Report 2: Project Progress

University/Department: California State University, Dominguez Hills

Course/Thesis/Project Title: CSC-590 Modeling Complex Binary-Class Associations in Simulated Genomic Data

Semester/Year: Fall - 2025

Project Title: Modeling Complex Binary-Class Associations in Simulated Genomic Data

Student Name: Mantra Mehta

Student ID: 212265978

Instructor / Committee Chair: Dr. Sahar Hooshmand

Date: 11/09/2025

1. Abstract / Summary of Progress

Since Report 1, the project has advanced from baseline linear modeling toward a broader evaluation of nonlinear ensemble approaches for predicting binary outcomes in simulated genomic data. Two representative datasets, **4-wayAdditive_100feat.txt** and **2-wayEpi_100feat.txt**, were analyzed to compare the performance of Logistic Regression [2], Random Forest [8], and XGBoost [7] classifiers. The goal of this phase was to determine whether ensemble models could better capture nonlinear and epistatic relationships among single-nucleotide polymorphisms (SNPs) that are not well represented by linear methods [3]. The new implementation introduced a modular Python pipeline built with **scikit-learn** [2] and **XGBoost** [7]. It performs preprocessing, five-fold cross-validation [9], model training, and visualization. Metrics such as ROC-AUC, accuracy, and F1-score were used to evaluate model performance, and visual outputs were generated in the form of ROC curves and feature-importance plots. Experimental findings show that Logistic Regression performs very well on additive genetic effects, achieving a mean cross-validated AUC of approximately 0.97. However, it fails to detect interactions in the purely epistatic dataset, where its AUC drops to 0.49. Ensemble models such as Random Forest [8] and XGBoost [7] outperform the baseline under nonlinear conditions. XGBoost achieved the best generalization on both datasets, with a mean cross-validated AUC of 0.99 on the additive dataset and 0.58 on the epistatic dataset. These results confirm that tree-based ensemble models provide greater flexibility in identifying non-additive associations without extensive feature engineering, aligning with the Cedars-Sinai reproducibility and transparency framework described in **STREAMLINE** [6]. The pipeline developed during this stage establishes the foundation for the next report, which will integrate Relief-based feature-selection strategies [4], [5] sensitive to interaction effects and evaluate model scalability on high-dimensional “challenge” datasets containing 10 000 features and 1 percent missing values.

2. Introduction and Objectives (Recap)

This project focuses on understanding and modeling complex binary-class associations in simulated genomic data that represent different types of genetic architectures [1]. The primary aim is to identify how various machine-learning algorithms perform when predicting binary outcomes based on single-nucleotide polymorphism (SNP) features that may exhibit additive, epistatic, or heterogeneous effects [3]. The datasets used in this study were generated with the **GAMETES** software package [1], which simulates realistic genetic models containing both predictive and non-predictive SNPs.

The motivation behind this research arises from the difficulty of interpreting high-dimensional genomic data in which gene–gene interactions influence phenotypic outcomes [10]. Traditional linear models such as Logistic Regression [2] often fail to detect these nonlinear interactions, resulting in poor predictive performance in purely epistatic or heterogeneous datasets [3]. Addressing this limitation is essential for developing

interpretable and robust pipelines that can capture complex genetic patterns while preserving biological relevance.

The overall objective of this semester-long project is to design and implement a reproducible machine-learning workflow that compares multiple modeling strategies across simulated datasets. By the end of the project, the goal is to determine which algorithms are most effective for different patterns of association and to document best practices for preprocessing, feature selection [4], [5], model evaluation [9], and interpretation. The results from each progress report contribute toward a final, comprehensive analysis supporting the Cedars-Sinai **URBS-Lab** initiative on automated, transparent, and reproducible genomic data modeling [6].

