Machine Learning Pipeline for Detecting PCR-Induced Chimeric Reads

MitoChime: Organellar Chimera Detection from Per-Read Features

Duran, Lin, Pailden

University of the Philippines Visayas
Philippine Genome Center Visayas

December 8, 2025

Outline

1 Objectives

2 Scope and Limitations

3 Methodology

General Objective

- Develop and evaluate a machine-learning pipeline (MitoChime) to detect PCR-induced chimeric reads in *S. lemuru* mitochondrial sequencing data to improve downstream assembly quality.

Specific Objectives

1. Construct simulated *Sardinella lemuru* Illumina paired-end datasets contain ing both clean and PCR-induced chimeric reads.

2. Extract alignment-based and sequence-based features such as k-mer composition, junction complexity, and split-alignment counts from both clean and chimeric reads

3. Train, validate, and compare supervised machine learning models for classifying reads as clean or chimeric.

4. Determine feature importance and identify indicators of PCR-induced chimerism.

5. Integrate the optimized classifier into a modular and interpretable pipeline deployable on standard computing environments at PGC Visayas.

Scope of the Study

- Focuses on PCR-induced chimeric reads in *Sardinella lemuru* mitochondrial sequencing data to:
    - to limit interspecific variation in mitochondrial genome size, GC content, and repetitive regions so that differences in read patterns can be attributed more directly to PCR-induced chimerism
    - to align the analysis with relevant *S. lemuru* sequencing projects at PGC Visayas
    - to take advantage of the availability of *S. lemuru* mitochondrial assemblies and raw datasets in public repositories such as the National Center for Biotechnology Information (NCBI), which facilitates reference selection and benchmarking
    - to develop a tool that directly supports local studies on *S. lemuru* population structure and fisheries management produce tools applicable to local population and fisheries studies

Scope of the Study

- Uses wgsim-based simulations and selected empirical mitochondrial datasets
- Analysis targets low-dimensional alignment and sequence features (k-mers, GC content, clipping, split alignments) to maintain interpretability and computational accessibility
- Long-read platforms (Nanopore, PacBio) and other taxa are not included

Key Exclusions

- Naturally occurring chimeras
- NUMTs
- Large-scale nuclear genome rearrangements
- High-dimensional deep learning embeddings

Other Limitations

- No simulations with variable sequencing error rates
- No testing of alternative parameter settings (k-mer length, microhomology windows)
- Reliance on supervised machine learning may limit detection of novel/unknown chimeric patterns

Methodology

Figure: Process Diagram of the Special Project

Data Collection

The *S. lemuru* mitochondrial reference genome (NCBI: NC_039553.1) was downloaded in FASTA format and used as the basis for generating simulated reads.

Data Preprocessing

- A Python script was used to generate the reads.
- Clean reads were produced with wgsim from the reference genome.
- A chimeric reference was created by creating a custom script to combine non-adjacent segments with microhomology
- Chimeric reads were simulated with wgsim.
- All reads were mapped with minimap2 to extract alignment information.
- SAM/BAM files were converted, sorted, and indexed with samtools.

Data Preprocessing

- Final dataset: 40k reads, roughly balanced between clean and chimeric (19,984 clean reads and 20,000 chimeric).
- Some of the clean reads failed to align due to the set error rate.

Data Preprocessing

Figure: SAM File of Clean Reads

Data Preprocessing

Figure: SAM File of Chimeric Reads

Feature Extraction Pipeline

- BAM files were processed with a Python script to build a TSV feature matrix.
- Used Pysam for parsing alignments and NumPy for computation.

Feature Extraction Pipeline

- Focused on three features linked to PCR-induced chimeras:
  1. **Supplementary Alignment (SA)**: Detects split alignments; counts and metrics extracted from SA tags
  2. **K-mer Composition Difference**: Breakpoints inferred; left/right segments compared using cosine and JS metrics.
  3. **Microhomology**: Overlap at junction quantified (length + GC content) within a defined window.
- Pipeline design and outputs to be validated by experts.

