Faculty of Mathematics and Information Sciences

WARSAW UNIVERSITY OF TECHNOLOGY

# Deliverable 2

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

*Authors:*
Norbert Frydrysiak
Antoni Kingston

*Supervisors:*
Mgr inż. Adam Chojecki
Dr hab. prof. uczelni Bartosz Kołodziejek

Version 1.0

November 15, 2025

## Contents

## 1 Abstract

Classical discriminant analysis methods, such as Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), are powerful classification tools which performance is critically dependent on the estimation of class covariance matrices. This estimation can be challenging in high-dimensional settings. This work introduces a novel family of classifiers, termed `gipsDA`, that enhances these methods by incorporating permutation group symmetries into the covariance estimation process. By leveraging the methodology of the `gips` R package, we impose structural constraints on the covariance matrices, assuming they are invariant under a specific permutation group. This approach acts as a form of model-based regularization, reducing the number of parameters to be estimated and potentially leading to more robust models.

We propose and describe four distinct models within this framework: `gipsQDA`, `gipsMultQDA`, and two variants of `gipsLDA` (`classic` and `weighted average`). These models offer a spectrum of flexibility, from a fully class-specific symmetry structure (`gipsQDA`) to a more constrained model with a shared symmetry across all classes (`gipsMultQDA`). The latter is facilitated by a proposed extension to the `gips` methodology, termed `gipsmult`. The performance of these models will be evaluated through a comprehensive simulation study based on five synthetic data generation scenarios, as well as on a diverse collection of real-world benchmark datasets from various domains. This document details the theoretical foundation of the proposed models, the structure of the accompanying R package, and the experimental design for their validation.

## 2 History of changes

| **Authors** | **Description** | **Version** | **Date** |
|---|---|---|---|
| **Antoni Kingston** | Initial draft of the document structure and content outline. | **0.2** | **04.11.2025** |
| **Antoni Kingston** | Added a detailed description of the solution proposal. | **0.4** | **05.11.2025** |
| **Norbert Frydrysiak** | Created the 'Data Report' section. | **0.5** | **09.11.2025** |
| **Norbert Frydrysiak** | Created sections: 'GUI Design', 'Initial Data Preparation', and 'Modules Description'. | **0.7** | **10.11.2025** |
| **Norbert Frydrysiak** | Made visual improvements and added the bibliography. | **0.8** | **11.11.2025** |
| **Norbert Frydrysiak** | Modified the 'Modules Description', 'Data Reports', and 'Solution Proposal' sections. | **0.9** | **14.11.2025** |
| **Antoni Kingston** | Modified the 'Solution Proposal' sections. | **1.0** | **15.11.2025** |

## 3 Solution Proposal

The central theme of our proposed solution is the enhancement of classical discriminant analysis methods by introducing a novel approach to estimating covariance matrices. We modify the standard Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) algorithms by leveraging the methodology of the `gips` R package [1]. The core function of the `gips` library is to identify the most probable permutation group under which a given covariance matrix remains invariant. By imposing such a symmetry constraint, we effectively introduce a form of model-based regularization. This reduces the number of free parameters that need to be estimated, which can be particularly advantageous when dealing with high-dimensional data or limited sample sizes, potentially leading to more stable and robust classifiers.

### 3.1 LDA and QDA

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are generative classifiers that model the probability density of each class, $g$, using a multivariate Gaussian distribution, $p(x|g) = \mathcal{N}(x|\mu_g, \Sigma_g)$. The key difference between them lies in their assumptions regarding the class covariance matrices, $\Sigma_g$.

- **Linear Discriminant Analysis (LDA)** assumes that all classes share a single, common covariance matrix, i.e., $\Sigma_g = \Sigma$ for all $g$. This assumption of homoscedasticity leads to a decision boundary that is a linear function of the input features $x$.

- **Quadratic Discriminant Analysis (QDA)** relaxes this assumption, allowing each class $g$ to have its own distinct covariance matrix, $\Sigma_g$. This assumption of heteroscedasticity results in a decision boundary that is a quadratic function of $x$, providing greater flexibility.

