constant intercept and should be excluded

method : string, optional

Determines the feature selection method to use.

Returns **features** : The X column indices to retain.

Notes

We typically exclude the first column since it is the intercept all-constant column.

SVM Module

class `qikify.controllers.SVM.SVM`

Bases: object

Methods

<code>getTEYL(gnd, predicted)</code>
<code>predict(X)</code>
<code>train(X, gnd[, gridSearch])</code>

getTEYL (*gnd*, *predicted*)

predict (*X*)

train (*X*, *gnd*, *gridSearch=False*)

identifyOutliers Module

`qikify.controllers.identifyOutliers.identifyOutliers` (*data*, *k=3*)

Compare a dataset against $\mu \pm k \cdot \sigma$ limits, and return a boolean vector with False elements denoting outliers.

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

`qikify.controllers.identifyOutliers.identifyOutliersSpecs` (*data*, *specs*, *ind*, *k=3*)

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.

interpolate Module

`qikify.controllers.interpolate.bilinear_interp` (*x*, *y*, *xlim*, *ylim*, *Q*)
 bilinear interpolation of *z* over 2d surface {*x*,*y*}

`qikify.controllers.interpolate.cart2polar` (*x*, *y*)

`qikify.controllers.interpolate.cart2polar_recenter` (*x*, *y*, *xmax*, *ymax*)

`qikify.controllers.interpolate.lerp` (*x*, *xlim*, *ylim*)
 linearly interpolate a value of *y* given ranges for *x*, *y*.

arguments: *x*: scalar *xlim*: array with *xmin*, *xmax* *ylim*: array with *ymin*, *ymax*

`qikify.controllers.interpolate.polar2cart` (*r*, *theta*)

`qikify.controllers.interpolate.polar2cart_recenter` (*r*, *theta*, *xmax*, *ymax*)

slicesample Module

`qikify.controllers.slicesample.inside` (*x*, *th*, *pdf*)

`qikify.controllers.slicesample.logpdf` (*x*, *pdf*)

`qikify.controllers.slicesample.outside` (*x*, *th*, *pdf*)

`qikify.controllers.slicesample.slicesample` (*x0*, *nsamples*, *pdf*, *width=10*, *maxiter=200*)
 Loosely based on `slicesample()` from MATLAB.

1.4.2 models Package

chip Module

class `qikify.models.chip.Chip` (*chip_dict*, *LCT_prefix=''*)
 Bases: `object`

This class encapsulates chip-level data.

dataset Module

Warning: Deprecated in version 0.2.

class `qikify.models.dataset.Dataset` (*filename=None*, *files=None*, *dataset=None*)
 Bases: `qikify.models.dotdict.dotdict`

This class is the fundamental data structure of the Qikify framework.

Methods

<code>clear()</code> -> <code>None</code> . Remove all items from <code>D</code> .)	
<code>copy()</code> -> a shallow copy of <code>D</code>)	
<code>fromkeys(...)</code>	<i>v</i> defaults to <code>None</code> .
<code>get((<i>k</i>,<i>d</i>))</code> -> <code>D[<i>k</i>]</code> if <i>k</i> in <code>D</code> , ...)	

Continued on next page

Table 1.7 – continued from previous page

<code>has_key(k)</code> -> True if D has a key k, else False)	
<code>items()</code> -> list of D's (key, value) pairs, ...)	
<code>iteritems()</code> -> an iterator over the (key, ...)	
<code>iterkeys()</code> -> an iterator over the keys of D)	
<code>itervalues(...)</code>	
<code>keys()</code> -> list of D's keys)	
<code>pop(k[,d])</code> -> v, ...)	If key is not found, d is returned if given, otherwise <code>KeyError</code> is raised
<code>popitem()</code> -> (k, v), ...)	2-tuple; but raise <code>KeyError</code> if D is empty.
<code>setdefault(k[,d])</code> -> D.get(k,d), ...)	
<code>update(E, ...)</code>	If E has a <code>.keys()</code> method, does: for k in E: D[k] = E[k]
<code>values()</code> -> list of D's values)	
<code>viewitems(...)</code>	
<code>viewkeys(...)</code>	
<code>viewvalues(...)</code>	

dotdict Module

class `qikify.models.dotdict.dotdict`

Bases: `dict`

We use `dotdict` to replace standard Python dictionaries. This is simply for the convenience of having `dict.property` access, instead of the messier `dict['property']` style.

