Faculty of Mathematics and Information Sciences

WARSAW UNIVERSITY OF TECHNOLOGY

Deliverable 4

Adapting the gips library for classification problem utilizing discriminant analysis – gipsDA

Authors:
Norbert Frydrysiak
Antoni Kingston

Supervisors:
MSc Eng. Adam Chojecki
PhD Bartosz Kołodziejek, Assoc. Prof.

Version 1.0

December 10, 2025

Contents

1 Abstract

This document serves as the evaluation report for the gipsDA project, presenting a comprehensive validation of the developed R package. The assessment is conducted through a multi-faceted approach, encompassing automated unit testing, empirical model evaluation, and qualitative usability analysis.

First, we verify the technical reliability of the software. The unit tests confirm the mathematical correctness of the novel gipsmult backend and the robustness of the user-facing modules, with a specific focus on the newly implemented JSON serialization engine designed to facilitate model deployment. Usability analysis further validates the package's adherence to standard R conventions, ensuring a seamless experience for users familiar with the MASS library.

Second, we present a detailed performance assessment of the proposed classifiers: gipsLDA, gipsQDA, and gipsMultQDA. By leveraging five synthetic data generation scenarios and real-world benchmarks, we demonstrate that imposing permutation symmetry constraints acts as an effective form of regularization. The results highlight that gips-based models significantly outperform standard QDA in low-sample, high-dimensional settings, providing a robust alternative when data is scarce.

2 History of changes

Authors	Description	Version	Date
Norbert Frydrysiak	Initial structure of the Document.	0.1	04.12.2025
Norbert Frydrysiak	Modified sections: 'References to Deliverable 1, 2, 3' and 'Unit Tests'.	0.4	06.12.2025
Norbert Frydrysiak	Modified section 'Usability Analysis'.	0.5	07.12.2025
Norbert Frydrysiak, Antoni Kingston	Modified section 'Model Tests'.	1.0	09.12.2025

3 References to Deliverable 1, 2, 3

This document evaluates the implementation described in the previous deliverables. Regarding the software architecture presented in Deliverable 3, we have introduced a significant extension to the models module's helper functions to support model serialization. To enable the saving and loading of trained models—a critical feature for deployment—we implemented a custom JSON serialization engine.

Since standard JSON parsers do not natively support R-specific objects like formulas, language calls, or the custom gips_perm class, we developed specific utility functions to handle these types. The serialize_for_json and deserialize_from_json functions were added to explicitly manage type conversion and metadata (e.g., preserving the recursive depth of permutation objects). These utilities are exposed to the user through gipsDA_to_json and gipsDA_from_json, allowing for the seamless persistence of trained model objects to disk and their exact restoration for inference.

4 Unit Tests

To ensure the reliability and correctness of the gipsDA package, we implemented a comprehensive suite of automated unit tests using the tinytest [2] framework. The tests are organized into logical groups covering the core backend logic, model interfaces, and utility functions.

4.1 Testing Customized Code (Backend)

The core logic of the package, contained within the `gipsmult` module, was rigorously tested to ensure mathematical correctness and robust object handling.

- **Constructor and Validation:**
  - Verified that the `gipsmult()` constructor correctly initializes objects with attributes `Ss`, `numbers_of_observations`, and `D_matrices`.
  - Implemented strict validation tests to ensure the system throws specific errors (matching pattern `"cannot be created"`) when inputs are invalid, such as:
    - Mismatch between the number of covariance matrices and observation counts.
    - Matrices with inconsistent dimensions.
    - Non-square or rectangular matrices.
- **Optimization Engines:**
  - Tested the `find_MAP()` function across all supported optimizers: Brute Force (`"BF"`), Hill Climbing (`"HC"`), and Metropolis-Hastings (`"MH"`).
  - Verified that the returned objects contain the correct `optimization_info` structure, including convergence flags and the algorithm used.
  - Validated argument handling, ensuring errors are raised for invalid optimizers or missing `max_iter` parameters.
- **Posterior Probability Logic:**
  - Verified the calculation of the log-posterior via `log_posteriori_of_gipsmult()`, ensuring it returns finite numeric values.
  - Performed consistency checks between the internal wrapper `log_posteriori_of_perm()` and the exported `log_posteriori_of_gipsmult()` function.
  - Tested the extraction of posterior probabilities using `get_probabilities_from_gipsmult()`, confirming that the probabilities sum to 1 (within tolerance).