Prediction for a new observation $x$ is made by assigning it to the class $g$ that maximizes the posterior probability $P(g|x)$. Using Bayes' theorem, this is equivalent to maximizing the discriminant function, $\delta_g(x)$:

$$\delta_g(x) = -\frac{1}{2}\log|\Sigma_g| - \frac{1}{2}(x-\mu_g)^T\Sigma_g^{-1}(x-\mu_g) + \log\pi_g \tag{1}$$

where $\mu_g$ is the mean vector for class $g$, $\Sigma_g$ is its covariance matrix, and $\pi_g$ is the prior probability of the class. In practice, these parameters are estimated from the training data. For a more detailed treatment of these methods, we refer the reader to "The Elements of Statistical Learning" [3]. Our work focuses on novel estimation techniques for the $\Sigma_g$ matrices.

### 3.2 gipsQDA

The `gipsQDA` model represents the most flexible approach within our proposed framework. In this method, the `gips` library is applied independently to the data of each class. This process yields a unique estimated covariance matrix, $\hat{\Sigma}_g$, and a unique optimal permutation group, $\hat{P}_g$, for each class $g$.

$$(\hat{\Sigma}_g, \hat{P}_g) = \texttt{gips}(\text{Data}_g) \quad \text{for each class } g = 1,\dots,m \tag{2}$$

This model is directly analogous to the classic QDA, as it allows for heteroscedasticity, but with the additional complexity that each class can exhibit its own distinct symmetry structure. Consequently, `gipsQDA` serves as a general model that contains all other proposed models as special, more constrained cases.

### 3.3 gipsLDA

The `gipsLDA` model adapts the core idea of LDA—a single, shared covariance matrix—to the `gips` framework. The general strategy is to first pool the covariance information from all classes into a single matrix, and then use `gips` to find a common symmetry structure for this pooled matrix. The process is as follows:

1. For each class $g$, estimate the sample covariance matrix, $S_g$, using the standard unbiased estimator:

$$S_g = \frac{1}{n_g-1}\sum_{i=1}^{n_g}(x_i^{(g)} - \overline{x}^{(g)})(x_i^{(g)} - \overline{x}^{(g)})^T \tag{3}$$

   where $n_g$ is the number of observations in class $g$, and $x_i^{(g)}$ and $\overline{x}^{(g)}$ are the $i$-th observation and the sample mean of class $g$, respectively.

2. Combine the individual $S_g$ matrices into a single pooled matrix, $S$, using a specific averaging method.

3. Supply the pooled matrix $S$ to the `gips` algorithm to find the optimal permutation group $\hat{P}$ and the projected covariance matrix $\hat{\Sigma}_{\text{gipsLDA}}$.

4. Use the resulting matrix $\hat{\Sigma}_{\text{gipsLDA}}$ as the shared covariance matrix in the LDA classification framework, assuming each class has a different mean vector $\mu_g$.

We propose two methods for pooling the covariance matrices.

#### 3.3.1 gipsLDA weighted average

This method is named "weighted average" because the contribution of each class's covariance matrix, $S_g$, to the final pooled matrix is weighted by its sample size, $n_g$. This gives more influence to classes with more data. The pooled matrix $S$ is calculated as:

$$S = \frac{1}{n}\sum_{g=1}^{m} n_g S_g \tag{4}$$

where $n = \sum_{g=1}^{m} n_g$ is the total number of observations and $m$ is the number of classes.

### 3.3.2 gipsLDA classic

This method uses the classic pooled covariance estimator, which is the standard approach in traditional LDA. The pooled matrix $S$ is defined as:

$$S = \frac{1}{n-m} \sum_{g=1}^{m} (n_g - 1) S_g \tag{5}$$

By substituting the formula for $S_g$ from Equation (3), this simplifies to the well-known pooled sample covariance matrix formula:

$$S = \frac{1}{n-m} \sum_{g=1}^{m} \sum_{i=1}^{n_g} (x_i^{(g)} - \overline{x}^{(g)})(x_i^{(g)} - \overline{x}^{(g)})^T \tag{6}$$

This estimator is an unbiased estimate of the common covariance matrix under the assumption of homoscedasticity.

## 3.4 gipsMultQDA

The `gipsMultQDA` model is an intermediate approach between the full flexibility of `gipsQDA` and the strong constraints of `gipsLDA`. The goal is to allow each class to have a different covariance matrix (like QDA), but enforce the constraint that all these matrices must share the same underlying permutation symmetry. This is useful for scenarios where classes might differ in scale or variance but are expected to share a common dependency structure.

