"Comparative Analysis of Bagging and Boosting Ensembles on Noisy Multi-Class Genomic Data"

**Project Summary**

**Part 1 – Data Loading and Noise Injection**

* Data Loading:  
  Loaded genomic features (2000 features) and labels (6 classes: BC, GBM, KI, LU, OV, U) from tab-separated files with no missing values.
* Noise Injection Procedure:  
  Created synthetic "noisy" data points by sampling from per-class normal distributions (based on original data mean and variance). Each synthetic sample inherits feature values from its genuine class distribution but is deliberately assigned a different, incorrect class label ("label noise").  
  Generated 5 datasets with noise levels: 0%, 20%, 45%, 70%.
* Data Storage:  
  Noisy datasets saved as numpy .npy files for faster reuse.

**Part 2 – Bagging with Random Forests**

* Concept:  
  Bagging (Bootstrap Aggregating) creates multiple bootstrap subsets from data, trains multiple Random Forests in parallel, and aggregates their votes for classification.
* Parameter Tuning:  
  Used GridSearchCV on 0% noise data to select n\_estimators=20 for optimal performance.
* Training:  
  Trained BaggingClassifier with Random Forest base estimators on noisy datasets at each noise level.
* Evaluation:  
  Evaluated with weighted recall (important for medical context due to false negative cost). Observed performance decreased linearly with noise, but baseline recall was very high (~99% at 0 noise).
* Feature Importance:  
  Used permutation importance to identify impactful features without black-box assumptions or retraining.
  + Computed importance for each noise level.
  + Found that noise increases the number of "important" features (from 53 to 842).
  + Top features at 0% noise mostly disappear with added noise; only a few features remain consistently important (e.g., Feature 3, 324).

**Part 3 – Boosting with Gradient Tree Boosting**

* Concept:  
  Used Gradient Boosting Classifier, which trains sequential weak learners (shallow trees), each correcting errors of the previous, combined into a strong classifier.
* Parameter Tuning:  
  GridSearchCV revealed best parameters on 0% noise: max\_depth=3, n\_estimators=15.
* Training and Evaluation:  
  Trained models for each noise level and evaluated weighted recall. Like bagging, recall decreased with noise but started near 98%.
* Feature Importance:  
  Computed permutation importance similarly to bagging for each noise level.
  + Noted greater feature importance stability and consistency across noise levels compared to bagging.
  + Number of important features ranges from 50 to 100, less sensitive to noise magnitude.

**Part 4 – Comparison and Insights**

* Performance:  
  Both bagging and boosting achieved near-identical recall at 0% noise. They both degrade gracefully as noise increases, with recall dropping roughly linearly.
* Feature Selection Consistency:  
  Boosting showed much better consistency in selecting top features across different noise levels, suggesting greater robustness for feature interpretability.
* Common Important Features:  
  Both methods agreed on several key features at 0% noise, including Features 3 and 1871 in the top 10.  
  Analysis of top 25% important features showed overlap on 5 key features, highlighting some biological signal captured by both methods.
* Implications:
  + While bagging is effective and simple, boosting offers more stable feature explanations, beneficial when interpreting genomic data for medical applications.
  + Noise makes models consider more features as important (potential spurious relationships), but boosting mitigates this effect better.

**Interview Preparation: Key Concepts**

**Data Processing & Noise Handling**

* Why add label noise via synthetic samples in this study?  
  To simulate real-world data imperfections and test model robustness under class mislabeling, which is common in medical datasets.
* How is this noise generated?  
  Per-class feature distributions modeled and synthetic points sampled; class labels purposely mismatched to mimic label noise.

**Bagging (Bootstrap Aggregating)**

* What is bagging, and how does it reduce variance?  
  Bagging trains multiple models on bootstrapped data subsets independently and aggregates their outputs, reducing overfitting by averaging out noise-induced variability.
* Why use Random Forests as base estimators?  
  They are robust, easy to train ensembles of decision trees, especially suited to high-dimensional data as used here.
* How do you select bagging parameters?  
  Grid search cross-validation helps find the optimal number of estimators balancing performance and compute.
* Why recall as a metric for evaluation?  
  In medical diagnosis, minimizing false negatives (high recall) is crucial because missing a positive case has severe consequences.

**Boosting (Gradient Boosting)**

* What distinguishes boosting from bagging?  
  Boosting trains weak learners sequentially, each correcting previous errors, producing a strong aggregated model, while bagging trains independent learners in parallel.
* Why shallow trees (max\_depth=3) for boosting?  
  Shallow trees prevent overfitting and ensure each learner focuses on learning residual errors incrementally.
* How does boosting balance bias and variance?  
  By sequentially correcting residuals, boosting reduces bias and variance simultaneously.

**Feature Importance & Model Interpretability**

* What is permutation importance, and why is it used here?  
  It measures feature impact by evaluating the effect on model performance when feature values are shuffled, requiring no retraining and applicable to any model.
* What limitations does permutation importance have?  
  Correlated features can affect interpretations; overfitting models may produce misleading importance scores.
* What does feature importance consistency indicate?  
  Stability of selected features across noise levels suggests a model’s interpretability and trustworthiness, critical in sensitive domains like genomics.

**General Machine Learning**

* Why store noisy datasets separately?  
  Avoid repeating computationally expensive noise generation, facilitating reproducibility and efficiency.
* How does noise affect model performance and interpretability?  
  Performance degrades as noise reduces the signal-to-noise ratio; model interpretability also suffers as noise introduces spurious feature relationships.