4.2 Testing Model Interfaces (Input-Output)

The user-facing models—`gipsLDA`, `gipsQDA`, and `gipsMultQDA`—were tested to ensure API consistency and correct integration with the standard R modeling paradigm.

- **Prediction Logic:**
  - Tested the `predict()` method for all models using the standard Iris dataset.
  - Validated the output structure, ensuring it contains class labels and posterior probabilities.
  - Verified that row-wise posterior probabilities sum to 1 (with a tolerance of 10^{-6}).
  - Confirmed that predictions work correctly on new subsets of data (e.g., predicting on just 5 rows).
- **Parameter Handling:**
  - Tested the pass-through of specific arguments such as `MAP` and `weighted_avg` to ensure the functions execute without errors under non-default configurations.

4.3 Utilities and Serialization

Additional tests were implemented to cover helper functions and file operations.

- **Serialization (JSON):**
  - Verified the full round-trip serialization process using `gipsDA_to_json()` and `gipsDA_from_json()`.
  - Confirmed that complex objects containing matrices and formulas are correctly restored from disk, maintaining their class structure and data integrity.
- **Projection Logic:**
  - Tested `project_covs()` for both `MAP = TRUE` (single permutation) and `MAP = FALSE` (weighted average).
  - Verified that `project_matrix_multiperm()` returns a matrix of the correct dimensions when applying weighted averaging based on posterior probabilities.
  - Validated edge cases, such as warning generation when no permutations exceed the probability tolerance threshold.

5 Model Tests

This section presents the empirical assessment of models trained with our proposed methodology. The tests were conducted on the hardware environments specified in Deliverable 3 (Apple M4 Max and AMD Ryzen 7).

5.1 Assessment Methodology

Models were evaluated using **Accuracy** metric. For the multi-class classification problem, Accuracy is defined as the ratio of correct predictions to the total number of observations. Let N be the total number of samples in the dataset, y_i be the true class label for the i-th observation, and $\hat{y}_i$ be the predicted class label. The accuracy is calculated as:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{(y_i=\hat{y}_i)}, \tag{1}$$

where $\mathbf{1}_{(\cdot)}$ is the indicator function, which takes the value 1 if the condition inside is true (i.e., the prediction is correct) and 0 otherwise. This metric provides a global measure of the model's performance across all K classes. For synthetic data, performance was evaluated across varying sample sizes using 80 logarithmically spaced intervals, starting from a lower bound of 18 observations. At each step, results were averaged over 20 repetitions on shuffled subsets. Similarly, for real-world data, the analysis was conducted over 60 logarithmically spaced sample sizes, starting from 10 observations. It is worth mentioning that accuracy of $\frac{1}{K}$ can be achieved by a random uniform model (assuming balance between classes).

5.2 Synthetic Data Results

We evaluated the models on the five synthetic data scenarios defined in Deliverable 2. For all scenarios, the data generation parameters were fixed at $p = 5$ features and $k = 6$ classes, with a sample size of $n_g = 150$ observations per class. These scenarios were designed to systematically test the models against specific ground-truth assumptions regarding covariance structure and symmetry:

Scenario gipsLDA (Homoscedastic / Symmetric): Data is generated assuming a single, common covariance matrix shared across all classes, which possesses the specific permutation symmetry (12)(35). This targets the `gipsLDA` model.

Scenario gipsMultQDA (Heteroscedastic / Shared Symmetry): Each class has a distinct covariance matrix (different scales/variances), but all matrices share the same underlying permutation symmetry group defined by (12)(35). This targets the `gipsMultQDA` model.

Scenario gipsQDA (Heteroscedastic / Unique Symmetry): The most flexible setting where each class has a unique covariance matrix with its own distinct permutation symmetry. The permutations assigned to the six classes are: (15342), (14)(25), (123), (135)(24), (1543), and (12)(345). This targets the `gipsQDA` model.

Scenario LDA (Classic Baseline): A standard homoscedastic scenario with a common covariance matrix, but with no specific permutation symmetry (unstructured).

