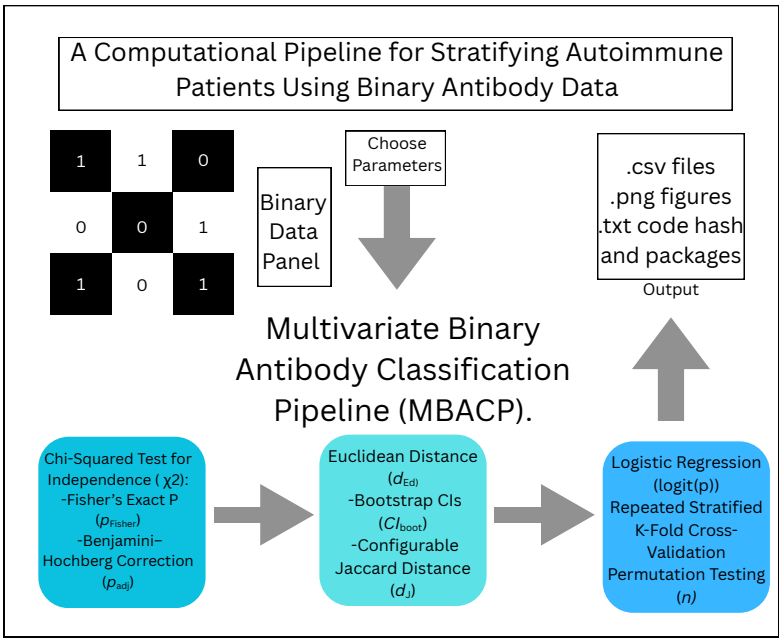
Graphical Abstract

A Computational Pipeline for Stratifying Autoimmune Patients Using Binary Antibody Data

Nathan Zhuang, Jacqueline Howells



A Computational Pipeline for Stratifying Autoimmune Patients Using Binary Antibody Data

Nathan Zhuang[a,b,*], Jacqueline Howells[a,c]

[a] *Polygence, Menlo Park, California, United States of America*
[b] *Western Canada High School, Calgary, Alberta, Canada*
[c] *Brown University, Providence, Rhode Island, United States of America*

Abstract

This article describes a comprehensive computational pipeline for stratifying autoimmune patient groups using exclusively binary (positive/negative) autoantibody data. The method addresses a significant methodological gap in computational immunology by providing a standardized framework for analyzing categorical serological data commonly found in electronic health records and resource-limited settings. The pipeline integrates three complementary analytical modules: (1) univariate association analysis with Chi-squared and Fisher's exact tests, (2) multivariate distance analysis with bootstrap confidence intervals, and (3) classification validation with logistic regression and permutation testing. We demonstrate the method's utility by applying it to autoimmune disorders, successfully recapitulating established clinical relationships. The pipeline is implemented in Python and includes detailed configuration options for custom antibody panels, disease groups, and stratification variables. This method enables researchers to extract meaningful immunological patterns from underutilized binary clinical data, serving as both a hypothesis-generation tool and stratification method in data-constrained research environments.

Keywords: Computational pipeline, Binary autoantibody data, Patient stratification, Autoimmune diseases, Statistical framework, Method validation

*Corresponding author. Email: nathan.zhuangzh@gmail.com

1. Method Name

Multivariate Binary Antibody Classification Pipeline (MBACP).

2. Background

2.1. Background and methodological gap

Binary autoantibody data represents an abundant yet underutilized resource in clinical datasets and electronic health records, particularly in resource-limited settings where continuous laboratory titers are unavailable [1, 2]. Currently, no standardized computational framework exists for multivariate analysis of binary autoantibody data, forcing researchers to either use simple univariate tests that lack power for complex pattern detection or apply methods designed for continuous data with suboptimal results [3, 4].

Our pipeline addresses this gap by providing an integrated analytical framework specifically optimized for binary serological data. The method builds upon established statistical approaches but combines them in a novel workflow that maximizes information extraction from categorical antibody data while providing robust validation at each analytical stage.

