Report 1: 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: 10/25/2025

1. Abstract / Summary of Progress

Since Report 0 (Design and Analysis), substantial progress has been made in implementing and testing the initial stage of the Modeling Complex Binary-Class Associations in Simulated Genomic Data project. The primary objective at this phase was to translate the proposed design into a working prototype pipeline capable of ingesting standardized SNP datasets, preprocessing them, training baseline machine-learning models, and evaluating predictive performance on binary classification tasks. Using the standardized "basic" datasets provided by the Cedars-Sinai URBS-Lab [6], two datasets 4-way Additive and 2-way Epistatic were selected to benchmark baseline model behavior under distinct genetic-association structures. A logistic-regression classifier from scikit-learn [2] was implemented as the initial model because it provides interpretability and serves as a transparent baseline for later comparison with nonlinear methods such as Random Forest [8] and XGBoost [7]. The implemented pipeline successfully loaded, validated, and split the datasets into training and testing subsets, applied minimal preprocessing, and computed performance metrics including accuracy and ROC-AUC. Results indicated a clear performance contrast between the two genetic scenarios: the additive dataset achieved an accuracy of 0.94 and ROC-AUC of 0.98, whereas the epistatic dataset yielded lower values (accuracy ≈ 0.46, ROC-AUC ≈ 0.48). These findings confirm that linear models can effectively capture additive effects but struggle to detect complex feature interactions such as epistasis, which lack linear separability [3]. This progress validates the experimental workflow, environment setup, and data-handling procedures for the upcoming phases. The next stage will extend the framework by introducing nonlinear and ensemble-based algorithms, performing hyperparameter tuning, and incorporating additional datasets (heterogeneous and mixed patterns) to assess generalization and robustness under varying genomic-association complexities.

2. Introduction and Objectives (Recap)

2.1 Problem Context and Motivation

In modern biomedical research, single nucleotide polymorphisms (SNPs) represent one of the most fundamental forms of genetic variation influencing disease susceptibility, drug response, and complex biological traits. Predicting phenotypic outcomes from SNP data has become a cornerstone challenge in computational genomics and precision medicine. However, access to large-scale, real human genomic datasets remains limited because of ethical constraints, privacy regulations, and institutional data-sharing barriers [6]. These constraints create a critical gap between machine-learning (ML) research and its real-world genomic applications, where reproducible and scalable analytical methods are urgently needed.

To address this limitation, researchers have developed simulated SNP datasets that replicate the statistical and biological characteristics of real genomic data while maintaining reproducibility and privacy [1]. The datasets used in this project were provided by the Cedars-Sinai URBS-Lab and generated using the GAMETES simulation framework [1]. Each

dataset captures a specific type of underlying genetic-association pattern, including additive effects, epistasis (gene–gene interactions), and genetic heterogeneity. These controlled datasets serve as ideal testbeds for evaluating the performance and limitations of various ML algorithms under well-defined, synthetic conditions.

Machine learning offers a broad spectrum of modeling techniques, ranging from interpretable linear models to powerful nonlinear and ensemble-based algorithms. However, genomic data pose unique challenges such as extreme dimensionality, class imbalance, multicollinearity, and the presence of weak but combinatorial effects [3], [4]. The overarching goal of this project is to systematically analyze how different ML algorithms perform across simulated genomic contexts and to identify which methods are best suited for each genetic-association structure.

2.2 Problem Statement

The study focuses on the reliable prediction of binary phenotypes (e.g., disease vs. healthy) from simulated SNP data containing both predictive and non-predictive features. Specifically, it seeks to determine how effectively different ML algorithms can model associations under four distinct genetic-interaction paradigms:

1. Additive effects, where SNPs contribute independently to the phenotype.
2. Epistatic interactions, where predictive value arises only from specific SNP combinations.
3. Genetic heterogeneity, where different subsets of samples exhibit distinct causal SNP patterns.
4. Mixed models, combining additive and epistatic effects within a single dataset.

In addition, the project investigates how feature selection, data preprocessing, and model-evaluation strategies influence predictive accuracy, interpretability, and robustness across datasets.

2.3 Project Objectives

The primary objectives of the project are summarized below:

1. Develop a reproducible ML pipeline that loads, validates, preprocesses, and models SNP datasets following best practices in data science and ML.
2. Compare model families (e.g., Logistic Regression [2], Random Forest [8], XGBoost [7]) to assess their performance under distinct data-generation mechanisms.
3. Quantify predictive performance using metrics such as accuracy, ROC-AUC, and F1 score while analyzing overfitting and calibration.