Methods

<code>clear()</code> -> None. Remove all items from D.)	
<code>copy()</code> -> a shallow copy of D)	
<code>fromkeys(...)</code>	v defaults to None.
<code>get(k[,d])</code> -> D[k] if k in D, ...)	
<code>has_key(k)</code> -> True if D has a key k, else False)	
<code>items()</code> -> list of D's (key, value) pairs, ...)	
<code>iteritems()</code> -> an iterator over the (key, ...)	
<code>iterkeys()</code> -> an iterator over the keys of D)	
<code>itervalues(...)</code>	
<code>keys()</code> -> list of D's keys)	
<code>pop(k[,d])</code> -> v, ...)	If key is not found, d is returned if given, otherwise <code>KeyError</code> is raised
<code>popitem()</code> -> (k, v), ...)	2-tuple; but raise <code>KeyError</code> if D is empty.
<code>setdefault(k[,d])</code> -> D.get(k,d), ...)	
<code>update(E, ...)</code>	If E has a <code>.keys()</code> method, does: for k in E: D[k] = E[k]
<code>values()</code> -> list of D's values)	
<code>viewitems(...)</code>	
<code>viewkeys(...)</code>	
<code>viewvalues(...)</code>	

class `qikify.models.dotdict.mdotmap(*args, **kwargs)`

Bases: `_abcoll.MutableMapping`

We use `mdotmap` to replace standard Python dictionaries. This is simply for the convenience of having `mdotmap.attr` access, instead of the `dict[attr]` style.

**** NOT YET WORKING ****

Methods

<code>clear()</code>
<code>get(key[, default])</code>
<code>items()</code>
<code>iteritems()</code>
<code>iterkeys()</code>
<code>intervalues()</code>
<code>keys()</code>
<code>pop(key[, default])</code>
<code>popitem()</code>
<code>setdefault(key[, default])</code>
<code>update(*args, **kws)</code>
<code>values()</code>

helpers Module

`qikify.models.helpers.gz_csv_read(file_path, pandasDF=False)`

`qikify.models.helpers.gz_csv_write(file_path, data)`

specs Module

class `qikify.models.specs.Specs(filename=None, specs=None)`
 Bases: `object`

Methods

<code>computePassFail(data)</code>	Compare a pandas Series or DataFrame structure to specification limits defined by
<code>genCriticalRegion(k_i, k_o)</code>	Takes specification boundary and generates two boundaries to define ‘critical’ device region.

computePassFail (*data*)

Compare a pandas Series or DataFrame structure to specification limits defined by this spec class instance.

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

genCriticalRegion (*k_i*, *k_o*)

Takes specification boundary and generates two boundaries to define ‘critical’ device region.

Parameters *k_i* : Inner critical region multiplier.

k_u : Outer critical region multiplier.

1.4.3 recipes Package

atesim Module

class `qikify.recipes.atesim.ATESimulator(data_src='filesystem')`
 Bases: `object`

Methods

`run([port])` This function runs the ATE simulator using CSV files in the current directory.

run (*port=5570*)

This function runs the ATE simulator using CSV files in the current directory. Currently, we only support loading .csv or .csv.gz files.

class qikify.recipes.atesim.**ChipDataIterator** (*data_dir*)
 Bases: object

Methods

`next()` The call to self.chip_iter.next() will raise StopIteration when done,

next ()

The call to self.chip_iter.next() will raise StopIteration when done, propagating through to the caller of ChipDataIterator().next().

basic_ML_testing Module

class qikify.recipes.basic_ML_testing.**BasicMLTesting**
 Bases: object

Methods

`run([port])`

run (*port=5570*)

two_tier_test Module

class qikify.recipes.two_tier_test.**TwoTierTest**
 Bases: object

Methods

`run([port])`

run (*port=5570*)

1.4.4 views Package

charts Module

`qikify.views.charts.coef_path` (*coefs*)
Plot the coefficient paths generated by elastic net / lasso.