To achieve this, we require a modification of the standard `gips` library, which we term `gipsmult`. This extension is designed to find a single, common permutation group that is most probable across multiple covariance matrices simultaneously.

### 3.4.1 gipsmult

Let us introduce some definitions

$$\mathcal{Z}_\Gamma := \left\{ S \in Sym(p;\mathbb{R}) : S_{ij} = S_{\sigma(i)\sigma(j)} \; for \; all \; \sigma \in \Gamma \right\} \tag{7}$$

$$\mathcal{P}_\Gamma := Z_\Gamma \cap Sym^+(p;\mathbb{R}) \tag{8}$$

The `gipsmult` methodology addresses the problem of finding a common symmetry structure across multiple groups of data. While the standard `gips` algorithm operates on a single covariance matrix, `gipsmult` is designed to take multiple covariance matrices as input. Its objective is to estimate a set of distinct covariance matrices, $\{\Sigma_1, \ldots, \Sigma_m\}$, under the constraint that they are all invariant under the same, shared permutation group, $\Gamma$ i.e. $\forall_{g\in[m]}\Sigma_g \in \mathcal{P}_\Gamma$. This allows for heteroscedasticity across classes while preserving a common structural assumption, providing a powerful trade-off between model flexibility and parameter parsimony. The detailed statistical formulation of this model is a key component of our research. Its main part being the reasoning for a single class is more deeply covered in [1].

We examine m multidimensional gaussian samples $Z_g^{(1)}, \ldots Z_g^{(n_g)}$ under set $\{\mathcal{K}_g = \kappa_g, \Gamma = c\}$ consisting of i.i.d. random vectors $\mathcal{N}_p(0, \kappa_g^{-1})$. Let $\Gamma$ be a discrete random variable uformly distributed over the set of $\mathcal{C} := \{\langle\sigma\rangle : \sigma \in \mathfrak{S}_p\}$ cyclic subgroups of $\mathfrak{S}_p$. We assume that each $\mathcal{K}_g$ under set $\Gamma = c$ follows conjugate Diaconis-Ylvisacker a priori distribution [2] defined by its PDF:

$$f_{\mathcal{K}_g|\Gamma=c}(k_g) = \frac{1}{I_c(\delta, D)} \operatorname{Det}(k_g)^{(\delta-2)/2} e^{-\frac{1}{2}\operatorname{Tr}[D^{-1}k_g]} \mathbf{1}_{\mathcal{P}_c}(k_g) \tag{9}$$

Where $\delta > 1$ and $D \in \mathcal{P}_c$ are hyperparameters and $I_c(\delta, D)$ is a normalizing constant We start with standard bayesian reasoning linking the probability of any given permutation conditioned by data with probability of specific data conditioned by given permutation:

$$\mathbb{P}(\Gamma = c \mid \mathbb{X} = \mathbb{x}, \mathbb{Y} = \mathbb{y}) = \frac{\mathbb{P}(\mathbb{X} = \mathbb{x} \mid \Gamma = c, \mathbb{Y} = \mathbb{y})}{\mathbb{P}(\mathbb{X} = \mathbb{x}, \mathbb{Y} = \mathbb{y})} \propto \mathbb{P}(\mathbb{X} = \mathbb{x} \mid \Gamma = c, \mathbb{Y} = \mathbb{y}) \tag{10}$$

Writing this in more probabilistic terms and conditioning $\mathbb{X}$ directly on precision matrices we get:

$$\mathbb{P}(\Gamma = c \mid \mathbb{X} = \mathbb{x}, \mathbb{Y} = \mathbb{y}) \propto f_{\mathbb{X} \mid \Gamma=c,\, \mathbb{Y}=\mathbb{y}}(\mathbb{x}) = \underset{\mathcal{P}_c^m}{\int \cdots \int} f_{\mathbb{X}|\mathcal{K}=\kappa, \mathbb{Y}=\mathbb{y}}(\mathbb{x})\, f_{\mathcal{K}|\Gamma=c}(\kappa)\, d\kappa \tag{11}$$

After some not to complex but pretty lengthy transformations we arrive at the result:

$$\mathbb{P}\left(\Gamma = c \mid \mathbb{X} = \mathbb{x}, \mathbb{Y} = \mathbb{y}\right) \propto \prod_{g=1}^{m} \frac{I_c\left(\delta + n_g, D + U_g\right)}{I_c\left(\delta, D\right)} \tag{12}$$

Where

$$U_g = \sum_{i=1}^{n_g} Z_g^{(i)} \left(Z_g^{(i)}\right)^T \tag{13}$$

Calculating factors of product on the RHS of (12) is something gips library already manages so we will refrain from examining it.