Scenario QDA (Classic Baseline): A standard heteroscedastic scenario with distinct, unstructured covariance matrices for each class.

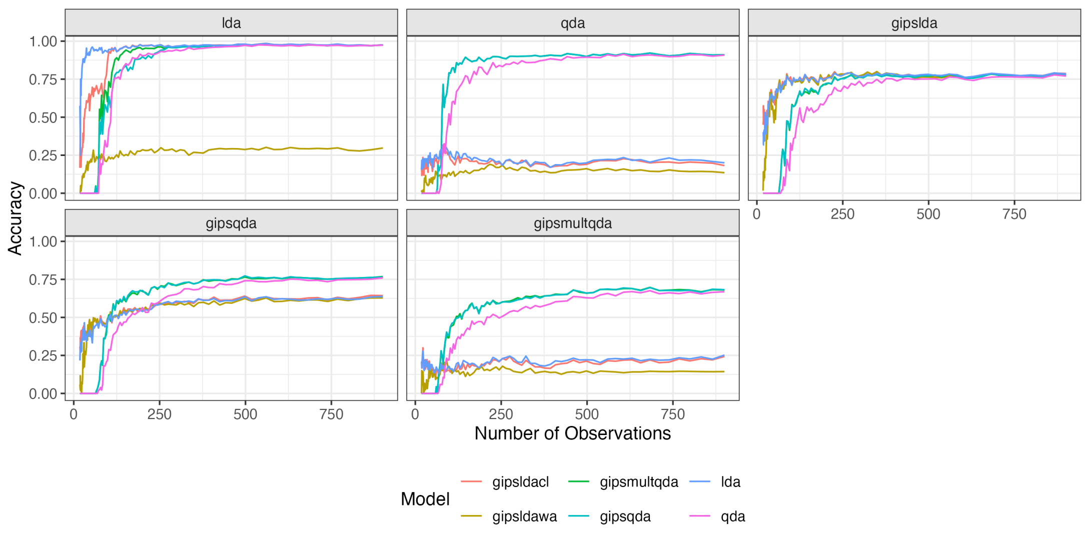

Figure 1: Performance comparison measured in classification accuracy across the five synthetic scenarios. Each subplot represents a distinct scenario as indicated by the facet title, while the specific models are identified in the legend at the bottom. All `gips`-based models are configured to utilize **the single most probable permutation (MAP estimation)**. An accuracy value of 0 indicates that the model failed to fit due to insufficient data.

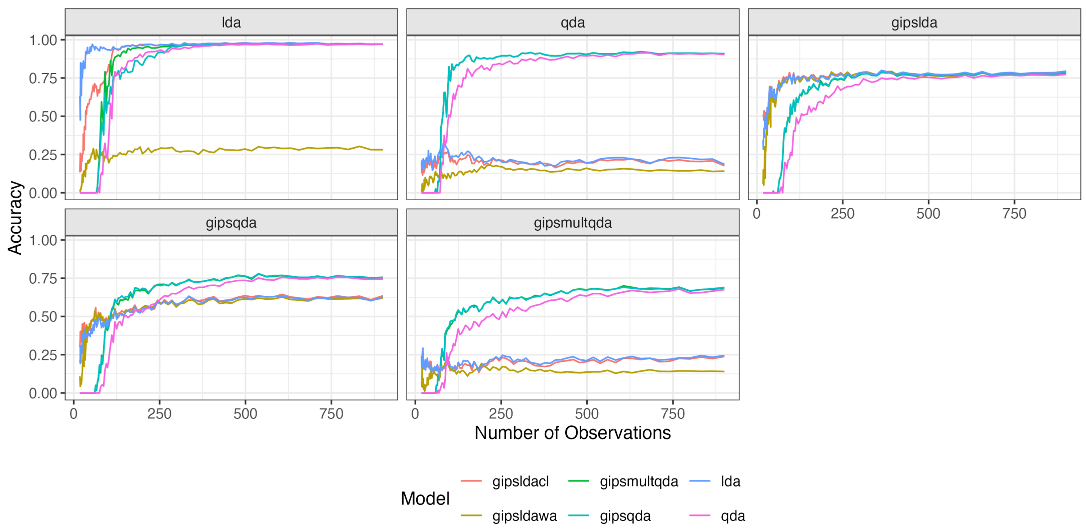

Figure 2: Performance comparison measured in classification accuracy across the five synthetic scenarios. Each subplot represents a distinct scenario as indicated by the facet title, while the specific models are identified in the legend at the bottom. All `gips`-based models are configured to utilize a weighted average estimator, where the projected covariance matrices are **weighted by the posterior probabilities of their corresponding permutation symmetries**. An accuracy value of 0 indicates that the model failed to fit due to insufficient data.