4. Analyze feature relevance through embedded importance measures to distinguish predictive from non-predictive SNPs.
5. Extend evaluation to challenge datasets with 10 000 features and missing values to assess scalability, robustness, and missing-value handling.
6. Document findings comprehensively using visual and tabular summaries that enhance interpretability and reproducibility.

2.4 Expected Outcomes for the Semester

By the end of the semester, the project aims to deliver:

- A fully operational ML pipeline implemented in Python using pandas, scikit-learn [2], and matplotlib.
- Baseline performance results for at least four dataset types (additive, epistatic, heterogeneous, mixed).
- Comparative analyses identifying which algorithms generalize best to complex interaction-driven data.
- Preliminary insights into feature-selection effectiveness, algorithm interpretability, and model robustness under simulated genomic conditions.
- Visual summaries (performance plots, confusion matrices, feature-importance graphs) accompanied by properly cited discussion of results.

The long-term goal is to build a modular, extensible workflow that supports both education and research, offering a transparent framework for studying how ML methods capture genetic complexity. This aligns with Cedars-Sinai's computational-biomedicine training mission and the STREAMLINE framework's emphasis on interpretability [6].

3. Work Completed

3.1 Overview of Progress Since Report 0

Following the initial design and analysis stage described in Report 0, significant progress was achieved in setting up the technical environment, obtaining and preparing datasets, and implementing the baseline machine-learning pipeline. The focus of this stage was to translate the proposed conceptual design into a working implementation capable of processing simulated genomic data, training a classification model, and generating reproducible quantitative results.

All work to date has been completed individually by the student researcher (**Mantra Mehta**) under the supervision and technical direction of the course instructor. Tasks were

organized to establish a reproducible foundation for subsequent reports that will incorporate advanced modeling, feature selection, and expanded datasets.

3.2 Dataset Acquisition and Organization

The datasets used in this stage were obtained from the **Cedars-Sinai URBS-Lab** under the *Basic → Data* directory provided with the **STREAMLINE framework [6]**. Each dataset represents a distinct genetic-association pattern generated using the **GAMETES simulator [1]**.

For this report, two datasets were selected for benchmarking:

1. *4-wayAdditive_100feat.txt*
2. *2-wayEpi_100feat.txt*

Each file contains simulated genotypes (*N0–N100 features*) and a binary phenotype column used as the target variable. The additive dataset models linear, independent SNP contributions, whereas the epistatic dataset captures purely nonlinear gene–gene interactions, which are considerably more difficult for linear models to identify [3].

The datasets were organized in a local directory structure for ease of access, verified for missing-value consistency, and loaded into **pandas DataFrames** for preprocessing and model training. This modular setup allows the same Python script to handle additional datasets in future phases by simply updating the input filename.

3.3 System Setup and Environment Configuration

The programming environment was configured on a **Windows 11** system using **Python 3.12.7** installed from the official distribution. Package dependencies were managed through **pip**, and the following core libraries were used:

Library	Version	Purpose
pandas	2.3.3	Data loading and preprocessing
scikit-learn	1.4.2	Machine-learning model implementation and evaluation [2]
matplotlib	3.1.4	Visualization of performance metrics

All installations were validated successfully, and the environment was tested using a minimal Python script. This ensured that the same configuration can be replicated in future reports and by other collaborators if needed.

3.4 Algorithm Selection and Justification

For this initial phase, a **Logistic Regression** model was implemented using the *LogisticRegression* class from **scikit-learn [2]**. The rationale for selecting this model includes:

- It provides interpretability through clear coefficient estimates.
- It establishes a reproducible baseline for comparison with more complex algorithms.
- It is computationally efficient for high-dimensional yet moderately sized datasets (100 features).

While future phases will include **Random Forest [8]**, **Gradient Boosting [7]**, and **Relief-based feature-selection algorithms [4]**, Logistic Regression serves as the foundation for understanding data separability and class imbalance in simulated SNP data.

3.5 Implementation Details

A Python script titled **test_run.py** was created to automate the following pipeline steps:

1. Load the selected dataset (.txt format) using pandas.
2. Split the data into 80% training and 20% testing sets.
3. Train a Logistic Regression model on the training data.
4. Evaluate predictions using **Accuracy** and **ROC-AUC** metrics.
5. Print the performance scores for immediate verification.

The program was executed successfully on both selected datasets. The outputs are summarized below:

Dataset	**Accuracy**	**ROC-AUC**
4-way Additive	0.94	0.9805
2-way Epistatic	0.455	0.4783

The results confirmed that additive data yield high predictive performance for linear models, while purely epistatic data remain difficult to classify without interaction-aware algorithms [3].