3. Method Details

3.1. Pipeline architecture and workflow

The computational pipeline follows a structured workflow consisting of three main phases: data preprocessing, multi-layered statistical analysis, and comprehensive output generation.

3.1.1. Data preprocessing module

The preprocessing module handles data loading, cleaning, and preparation for analysis through several key functions. The module provides flexible data input capabilities that accept CSV files with customizable column names. It performs automatic data cleaning by handling missing values, normalizing column names, and ensuring data consistency across the dataset. The system includes patient deduplication functionality that identifies and removes duplicate patient records based on unique identifiers and antibody profiles. Diagnosis standardization converts various diagnosis formats into standardized groups using configurable matching rules, enabling consistent group definitions across different data sources. Finally, the module supports subgroup stratification by clinical variables such as rheumatoid factor status, allowing for detailed subgroup analyses within broader diagnostic categories.

3.1.2. Analytical modules

The pipeline implements three complementary analytical modules that provide progressively sophisticated insights into the binary antibody data.

Module 1 performs univariate association analysis on the overall diagnostic groups (SS vs RA) to assess independence between individual antibodies and diagnosis. This module serves as an exploratory screening step to identify preliminary associations before detailed subgroup analysis. Chi-squared tests assess independence between individual antibodies and diagnostic groups [5]. The module complements Chi-squared tests with Fisher's exact tests, which provide exact p-values for small sample sizes or when expected cell counts are low [6]. The module automatically applies false discovery rate adjustment using the Benjamini-Hochberg procedure to control for multiple testing across all antibodies in the panel [7].

Module 2 conducts multivariate distance analysis to quantify overall immunological similarity between patient groups, including RF-stratified subgroups. This module calculates Euclidean distances between group centroids in the multidimensional antibody space, providing an intuitive measure of overall profile similarity [8]. To quantify uncertainty in these distance estimates, the module implements bootstrap confidence intervals through nonparametric resampling with 1000 iterations [9]. Additionally, the module computes Jaccard distance as an alternative similarity metric based on presence/absence patterns, offering complementary perspectives on group relationships.

Module 3 focuses on classification and validation through multivariate analysis of multiple group comparisons, including RF-stratified SS subgroups versus RA. The module employs logistic regression with default L2 regularization (C=1.0) and balanced class weighting to prevent overfitting while maintaining interpretability and handling class imbalance [10]. Model evaluation uses repeated stratified k-fold cross-validation with different iteration schemes optimized for computational efficiency: main performance metrics (accuracy, balanced accuracy, ROC AUC) employ 5 folds with 10 repeats (50 total iterations) for robust estimation, while ROC curve generation uses 5 folds with 2 repeats (10 iterations) to balance computational cost with adequate sampling [11]. Statistical significance of classification performance is assessed through permutation testing with 100 label randomizations, each evaluated using 5-fold cross-validation with 2 repeats, ensuring that observed performance exceeds chance levels [12]. The module reports comprehensive

performance metrics including accuracy, balanced accuracy, and ROC AUC with appropriate uncertainty estimates [13].

3.1.3. Output generation

The pipeline generates comprehensive outputs designed to support both methodological validation and scientific interpretation. These outputs include statistical summary tables in CSV format for easy integration with other analysis tools, publication-ready visualizations such as ROC curves, bootstrap distributions, and heatmaps that illustrate key findings, detailed performance metrics with uncertainty estimates to support robust conclusions, and complete reproducibility information including code hashes and package versions to ensure methodological transparency.

3.2. Software requirements and installation

The pipeline requires Python 3.9 or higher with several essential scientific computing packages. The core dependencies include pandas ($\geq$1.4.0) for data manipulation, numpy ($\geq$1.21.0) for numerical computations, scikit-learn ($\geq$1.0.0) for machine learning algorithms, scipy ($\geq$1.7.0) for statistical functions, matplotlib ($\geq$3.5.0) and seaborn ($\geq$0.11.0) for visualization, and statsmodels ($\geq$0.13.0) for advanced statistical testing. Installation is straightforward as the pipeline consists of a single Python script with no complex dependencies or external compilation requirements.