`qikify.views.charts.histogram` (*sData*, *bData*, *i*, *filename=None*)

`qikify.views.charts.laplacianScores` (*filename*, *Scores*, *Ranking*)

`qikify.views.charts.pairs` (*data*, *labels=None*, *filename=None*)
Generates something similar to R pairs()

`qikify.views.charts.percentFormatter` (*x*, *pos=0*)

`qikify.views.charts.qq` (*x*, *filename=None*)

`qikify.views.charts.syntheticAndReal` (*sData*, *bData*, *d1*, *d2*, *filename*)

`qikify.views.charts.te_and_y1` (*error*, *errorSyn*, *filename*, *description*)

`qikify.views.charts.wafermap` (*x*, *y*, *val*, *filename=None*)
Plots a heatmap of argument val over wafer coordinates.

`qikify.views.charts.yp_vs_y` (*yp*, *y*, *filename=None*)
This method plots y predicted vs. y actual on a 45-degree chart.

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

BIBLIOGRAPHY

[R1] He, X. and Cai, D. and Niyogi, P., “Laplacian Score for Feature Selection”, NIPS 2005.

PYTHON MODULE INDEX

q

- [qikify.__init__, 3](#)
- [qikify.controllers.identifyOutliers, 8](#)
- [qikify.controllers.interpolate, 9](#)
- [qikify.controllers.KDE, 5](#)
- [qikify.controllers.LSFS, 6](#)
- [qikify.controllers.OLS, 6](#)
- [qikify.controllers.QFFS, 7](#)
- [qikify.controllers.slicesample, 9](#)
- [qikify.controllers.SVM, 8](#)
- [qikify.helpers, 3](#)
- [qikify.models.chip, 9](#)
- [qikify.models.dataset, 9](#)
- [qikify.models.dotdict, 10](#)
- [qikify.models.helpers, 11](#)
- [qikify.models.specs, 11](#)
- [qikify.recipes.atesim, 11](#)
- [qikify.recipes.basic_ML_testing, 12](#)
- [qikify.recipes.two_tier_test, 12](#)
- [qikify.term_helpers, 5](#)
- [qikify.views.charts, 13](#)

PYTHON MODULE INDEX

q

- `qikify.__init__`, 3
- `qikify.controllers.identifyOutliers`, 8
- `qikify.controllers.interpolate`, 9
- `qikify.controllers.KDE`, 5
- `qikify.controllers.LSFS`, 6
- `qikify.controllers.OLS`, 6
- `qikify.controllers.QFFS`, 7
- `qikify.controllers.slicesample`, 9
- `qikify.controllers.SVM`, 8
- `qikify.helpers`, 3
- `qikify.models.chip`, 9
- `qikify.models.dataset`, 9
- `qikify.models.dotdict`, 10
- `qikify.models.helpers`, 11
- `qikify.models.specs`, 11
- `qikify.recipes.atesim`, 11
- `qikify.recipes.basic_ML_testing`, 12
- `qikify.recipes.two_tier_test`, 12
- `qikify.term_helpers`, 5
- `qikify.views.charts`, 13