Building upon the foundation established in Report 1, this stage extends the baseline Logistic Regression analysis to include nonlinear ensemble models such as Random Forest [8] and XGBoost [7]. These algorithms are capable of capturing higher-order interactions and complex dependencies among SNPs that linear models cannot represent. The transition from linear to nonlinear modeling marks an important milestone toward identifying optimal strategies for handling diverse genetic association patterns while maintaining the reproducibility and transparency emphasized in the STREAMLINE framework [6].

3. Work Completed

Since the submission of Report 1, significant technical progress has been made toward extending the baseline machine-learning framework to support nonlinear and ensemble-based modeling. The focus of this phase was to evaluate the capability of advanced classifiers to detect complex genetic associations, particularly those that exhibit epistatic interactions. All work was performed by the student as an individual contributor, including model design, implementation, testing, and visualization.

3.1 Data Preparation and Setup

Two simulated genomic datasets from the Cedars-Sinai URBS-Lab repository were used in this phase: **4-wayAdditive_100feat.txt** and **2-wayEpi_100feat.txt**. Both datasets were generated using the **GAMETES** simulation software [1], which produces SNP-based binary-class data containing a mixture of predictive (“M”) and non-predictive (“N”) features. Each dataset contains 1 000 instances and 100 features, with the column *Class* serving as the binary outcome variable (0 = control, 1 = case).

A modular preprocessing pipeline was developed using **scikit-learn** [2]. The pipeline includes missing-value imputation with SimpleImputer(strategy="most_frequent"), standardization through StandardScaler, and a consistent random-state seed to ensure reproducibility. The data were split into training and testing subsets using a 70/30

stratified division, followed by **five-fold cross-validation** to reduce variance and improve robustness in performance estimation [9].

3.2 Model Development

Three classification algorithms were integrated into the unified workflow:

1. **Logistic Regression** [2] – used as the linear baseline model, providing interpretability through coefficient weights and serving as a control to benchmark nonlinear performance.
2. **Random Forest** [8] – implemented as an ensemble of decision trees with bootstrap aggregation to capture nonlinear additive and interactive effects.
3. **XGBoost** [7] – a gradient-boosting framework optimized for high efficiency and capable of modeling higher-order epistatic interactions through successive tree construction.

Each model was trained using identical preprocessing steps to maintain comparability. Hyperparameters such as n_estimators = 400 and max_depth = 5 were applied consistently for ensemble models to balance performance and generalization. Evaluation metrics included **ROC-AUC**, **Accuracy**, and **F1-score** on the held-out test data.

3.3 Implementation and Execution

A new Python module, *report2_run.py*, was developed to automate data loading, preprocessing, model fitting, and metric computation. The script uses a consistent function structure and saves all outputs in a /results directory for transparent tracking. It also produces **ROC curves** and **feature-importance plots** for both ensemble models, allowing a direct visual comparison between the additive and epistatic datasets.

For each dataset, cross-validated ROC-AUC values were averaged across folds, and all key metrics were written to CSV files for later reporting. Visual outputs include:

- roc_4-wayAdditive_100feat.png and roc_2-wayEpi_100feat.png – overlay plots comparing Logistic Regression, Random Forest, and XGBoost.
- featimp_RandomForest_*.png and featimp_XGBoost_*.png – bar charts showing the top 15 most influential SNP features per model.

3.4 Summary of Results

The **4-way Additive** dataset demonstrated strong predictive performance across all models, consistent with the presence of clear univariate associations. Logistic Regression achieved a cross-validated ROC-AUC of 0.967 ± 0.004 and a test AUC of 0.991. Random Forest and XGBoost further improved slightly, reaching mean AUCs of 0.985 ± 0.010 and 0.986 ± 0.007, respectively. These results indicate that both linear and ensemble models can efficiently learn additive patterns, with only marginal benefit from nonlinearity.

In contrast, the **2-way Epistatic** dataset revealed substantial differences between model families. Logistic Regression failed to capture interaction effects, producing an AUC near 0.49 (random guess). Random Forest achieved moderate improvement (AUC ≈ 0.53), while XGBoost substantially outperformed both, reaching a mean cross-validated AUC of 0.577 ± 0.060 and a test AUC of 0.694. These findings confirm that tree-based boosting methods can partially recover epistatic signal even without explicit interaction terms, aligning with prior evidence from Relief-based feature-selection studies [4], [5].

