PLSA Documentation

Release 0.0.2

Pei Liu

Nov 21, 2018

Contents:

CHAPTER 1

PLSA package

1.1 Subpackages

1.1.1 PLSA.data package

Submodules

PLSA.data.processing module

Module for processing data

The function of this Module is served for processing data.

`PLSA.data.processing.`**`cut_groups`**(*data*, *col*, *cutoffs*)

Cut data into subsets according to cutoffs

Parameters

- **data** (*pandas.DataFrame*) – Data to split.
- **col** (*str*) – Name of column in data to compare with.
- **cutoffs** (*list (int)*) – List of cutoffs, like as [min-value, 30, 60, max-value].

Returns List of sub-data as DataFrame.

Return type list(pandas.DataFrame)

Examples

```
>>> cut_groups(data, "X", [0, 0.4, 0.6, 1.0])
[pandas.DataFrame, pandas.DataFrame, pandas.DataFrame]
```

`PLSA.data.processing.`**`parse_surv`**(*x*, *label*)

Parse raw-data for survival analyze(Deep Surival).

Parameters

- **x** (*np.array*) – two-dimension array indicating variables.
- **label** (*dict*) – Contain ‘e’, ‘t’.

  examples as {‘e’: np.array, ‘t’: np.array}.

Returns Sorted (x, e, t) tuple, index of people who is failure or at risk, and type of ties.

Return type tuple

Examples

```
>>> parse_surv(data[x_cols].values, {'e': data['e'].values, 't': data['t'].values}
↪)
```

PLSA.data.processing.**prepare_surv**(*x*, *label*)

Prepare data for survival analyze(Deep Surival).

Parameters

- **x** (*numpy.array*) – Two-dimension array indicating variables.
- **label** (*dict*) – Contain ‘e’, ‘t’.

  examples as {‘e’: np.array, ‘t’: np.array}.

Returns Sorted (x, label) tuple of survival data.

Return type tuple

Examples

```
>>> prepare_surv(data[x_cols].values, {'e': data['e'].values, 't': data['t'].
↪values})
```

Module contents

1.1.2 PLSA.qcal package

Submodules

PLSA.qcal.func module

Module for quick calling

The function of this Module is served for quick calling functions, and functions of other modules will be called by it.

PLSA.qcal.func.**div_three_groups**(*data*, *pred_col*, *duration_col*, *event_col*, *cutoffs=None*, *methods='youden'*, *pt=None*, ***kws*)

Divide data into three groups using methods and summarize result.

Parameters

- **data** (*pandas.DataFame*) – Full survival data.
- **pred_col** (*str*) – Name of column to reference for dividing groups.

- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **cutoffs** (default *None* or *tuple*) – Given cutoffs for risk groups. If *cutoffs* is not None, then methods will not be called.
- **methods** (*str*) – Methods for selecting cutoffs, default "youden".
- **pt** (*int*) – Predicted time.

Returns Print summary of result and plot KM-curve of each groups.

Return type None

Examples

```
>>> # Youden index to give cutoffs
>>> div_three_groups(data, "X", "T", "E")
>>> # Give cutoffs explicitly
>>> div_three_groups(data, "X", "T", "E", cutoffs=(20, 50))
```

PLSA.qcal.func.**surv_calibration**(*data*, *duration_col*, *event_col*, *pred_proba*, *pt=None*, *n_bins=10*, *xlabel='Predicted Risk Probability'*, *ylabel='Observed Risk Probability'*, *title='Model Performance'*, *save_fig_as="*)

Evaluate calibration of predicted survival probability at time pt.

Parameters

- **data** (*pandas.DataFame*) – Full survival data.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **pred_proba** (*np.array*) – Predicted survival probability at time pt.
- **pt** (*int*) – Predicted time.

Returns Print summary of result and plot curve of calibration.

Return type None

Examples

```
>>> surv_calibration(data, "T", "E", surv_function[10], pt=10)
```

PLSA.qcal.func.**surv_coxph**(*data_train*, *x_cols*, *duration_col*, *event_col*, *data_test=None*, *pt=None*, *show_extra=True*)

Integrate functions that include modeling using Cox Regression and evaluating

Parameters

- **data_train** (*pandas.DataFame*) – Full survival data for train.
- **x_cols** (*list of str*) – Name of column indicating variables.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.