3.3. Data preparation and formatting

3.3.1. Required data structure

The pipeline expects input data in CSV format with a specific structure to ensure proper analysis. The data should be organized with patients as rows and variables as columns, where the columns represent the autoantibodies being tested. Table 1 outlines the required columns and their specifications. The data must include a unique patient identifier column, a diagnosis column containing disease labels, one or more stratification variables encoded as binary values, and multiple antibody columns representing the autoantibody panel, all encoded as binary (0/1) variables indicating presence or absence.

Table 1: Required data structure for pipeline input

Column Name	Data Type	Description	Example
patient_id	String/Integer	Unique patient identifier	P001, 12345
diagnosis	String	Disease diagnosis	"Rheumatoid Arthritis", "Sjögren's"
rheumatoid_factor	Binary (0/1)	Stratification variable	1 (positive), 0 (negative)
acpa	Binary (0/1)	Autoantibody 1	1 (positive), 0 (negative)
ana	Binary (0/1)	Autoantibody 2	1 (positive), 0 (negative)
anti_dsdna	Binary (0/1)	Autoantibody 3	1 (positive), 0 (negative)
anti_sm	Binary (0/1)	Autoantibody 4	1 (positive), 0 (negative)

3.3.2. Customizing antibody panels

Users can easily modify the antibody panel to match their specific research needs by editing the antibodies list in the main code. The default configuration includes four common autoantibodies, which can be customized as shown in the following code example:

```
# DEFAULT CONFIGURATION:
antibodies = ['acpa', 'ana', 'anti_dsdna', 'anti_sm']

# CUSTOM CONFIGURATION EXAMPLE:
antibodies = ['ana', 'anti_ro', 'anti_la', 'rnp', 'scl70']
```

The pipeline automatically adapts to the specified antibody columns, provided they exist in the input data and contain properly encoded binary values. The antibody names should match the column names in the input CSV file exactly.

3.3.3. Customizing disease groups

Disease group definitions can be modified in the `standardize_diagnosis` function to accommodate different research contexts. The function uses regular expressions to map various diagnosis formats to standardized group labels, as demonstrated in this code implementation:

```
def get_diagnosis_group(diagnosis_str):
    if pd.isna(diagnosis_str):
        return None
    s = str(diagnosis_str).lower()

    # CUSTOMIZE DISEASE MAPPING HERE:
    if re.search(r'\bra\b', s) or ('rheumatoid' in s and '
   arthritis' in s):
        return 'ra'
```

```
    elif any(x in s for x in ['sjogren', 'sjogren', 'sjogrens', 'sjogrens']):
        return 'ss'
    elif 'lupus' in s or 'sle' in s:
        return 'sle'  # ADD NEW DISEASE GROUP
    elif 'scleroderma' in s or 'systemic sclerosis' in s:
        return 'ssc'  # ADD NEW DISEASE GROUP
    else:
        return None
```

Researchers can extend the existing mapping by adding new disease patterns or modify the existing patterns to better match their specific dataset terminology. This flexibility allows the pipeline to be applied to diverse autoimmune disease research questions beyond the original RA and SS focus.

3.3.4. Customizing stratification variables

The pipeline supports stratification by any binary clinical variable present in the dataset, enabling detailed subgroup analyses. The default implementation uses rheumatoid factor status to stratify Sjögren's syndrome patients, but researchers can easily modify this to use other clinically relevant variables. Stratification is defined separately in two functions: `run_logistic_analysis` and `distance_analysis`, as shown in this code example:

```
# DEFAULT STRATIFICATION (Rheumatoid Factor):
# In run_logistic_analysis() and distance_analysis()
rf_pos = ss[ss['rheumatoid_factor'] == 1].copy()
rf_neg = ss[ss['rheumatoid_factor'] == 0].copy()

# CUSTOM STRATIFICATION EXAMPLE (ACPA status):
acpa_pos = ra[ra['acpa'] == 1].copy()
acpa_neg = ra[ra['acpa'] == 0].copy()
```

The stratification logic is implemented in clearly marked sections of the analysis functions, making customization straightforward for users with basic Python knowledge. Any binary column in the dataset can be used for stratification purposes.

