# Package 'immunaut'

April 9, 2025

**Version** 1.0.2

**Date** 2025-04-09

**Title** Machine Learning Immunogenicity and Vaccine Response Analysis

**Author** Ivan Tomic [aut, cre, cph] (<https://orcid.org/0000-0003-3596-681X>),
Adriana Tomic [aut, ctb, cph, fnd]
(<https://orcid.org/0000-0001-9885-3535>),
Stephanie Hao [aut] (<https://orcid.org/0000-0002-3760-8234>)

**Description** Used for analyzing immune responses and predicting vaccine efficacy using machine learning and advanced data processing techniques. 'Immunaut' integrates both unsupervised and supervised learning methods, managing outliers and capturing immune response variability. It performs multiple rounds of predictive model testing to identify robust immunogenicity signatures that can predict vaccine responsiveness. The platform is designed to handle high-dimensional immune data, enabling researchers to uncover immune predictors and refine personalized vaccination strategies across diverse populations.

**Maintainer** Ivan Tomic <info@ivantomic.com>

**Imports** cluster, plyr, dplyr, caret, pROC, PRROC, stats, rlang, Rtsne, dbscan, FNN, igraph, fpc, mclust, ggplot2, grDevices, RColorBrewer, R.utils, clusterSim, parallel, doParallel

**Depends** R (>= 3.4.0)

**URL** https://github.com/atomiclaboratory/immunaut, <https://atomic-lab.org>

**BugReports** https://github.com/atomiclaboratory/immunaut/issues

**License** GPL-3

**Encoding** UTF-8

**LazyLoad** yes

**LazyData** yes

**RoxygenNote** 7.3.2.9000

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2025-04-09 17:10:02 UTC

# Contents



---

`auto_simon_ml` *Automated Machine Learning Model Building*

---

## Description

This function automates the process of building machine learning models using the caret package. It supports both binary and multi-class classification and allows users to specify a list of machine learning algorithms to be trained on the dataset. The function splits the dataset into training and testing sets, applies preprocessing steps, and trains models using cross-validation. It computes relevant performance metrics such as confusion matrix, AUROC (for binary classification), and prAUC (for binary classification).

## Usage

```
auto_simon_ml(dataset_ml, settings)
```

## Arguments

| | |
|---|---|
| `dataset_ml` | A data frame containing the dataset for training. All columns except the outcome column should contain the features. |
| `settings` | A list containing the following parameters: |

- `outcome`: A string specifying the name of the outcome column in `dataset_ml`. Defaults to "immunaut" if not provided.
- `excludedColumns`: A vector of column names to be excluded from the training data. Defaults to `NULL`.
- `preProcessDataset`: A vector of preprocessing steps to be applied (e.g., `c("center", "scale", "medianImpute")`). Defaults to `NULL`.
- `selectedPartitionSplit`: A numeric value specifying the proportion of data to be used for training. Must be between 0 and 1. Defaults to 0.7.
- `selectedPackages`: A character vector specifying the machine learning algorithms to be used for training (e.g., `"nb"`, `"rpart"`). Defaults to `c("nb", "rpart")`.

## Details

The function performs preprocessing (e.g., centering, scaling, and imputation of missing values) on the dataset based on the provided settings. It splits the data into training and testing sets using the specified partition, trains models using cross-validation, and computes performance metrics.

For binary classification problems, the function calculates AUROC and prAUC. For multi-class classification, it calculates macro-averaged AUROC, though prAUC is not used.

The function returns a list of trained models along with their performance metrics, including confusion matrix, variable importance, and post-resample metrics.

## Value

A list where each element corresponds to a trained model for one of the algorithms specified in `settings$selectedPackages`. Each element contains:

- `info`: General information about the model, including resampling indices, problem type, and outcome mapping.
- `training`: The trained model object and variable importance.
- `predictions`: Predictions on the test set, including probabilities, confusion matrix, post-resample statistics, AUROC (for binary classification), and prAUC (for binary classification).