- **data_test** (*pandas.DataFame*) – Full survival data for test, default None.
- **pt** (*float*) – Predicted time for AUC.

Returns Object of cox model in *lifelines.CoxPHFitter*.

Return type object

Examples

```
>>> surv_coxph(train_data, ['x1', 'x2'], 'T', 'E', test_data, pt=5*12)
```

PLSA.qcal.func.**surv_time_auc**(*data_train, data_test, pred_col, duration_col, event_col, pt=[], labels=['Train', 'Validation'], **kws*)

Plot curve of auc at some predicted time.

Parameters

- **data_train** (*pandas.DataFame*) – Full survival data for train.
- **data_test** (*pandas.DataFame*) – Full survival data for test.
- **pred_col** (*str*) – Name of column indicating target value.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **pt** (*list(int)*) – Predicted time indicating list of watching.

Returns Print summary of result and plot curve of auc with time.

Return type None

Examples

```
>>> surv_time_auc(train_data, test_data, 'X', 'T', 'E', pt=[1, 3, 5, 10])
```

Module contents

1.1.3 PLSA.surv package

Submodules

PLSA.surv.cutoff module

Module for determinding cutoffs in survival analyze

The function of this Module is served for determinding cutoffs by different methods in survival analyze.

PLSA.surv.cutoff.**coxph_coef**(*data, duration_col, event_col, silence=True*)

PLSA.surv.cutoff.**hazards_ratio**(*data, pred_col, duration_col, event_col, score_min=0, score_max=100, balance=True*)

Cutoff maximize HR or BHR.

Parameters

- **data** (*DataFrame*) – full survival data.

- **pred_col** (*str*) – Name of column to reference for dividing groups.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **score_min** (int, optional) – min value in pred_col.
- **score_max** (int, optional) – max value in pred_col.
- **balance** (*bool*) – True if using BHR as metrics, otherwise HR.

Returns Optimal cutoffs according to ratio of hazards methods.

Return type float

Examples

```
>>> hazards_ratio(data, 'score', 'T', 'E', balance=True)
```

PLSA.surv.cutoff.**loss_bhr**(*data_list*, *duration_col*, *event_col*, *base_val=2*, *silence=True*)

PLSA.surv.cutoff.**loss_dis**(*data*, *data_list*, *col*)

PLSA.surv.cutoff.**loss_hr**(*data_list*, *duration_col*, *event_col*, *base_val=0*, *silence=True*)

PLSA.surv.cutoff.**stats_var**(*data*, *x_col*, *y_col*, *score_min=0*, *score_max=100*)

Cutoff maximize distant between groups, minimize variance in group

Parameters

- **data** (*pd.DataFrame*) – Data set.
- **x_col** (*str*) – Name of column to reference for dividing groups.
- **y_col** (*str*) – Name of column to measure differences.
- **score_min** (int, optional) – Min value in x_col.
- **score_max** (int, optional) – Max value in x_col.

Returns Optimal cutoffs according to statistical methods.

Return type float

Examples

```
>>> stats_var(data, 'score', 'y')
```

PLSA.surv.cutoff.**youden_onecut**(*data*, *pred_col*, *duration_col*, *event_col*, *pt=None*)

Cutoff maximize Youden Index.

Parameters

- **data** (*pandas.DataFrame*) – full survival data.
- **pred_col** (*str*) – Name of column to reference for dividing groups.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **pt** (*int, default None*) – Predicted time.

Returns Value indicating cutoff for pred_col of data.

Return type float

Examples

```
>>> youden_onecut(data, 'X', 'T', 'E')
```

PLSA.surv.cutoff.**youden_twocut**(*data*, *pred_col*, *duration_col*, *event_col*, *pt=None*)

Two values of cutoff maximize Youden Index.

Parameters

- **data** (*pandas.DataFrame*) – Full survival data.
- **pred_col** (*str*) – Name of column to reference for dividing groups.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **pt** (*int*) – Predicted time.