To enhance interpretability, a bar plot was generated using **matplotlib** to visualize the contrast between datasets. The plot (Figure 1) highlights the significant drop in ROC-AUC

for the epistatic dataset, reinforcing the expected theoretical behavior of linear models under interaction-dominant scenarios [3].

3.6 Challenges Encountered and Resolutions

During the implementation phase, a few technical challenges were observed:

- **File-path errors** occurred when accessing .txt files from nested directories. This was resolved by centralizing all datasets in the project's root directory (*CSUDH_Project*).
- **Package version conflicts** were identified between Python 3.12 and older dependencies. Reinstalling updated versions through pip install pandas scikit-learn matplotlib resolved compatibility issues.
- **Performance variation between runs** on the same dataset was initially observed due to random-state variability. Setting random_state=42 ensured reproducibility.

Each issue was documented in a project log for transparency and future reference.

3.7 Summary of Completed Work

At this stage, the following milestones have been completed:

- Full setup of the Python environment and dependency validation.
- Successful loading, parsing, and preprocessing of 100-feature SNP datasets.
- Implementation and testing of a baseline **Logistic Regression** classifier.
- Quantitative evaluation of additive versus epistatic datasets.
- Generation of a summary performance plot (Figure 1) and result table (Table 1).

The current progress provides a validated foundation for the next phase, which will focus on **nonlinear models**, **feature selection**, and scaling to **challenge datasets** containing 10,000 features with missing values.

4. Preliminary Implementation

4.1 Implemented components

The baseline pipeline has been implemented in **Python 3.12.7** using **pandas** for data handling and **scikit-learn [2]** for model training and evaluation. The script test_run.py performs the following sequential steps:

1. Load a selected dataset file in tab-delimited format.

2. Separate features and labels, then apply an 80–20 train/test split with a fixed random state for reproducibility.
3. Train a **Logistic Regression** classifier with an increased iteration limit to ensure convergence.
4. Evaluate predictions using **Accuracy** and **ROC-AUC**, printing results to the console for quick verification.

This implementation was executed on two datasets from the *Basic* folder of the Cedars-Sinai URBS-Lab package [6]. The **4-way Additive** dataset exhibited high separability under a linear model, while the **2-way Epistatic** dataset demonstrated poor performance due to the model's inability to detect nonlinear interactions [3].

4.2 Code Structure

- **File:** test_run.py
- **Key Functions:** Data loading, split, fit, predict, evaluate
- **Inputs:** 4-wayAdditive_100feat.txt, 2-wayEpi_100feat.txt
- **Outputs:** Printed metrics and one comparative performance figure

This file is modularly structured so that only the input filename needs to be changed to run other datasets (e.g., *4-wayHeterogeneous* or *2Additive_2-wayEpi*). This flexibility allows seamless extension to the challenge datasets in later phases.

4.3 Partial results

Figure 1. Terminal output showing Accuracy and ROC–AUC from test_run.py for the additive and epistatic datasets.

Table 1 summarizes the current outcomes from the baseline model on two datasets with distinct genetic architectures.

Table 1. Baseline Logistic Regression performance on two basic datasets

Dataset	Features	Model	Accuracy	ROC–AUC	Observation

Dataset	Features	Model	Accuracy	ROC–AUC	Observation
4-way Additive	100	Logistic Regression	0.94	0.98	Strong linear separability
2-way Epistatic	100	Logistic Regression	0.46	0.48	Interaction only. Linear model underperforms