3.4. Running the analysis

3.4.1. Basic execution

The pipeline can be executed with default settings using a simple command line invocation:

```
python binary_antibody_pipeline.py
```

Running the script automatically triggers the complete analytical workflow: loading data from the specified CSV file path, performing all analytical steps including preprocessing, statistical testing, distance calculation, and classification, and generating comprehensive outputs in a designated directory. The default configuration uses established best practices for statistical parameters, making it suitable for most research applications without requiring immediate customization.

3.4.2. Configuration options

Several key configuration parameters are hardcoded within specific functions and can be modified to adapt the pipeline to specific research needs. Advanced users can adjust these parameters directly in the source code:

```
# CROSS-VALIDATION SETTINGS (in run_logistic_analysis)
# Main performance metrics - robust estimation
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10,
    random_state=42)  # 50 iterations

# ROC curve generation - computational efficiency
cv_roc = RepeatedStratifiedKFold(n_splits=5, n_repeats=2,
    random_state=42)  # 10 iterations

# PERMUTATION TEST SETTINGS (in run_permutation_tests)
n_permutations = 100  # Number of label randomizations
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=2,
    random_state=42)  # 10 iterations per permutation

# BOOTSTRAP SETTINGS (in bootstrap_centroid_distance function
    )
n_bootstrap = 1000  # Number of bootstrap iterations
ci = 95            # Confidence interval percentage
seed = 42          # Random seed for reproducibility

# OUTPUT SETTINGS (in main function)
output_dir = "output"  # Output directory name
```

Cross-validation settings control the robustness of performance estimation, with main metrics using 50 iterations (5 folds $\times$ 10 repeats) for stable estimates, while ROC curves and permutation tests use 10 iterations (5 folds $\times$ 2 repeats) for computational efficiency. Permutation testing performs 100 label randomizations, each evaluated with 10-fold cross-validation. Bootstrap resampling uses 1000 iterations with 95% confidence intervals by default. Output settings allow researchers to specify custom directory names

for organizing results.

3.4.3. Handling special cases

The pipeline includes specific features to handle common data challenges encountered in real-world research. For datasets with severe class imbalance, the pipeline automatically adjusts class weights inversely proportional to class frequencies using the `class_weight='balanced'` parameter in logistic regression. This ensures equal importance of minority and majority classes during model training:

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(max_iter=1000,
                                 class_weight='balanced',  # Handles imbalance
                                 random_state=42))
])
```

When working with small sample sizes, researchers may benefit from increasing the number of permutation tests for more precise p-value estimation and should rely more heavily on Fisher's exact tests in the univariate analysis, which the pipeline automatically computes alongside Chi-squared tests. For studies with expected cell counts below 5 in contingency tables, the pipeline provides warnings and prioritizes Fisher's exact results.

3.5. Output interpretation

3.5.1. Key output files

The pipeline generates a comprehensive set of output files designed to support complete methodological documentation and result interpretation. The `classifier_performance.csv` file contains cross-validated performance metrics for all group comparisons, including accuracy, balanced accuracy, and ROC AUC with standard deviations (based on 50 CV iterations). The `statistical_analysis.csv` file provides results of univariate association tests for each antibody comparing overall SS vs RA groups, including Chi-squared and Fisher's exact test statistics, p-values, adjusted p-values, and odds ratios. Distance analysis results are saved in `distance_analysis.csv` containing Euclidean and Jaccard distance metrics with bootstrap confidence intervals for multiple group comparisons including RF-stratified subgroups. Raw 2×2 contingency tables for each antibody are exported in

`contingency_tables.csv` for additional verification. Reproducibility information including code hashes, timestamps, and package versions is documented in `version_info.txt`. Additionally, the pipeline generates multiple visualization plots including ROC curves with confidence intervals, bootstrap distribution histograms, confusion matrices, and permutation test results for intuitive result interpretation. All results are also saved in `all_results.pkl` as a Python pickle file for programmatic access.

3.5.2. Interpreting distance metrics

Euclidean distances provide a quantitative measure of immunological similarity between patient groups in the multidimensional antibody space. These distances range from 0, indicating identical antibody profiles between groups, to $\sqrt{p}$ (where p is the number of antibodies) in extreme cases of completely dissimilar profiles. Smaller distance values suggest greater immunological similarity, which may reflect shared pathogenic mechanisms or clinical overlap between conditions. The bootstrap confidence intervals accompanying each distance estimate provide important information about measurement precision, with wider intervals typically indicating greater uncertainty often associated with smaller sample sizes. Jaccard distance complements Euclidean distance by providing a binary similarity metric ranging from 0 (identical presence/absence patterns) to 1 (completely different patterns), based solely on which antibodies show any prevalence in each group.

3.5.3. Interpreting classification performance

Classification performance metrics offer insights into the discriminative power of antibody profiles for distinguishing between patient groups. ROC AUC values close to 0.5 indicate classification performance indistinguishable from random guessing, while values approaching 1.0 suggest strong discriminative ability. Values between 0.7-0.8 are often considered acceptable discrimination, 0.8-0.9 indicate excellent discrimination, and above 0.9 suggest outstanding discrimination. Permutation p-values below 0.05 indicate that the observed classification performance significantly exceeds chance levels, providing statistical validation of the multivariate patterns. These p-values are calculated by comparing the true ROC AUC (from 10 CV iterations) against the distribution of 100 permuted label AUCs. Confidence intervals around performance metrics (expressed as standard deviations from 50 CV iterations) help assess estimate stability, with smaller standard deviations generally reflecting more reliable performance estimates. The relationship