Returns (cutoff-1, cutoff-2) value indicating cutoff for pred_col of data.

Return type tuple

Examples

```
>>> youden_twocut(data, 'X', 'T', 'E')
```

PLSA.surv.utils module

Module for utilitize function of survival analyze.

The function of this Module is served as utility of survival analyze.

PLSA.surv.utils.**surv_data_at_risk**(*data*, *duration_col*, *points=None*)

Get number of people at risk at some timing.

Parameters

- **data** (*pandas.DataFrame*) – Full survival data.
- **duration_col** (*str*) – Name of column indicating time.
- **points** (*list(int)*) – Points of Time selected to watch.

Returns Number of people at risk.

Return type *pandas.DataFrame*

Examples

```
>>> surv_data_at_risk(data, "T", points=[0, 10, 20, 30, 40, 50])
```

PLSA.surv.utils.**surv_roc**(*data*, *pred_col*, *duration_col*, *event_col*, *pt=None*)

Get survival ROC at predicted time.

Parameters

- **data** (*pandas.DataFrame*) – Full survival data.
- **pred_col** (*str*) – Name of column to reference for dividing groups.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **pt** (*int*) – Predicted time.

Returns Object of dict include “FP”, “TP” and “AUC” in ROC.

Return type *dict*

Examples

```
>>> surv_roc(data, 'X', 'T', 'E', pt=5)
```

PLSA.surv.utils.**survival_by_hr**(*T0*, *S0*, *pred*)

Get survival function of patients according to giving hazard ratio.

Parameters

- **T0** (*np.array*) – time.
- **S0** (*np.array*) – based estimated survival function of patients.
- **pred** (*pandas.Series*) – hazard ratio of patients.

Returns T0, ST indicating survival function of patients.

Return type *tuple*

Examples

```
>>> survival_by_hr(T0, S0, data['hazard_ratio'])
```

PLSA.surv.utils.**survival_status**(*data*, *duration_col*, *event_col*, *end_time*, *inplace=False*)

Get status of event at a specified time.

0: status = 0, Time = end_time (T >= end_time) status = 0, Time = T (T < end_time)

1: status = 1, Time = T (T <= end_time) status = 0, Time = end_time (T > end_time)

Parameters

- **data** (*pandas.DataFrame*) – Full survival data.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **end_time** (*int*) – End time of study.
- **inplace** (*bool, default False*) – Do replace original data.

Returns

data indicates status of survival.

None or tuple(time(pandas.Series), status(pandas.Series))

Return type None or tuple

Examples

```
>>> survival_status(data, 'T', 'E', 10, inplace=False)
```

Module contents

1.1.4 PLSA.utils package

Submodules

PLSA.utils.cutoff module

Module for determinding cutoffs in common

The function of this Module is served for determinding cutoffs by different methods in common.

`PLSA.utils.cutoff.`**`accuracy`**(*y_true*, *y_prob*)

Cutoff maximize accuracy.

Parameters

- **`y_true`** (*np.array* or *pandas.Series*) – True value.
- **`y_prob`** (*np.array* or *pandas.Series*) – Predicted value.

Returns Optimal cutoff and max metrics.

Return type tuple(float, float)

Examples

```
>>> accuracy(y_true, y_prob)
```

`PLSA.utils.cutoff.`**`youden`**(*target*, *predicted*)

Cutoff maximize Youden Index.

Parameters

- **`target`** (*np.array* or *pandas.Series*) – True value.
- **`predicted`** (*np.array* or *pandas.Series*) – Predicted value.

Returns optimal cutoff and max metrics.

Return type tuple(float, float)

Examples

```
>>> youden(y_true, y_prob)
```

PLSA.utils.metrics module

Module for evaluating model by many kinds of metrics

The function of this Module is served for evaluating model by many kinds of metrics.

`PLSA.utils.metrics.`**`calibration`**(*y_true*, *pred_proba*, *n_bins=10*, *in_sample=False*)

Calibration and test of predictive model.

Parameters

- **`y_true`** (*np.array* or *pandas.Series*) – True label.
- **`pred_proba`** (*np.array* or *pandas.Series*) – Predicted label.
- **`n_bins`** (*`int`*) – Number of groups.
- **`in_sample`** (bool, default *False*) – Is Calibration-Test in sample.