## 4 Data Reports

### 4.1 Synthetic Data

To rigorously evaluate the performance of the proposed classification models, a comprehensive simulation study was designed. This involved generating synthetic datasets with controlled and well-defined properties. The data generation process was structured to systematically explore different assumptions about the underlying covariance structures of the classes. The generation process consists of the following steps:

1. **Mean Vector Sampling:** For each of the $k$ classes, a mean vector $\mu_k$ is sampled as a random point from a $p$-dimensional hypercube, namely $[0,1]^p$.

2. **Scale Matrix Generation:** The process begins by creating a scale matrix, denoted as $V$, for the Wishart distribution. A random $p \times p$ matrix $A$ is generated from a standard normal distribution, and $S$ is computed as $V = A^T A$, ensuring it is positive semi-definite. This matrix serves as the basis for generating structured covariance matrices.

3. **Covariance Matrix Sampling:** Using the scale matrix $V$, one or more covariance matrices $\Sigma$ are drawn from a Wishart distribution, $W_p(V, \nu)$, where $p$ is the data dimensionality and $\nu$ are the degrees of freedom.

4. **Covariance Matrix Projection (for gips models):** For scenarios involving the `gips` methodology, the sampled covariance matrices are projected onto a specific permutation group structure. This step imposes predefined symmetries onto the covariance structure, which is a core feature of the `gips` models.

5. **Separability Control:** To ensure a consistent level of classification difficulty across all simulation scenarios, a parameter, denoted by $\psi$, is introduced to control the degree of overlap between classes. This parameter directly scales the base covariance matrix ($\Sigma_{\text{base}}$), where smaller values of $\psi$ reduce the variance, making classes more compact and separable. The optimal value of $\psi$ is determined through an automated, iterative search procedure. The search begins with a predefined initial value, $\psi_{\text{initial}}$. In each iteration, $i$, a new candidate value is calculated as $\psi_{\text{new}} = \frac{\psi_{\text{initial}}}{2^i}$. The base covariance matrix is then scaled by this new value: $\Sigma_{\text{scaled}} = \Sigma_{\text{base}} \cdot \psi_{\text{new}}$. This process is repeated until the generated dataset allows a baseline LDA model or QDA model to achieve a predefined target accuracy. This method guarantees that each scenario is calibrated to a comparable level of difficulty before the main models are evaluated.

6. **Data Generation:** Finally, for each class $k$, $n_k$ data points are sampled from a multivariate normal distribution $\mathcal{N}_p(\mu_k, \Sigma_k)$.

Based on this framework, five distinct data generation scenarios were defined to test the models under different assumptions:

**Scenario 1: gipsLDA Model** This model assumes a single covariance matrix and a single permutation structure shared across all classes. It is analogous to the classic LDA assumption of homoscedasticity but with an added layer of permutation symmetry.

**Scenario 2: gipsMultQDA Model** In this scenario, each class possesses a unique covariance matrix, but all matrices share the same underlying permutation structure. This allows for different class shapes while maintaining a common dependency model.

**Scenario 3: gipsQDA Model** This is the most flexible `gips`-based model, where each class has a distinct covariance matrix and a distinct permutation structure. It corresponds to the QDA assumption of heteroscedasticity, with each class having its own unique symmetry.

**Scenario 4: Classic LDA Model** This serves as a baseline, generated under the standard LDA assumption of a single, shared covariance matrix for all classes, with no permutation projection.

**Scenario 5: Classic QDA Model** This second baseline is generated under the standard QDA assumption, where each class has its own unique covariance matrix, with no permutation projection.