Feature Extraction Pipeline

../figures/clean_excel.png

Feature Extraction Pipeline

../figures/chimeric_excel.png

Dataset construction and split

- Simulated feature tables:
  - Clean reads (label 0)
  - PCR-induced chimeras (label 1)
- `build_datasets.py`:
  - Concatenate tables
  - Shuffle rows (avoid file-order artefacts)
- 80/20 **stratified** train–test split
- Test set held out and used **only once** at the end

Validation strategy

- Layer 1: 80/20 stratified train–test split
- Layer 2: 5-fold stratified cross-validation on training set
  - Train on 4 folds, validate on 1
  - Rotate so each fold is validation once
- Layer 3: Final evaluation on held-out test set
- Hyperparameter tuning:
  - RandomizedSearchCV inside CV for top models
- Goal: stable estimates and **unbiased** test performance

Model zoo and preprocessing pipeline

- **Baseline:** dummy majority-class classifier
- **Linear models:** logistic regression, calibrated linear SVM
- **Tree ensembles:**
    - Random Forest, Extra Trees
    - Gradient Boosting, XGBoost, LightGBM, CatBoost
- **Others:** bagging trees, k-NN, Gaussian NB, shallow MLP
- Common scikit-learn pipeline:
    - Median imputation (numeric missing values)
    - Standardisation (zero mean, unit variance)
- Ensures a **fair comparison** across models
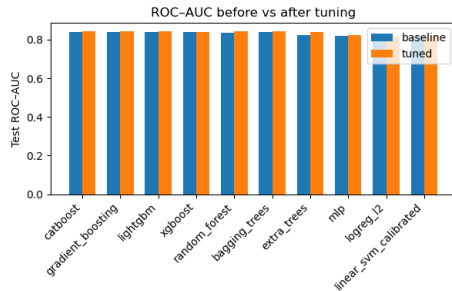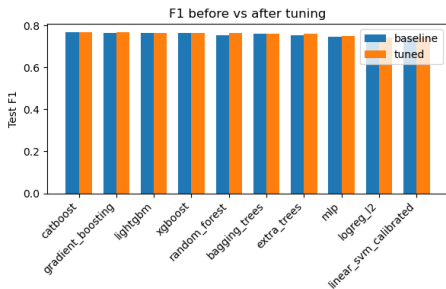
Effect of hyperparameter tuning

Figure: Test F1: baseline vs tuned.

Figure: Test ROC–AUC: baseline vs tuned.

- Tuning done with `RandomizedSearchCV` on training set
- Small but consistent gains (F1, AUC ≈ 0.001–0.01)
- Top-ranked models remain the same (CatBoost, Gradient Boosting, LightGBM)
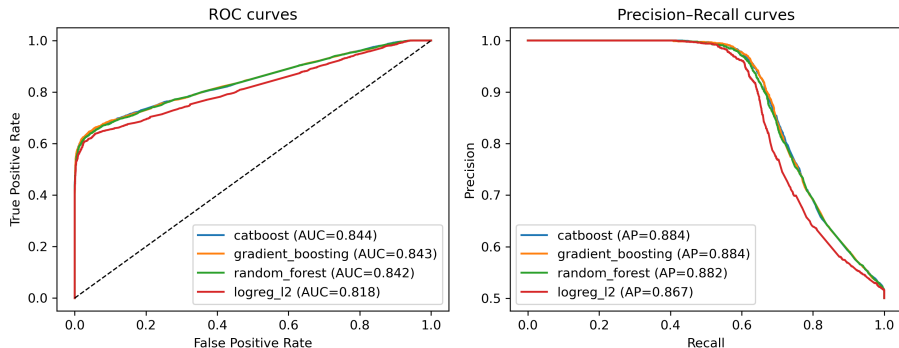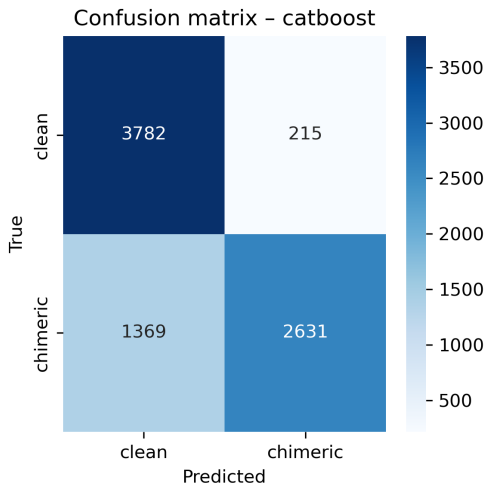
ROC and precision–recall curves