Returns Table of calibration.

Return type pandas.DataFrame

Examples

```
>>> calibration(y_test, y_pred, n_bins=5)
```

`PLSA.utils.metrics.`**`calibration_table`**(*y_true*, *y_prob*, *normalize=False*, *n_bins=10*)

Calibration table of predictive model.

Parameters

- **`y_true`** (*np.array* or *pandas.Series*) – True label.
- **`y_prob`** (*np.array* or *pandas.Series*) – Predicted label.
- **`n_bins`** (*`int`*) – Number of groups.

Returns true, sum and total number of each group.

Return type tuple(*numpy.array*)

Examples

```
>>> calibration_table(y_test, y_pred, n_bins=5)
```

`PLSA.utils.metrics.`**`discrimination`**(*y_true*, *y_pred_proba*, *threshold=None*, *name='Model X'*)

Discrimination of classification model.

Parameters

- **`y_true`** (*np.array* or *pandas.Series*) – True label.
- **`pred_proba`** (*np.array* or *pandas.Series*) – Predicted label.
- **`threshold`** (*`float`*) – Cutoff value.
- **`name`** (*`str`*) – Title for printing.

Returns

Dict with kinds of metrics.

{ "points": threshold, "Sen": Re, "Spe": Spe, "Acc": Accuracy, "F1": F1

}

Return type dict

Examples

```
>>> discrimination(y_true, y_pred_proba, threshold=0.21)
```

`PLSA.utils.metrics.`**`discrimination_ver`**(*y_true*, *y_pred_proba*, *threshold=None*, *name='Model X'*)

Discrimination of classification model in version 2.

Parameters

- **`y_true`** (*np.array* or *pandas.Series*) – True label.
- **`pred_proba`** (*np.array* or *pandas.Series*) – Predicted label.
- **`threshold`** (*`float`*) – Cutoff value.
- **`name`** (*`str`*) – Title for printing.

Returns

Dict with kinds of metrics.

{ "points": threshold, "Sen": Sen, "Spe": Spe, "PPV": ppv, "NPV": npv

}

Return type dict

Examples

```
>>> discrimination_ver(y_true, y_pred_proba, threshold=0.21)
```

PLSA.utils.test module

Module for statistical test

The function of this Module is served for statistical test.

`PLSA.utils.test.`**`Delong_Test`**(*y_true*, *pred_a*, *pred_b*)

Delong-Test for comparing two predictive model.

Parameters

- **`y_true`** (*`numpy.array or pandas.Series.`*) – True label.
- **`pred_a`** (*`numpy.array or pandas.Series.`*) – Prediction of model A.
- **`pred_b`** (*`numpy.array or pandas.Series.`*) – Prediction of model B.

Returns chi2 value and P-value.

Return type tuple

Examples

```
>>> # pred_proba1 = xgb1.predict_proba(test_X)
>>> # pred_proba2 = xgb2.predict_proba(test_X)
>>> Delong_test(test_y, pred_proba1[:, 1], pred_proba2[:, 1])
```

PLSA.utils.test.**Hosmer_Lemeshow_Test**(*bins_true*, *bins_pred*, *bins_tot*, *n_bins=10*, *in_sample=False*)

Hosmer-Lemeshow Test for testing calibration.

Parameters

- **bins_true** (*numpy.array*) – True Number of people in each group.
- **bins_pred** (*numpy.array*) – Pred Number of people in each group.
- **bins_tot** (*numpy.array*) – Totol Number of people in each group.
- **n_bins** (*int*) – Number of groups.
- **in_sample** (*bool, default False*) – Is Calibration-Test in sample.

Returns chi2 value and P value.

Return type tuple

Examples

```
>>> Hosmer_Lemeshow_Test(bins_true, bins_pred, bins_tot, n_bins=5)
```

PLSA.utils.test.**VIF_Test**(*data*, *cols=None*)

Variance Inflation Factors for each variable.

Parameters

- **data** (*pandas.DataFrame*) – Targeted data.
- **cols** (list(str), default *None*) – Given columns to calculate VIF.