between distance metrics and classification performance often shows an inverse pattern, where immunologically distinct groups (larger Euclidean distances) are typically easier to classify (higher AUC values), providing internal validation of the analytical consistency. However, this relationship is not deterministic, as classification performance also depends on the separability of groups along the specific antibody dimensions used by the logistic regression model.

4. Method validation

We validated the pipeline using both synthetic and clinical autoimmune datasets to assess performance under controlled and real-world conditions.

4.1. Validation datasets

The synthetic validation used the Comprehensive Autoimmune Disorder Dataset [14], which contained 176 patients after deduplication, including 91 rheumatoid arthritis (RA) patients and 85 Sjögren's syndrome (SS) patients with rheumatoid factor stratification (RF-positive SS: n=38, RF-negative SS: n=47). For real-world validation, we applied the pipeline to the Diagnosis of Rheumatic and Autoimmune Diseases dataset [15], which included 1,229 patients (RA: n=766, SS: n=463) with substantial class imbalance in RF stratification among SS patients (RF-positive SS: n=433, RF-negative SS: n=30).

4.2. Statistical analysis validation

The pipeline successfully performed comprehensive statistical analyses on both datasets. For the synthetic dataset, univariate association tests between RA and SS groups revealed no statistically significant associations after Benjamin-Hochberg correction. ANA showed the strongest trend toward association (χ^2=2.910, p=0.0880, adjusted p=0.3242; Fisher's OR=0.531, p=0.0689, adjusted p=0.2651), while other antibodies showed minimal association: ACPA (χ^2=0.005, p=0.9419; Fisher's OR=0.933, p=0.8790), anti-dsDNA (χ^2=1.954, p=0.1621; Fisher's OR=0.624, p=0.1325), and anti-Sm (χ^2=0.167, p=0.6830; Fisher's OR=1.185, p=0.6497).

In the clinical dataset, similar non-significant patterns emerged for the available antibody panel: HLA-B27 (χ^2=0.000, p=0.9823; Fisher's OR=1.010, p=0.9531), anti-dsDNA (χ^2=0.201, p=0.6536; Fisher's OR=0.942, p=0.6378),

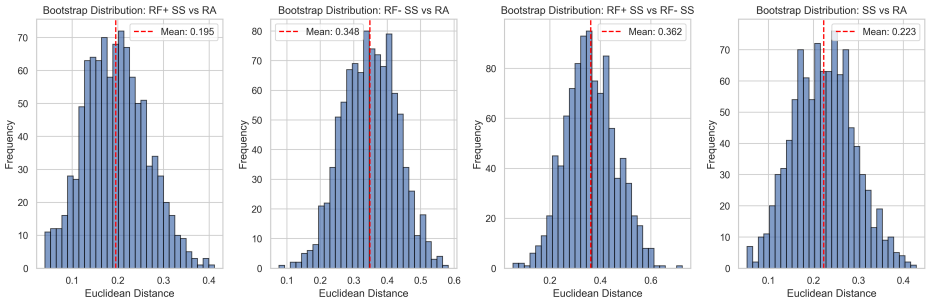
and anti-Sm (χ^2=0.657, p=0.4176; Fisher's OR=1.108, p=0.4091). All clinical dataset p-values remained non-significant after multiple testing correction.

4.3. Multivariate distance analysis

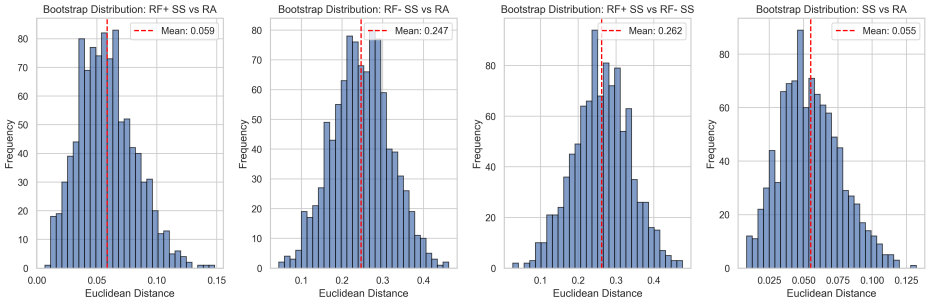
Euclidean distance analysis with bootstrap confidence intervals revealed coherent immunological similarity patterns that aligned with established clinical knowledge in both datasets (Table 2). In the synthetic dataset, RF-positive SS patients showed significantly closer immunological proximity to RA (distance = 0.195, 95% CI: 0.068-0.335) compared to RF-negative SS patients (distance = 0.348, 95% CI: 0.197-0.507). The RF-positive SS vs RF-negative SS comparison showed the largest distance (0.362, 95% CI: 0.173-0.558), while the overall SS vs RA distance was intermediate (0.223, 95% CI: 0.095-0.362).

Table 2: Bootstrap-validated Euclidean distances between diagnostic groups

Comparison	**Synthetic Dataset**	**Clinical Dataset**
RF-positive SS vs RA	0.195 (0.068-0.335)	0.059 (0.019-0.108)
RF-negative SS vs RA	0.348 (0.197-0.507)	0.247 (0.108-0.389)
RF-positive vs RF-negative SS	0.362 (0.173-0.558)	0.262 (0.112-0.409)
SS vs RA	0.223 (0.095-0.362)	0.055 (0.018-0.101)



(a) Synthetic dataset results.



(b) Clinical dataset results.

Figure 1: **Bootstrap analyses of immunological distance metrics.** Empirical Euclidean-distance distributions from 1000 bootstrap iterations. Panel (a) shows results obtained from the synthetic dataset, while panel (b) shows corresponding distributions from the clinical dataset. Together they illustrate the reproducibility and clinical relevance of the immunological similarity metrics.