INDEX

A

ATESimulator (class in qikify.recipes.atesim), 11

B

BasicMLTesting (class in qikify.recipes.basic_ML_testing), 12

bilinear_interp() (in module qikify.controllers.interpolate), 9

bool2symmetric() (in module qikify.helpers), 3

C

cart2polar() (in module qikify.controllers.interpolate), 9

cart2polar_recenter() (in module qikify.controllers.interpolate), 9

Chip (class in qikify.models.chip), 9

ChipDataIterator (class in qikify.recipes.atesim), 12

coef_path() (in module qikify.views.charts), 13

colors (class in qikify.term_helpers), 5

computeCorrCoefs() (qikify.controllers.QFFS.QFFS method), 7

computePassFail() (qikify.models.specs.Specs method), 11

computeR2() (in module qikify.helpers), 3

computeStatistics() (qikify.controllers.OLS.OLS method), 7

constructS() (qikify.controllers.LSFS.LSFS method), 6

create_logger() (in module qikify.helpers), 3

D

Dataset (class in qikify.models.dataset), 9

disable() (qikify.term_helpers.colors method), 5

dotdict (class in qikify.models.dotdict), 10

dw() (qikify.controllers.OLS.OLS method), 7

G

genCriticalRegion() (qikify.models.specs.Specs method), 11

getParetoFront() (in module qikify.helpers), 3

getTEYL() (qikify.controllers.SVM.SVM method), 8

gz_csv_read() (in module qikify.models.helpers), 11

gz_csv_write() (in module qikify.models.helpers), 11

H

histogram() (in module qikify.views.charts), 13

I

qik-identifyOutliers() (in module qikify.controllers.identifyOutliers), 8

qik-identifyOutliersSpecs() (in module qikify.controllers.identifyOutliers), 8

inside() (in module qikify.controllers.slicesample), 9

is1D() (in module qikify.helpers), 3

J

JB() (qikify.controllers.OLS.OLS method), 7

K

KDE (class in qikify.controllers.KDE), 5

L

laplacianScores() (in module qikify.views.charts), 13

lerp() (in module qikify.controllers.interpolate), 9

ll() (qikify.controllers.OLS.OLS method), 7

logpdf() (in module qikify.controllers.slicesample), 9

LSFS (class in qikify.controllers.LSFS), 6

M

mdotmap (class in qikify.models.dotdict), 10

N

next() (qikify.recipes.atesim.ChipDataIterator method), 12

nmse() (in module qikify.helpers), 3

O

OLS (class in qikify.controllers.OLS), 6

omni() (qikify.controllers.OLS.OLS method), 7

outputPassFail() (in module qikify.term_helpers), 5

outside() (in module qikify.controllers.slicesample), 9

P

pairs() (in module qikify.views.charts), 13

partition() (in module qikify.helpers), 4
 percentFormatter() (in module qikify.views.charts), 13
 polar2cart() (in module qikify.controllers.interpolate), 9
 polar2cart_recenter() (in module qikify.controllers.interpolate), 9
 predict() (qikify.controllers.SVM.SVM method), 8

Q

QFFS (class in qikify.controllers.QFFS), 7
 qikify.__init__ (module), 3
 qikify.controllers.identifyOutliers (module), 8
 qikify.controllers.interpolate (module), 9
 qikify.controllers.KDE (module), 5
 qikify.controllers.LSFS (module), 6
 qikify.controllers.OLS (module), 6
 qikify.controllers.QFFS (module), 7
 qikify.controllers.slicesample (module), 9
 qikify.controllers.SVM (module), 8
 qikify.helpers (module), 3
 qikify.models.chip (module), 9
 qikify.models.dataset (module), 9
 qikify.models.dotdict (module), 10
 qikify.models.helpers (module), 11
 qikify.models.specs (module), 11
 qikify.recipes.atesim (module), 11
 qikify.recipes.basic_ML_testing (module), 12
 qikify.recipes.two_tier_test (module), 12
 qikify.term_helpers (module), 5
 qikify.views.charts (module), 13
 qq() (in module qikify.views.charts), 13

R

run() (qikify.controllers.KDE.KDE method), 5
 run() (qikify.controllers.LSFS.LSFS method), 6
 run() (qikify.controllers.QFFS.QFFS method), 7
 run() (qikify.recipes.atesim.ATESimulator method), 12
 run() (qikify.recipes.basic_ML_testing.BasicMLTesting method), 12
 run() (qikify.recipes.two_tier_test.TwoTierTest method), 12

S

slicesample() (in module qikify.controllers.slicesample), 9
 Specs (class in qikify.models.specs), 11
 standardize() (in module qikify.helpers), 4
 SVM (class in qikify.controllers.SVM), 8
 syntheticAndReal() (in module qikify.views.charts), 13

T

te_and_y1() (in module qikify.views.charts), 13
 threshold() (qikify.controllers.LSFS.LSFS method), 6
 train() (qikify.controllers.OLS.OLS method), 7

train() (qikify.controllers.SVM.SVM method), 8
 TwoTierTest (class in qikify.recipes.two_tier_test), 12

W

wafermap() (in module qikify.views.charts), 13

Y

yp_vs_y() (in module qikify.views.charts), 13

Z

zeroMatrixDiagonal() (in module qikify.helpers), 4