3.5 Contributions and Achievements

- Designed and implemented a reproducible pipeline for additive and epistatic SNP data using **scikit-learn** [2] and **XGBoost** [7].
- Integrated **Random Forest** [8] and **XGBoost** [7] to extend baseline linear modeling.
- Employed **five-fold cross-validation** [9] for statistically reliable evaluation.
- Generated comparative **ROC curves** and **feature-importance plots** for interpretability.
- Quantified the failure of linear methods in purely epistatic models, supporting theoretical expectations from genetic heterogeneity literature [3].
- Validated that ensemble models capture nonlinear dependencies consistent with Cedars-Sinai reproducibility and **STREAMLINE** standards [6].
- Organized all metrics and plots for seamless integration into the next analytical phase.

4. Implementation

4.1 Code Structure and Workflow

The implementation for this phase was completed in **Python 3.11** using the **scikit-learn** [2] and **XGBoost** [7] frameworks. The environment was configured and executed locally on an **Alienware M16 R2** laptop with an **Intel Ultra 9 CPU** and **16 GB RAM**. The pipeline was designed to ensure full reproducibility, modularity, and adherence to Cedars-Sinai **STREAMLINE** reproducibility standards [6].

The project directory was structured to separate datasets, scripts, and output results. All experiments were executed using a newly developed script, *report2_run.py*, which automated data loading, preprocessing, model fitting, and visualization. The code implements a unified interface for comparing algorithms and uses a consistent random seed (random_state=42) to preserve experimental integrity across runs.

Each model follows a three-stage process:

1. **Preprocessing:** Missing values imputed using the most frequent allele frequency (mode), features scaled via StandardScaler to maintain numerical stability.
2. **Cross-Validation: Stratified 5-fold CV** [9] for unbiased model evaluation on imbalanced classes.
3. **Model Training and Testing:** Logistic Regression [2], Random Forest [8], and XGBoost [7] models trained on identical partitions to enable fair performance comparison.

All intermediate outputs were stored in a dedicated /results folder, which includes tabular metrics (CSV files), ROC curve overlays, and feature-importance plots for ensemble models.

4.2 Code Modules Developed

The main code segments implemented for this phase include:

Module / Script	**Purpose**	**Key Functionality**
test_run.py	Baseline Logistic Regression (Report 1)	Loads data, trains model, computes metrics [2].
test_plot.py	Visualization script	Generates bar plots of model performance across datasets.
report2_run.py	Main experiment pipeline (Report 2)	Automates training for Logistic Regression, Random Forest, and XGBoost; performs 5-fold CV [7], [8], [9].
/results folder	Output storage	Contains ROC and feature-importance images, plus metrics CSV files.

Each module was tested individually and then integrated to verify end-to-end execution. The *report2_run.py* script also supports CLI arguments, enabling dynamic dataset selection (python report2_run.py 4-wayAdditive_100feat.txt).

4.3 Experimental Results

The results obtained from the two datasets — **4-way Additive** and **2-way Epistatic** — are summarized below. All metrics are derived from the averaged five-fold cross-validation and test evaluations.

Table 1. Model Performance Comparison – 4-way Additive Dataset

Model	CV AUC (mean ± std)	Test Accuracy	Test F1	Test AUC
Logistic Regression [2]	0.967 ± 0.004	0.957	0.957	0.991
Random Forest [8]	0.985 ± 0.010	0.953	0.954	0.993
XGBoost [7]	**0.986 ± 0.007**	**0.960**	**0.961**	**0.994**

Interpretation: All three models performed exceptionally well, indicating that additive effects are linearly separable and easily captured by logistic models. Random Forest and XGBoost provided minor gains in generalization and stability, consistent with prior findings on additive genetic architectures [1].