The clinical dataset reproduced this directional pattern despite different antibody panels and substantial class imbalance. RF-positive SS patients showed closer serological proximity to RA (distance = 0.059, 95% CI: 0.019-0.108) compared to RF-negative SS patients (distance = 0.247, 95% CI: 0.108-0.389). The consistency across datasets demonstrates the pipeline's robustness to different data characteristics.

4.4. Classification performance

Logistic regression models demonstrated performance patterns that directly reflected the immunological relationships identified through distance analysis (Table 3). In the synthetic dataset, the RF-negative SS vs RA comparison showed the highest discriminative ability (ROC AUC = 0.639 $\pm$ 0.093), consistent with their greater immunological distance. Conversely,

RF-positive SS vs RA showed minimal predictive power (ROC AUC = 0.362 ± 0.102), accurately reflecting their serological similarity. The RF-positive SS vs RF-negative SS comparison showed intermediate performance (ROC AUC = 0.618 ± 0.130), while overall SS vs RA discrimination was modest (ROC AUC = 0.557 ± 0.088).

Table 3: Classification Performance with Permutation Test Results

Comparison	Accuracy	Balanced Accuracy	ROC AUC	Permutation p	Class Ratio
Synthetic Dataset					
RF-positive SS vs RA	0.391 ± 0.091	0.364 ± 0.080	0.362 ± 0.102	0.9406	1:2.4
RF-positive vs RF-negative SS	0.608 ± 0.113	0.602 ± 0.115	0.618 ± 0.130	0.0594	1:1.2
RF-negative SS vs RA	0.635 ± 0.073	0.639 ± 0.073	0.639 ± 0.093	0.0396	1:1.9
SS vs RA	0.566 ± 0.085	0.568 ± 0.085	0.557 ± 0.088	0.1881	1:1.1
Clinical Dataset					
RF-positive SS vs RA	0.485 ± 0.028	0.485 ± 0.027	0.490 ± 0.033	0.4752	1:1.8
RF-positive vs RF-negative SS	0.591 ± 0.085	0.547 ± 0.097	0.580 ± 0.102	0.0891	14.4:1
RF-negative SS vs RA	0.553 ± 0.071	0.547 ± 0.087	0.572 ± 0.081	0.0693	1:25.5
SS vs RA	0.486 ± 0.029	0.486 ± 0.024	0.482 ± 0.026	0.8020	1:1.7

In the clinical dataset, classification performance was substantially impacted by class imbalance despite class-weighted logistic regression. RF-positive SS vs RA achieved near-chance performance (ROC AUC = 0.490 ± 0.033, permutation p = 0.4752), while RF-negative SS vs RA showed modest but non-significant discrimination (ROC AUC = 0.572 ± 0.081, permutation p = 0.0693). The extreme class imbalance in RF-positive vs RF-negative SS comparison (14.4:1 ratio) resulted in challenging classification (ROC AUC = 0.580 ± 0.102, permutation p = 0.0891).

4.4.1. Permutation testing validation

Permutation testing provided crucial statistical context for classification performance across both datasets. In the synthetic dataset, only the RF-negative SS vs RA comparison achieved statistical significance (p = 0.0396), while the RF-positive SS vs RA comparison showed performance indistinguishable from chance (p = 0.9406). The RF-positive SS vs RF-negative SS comparison approached but did not reach significance (p = 0.0594). In the clinical dataset, no comparisons reached statistical significance, with all permutation p-values exceeding 0.05, reflecting the greater analytical challenge posed by real-world data characteristics.

4.5. Antibody prevalence patterns

Analysis of autoantibody prevalence revealed distinct serological profiles across diagnostic groups in both datasets. In the synthetic dataset,

RF-negative SS patients demonstrated higher prevalence of ANA (83.0% vs 71.1%) and anti-dsDNA (72.3% vs 44.7%) compared to RF-positive SS patients, while ACPA and anti-Sm prevalence remained similar across subgroups. In the clinical dataset, RF-negative SS patients exhibited higher HLA-B27 prevalence (63.3% vs 49.0%) but lower anti-dsDNA prevalence (36.7% vs 52.0%) compared to RF-positive SS patients. These patterns demonstrate the pipeline's ability to capture clinically relevant serological heterogeneity across different antibody panels and population characteristics.

5. Limitations