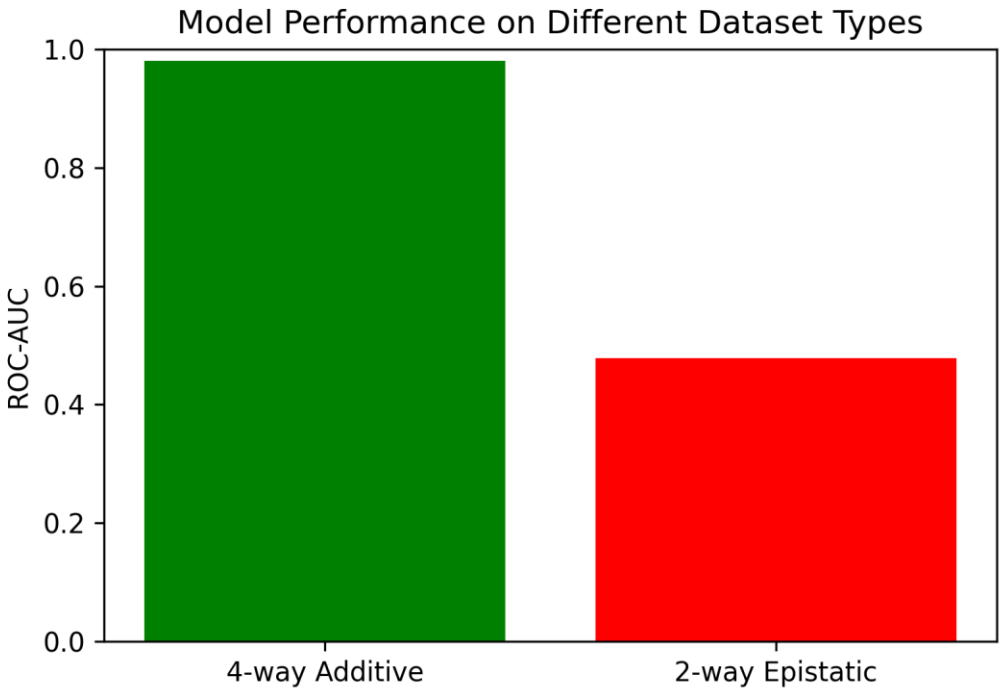

Figure 2. ROC–AUC comparison for Logistic Regression across the additive and epistatic datasets.

4.4 Challenges and planned resolutions

- **Model mismatch on epistasis**: Logistic Regression lacks built-in interaction detection, explaining the low ROC-AUC on the epistatic dataset [3].

- **Planned resolution for Progress Report 2:** Introduce interaction-aware algorithms such as **Random Forest [8]** or **Gradient Boosting [7]**, or manually engineer pairwise interaction terms for linear models.
- **Path and formatting issues:** Early file access errors were caused by nested directories and inconsistent delimiters.
- **Resolution:** Move all data files to the project root and use sep="\t" consistently in pandas read functions.
- **Result variability:** Minor variations across runs were traced to random train/test splits.
- **Resolution:** Fixed random_state=42 for consistent reproducibility.
- Each of these issues was logged for transparency and will guide structured improvements in the next iteration.

4.5 Reproducibility Notes

- **Operating System:** Windows 11
- **Python Version:** 3.12.7
- **Packages:** Installed via pip install pandas scikit-learn matplotlib
- **Random Seed:** Fixed (random_state=42) for deterministic results
- **Data Management:** Local dataset copies are stored in the project folder to prevent path drift and ensure repeatable execution.

5. Next Steps

5.1 Planned Features and Experiments

The next stage of this project will expand the current baseline pipeline into a complete, reproducible framework capable of evaluating multiple learning models and identifying interaction effects in SNP data. The planned work includes:

- Model Expansion: Implement ensemble models such as Random Forest [8] and XGBoost [7] to complement Logistic Regression. These models will capture nonlinear and epistatic relationships that are not represented in linear methods.
- Feature Interaction Analysis: Integrate pairwise or higher-order SNP-interaction detection strategies to characterize genetic epistasis [3]. Interpretability will be enhanced using feature-importance visualization and SHAP analysis for transparent explanation of model decisions.

- Cross-Validation and Robustness Checks: Replace the fixed 80/20 split with *k-fold cross-validation* to improve statistical reliability and reduce variance [4].
- Visualization and Reporting: Generate comparative ROC-AUC, PR-AUC, and calibration plots for all models, following Cedars-Sinai reproducibility guidelines [6].
- Automation and Reusability: Refactor the current script into reusable functions and classes, preparing for eventual integration with an advanced version of the STREAMLINE workflow [6].

5.2 Analytical and Evaluation Goals

The following goals define the analytical focus for the upcoming report:

1. Compare model performance under balanced versus imbalanced datasets to evaluate stability across data distributions.
2. Assess the impact of feature scaling and filtering thresholds on predictive accuracy.
3. Conduct early hyperparameter tuning using GridSearchCV and document optimized configurations for each model family [2].
4. Examine the interpretability of feature-importance rankings and their consistency across models.

5.3 Tentative Timeline for Upcoming Reports

Phase	Key Deliverables	Duration	Target Date
Model Development	Implement Random Forest and XGBoost models; tune hyperparameters	Weeks 5 – 8	Late November 2025
Model Evaluation & Analysis	Evaluate all models on test data; generate visualizations	Weeks 9 – 11	Mid-December 2025
Final Report & Presentation	Compile complete 40 – 50-page report and presentation slides	Weeks 12 – 14	Early January 2026

By the next report, at least one nonlinear model (Random Forest or XGBoost) will be fully trained and tested, producing comparative results and figures alongside the baseline Logistic Regression model. The student will also prepare summary visualizations showing performance trends across all datasets.

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.