Major Results Review:

- **Convergence with Sample Size:** A consistent trend across all scenarios is that as the volume of training data increases, the performance gap between standard models and `gips`-based models narrows. With sufficient data, standard LDA and QDA are able to estimate the covariance matrices accurately enough to achieve satisfactory performance, diminishing the relative advantage of the regularization provided by `gips`.

- **Performance in Symmetry-Based Scenarios:** In the scenarios designed for `gipsMultQDA` and `gipsQDA`, both `gips`-based models perform best and exhibit very similar accuracy profiles. As expected, as the sample size grows, the standard QDA model eventually catches up to the `gipsLDA` and `gipsMultQDA` variants in terms of accuracy.

- **Regularization Benefit in Data-Scarce Regimes:** A highly encouraging result is observed in the scenario designed for standard QDA. Even though this scenario does not strictly enforce a permutation symmetry, `gipsQDA` and `gipsMultQDA` significantly outperform standard QDA when the sample size is small. This confirms that the symmetry imposition acts as an effective form of model-based regularization, preventing the overfitting that typically plagues QDA in high-dimensional, low-sample settings.

- **Comparison of `gipsLDA` Variants:** We observed a significant performance disparity between the two proposed pooling strategies for `gipsLDA`. The `weighted_average` variant frequently performed poorly, yielding results significantly inferior to the `classic` (unbiased pooled) approach across multiple scenarios.

- **Homoscedastic Scenario Performance:** In the scenario designed for the `gipsLDA` model, the results were surprisingly uniform. `gipsLDA classic`, `gipsLDA weighted average`, and standard LDA achieved nearly identical accuracy. This suggests that when the homoscedasticity assumption holds strong, the standard pooled estimator is already highly efficient, and the additional symmetry constraint provides marginal gain in this specific setup.

5.3 Real-World Datasets

The models were applied to heart failure [1] and breast cancer [3] datasets.