## Examples

```
## Not run:
dataset <- read.csv("fc_wo_noise.csv", header = TRUE, row.names = 1)

# Generate a file header for the dataset to use in downstream analysis
file_header <- generate_file_header(dataset)

settings <- list(
    fileHeader = file_header,
    # Columns selected for analysis
    selectedColumns = c("ExampleColumn1", "ExampleColumn2"),
    clusterType = "Louvain",
    removeNA = TRUE,
    preProcessDataset = c("scale", "center", "medianImpute", "corr", "zv", "nzv"),
    target_clusters_range = c(3,4),
    resolution_increments = c(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5),
    min_modularities = c(0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9),
    pickBestClusterMethod = "Modularity",
    seed = 1337
)

result <- immunaut(dataset, settings)
dataset_ml <- result$dataset$original
dataset_ml$pandora_cluster <- tsne_clust[[i]]$info.norm$pandora_cluster
dataset_ml <- dplyr::rename(dataset_ml, immunaut = pandora_cluster)
dataset_ml <- dataset_ml[, c("immunaut", setdiff(names(dataset_ml), "immunaut"))]
settings_ml <- list(
    excludedColumns = c("ExampleColumn0"),
    preProcessDataset = c("scale", "center", "medianImpute", "corr", "zv", "nzv"),
```

```
    selectedPartitionSplit = split,  # Use the current partition split
    selectedPackages = c("rf", "RRF", "RRFglobal", "rpart2", "c5.0", "sparseLDA",
    "gcvEarth", "cforest", "gaussPRPoly", "monmlp", "slda", "spls"),
    trainingTimeout = 180  # Timeout 3 minutes
)
ml_results <- auto_simon_ml(dataset_ml, settings_ml)

## End(Not run)
```

---

`generate_demo_data` *Generate a Demo Dataset with Specified Number of Clusters and Overlap*

---

## Description

This function generates a demo dataset with a specified number of subjects, features, and desired number of clusters, ensuring that the generated clusters are not too far apart and have some degree of overlap to simulate real-world data. The generated dataset includes demographic information (`outcome`, `age`, and `gender`), as well as numeric features with a specified probability of missing values.

## Usage

```
generate_demo_data(
  n_subjects = 1000,
  n_features = 200,
  missing_prob = 0.1,
  desired_number_clusters = 3,
  cluster_overlap_sd = 15
)
```

## Arguments

| | |
|---|---|
| `n_subjects` | Integer. The number of subjects (rows) to generate. Defaults to 1000. |
| `n_features` | Integer. The number of features (columns) to generate. Defaults to 200. |
| `missing_prob` | Numeric. The probability of introducing missing values (NA) in the feature columns. Defaults to 0.1. |
| `desired_number_clusters` | Integer. The approximate number of clusters to generate in the feature space. Defaults to 3. |
| `cluster_overlap_sd` | Numeric. The standard deviation to control cluster overlap. Defaults to 15 for more overlap. |

### Details

The function generates `n_features` numeric columns based on Gaussian clusters with some overlap between clusters to simulate more realistic data. Missing values are introduced in each feature column based on the `missing_prob`.

### Value

A data frame containing the generated demo dataset, with columns:

- `outcome`: A categorical variable with values "low" or "high".
- `age`: A numeric variable representing the age of the subject (range 18-90).
- `gender`: A categorical variable with values "male" or "female".
- `Feature X`: Numeric feature columns with random values and some missing data.

### Examples

```
# Generate a demo dataset with 1000 subjects, 200 features, and 3 clusters
demo_data <- generate_demo_data(n_subjects = 1000, n_features = 200,
                                desired_number_clusters = 3,
                                cluster_overlap_sd = 15, missing_prob = 0.1)

# View the first few rows of the dataset
head(demo_data)
```

---

## `generate_file_header` *Generate a File Header*

---

### Description

This function generates a fileHeader object from a given data frame which includes original names and remapped names of the data frame columns.

### Usage

```
generate_file_header(dataset)
```

### Arguments

| | |
|---|---|
| `dataset` | The input data frame. |

### Value

A data frame containing original and remapped column names.

`immunaut` *Main function to carry out Immunaut Analysis*

## Description

This function performs clustering and dimensionality reduction analysis on a dataset using user-defined settings. It handles various preprocessing steps, dimensionality reduction via t-SNE, multiple clustering methods, and generates associated plots based on user-defined or default settings.

## Usage

```
immunaut(dataset, settings = list())
```

## Arguments

`dataset` A data frame representing the dataset on which the analysis will be performed. The dataset must contain numeric columns for dimensionality reduction and clustering.