Returns Return VIF for each variable included in cols.

Return type pandas.Series

Examples

```
>>> VIF_Test(data[x_cols])
```

PLSA.utils.write module

Module for outputting result

The function of this Module is served for outputting result.

PLSA.utils.write.**xgboost_to_pmml**(*data_X*, *data_y*, *par_file*, *save_model_as*)

Save Xgboost Model to PMMl file.

Parameters

- **data_X** (*pandas.DataFrame*) – Variables of train data.

- **date_y** (*pandas.DataFrame*) – Lables of train data.
- **par_file** (*str*) – File path of model's parameters.
- **save_model_as** (*str*) – File path of PMML.

Returns Generate PMML file locally as *save_model_as* given.

Return type None

Examples

```
>>> xgboost_to_pmml(data_x, data_y, "par.json", "model.pmml")
```

Module contents

1.1.5 PLSA.vision package

Submodules

PLSA.vision.calibration module

Module for visualizing curve of calibration test

The function of this Module is served for visualizing curve of calibration test.

PLSA.vision.calibration.**plot_DCalibration**(*y_true*, *pred_proba*, *n_bins=10*, *summary=True*, *xlabel='Predicted value'*, *ylabel='Observed average'*, *title='Hosmer-Lemeshow Test'*, *save_fig_as="*)

Plot calibration curve.

Parameters

- **y_true** (*numpy.array*) – True label.
- **y_prob** (*numpy.array*) – Predicted label.
- **n_bins** (*int*) – Number of groups.

Returns

Summary table of result.

Plot figure of calibration curve.

Return type None

Examples

```
>>> plot_DCalibration(test_y, test_pred, n_bins=5)
```

PLSA.vision.lib module

Module for visualizing common curve

The function of this Module is served for visualizing common curve.

`PLSA.vision.lib.`**`plot_cphCoef`**(*dfx, coef_col=’coef’, se_col=’se(coef)’, c_col=’p’, name_col=None, ci=0.95, error_bar=’hr’, xlabel=’Name of variable’, ylabel=”, title="Variable’s coefficient of CPH model", figsize=(8, 6), save_fig_as=”*)

Visualize variables’ coefficient in lifelines.CPH model

Parameters

- **`dfx`** (*`pandas.DataFrame`*) – Object equals to cph.summary.
- **`coef_col`** (*`str`*) – Name of column indicating coefficient.
- **`se_col`** (*`str`*) – Name of column indicating standard error.
- **`c_col`** (*`str`*) – Name of column indicating color.
- **`name_col`** (*`str`*) – Name of x-axis’s column.
- **`ci`** (*`float`*) – Confidence interval, default 0.95.
- **`error_bar`** (*`str`*) – Type of error bars, ‘hr’ for asymmetrical error bars, ‘log-hr’ for symmetrical error bars.

Returns Plot figure of coefficient.

Return type None

Examples

```
>>> plot_cphCoef(cph.summary, 'coef', 'se(coef)', 'p')
```

PLSA.vision.roc module

Module for visualizing ROC curve

The function of this Module is served for visualizing ROC curve.

`PLSA.vision.roc.`**`plot_DROC`**(*y_true, y_pred, x_true=None, x_pred=None, **kws*)

Plot ROC curve for giving data.

Parameters

- **`y_true`** – True label in train data.
- **`y_pred`** – Predict label in train data.
- **`x_true`** – True label in test data.
- **`x_pred`** – Predict label in test data.
- **`**kws`** – Arguments for plotting.

Returns Plot figure of AUC

Return type None

Examples

```
>>> plot_DROC(train_y, train_pred, test_y, test_pred)
```

PLSA.vision.roc.**plot_ROC**(*data_roc*, *xlabel='1 - Specificity'*, *ylabel='Sensitivity'*, *title='Model Performance'*, *save_fig_as="*)

Plot one ROC curve in one figure.

Parameters

- **data_roc** (*dict*) – Python dict contains values about 'FP', 'TP', 'AUC'.
- **save_fig_as** (*str*) – Name of file for saving in local.