Figure: ROC (left) and PR (right) curves for CatBoost, Gradient Boosting, Random Forest, and logistic regression.

- Ensembles: ROC–AUC ≈ 0.84; logreg: ≈ 0.82
- Average precision ≈ 0.88 for ensembles
- Precision > 0.9 up to recall ≈ 0.5–0.6

Figure: Confusion matrix heatmap for CatBoost.

- Clean reads:
  - Recall ≈ 0.95 (3782 / 3997)
- Chimeric reads:
  - Precision ≈ 0.92
  - Recall ≈ 0.66 (2631 / 4000)
- Behaviour at default threshold:
  - **Conservative chimera filter**
  - Protects clean reads, misses some subtle chimeras
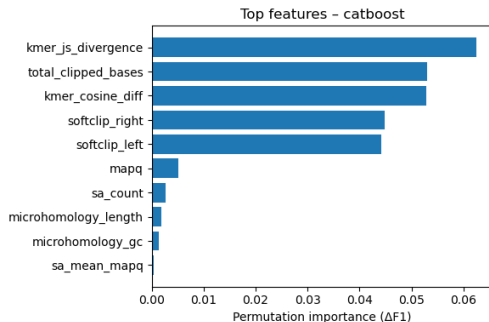
Top features for CatBoost

Figure: Permutation importance (F1) for CatBoost.

- Strongest signals:
  - `kmer_js_divergence`
  - `total_clipped_bases`
  - `kmer_cosine_diff`
- Also important:
  - Left/right soft-clipping
  - Mapping quality (MAPQ)
  - SA count (supplementary alignments)
- Consistent with PCR chimera junctions
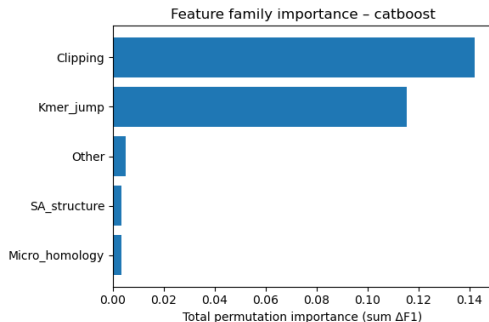
Feature family importance

Figure: Aggregated feature families for CatBoost.

- Aggregated permutation importance:
  - **Clipping** features dominate
  - **K-mer jump** features also strong
- Smaller contributions:
  - SA structure
  - Micro-homology
  - Other alignment context
- Same pattern for Gradient Boosting and Random Forest

ML component: summary and implications

- Per-read classifier for clean vs PCR chimeric reads
- Evaluation:
    - Stratified 80/20 split, 5-fold CV, held-out test set
- Best models: tree-based ensembles
    - CatBoost, Gradient Boosting, LightGBM
    - Test F1 $\approx$ 0.77, ROC–AUC $\approx$ 0.84
- Default threshold:
    - Conservative chimera filter (high clean recall, high chimera precision)
    - Removes $\sim$2/3 of chimeras
- Features match PCR chimera biology
- Practical, interpretable pre-filter before mitochondrial assembly