`settings` A named list containing settings for the analysis. If NULL, defaults will be used. The settings list may contain:

**fileHeader** A data frame mapping the original column names to remapped column names. Used for t-SNE input preparation.

**selectedColumns** Character vector of columns to be used for the analysis. Defaults to NULL.

**cutOffColumnSize** Numeric. The maximum size of the dataset in terms of columns. Defaults to 50,000.

**excludedColumns** Character vector of columns to exclude from the analysis. Defaults to NULL.

**groupingVariables** Character vector of columns to use for grouping the data during analysis. Defaults to NULL.

**colorVariables** Character vector of columns to use for coloring in the plots. Defaults to NULL.

**preProcessDataset** Character vector of preprocessing methods to apply (e.g., scaling, normalization). Defaults to NULL.

**fontSize** Numeric. Font size for plots. Defaults to 12.

**pointSize** Numeric. Size of points in plots. Defaults to 1.5.

**theme** Character. The ggplot2 theme to use (e.g., "theme_gray"). Defaults to "theme_gray".

**colorPalette** Character. Color palette for plots (e.g., "RdPu"). Defaults to "RdPu".

**aspect_ratio** Numeric. The aspect ratio of plots. Defaults to 1.

**clusterType** Character. The clustering method to use. Options are "Louvain", "Hierarchical", "Mclust", "Density". Defaults to "Louvain".

**removeNA** Logical. Whether to remove rows with NA values. Defaults to FALSE.

**datasetAnalysisGrouped** Logical. Whether to perform grouped dataset analysis. Defaults to FALSE.

**plot_size** Numeric. The size of the plot. Defaults to 12.

**knn_clusters** Numeric. The number of clusters for KNN-based clustering. Defaults to 250.

**perplexity** Numeric. The perplexity parameter for t-SNE. Defaults to NULL (automatically determined).

**exaggeration_factor** Numeric. The exaggeration factor for t-SNE. Defaults to NULL.

**max_iter** Numeric. The maximum number of iterations for t-SNE. Defaults to NULL.

**theta** Numeric. The Barnes-Hut approximation parameter for t-SNE. Defaults to NULL.

**eta** Numeric. The learning rate for t-SNE. Defaults to NULL.

**clustLinkage** Character. Linkage method for hierarchical clustering. Defaults to "ward.D2".

**clustGroups** Numeric. The number of groups for hierarchical clustering. Defaults to 9.

**distMethod** Character. Distance metric for clustering. Defaults to "euclidean".

**minPtsAdjustmentFactor** Numeric. Adjustment factor for the minimum points in DBSCAN clustering. Defaults to 1.

**epsQuantile** Numeric. Quantile to compute the epsilon parameter for DBSCAN clustering. Defaults to 0.9.

**assignOutliers** Logical. Whether to assign outliers in the clustering step. Defaults to TRUE.

**excludeOutliers** Logical. Whether to exclude outliers from clustering. Defaults to TRUE.

**legendPosition** Character. Position of the legend in plots (e.g., "right", "bottom"). Defaults to "right".

**datasetAnalysisClustLinkage** Character. Linkage method for dataset-level analysis. Defaults to "ward.D2".

**datasetAnalysisType** Character. Type of dataset analysis (e.g., "heatmap"). Defaults to "heatmap".

**datasetAnalysisRemoveOutliersDownstream** Logical. Whether to remove outliers during downstream dataset analysis (e.g., machine learning). Defaults to FALSE.

**datasetAnalysisSortColumn** Character. The column used to sort dataset analysis results. Defaults to "cluster".

**datasetAnalysisClustOrdering** Numeric. The order of clusters for analysis. Defaults to 1.

**anyNAValues** Logical. Whether the dataset contains NA values. Defaults to FALSE.

**categoricalVariables** Logical. Whether the dataset contains categorical variables. Defaults to FALSE.

**resolution_increments** Numeric vector. The resolution increments to be used for Louvain clustering. Defaults to `c(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5)`.

**min_modularities** Numeric vector. The minimum modularities to test for clustering. Defaults to `c(0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9)`.

**target_clusters_range** Numeric vector. The range of acceptable clusters to identify. Defaults to `c(3, 6)`.

**pickBestClusterMethod** Character. The method to use for picking the best clustering result ("Modularity", "Silhouette", or "SIMON"). Defaults to "Modularity".