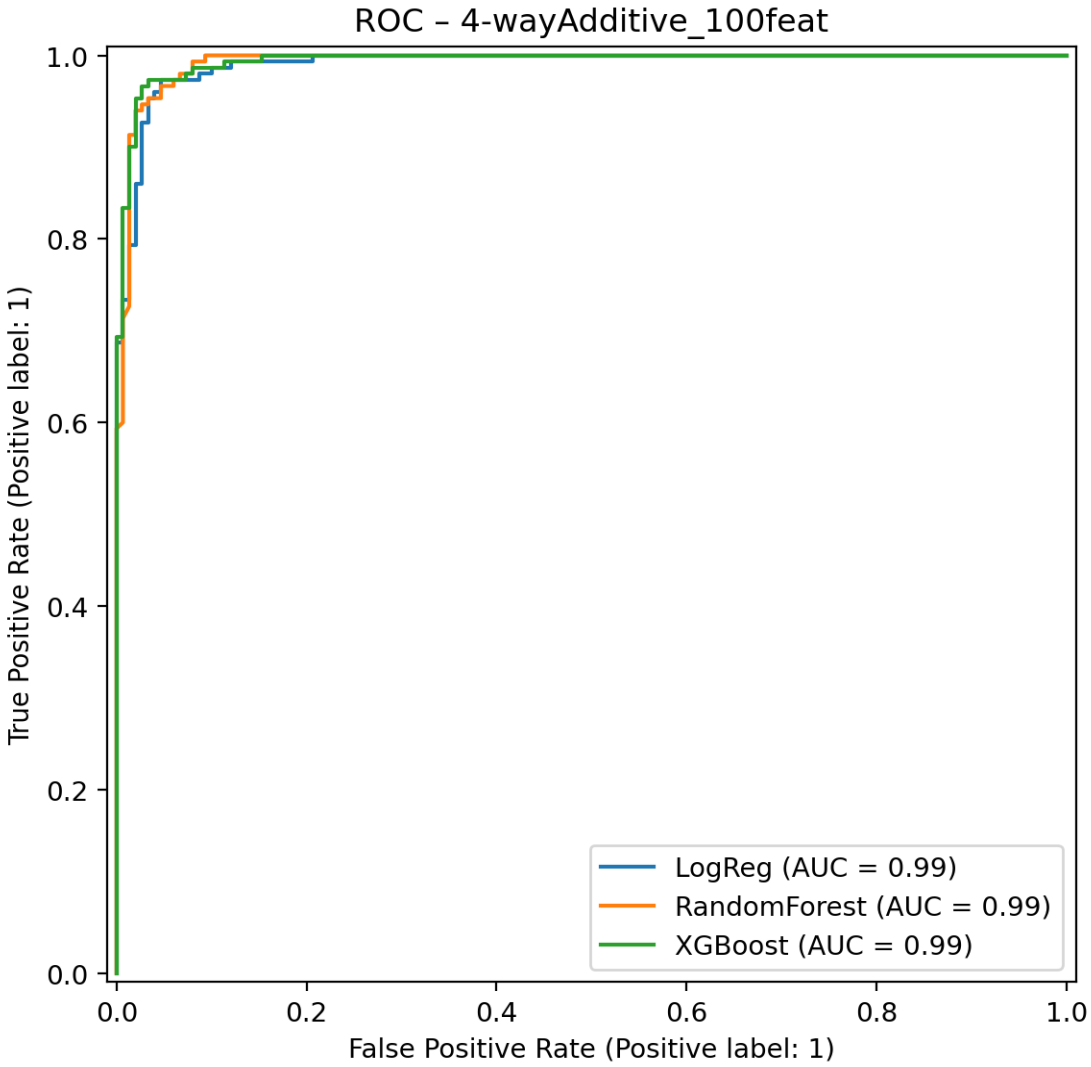

Figure 1. ROC – 4-way Additive Dataset

Table 2. Model Performance Comparison – 2-way Epistatic Dataset

Model	CV AUC (mean ± std)	Test Accuracy	Test F1	Test AUC
Logistic Regression [2]	0.451 ± 0.040	0.487	0.476	0.492
Random Forest [8]	0.523 ± 0.030	0.527	0.536	0.532
XGBoost [7]	**0.577 ± 0.060**	**0.640**	**0.645**	**0.694**