Examples

```
>>> plot_ROC(data_roc)
```

PLSA.vision.roc.**plot_SROC**(*data_train, data_test, pred_col, duration_col, event_col, pt=None, labels=['Train', 'Validation'], **kws*)

Plot Time-Dependent survival ROC curve for giving data.

Parameters

- **data_train** (*pandas.DataFrame*) – Train DataFrame included columns of Event, Duration, Pred.
- **data_train** – Test DataFrame included columns of Event, Duration, Pred.
- **pred_col** (*str*) – Name of column indicating predicted value.
- **duration_col** (*str*) – Name of column indicating time.
- **event_col** (*str*) – Name of column indicating event.
- **pt** (*int*) – Predicte time.
- ****kws** – Arguments for plotting.

Returns Plot figure of AUC

Return type None

Examples

```
>>> plot_SROC(data_train, data_test, "X", "T", "E", pt=5)
```

PLSA.vision.roc.**plot_twoROC**(*train_roc, test_roc, labels=['Train', 'Validation'], xlabel='1 - Specificity', ylabel='Sensitivity', title='Model Performance', save_fig_as="*)

Plot two ROC curve in one figure.

Parameters

- **train_roc** (*dict*) – Python dict contains values about 'FP', 'TP', 'AUC'.
- **test_roc** (*dict*) – Python dict contains values about 'FP', 'TP', 'AUC'.
- **save_fig_as** (*str*) – Name of file for saving in local.

Examples

```
>>> plot_twoROC(train_roc, test_roc)
```

PLSA.vision.survrisk module

Module for visualizing a kind of curves in survival analyze

The function of this Module is served for visualizing a kind of curves in survival analyze.

PLSA.vision.survrisk.**plot_riskGroups**(*data_groups*, *event_col*, *duration_col*, *labels=[]*, *plot_join=False*, *xlabel='Survival time (Month)'*, *ylabel='Survival Rate'*, *title='Survival function of Risk groups'*, *save_fig_as=”*)

Plot survival curve for different risk groups.

Parameters

- **data_groups** (list(*pandas.DataFame*)) – list of DataFame[['E', 'T']], risk groups from lowest to highest.
- **event_col** (*str*) – column in DataFame indicating events.
- **duration_col** (*atr*) – column in DataFame indicating durations.
- **labels** (*list(str), default []*) – One text label for one group.
- **plot_join** (*bool, default False*) – Is plotting for two adjacent risk group, default False.
- **save_fig_as** (*str*) – Name of file for saving in local.

Returns Plot figure of each risk-groups.

Return type None

Examples

```
>>> plot_riskGroups(df_list, "E", "T", labels=["Low", "Mid", "High"])
```

PLSA.vision.survrisk.**plot_rsRisk**(*data*, *x_col*, *y1_col*, *y2_col*, *labels=['Line-1', 'Line2']*, *xlabel='Risk Score'*, *ylabel='Rate of Risk'*, *title='Curve of risk score and rate of risk'*, *save_fig_as=”*)

Plot continues function between risk score and rate of risk.

Parameters

- **data** (*pandas.DataFame*) – Full survival data.
- **x_col** (*str*) – Name of column indicating risk score.
- **y1_col** (*str*) – Name of column indicating rate of risk at t1.
- **y2_col** (*str*) – Name of column indicating rate of risk at t2.
- ****kws** – Setting of plot.

Returns Plot figure of RS-rate.

Return type None

Examples

```
>>> plot_rsRisk(data, 'RS', 'pred_idfs_y5', 'pred_idfs_y10', labels=['5 Year.',
↪'10 Year.'])
```

PLSA.vision.survrisk.**plot_timeAUC**(*x, y_train, y_test, labels=['Train', 'Validation'], xlabel='Time', ylabel='AUC', title='Model Performance', save_fig_as="*)

Plot line chart about time and AUC.

Parameters

- **x** (*list*) – Time.
- **y_train** (*list*) – AUC of train.
- **y_test** (*list*) – AUC of test.
- ****kws** – Setting of plot.

Returns Plot figure of auc with time.

Return type None

Examples

```
>>> plot_timeAUC([1, 3, 5, 10], train_list, test_list)
```

Module contents

1.2 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index