**weights** List. Weights for evaluating clusters based on `AUROC`, `modularity`, and `silhouette`. Defaults to `list(AUROC = 0.5, modularity = 0.3, silhouette = 0.2)`. These weights are applied to help choose the most relevant clusters based on user goals:

`AUROC` Weight for predictive performance (area under the receiver operating characteristic curve). Prioritize this when predictive accuracy is the main goal. For predictive analysis, a recommended configuration could be `list(AUROC = 0.8, modularity = 0.1, silhouette = 0.1)`.

`modularity` Weight for modularity score, which indicates the strength of clustering. Higher modularity suggests that clusters are well-separated. To prioritize well-separated clusters, use a configuration like `list(AUROC = 0.4, modularity = 0.4, silhouette = 0.2)`.

`silhouette` Weight for silhouette score, a measure of cohesion within clusters. Useful when cluster cohesion and interpretability are desired. For balanced clusters, a suggested configuration is `list(AUROC = 0.4, modularity = 0.3, silhouette = 0.3)`.

## Value

A list containing the following:

- `tsne_calc`: The t-SNE results object.
- `tsne_clust`: The clustering results.
- `dataset`: A list containing the original dataset, the preprocessed dataset, and a dataset with machine learning-ready data.
- `clusters`: The final cluster assignments.
- `settings`: The list of settings used for the analysis.

## Examples

```
data <- matrix(runif(2000), ncol=20)
settings <- list(clusterType = "Louvain",
resolution_increments = c(0.05, 0.1),
min_modularities = c(0.3, 0.5))
result <- immunaut(data.frame(data), settings)
print(result$clusters)
```

immunautDemo *Demo data set from immunaut package. This data is used in this package examples. It consist of 4x4 feature matrix + additional dummy columns that can be used for testing.*

## Description

Demo data set from immunaut package. This data is used in this package examples. It consist of 4x4 feature matrix + additional dummy columns that can be used for testing.

## Usage

```
data(immunautDemo)
```

## Format

An object of class `data.frame` with 4 rows and 7 columns.

## Examples

```
## Not run:
data(immunautDemo)
## define settings variable
settings <- list()
settings$fileHeader <- generate_file_header(immunautDemo)
# ... and other settings
results <- immunaut(immunautDemo, settings)

## End(Not run)
```

immunautLAIV *Demo data set from immunaut package. This data is used in this package examples.*

## Description

Demo data set from immunaut package. This data is used in this package examples.

## Usage

```
data(immunautLAIV)
```

## Format

An object of class `data.frame` with 244 rows and 32 columns.

## Examples

```
## Not run:
data(immunautLAIV)
## define settings variable
settings <- list()
settings$fileHeader <- generate_file_header(immunautLAIV)
# ... and other settings
results <- immunaut(immunautLAIV, settings)

## End(Not run)
```

---

`plot_clustered_tsne` *Plot Clustered t-SNE Results*

---

## Description

This function generates a t-SNE plot with cluster assignments using consistent color mappings. It includes options for plotting points based on their t-SNE coordinates and adding cluster labels at the cluster centroids. The plot is saved as an SVG file in a temporary directory.

## Usage

```
plot_clustered_tsne(info.norm, cluster_data, settings)
```

## Arguments

| | |
|---|---|
| `info.norm` | A data frame containing t-SNE coordinates (`tsne1`, `tsne2`) and cluster assignments (`pandora_cluster`) for each point. |
| `cluster_data` | A data frame containing the cluster centroids and labels, with columns `tsne1`, `tsne2`, `label`, and `pandora_cluster`. |
| `settings` | A list of settings for the plot, including:<br>• `theme`: The ggplot2 theme to use (e.g., `"theme_classic"`).<br>• `colorPalette`: The color palette to use for clusters (e.g., `"RdPu"`).<br>• `pointSize`: The size of points in the plot.<br>• `fontSize`: The font size used in the plot.<br>• `legendPosition`: The position of the legend (e.g., `"right"`).<br>• `plot_size`: The size of the plot.<br>• `aspect_ratio`: The aspect ratio of the plot. |

## Value

ggplot2 object representing the clustered t-SNE plot.

## Examples

```
## Not run:
# Example usage
plot <- plot_clustered_tsne(info.norm, cluster_data, settings)
print(plot)

## End(Not run)
```

# Index