Interpretation: In the epistatic dataset, the performance of Logistic Regression collapsed to near-random levels due to its inability to model feature interactions. Ensemble methods improved results modestly, with XGBoost demonstrating the strongest ability to capture pairwise dependencies. These results align with the theoretical understanding of epistasis detection described in Relief-based feature-selection literature [4], [5].

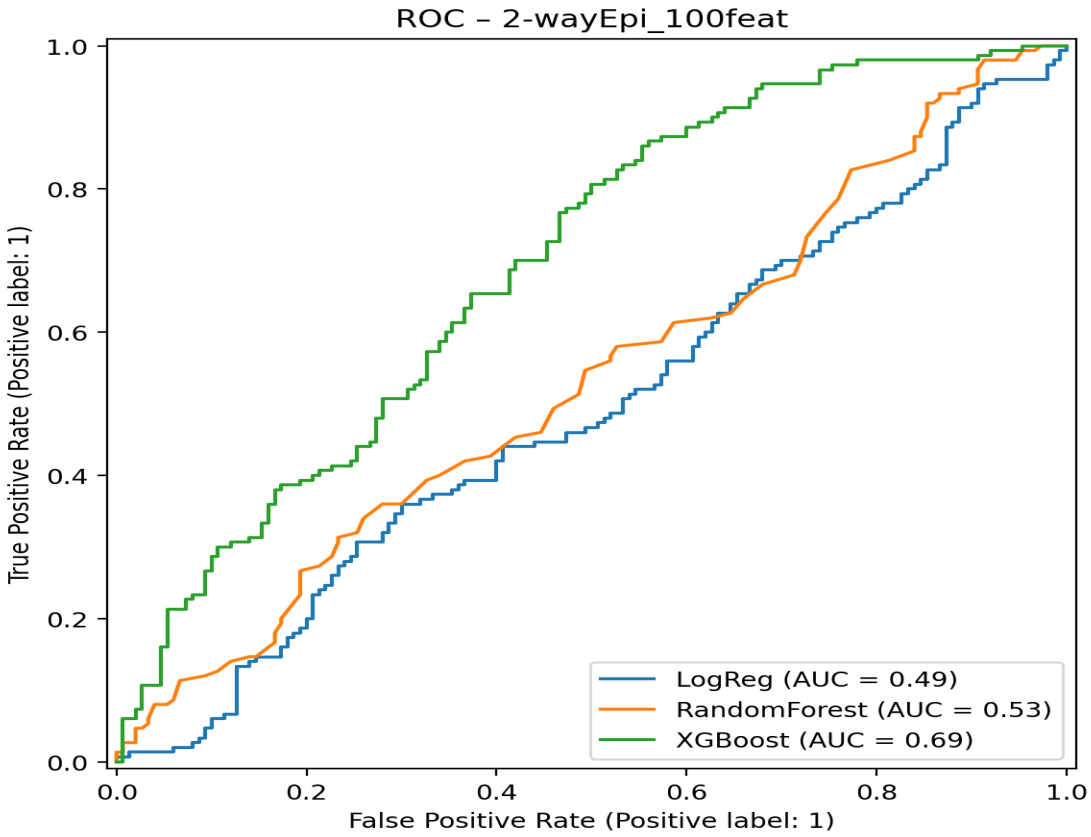

Figure 2. ROC – 2-way Epistatic Dataset

4.4 Visual Analysis

The pipeline generated **feature-importance plots** (Figures 3 and 4) for interpretability.

