Package ‘BioMoR’

October 3, 2025

Type Package

Title Bioinformatics Modeling with Recursion and Autoencoder-Based Ensemble

Version 0.1.0

Description Tools for bioinformatics modeling using recursive transformer-inspired architectures, autoencoders, random forests, XGBoost, and stacked ensemble models. Includes utilities for cross-validation, calibration, benchmarking, and threshold optimization in predictive modeling workflows. The methodology builds on ensemble learning (Breiman 2001 <doi:10.1023/A:1010933404324>), gradient boosting (Chen and Guestrin 2016 <doi:10.1145/2939672.2939785>), autoencoders (Hinton and Salakhutdinov 2006 <doi:10.1126/science.1127647>), and recursive transformer efficiency approaches such as Mixture-of-Recursions (Bae et al. 2025 <doi:10.48550/arXiv.2507.10524>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports caret, recipes, themis, xgboost, magrittr, dplyr, pROC

Suggests randomForest, testthat (>= 3.0.0), PRROC, ggplot2, purrr, tibble, yardstick, knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author MD. Arshad [aut, cre]

Maintainer MD. Arshad <`arshad10867c@gmail.com`>

Repository CRAN

Date/Publication 2025-10-03 13:50:02 UTC

Contents

`biomor_benchmark` *Benchmark a trained model*

Description

Evaluates a trained caret model on test data, returning Accuracy, F1 score, and ROC-AUC. If only one class is present in the test set, ROC-AUC is returned as NA.

Usage

```
biomor_benchmark(model, test_data, outcome_col)
```

Arguments

`model`	A trained caret model
`test_data`	Dataframe containing predictors and outcome
`outcome_col`	Name of outcome column

Value

A named list of metrics

`biomor_run_pipeline` *Run full BioMoR pipeline*

Description

Run full BioMoR pipeline

Usage

```
biomor_run_pipeline(data, feature_cols = NULL, epochs = 50)
```

Arguments

`data`	dataframe with Label + descriptors
`feature_cols`	optional feature set
`epochs`	autoencoder epochs

Value

list of trained models + benchmark reports

`brier_score` *Compute Brier Score*

Description

The Brier score is the mean squared error between predicted probabilities and the true binary outcome (0/1). Lower is better.

Usage

```
brier_score(y_true, y_prob, positive = "Active")
```

Arguments

`y_true`	True factor labels.
`y_prob`	Predicted probabilities for the positive class.
`positive`	Name of the positive class (default `"Active"`).

Value

Numeric Brier score.

`calibrate_model` *Calibrate model probabilities*

Description

Calibrate model probabilities

Usage

```
calibrate_model(model, test_data, method = "platt")
```

Arguments

`model`	caret or xgboost model
`test_data`	test dataframe
`method`	"platt" or "isotonic"

Value

calibrated probs

`compute_f1_threshold` *Compute optimal threshold for maximum F1 score*

Description

Sweeps thresholds between 0 and 1 to find the one that maximizes F1.

Usage

```
compute_f1_threshold(y_true, y_prob, positive = "Active")
```

Arguments

`y_true`	True factor labels.
`y_prob`	Predicted probabilities for the positive class.
`positive`	Name of the positive class (default `"Active"`).

Value

A list with elements:

threshold Best probability cutoff.

best_f1 Maximum F1 score achieved.

`get_cv_control` *Get caret cross-validation control*

Description

Creates a caret::trainControl object for cross-validation, configured for two-class problems, ROC-based performance, and optional sampling strategies such as SMOTE or ROSE.

Usage

```
get_cv_control(cv = 5, sampling = NULL)
```

Arguments

`cv`	Number of folds (default 5).
`sampling`	Sampling method (e.g., "smote", "rose", or NULL).

Value

A caret::trainControl object.

`get_embeddings` *Get Embeddings from Autoencoder (stub)*

Description

Placeholder for extracting embeddings from a trained autoencoder.

Usage

```
get_embeddings(ae_obj, data, feature_cols = NULL)
```

Arguments

`ae_obj`	Autoencoder object
`data`	Input data
`feature_cols`	Columns to use as features

Value

Matrix of embeddings (currently NULL since this is a stub)

`prepare_model_data` *Prepare dataset for modeling*

Description

Prepare dataset for modeling

Usage

```
prepare_model_data(df, outcome_col = "Label")
```

Arguments

`df`	A data.frame
`outcome_col`	Name of the outcome column

Value

A processed data.frame with factor outcome

`train_autoencoder` *Train Autoencoder (stub)*

Description

Placeholder for future autoencoder integration in BioMoR.

Usage

```
train_autoencoder(
  data,
  feature_cols = NULL,
  epochs = 10,
  batch_size = 32,
  lr = 0.001
)
```

Arguments

`data`	Input data (matrix or data frame)
`feature_cols`	Columns to use as features
`epochs`	Number of training epochs
`batch_size`	Mini-batch size
`lr`	Learning rate

Value

A placeholder list with class "autoencoder"

`train_biomor` *Train BioMoR Autoencoder*

Description

Train BioMoR Autoencoder

Usage

```
train_biomor(data, feature_cols, epochs = 100, batch_size = 50, lr = 0.001)
```

Arguments

`data`	Dataframe with numeric features + Label
`feature_cols`	Character vector of feature columns
`epochs`	Number of training epochs
`batch_size`	Batch size
`lr`	Learning rate

Value

list(model, dataset, embeddings)

`train_rf` *Train a Random Forest model with caret*

Description

Train a Random Forest model with caret

Usage

```
train_rf(df, outcome_col = "Label", ctrl)
```

Arguments

`df`	A data.frame containing predictors and outcome
`outcome_col`	Name of the outcome column (binary factor)
`ctrl`	A caret::trainControl object

Value

A caret train object

`train_xgb_caret` *Train an XGBoost model with caret*

Description

Train an XGBoost model with caret

Usage

```
train_xgb_caret(df, outcome_col = "Label", ctrl)
```

Arguments

`df`	A data.frame containing predictors and outcome
`outcome_col`	Name of the outcome column (binary factor)
`ctrl`	A caret::trainControl object

Value

A caret train object

Index