This method will not work if the antibodies selected for stratification are used as diagnostic specifiers in the dataset being used. For example, conducting a test on the potential stratification of Disease X and Disease Y using Z stratification and antibodies N and M will not work if all of Disease X is positive for N due to the diagnostic procedure of the data collection. Same goes for if the stratification is all positive for a certain tested disease.

Ethics Statement

This method uses de-identified data; no ethical approval was required for the development and validation of the computational pipeline. Users should ensure appropriate ethical approvals for their specific datasets and research contexts.

CRediT Authorship Contribution Statement

Nathan Zhuang: Methodology, Software, Validation, Formal analysis, Data curation, Writing - original draft, Visualization.

Jacqueline Howells: Supervision, Writing - review & editing, Project administration.

Declaration of Competing Interests

The authors declare no competing interests.

Acknowledgments

The authors thank Polygence for providing a platform and support to conduct this research. The method development was supported by the Polygence mentorship program.

Data and Code Availability

The complete pipeline code, configuration files, and usage examples are available at: GitHub Repository

References

[1] Z. Wu, L. Casciola-Rosen, A. A. Shah, A. Rosen, S. L. Zeger, Estimating autoantibody signatures to detect autoimmune disease patient subsets, Biostatistics 20 (1) (2017) 30–47. doi:10.1093/biostatistics/kxx061.

[2] L. H. Carlton, R. McGregor, N. J. Moreland, Human antibody profiling technologies for autoimmune disease, Immunologic Research 71 (4) (2023) 516–527. doi:10.1007/s12026-023-09362-8.

[3] S. M. Al-Mayouf, A. Hamad, W. Kaidali, R. Alhuthil, A. Alsaleem, Clinical characteristics and prognostic value of autoantibody profile in children with monogenic lupus, Journal of Rheumatic Diseases 31 (3) (2023) 143–150. doi:10.4078/jrd.2023.0065.

[4] J. H. Jones, N. Fleming, The problem with dichotomizing quality improvement measures, BMC Anesthesiology 22 (1) (9 2022). doi:10.1186/s12871-022-01833-z.

[5] M. L. McHugh, The Chi-square test of independence, Biochemia Medica (2013) 143–149doi:10.11613/bm.2013.018.

[6] H.-Y. Kim, Statistical notes for clinical researchers: Chi-squared test and Fisher's exact test, Restorative Dentistry & Endodontics 42 (2) (2017) 152. doi:10.5395/rde.2017.42.2.152.

[7] Y. Benjamini, Y. Hochberg, Controlling the false discovery Rate: A practical and powerful approach to multiple testing, Journal of the Royal Statistical Society Series B (Statistical Methodology) 57 (1) (1995) 289–300. doi:10.1111/j.2517-6161.1995.tb02031.x.
URL https://doi.org/10.1111/j.2517-6161.1995.tb02031.x

[8] G. Glazko, A. Mushegian, Measuring gene expression divergence: the distance to keep, Biology Direct 5 (1) (2010) 51. doi:10.1186/1745-6150-5-51.

[9] T. J. DiCiccio, B. Efron, Bootstrap confidence intervals, Statistical Science 11 (3) (9 1996). doi:10.1214/ss/1032280214.

[10] S. Sperandei, Understanding logistic regression analysis, Biochemia Medica (2014) 12–18doi:10.11613/bm.2014.003.

[11] D. Wilimitis, C. G. Walsh, Practical Considerations and Applied Examples of Cross-Validation for model development and Evaluation in Health Care: tutorial, JMIR AI 2 (2023) e49023. doi:10.2196/49023.

[12] A. I. Bandos, H. E. Rockette, D. Gur, A permutation test sensitive to differences in areas for comparing ROC curves from a paired design, Statistics in Medicine 24 (18) (2005) 2873–2893. doi:10.1002/sim.2149.

[13] F. S. Nahm, Receiver operating characteristic curve: overview and practical use for clinicians, Korean journal of anesthesiology 75 (1) (2022) 25–36. doi:10.4097/kja.21209.

[14] A. Ragheb, Comprehensive Autoimmune Disorder Dataset (11 2024).
URL https://www.kaggle.com/datasets/abdullahragheb/all-autoimmune-disorder-10k/data

[15] M. F. Mahdi, A. Jahani, D. H. Abd, Diagnosis of Rheumatic and Autoimmune Diseases Dataset, Data in Brief 60 (2025) 111623. doi:10.1016/j.dib.2025.111623.