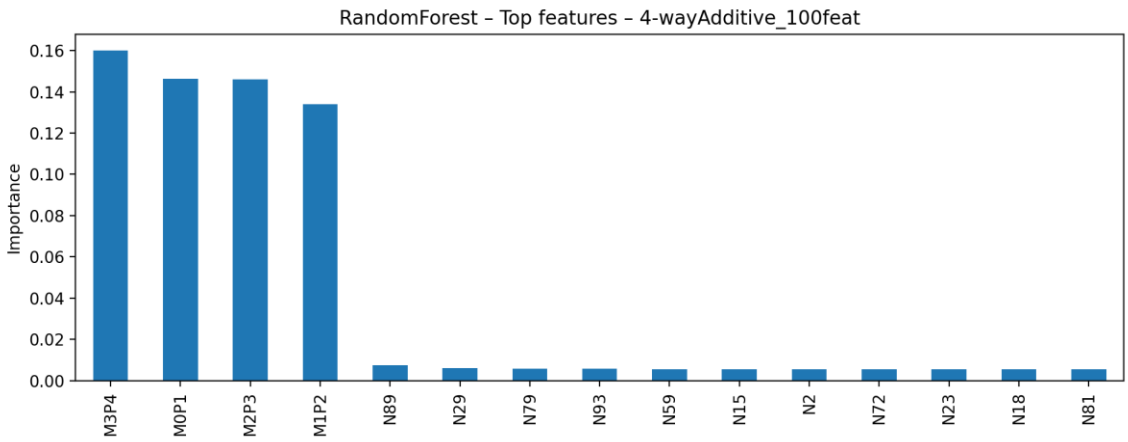

Figure 3. Feature Importance (Random Forest)
(Top 15 SNPs by importance for additive and epistatic datasets.)

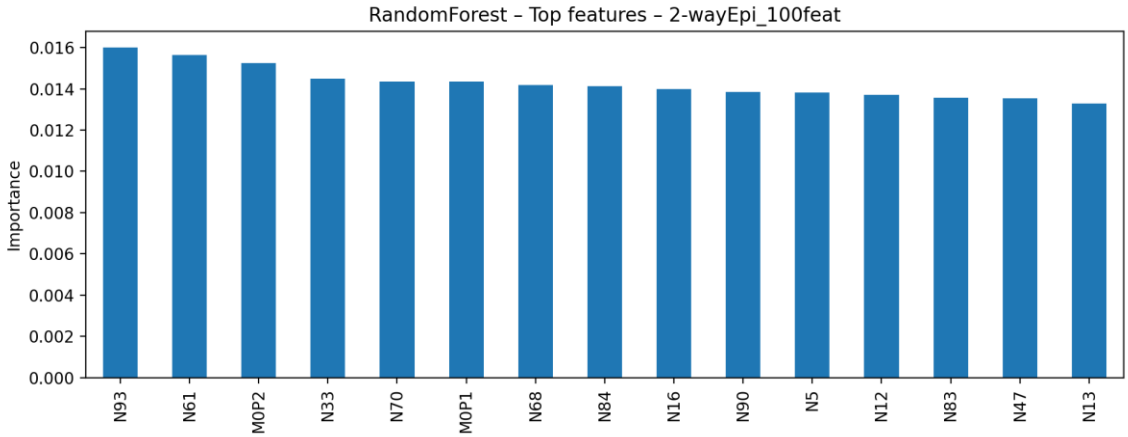

Figure 4. Random Forest – 2-way Epistatic Dataset
(Random Forest achieved weaker separation (AUC = 0.53) with dispersed importance across non-additive loci.)