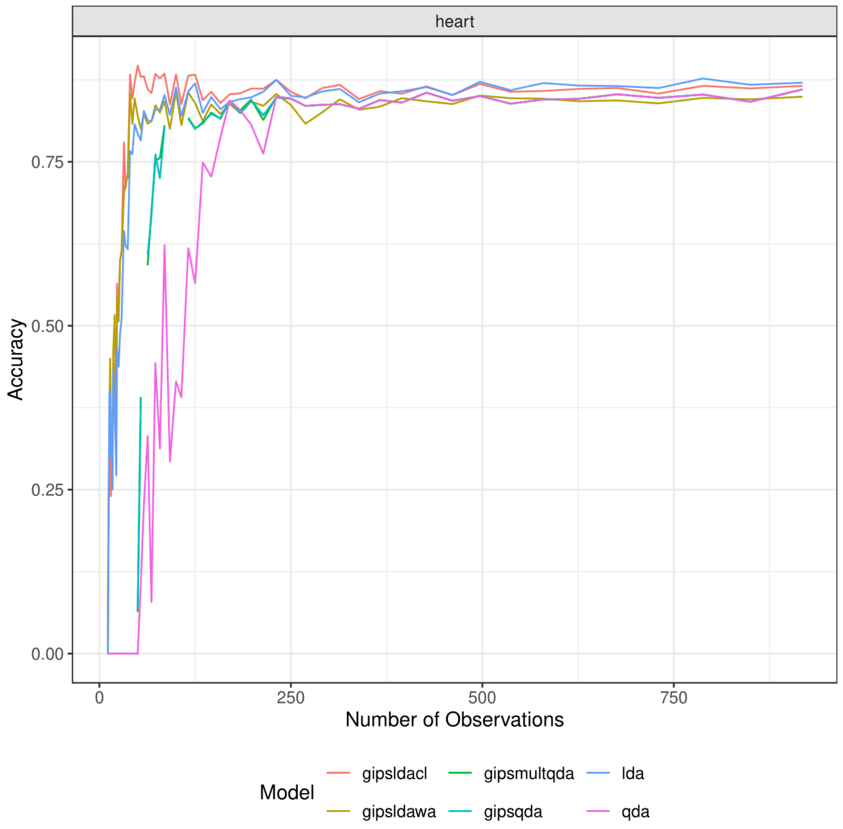

Figure 3: Performance comparison measured in classification accuracy. All `gips`-based models are configured to utilize a weighted average estimator, where the projected covariance matrices are weighted by the posterior probabilities of their corresponding permutation symmetries. An accuracy value of 0 indicates that the model failed to fit due to insufficient data.

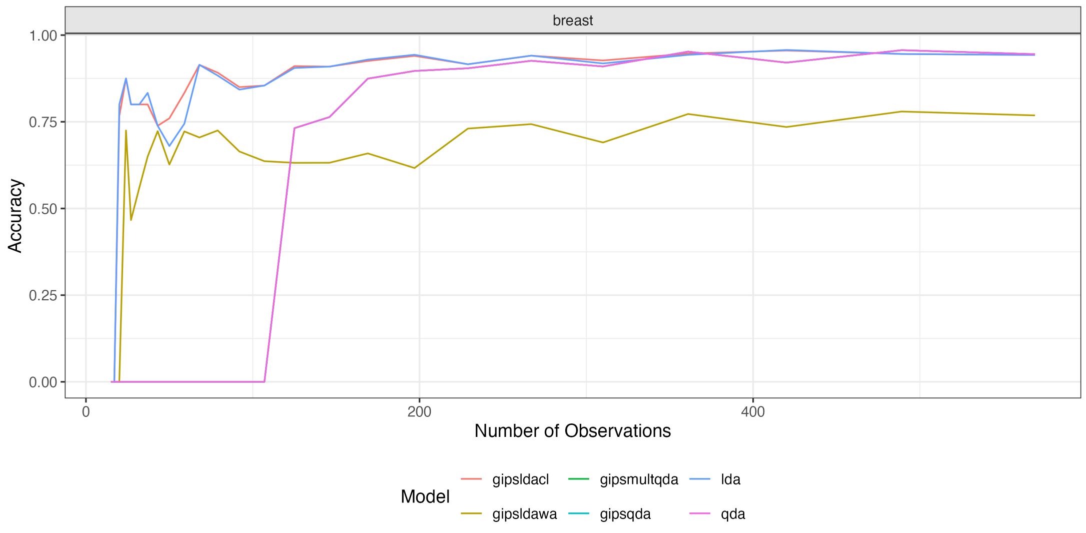

Figure 4: Performance comparison measured in classification accuracy. All `gips`-based models are configured to utilize a weighted average estimator, where the projected covariance matrices are weighted by the posterior probabilities of their corresponding permutation symmetries. An accuracy value of 0 indicates that the model failed to fit due to insufficient data.

Results Review:

- **Convergence at Scale:** Consistent with synthetic results, we observe that for sufficiently large data volumes, nearly all models converge to a satisfying level of accuracy. The notable exception is the `gipsLDA weighted average` variant, which tends to underperform even as sample sizes increase.
- **Data Efficiency (Heart Failure):** In the Heart Failure dataset, the `gips`-based models demonstrate superior data efficiency, reaching their accuracy plateau significantly faster (i.e., at smaller sample sizes) than their non-`gips` counterparts.
- **Low-Data Stability:** Specifically within the Heart Failure analysis, the `gipsLDA classic` model performs remarkably well in extremely low-data regimes, offering stable predictions where other models exhibit high variance.
- **Sample Size Thresholds (Breast Cancer):** A striking contrast is observed in the Breast Cancer dataset. Both `gipsLDA classic` and standard `LDA` achieve robust performance with as few as 30 observations. In comparison, the standard `QDA` model requires approximately 200 observations to reach a comparable accuracy level, underscoring the benefit of the parsimonious parameter estimation in our proposed models.

