Comprehensive Methodology Report: Supervised Learning on High-Dimensional Data

Aayush Adhikari

May 31, 2025

1 Introduction

This report describes in detail the machine learning workflow for predicting a binary outcome from a high-dimensional biomedical dataset. Our goal was to maximize model generalization and produce class probability predictions for unseen data (test, blinded test). This document provides an honest, technical, and realistic account of the analytical decisions, challenges encountered, and lessons learned.

2 Data Description and Initial Challenges

The dataset consisted of three main files: a training set (315 samples), a public test set (100 samples), and a blinded test set (36 samples). Each record included thousands of engineered features (over 3200), an `ID` column, and a binary `CLASS` label (for train only). The feature names were anonymized (`Feature_1`, `Feature_2`, ...), indicating no prior domain knowledge.

Immediate challenges included:

- **Very high feature-to-sample ratio** (curse of dimensionality).
- **High missingness**: Many features had substantial missing values.
- **Potential data leakage** due to feature redundancy and constant columns.
- **Unknown feature meaning**: All features were anonymized, preventing direct interpretation.
- **Class imbalance** and risk of overfitting, particularly on such a small sample size.

3 Data Cleaning and Preprocessing

Feature Filtering

- **Constant and near-constant features** were removed to reduce computation and avoid misleading importance scores.

- **Missing value thresholding:** Features with over 30% missingness were discarded. This aggressive cut-off was necessary due to the small sample size and because imputation with too few non-missing samples can create noise.
- **Outlier and non-finite handling:** All `inf`/`-inf` values were set to NaN, then all remaining missing values were imputed using the median—a robust strategy that preserves distributions for both continuous and categorical encodings.
- **Correlation pruning:** To avoid multi-collinearity, we computed the absolute Pearson correlation matrix and removed features with a correlation above 0.95 to any other remaining feature (upper triangle). This is especially crucial for tree models and logistic regression, where highly collinear features can bias importance and inflate generalization error.

Dataset Alignment

After feature selection, test and blind test sets often lacked some features selected in the train set, due to missing values or construction. To ensure prediction pipelines never fail:

- Any missing features in the test/blind set were added as columns with all values set to NaN (so the imputer can handle them).
- All columns were always ordered identically to the selected feature list, enforcing robust pipeline operation across splits and datasets.

4 Feature Selection

Given the extreme dimensionality and low sample size, reducing the number of input features was critical. We evaluated:

- **Recursive Feature Elimination with Cross Validation (RFECV):** This was our primary strategy. RFECV works by recursively removing the least important features (as determined by model coefficients or importance scores), using cross-validated performance (AUC) to select the optimal subset size. We used it with several estimators: Logistic Regression, Random Forest, and XGBoost.
- **Univariate Feature Selection:** For algorithms without built-in feature importance (e.g., KNN), we used SelectKBest based on ANOVA F-score. This approach is fast and often surprisingly effective in high-dimensional but low-sample scenarios.

Reflection: The number of selected features varied widely by algorithm. For example, RFECV with Logistic Regression often selected ∼150 features, while with Random Forest or XGBoost it sometimes retained hundreds, and SelectKBest with KNN typically selected 30. This demonstrates how model bias, feature redundancy, and validation metric interact—the pipeline must always validate feature choices to avoid overfitting.

5 Modeling Pipeline and Algorithm Choices

Common Pipeline Steps

All models used a standardized `scikit-learn` pipeline:

1. **Imputation:** Median imputer handles all NaNs (critical for robust generalization and to deal with intentionally added missing columns).
2. **Scaling:** StandardScaler applied (required for KNN and logistic regression, included for consistency elsewhere).
3. **Feature selection:** Only for training; final prediction pipelines use the selected feature set.
4. **Classifier:** Algorithm-specific.

Algorithm Details and Rationale

- **Logistic Regression:** Chosen for its interpretability and suitability for high-dimensional binary classification. Used L1 penalty (Lasso) to encourage sparsity.
- **Random Forest:** Robust to noise and irrelevant features, handles non-linearities and interactions. Class weighting used to address imbalance.
- **XGBoost:** More flexible boosting model, can capture subtle feature interactions. Used scale_pos_weight for class imbalance.
- **KNN:** Simple, non-parametric baseline. Requires scaling and careful feature selection due to the curse of dimensionality.

Hyperparameters: All hyperparameters were kept conservative and reproducible (e.g., `random_state=42`, default or moderate complexity settings for estimators). For KNN, $k = 5$ neighbors was used.

Validation Strategy

- **Feature selection validation:** All feature selection used 5-fold stratified cross-validation, optimizing AUROC.
- **Final model validation:** An 80/20 stratified train/validation split was used for unbiased final metric reporting.
- **Prediction for test/blinded sets:** Models were re-fitted on all training data and used to generate class probabilities for the test and blind sets. True labels were unavailable for these splits.

6 Reflections on Problems and Solutions

Major Issues

- **High dimensionality:** Models rapidly overfit if feature selection was not stringent. We solved this by aggressive feature reduction and cross-validation.
- **Feature alignment:** Test and blind sets were missing columns after feature selection, causing errors. Our fix was to forcibly add any missing features as all-NaN columns before imputation, guaranteeing full compatibility.
- **Imbalanced classes:** Most models included class_weight='balanced' or scale_pos_weight for XGBoost, ensuring sensitivity/specificity tradeoff was not ignored.
- **Inconsistent performance across models:** Some algorithms (e.g., Random Forest) showed signs of overfitting even after feature selection, due to their capacity and the low sample size. Validation scores were therefore always compared, and model selection was never made based on training scores alone.
- **Interpretability:** Feature names were anonymized, so interpretation was limited to importance rankings. Our focus was thus on reproducibility and validation.

Algorithm Comparison and Lessons Learned

- **Logistic Regression** gave the most stable, generalizable performance, especially when paired with careful feature selection (L1 or RFECV).
- **Random Forest** often selected a large number of features and tended to overfit unless hyperparameters were strongly regularized.
- **XGBoost** was sensitive to feature selection strategy; careful tuning of scale_pos_weight and feature count was needed.
- **KNN** was only competitive with very aggressive dimensionality reduction; otherwise, the curse of dimensionality rendered its predictions nearly random.
- **Univariate selection** (SelectKBest) worked surprisingly well as a baseline, showing that a handful of strong features can sometimes outperform complex selection methods when data is limited.

Best Practices and Reproducibility

Every step was wrapped in a pipeline, with strict train/validation/test isolation and reproducible random seeds. We exported all model predictions and processed feature sets to CSV for downstream evaluation.

7 Future Directions

Given more time, further improvements could include:

- Advanced ensembling (e.g., stacking, blending).
- Deep learning models (with caution, given small data).
- Automated hyperparameter tuning (e.g., grid or Bayesian search).
- Synthetic data augmentation.
- If feature meanings become available, domain-driven feature engineering.

8 Deliverables

- Predicted class probabilities for all models and datasets (as CSV).
- Methodology report (this file).
- Scripts and Jupyter notebooks for all code and data processing.

9 Conclusion

The project illustrates the complexity and risk of overfitting inherent in high-dimensional, low-sample machine learning. Through systematic preprocessing, stringent feature selection, model validation, and honest metric reporting, we produced robust and reproducible results, even under severe constraints. The lessons learned here generalize to a wide range of real-world biomedical and high-dimensional ML applications.