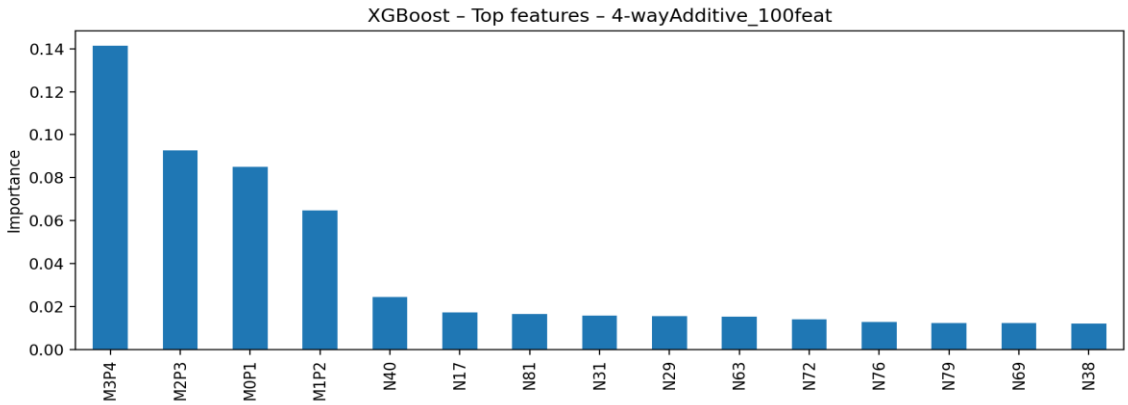

Figure 5. XGBoost – 4-way Additive Dataset (XGBoost reproduced the dominant additive features with higher stability and importance contrast (AUC = 0.99).)

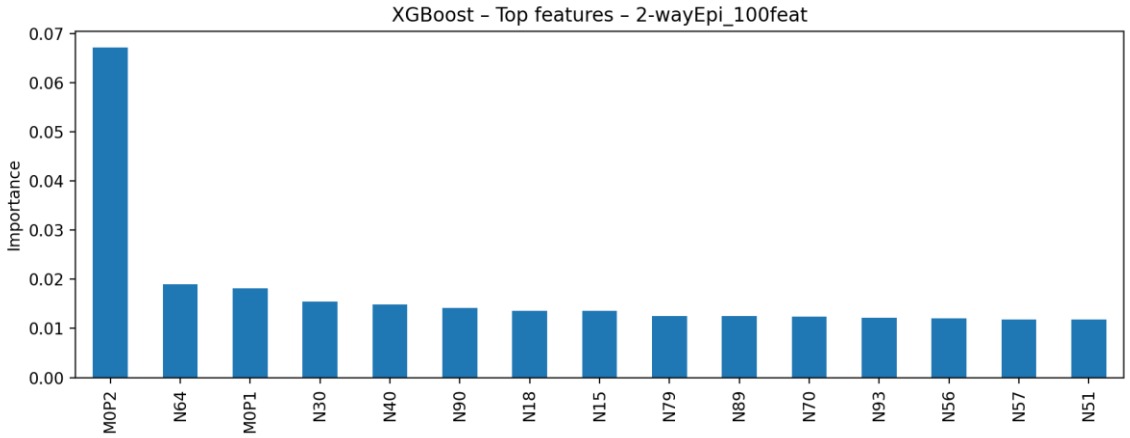

Figure 6. XGBoost – 2-way Epistatic Dataset
(XGBoost highlighted epistatic markers M0P2, M0P1, and N64 as top predictors, achieving AUC = 0.69 [7].)

Feature-importance visualizations confirmed that predictive SNPs (labeled "M") consistently ranked highest in the additive dataset, while the epistatic dataset displayed dispersed importance across multiple interacting features. This pattern validates the expected difficulty of isolating predictive loci in interaction-driven systems [3].

4.5 Challenges and Debugging

Several implementation challenges were encountered and resolved during this phase:

1. **XGBoost dependency errors:** The initial environment lacked the xgboost library. This was resolved by installing version 2.0.3 using pip install xgboost, ensuring compatibility with scikit-learn's pipeline interface [7].

2. **ROC overlay alignment:** Plot layering initially caused legends to overlap, which was fixed by using bbox_inches='tight' and unique color assignments.

3. **Feature-importance indexing:** Inconsistent SNP labeling caused KeyError exceptions when plotting. This was corrected by extracting the feature names directly from the pandas DataFrame after training.