## 4.2 Real-World Datasets

In addition to synthetic data, the models will be evaluated on a collection of real-world datasets sourced from public repositories. These datasets span various domains, including medical diagnostics, finance, and industrial quality control, providing a diverse set of challenges in terms of dimensionality, class balance, and complexity.

**Heart Failure Prediction Dataset**
- *Description:* This dataset contains clinical features that can be used to predict whether a patient has heart disease. It is a classic binary classification problem.
- *Target Variable:* `HeartDisease` (1 for heart disease, 0 for normal).
- *Details:* 11 features. Class distribution: Normal (410), Heart Disease (508).
- *Link:* https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

**Breast Cancer Wisconsin (Diagnostic) Data Set**
- *Description:* The features in this dataset are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The task is to classify tumors as malignant or benign.
- *Target Variable:* `diagnosis` (M = malignant, B = benign).
- *Details:* 30 features. Class distribution: Benign (357), Malignant (212).
- *Link:* https://archive.ics.uci.edu/dataset/17/breast-cancer-wisconsin-diagnostic

**EEG Brainwave Dataset: Feeling Emotions**
- *Description:* This dataset contains EEG brainwave data that has been processed using statistical feature extraction. The data was collected from two individuals for three minutes per emotional state: positive, neutral, and negative.
- *Target Variable:* `emotion` (Positive, Neutral, Negative).
- *Details:* 2548 features. Class distribution: Neutral (716), Positive (708), Negative (708).
- *Link:* https://www.kaggle.com/datasets/birdy654/eeg-brainwave-dataset-feeling-emotions

**Credit Card Fraud Detection**
- *Description:* This dataset contains credit card transactions, where the challenge is to identify fraudulent transactions, which constitute a very small fraction of all transactions.
- *Target Variable:* `Class` (1 for fraud, 0 for a legitimate transaction).
- *Details:* 30 features. Class distribution: Legitimate (284315), Fraud (492).
- *Link:* https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

**Steel Plates Faults Dataset**
- *Description:* This dataset is from experiments on steel plates to identify different types of faults. This is a multi-class classification problem.
- *Target Variable:* 7 different types of faults (e.g., `Pastry`, `Z_Scratch`).
- *Details:* 27 features. Class distribution: Pastry (158), Z_Scratch (190), K_Scatch (391), Stains (72), Dirtiness (55), Bumps (402), Other_Faults (673).
- *Link:* https://archive.ics.uci.edu/dataset/198/steel-plates-faults

**Sensorless Drive Diagnosis Data Set**
- *Description:* Data was collected from a sensorless drive to diagnose its condition based on features from the current signal.
- *Target Variable:* The drive's condition (11 different fault classes).
- *Details:* 48 features. Class distribution: Balanced, with 5,319 instances for each of the 11 classes.
- *Link:* https://archive.ics.uci.edu/dataset/325/sensorless+drive+diagnosis

# 5 Modules Description

The project will be structured into two primary modules, each with a distinct responsibility.

## 5.1 gipsmult Module

This module extends the core functionality of the `gips` library. Its primary purpose is to enable the estimation of different covariance matrices for each class under the constraint that they all share the same underlying permutation symmetry. This is designed for scenarios where the classes may have different scales or variances but are assumed to possess a common dependency structure.

## 5.2 models Module

This is the main user-facing module of the package, providing the tools to create, train, and use the classification models. The interaction will be based on an object-oriented paradigm. A user will instantiate a model object, which can then be used to perform standard machine learning operations. Key methods will include training the model on data, making predictions on new observations, and introspecting the fitted model to examine its internal components (e.g., estimated covariance matrices, permutations).

The specific algorithm used by the model object is determined at the time of its creation via a key hyperparameter. This allows the user to select one of the following four implemented models:

**gipsLDA weighted average** In this approach, a separate covariance matrix is first estimated for each class using the standard `gips` library. A final, single covariance matrix is then computed as a weighted average of these individual matrices. This pooled matrix is subsequently used for classification, adhering to the LDA framework.

**gipsLDA classic** This model follows a more traditional approach by estimating a single, pooled covariance matrix directly from all classes at once using the `gips` library. This is analogous to the classic LDA assumption but with the added layer of permutation symmetry discovery.