5.4 Identified Issues and Anomalies

During the analysis of the experimental results, several anomalies were observed that require further investigation and potential code refinement:

- **Underperformance of gipsLDA Weighted Average:** In the synthetic data benchmarks, the `gipsLDA weighted average` variant (labeled `gipsldawa` in the plots) consistently underperforms compared to the `gipsLDA classic` variant (`gipsldacl`). Theoretically, these two pooling strategies should yield comparable results. This significant discrepancy suggests a potential implementation defect in the weighted average logic, specifically in how the individual class covariance matrices are combined or projected.

- **Discontinuities in Heart Failure Experiments:** The performance curves for `gipsMultQDA` and `gipsQDA` in the Heart Failure dataset analysis exhibit noticeable gaps and discontinuities (broken lines). This behavior likely originates from the experimental framework rather than the package itself. We hypothesize that the random subsampling procedure, combined with the dataset's class imbalance, occasionally produced training subsets where specific classes were insufficiently represented, leading to model fitting failures in those specific iterations. This necessitates further testing and refinement of the data partitioning strategy to ensure robust evaluation.
- **Model Overlap in Breast Cancer Dataset:** In the Breast Cancer dataset results, the performance lines for `gipsQDA`, `gipsMultQDA` and standard `QDA` overlap perfectly, rendering them indistinguishable (missing lines in the plot). This is unexpected and requires deep verification. A primary hypothesis is that the `gips` optimization process is consistently identifying the Identity permutation (trivial symmetry) as the optimal structure. Consequently, the projected covariance matrices would remain identical to the empirical ones, causing the `gips`-based models to mathematically reduce to the standard QDA model.

6 Usability Analysis

We performed a qualitative usability analysis based on human-driven tests. The goal was to validate the "Supportability" and "Usability" non-functional requirements defined in Deliverable 1, ensuring the package effectively serves both Data Scientist and Machine Learning Engineer actors.

- **API Consistency and Familiarity:** The package successfully mimics the interface of the standard `MASS` library. Users familiar with `MASS::lda` found the transition to `gipsDA` seamless. The function signatures (e.g., `gipsLDA(formula, data)`) and the structure of the returned objects match standard R conventions. Crucially, the `predict()` method behaves identically to established standards, returning class labels and posterior probabilities without requiring the user to learn a new syntax.
- **Introspection and Interpretability:** To aid in exploratory data analysis, the `print()` methods were customized to provide immediate insight into the model's internal logic. Unlike standard black-box models, `gipsDA` clearly displays the "Found Permutation" and "Log-Posterior" values in the console output. This allows the Data Scientist to immediately verify if the discovered symmetries make intuitive sense for the given problem domain.
- **Serialization and Deployment Workflow:** Addressing the specific requirement for the Machine Learning Engineer actor, we validated the full end-to-end deployment cycle. We successfully trained a model in one R session, saved it to a JSON file using the newly implemented `gipsDA_to_json()` function, and restored it in a completely fresh session using `gipsDA_from_json()`. The restored model retained all learned parameters and successfully performed predictions on new data, confirming the library's readiness for production integration.
- **Error Handling and Stability:** We conducted "negative testing" by intentionally providing invalid inputs (e.g., non-numeric matrices, mismatched dimensions, or invalid optimizer names). The system consistently caught these exceptions and returned informative, human-readable error messages (e.g., *"Input x must be numeric"*) rather than generic R execution errors. This facilitates faster debugging for the user.
- **Performance and Responsiveness:** We observed that for datasets with a small number of features ($p \leq 9$), where the exhaustive Brute Force search is applicable, the model

fitting process is immediate and efficient. This performance consistency was verified on both testing platforms (Apple Silicon and Linux x86_64), meeting the requirement for interactive use and allowing researchers to iterate quickly during the model prototyping phase.

- **Documentation:** All exported functions are documented using the standard `roxygen2` format. Users reported that the built-in help pages (accessible via for instance `?gipsLDA`) provided sufficient information, including parameter descriptions and runnable examples, to use the package without external guidance.

References

[1] fedesoriano. Heart failure prediction dataset. https://www.kaggle.com/fedesoriano/heart-failure-prediction, September 2021. Kaggle dataset. Accessed: 2025-12-05.

[2] MPJ van der Loo. A method for deriving information from running r code. *The R Journal*, page Accepted for publication, 2020.

[3] Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.