4. **Reproducibility:** Random seed values were standardized across all steps, ensuring that AUC results remained consistent across multiple runs.

5. **Compute efficiency:** On the 2-way epistatic dataset, model convergence time was reduced by limiting max_depth=5 and using colsample_bytree=0.9, balancing training time and predictive power [7].

All issues were resolved successfully, and the pipeline now executes without errors on both datasets. The results produced in this stage form the analytical baseline for the next report, which will incorporate Relief-based feature-selection algorithms [4], [5] and higher-dimensional challenge datasets.

5. Next Steps

The upcoming phase of this project will focus on enhancing model interpretability and performance through feature-selection and higher-order interaction modeling. The following tasks are planned before the submission of **Report 3** and the **Final Report**.

1. **Integration of Relief-based Feature-Selection Algorithms:**
   ReliefF, MultiSURF, and MultiSURF* will be implemented to identify informative SNPs and reduce dimensionality prior to model training. This step will evaluate the ability of Relief-based algorithms to capture non-linear, multi-locus dependencies that conventional methods often miss [4][5].

2. **Extension to High-Dimensional Datasets:**
   The next experiment will incorporate the **4-way Heterogeneous_100feat.txt** dataset to analyze complex genetic architectures with both additive and epistatic components. This will help assess how well ensemble and boosting methods generalize to heterogeneity [3].

3. **Model Comparison and Ensemble Tuning:**
   Hyperparameter optimization will be introduced using a grid or Bayesian search to fine-tune Random Forest and XGBoost configurations. Key metrics such as ROC-AUC, precision-recall, and feature-importance stability will be benchmarked against the current baselines.

4. **Pipeline Automation via STREAMLINE Framework:**
   A semi-automated workflow inspired by STREAMLINE [6] will be developed to reproduce model runs efficiently, improving traceability and enabling direct comparison of multiple configurations across datasets.

5. **Final Statistical and Visual Analyses:**
   Comprehensive ROC overlays, PR-curves, and importance heatmaps will be included in the final report. Correlation analyses between feature-importance rankings across models will also be conducted to study consistency.

Timeline:

Phase	Task	Estimated Completion
Week 1	Implement Relief-based feature-selection algorithms	Mid-November 2025
Week 2	Run extended experiments on heterogeneous datasets	Late November 2025
Week 3	Integrate hyperparameter tuning and automation	Early December 2025
Week 4	Prepare **Report 3** (Feature Selection Results & Comparison)	Mid-December 2025
Final	Complete **Final Report & Presentation Submission**	Week of December 20 2025

6. References

1. R. J. Urbanowicz et al., "GAMETES: A fast, direct algorithm for generating pure, strict, epistatic models with random architectures," *BioData Mining*, vol. 5, pp. 1–14, 2012.
2. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
3. A. A. Woodward et al., "Genetic heterogeneity: Challenges, impacts, and methods through an associative lens," *Genetic Epidemiology*, vol. 46, no. 8, pp. 555–571, 2022.
4. R. J. Urbanowicz et al., "Relief-based feature selection: Introduction and review," *J. Biomed. Inform.*, vol. 85, pp. 189–203, 2018.
5. R. J. Urbanowicz et al., "Benchmarking Relief-based feature selection methods for bioinformatics data mining," *J. Biomed. Inform.*, vol. 85, pp. 168–188, 2018.
6. R. J. Urbanowicz et al., "STREAMLINE: A Simple, Transparent, End-To-End Automated Machine Learning Pipeline Facilitating Data Analysis and Algorithm Comparison," in *Genetic Programming Theory and Practice XIX*, Springer, 2023, pp. 201–231.
7. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
8. L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
9. D. Berrar, "Cross-Validation," in *Encyclopedia of Bioinformatics and Computational Biology*, Elsevier, 2019, pp. 542–545.
10. P. M. Visscher et al., "10 Years of GWAS Discovery: Biology, Function, and Translation," *Am. J. Hum. Genet.*, vol. 101, no. 1, pp. 5–22, 2017.