**gipsMultQDA** This model leverages the `gipsmult` module. The process involves first identifying the single most probable permutation structure that is common across all classes. Subsequently, a separate covariance matrix is estimated for each class, with each matrix being projected onto this shared permutation. This allows for class-specific covariance while maintaining a unified dependency model.

**gipsQDA** This represents the most flexible model. The `gips` library is applied independently to each class. Consequently, each class can have its own uniquely estimated permutation structure and its own distinct covariance matrix, analogous to the classic QDA framework but with individualized symmetry discovery for each class.

## 5.3 Code Organization

To ensure a clear separation between the reusable software package and the research-specific materials, the project is organized across two distinct repositories. This structure promotes modularity, reproducibility, and maintainability.

### 5.3.1 The R Package Repository

This repository contains the source code for the `gipsDA` R package itself, designed to be a self-contained and installable piece of software. Its main components are:

- **Source Code:** The core logic of the package. To adhere to the standard structure of an R package, all source code files are located within a single `R/` directory. To maintain logical modularity within this flat structure, a file naming convention has been adopted. Source files are prefixed to indicate their module affiliation:
  - `gipsmult_`: for files belonging to the 'gipsmult' module.
  - `models_`: for files belonging to the 'models' module.
- **Unit Tests:** Automated tests that verify the correctness of individual functions and methods within the package, ensuring its reliability.

### 5.3.2 The Engineering Thesis Repository

This repository houses all materials related to the engineering thesis, including data, experiments, and the written report. It is structured as follows:

- **Thesis Document:** The LaTeX source code for the written thesis.
- **Manual Tests:** R scripts used for higher-level validation, exploratory analysis, and the replication of the simulation studies described in this document.
- **Data Directory:** This directory does not store the raw dataset files directly, especially the large ones, to keep the repository lightweight. Instead, it is organized as follows:
  - `README.md`: A file providing detailed instructions and direct links for downloading the real-world datasets from their original sources.
  - `generate/`: A subdirectory containing the R scripts used to generate the five synthetic data scenarios for the simulation study.

## 6 GUI Design

A graphical user interface (GUI) will not be developed for this project. The primary reason for this decision is that the project is intended to be a typical software package, designed for integration into data analysis scripts and workflows, rather than a standalone application.

The user interface (UI) will be programmatic, following the standard design patterns of popular machine learning libraries in R and Python. Users will interact with the package by creating model objects. These objects will expose methods to perform key actions, such as training a model on a dataset (i.e., fitting) and using the fitted model to make predictions on new data. Furthermore, the model objects will be introspectable, allowing users to access internal parameters and results directly. This approach provides a user experience consistent with established tools like `scikit-learn` (Python) or `MASS` (R), offering a flexible and powerful interface for researchers and data scientists.

## 7 Initial Data Preparation

The data preparation strategy differs based on the nature of the dataset, whether it is synthetically generated or a real-world benchmark.

**Synthetic Data** For the synthetic datasets generated in our simulation study, no data preparation is required. The generation process is designed to produce clean, complete, and purely numerical data, which is immediately ready for model training and evaluation.

**Real-World Data** For the real-world datasets, a preprocessing pipeline is applied to handle non-numerical features and ensure compatibility with the models. The strategy for feature transformation is as follows:

- **Categorical Features:** These are processed based on their cardinality (the number of unique values):
  - If a feature has exactly two categories (binary), it is converted into a single numerical column with values 0 and 1.
  - If a feature has more than two categories (multi-class), the encoding strategy depends on the nature of the variable:
    - *Ordinal Encoding* is applied if the categories have a clear, justifiable intrinsic order.
    - *One-Hot Encoding* is applied otherwise for nominal variables where no such order exists. This method creates a new binary column for each category to prevent the model from inferring a false ordinal relationship.
- **Numerical Features:** Numerical features are used directly without any transformation. No scaling methods, such as standardization or min-max scaling, are applied.

## References

[1] Adam Chojecki, Paweł Morgen, and Bartosz Kołodziejek. Learning permutation symmetry of a gaussian vector with gips in r. *Journal of Statistical Software*, 112(7):1–38, 2025.

[2] P. Diaconis and D. Ylvisaker. Conjugate Priors for Exponential Families. *The Annals of Statistics*, 7(2):269–281, 1979.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition, 2009.