Project:
Machine Learning Competition

Student BOUNOUA Ilyas (28372410)
Professors Pierre Dupont - Ronval Benoît

Introduction

This report summarizes the methodology, design choices, algorithms, and evaluation methods employed to predict gene mutation status (active or inactive) using machine learning techniques. The task is inspired by a biomedical research scenario aiming to classify gene mutations based on provided features.

I Design Choices and Preprocessing

Given a dataset of 14,958 input features (14,908 numerical and 50 categorical), preprocessing was essential. The chosen pipeline was:

- **Numerical features**: Imputed missing values using median imputation and applied Robust Scaling to reduce the influence of outliers.
- **Categorical features**: Missing values were imputed using the most frequent category, followed by one-hot encoding, ensuring consistent handling of unseen categories in the test set.
- **Dimensionality Reduction**: Principal Component Analysis (PCA) was applied to reduce dimensionality, selecting the first 100 principal components based on randomized SVD.

This comprehensive preprocessing pipeline ensured robustness to outliers, handled missing values appropriately, and significantly reduced computational complexity by feature reduction.

II Algorithms and Software

The classification task employed an ensemble learning approach to leverage the predictive strengths of multiple models:

II.1 Base Learners

Three robust classifiers were selected:

- **Random Forest**: Utilized due to its efficiency in handling complex data and feature interactions, configured with 200 estimators, maximum depth of 20, and balanced class weights.
- **XGBoost**: Known for high performance in classification tasks, set with 200 estimators, maximum depth of 6, a learning rate of 0.1, and automatic balancing of classes via *scale_pos_weight*.
- **LightGBM**: Chosen for its speed and efficiency on large datasets, configured similarly to XGBoost with 200 estimators, maximum depth of 6, and a learning rate of 0.1, maintaining balanced class weights.

II.2 Stacked Ensemble

A stacking ensemble was implemented, combining predictions from these base learners using logistic regression as the meta-learner. This approach captures diverse patterns from each model and significantly improves predictive performance.

III Evaluation Methodology

Performance was assessed using Stratified 5-fold cross-validation with the Balanced Classification Rate (BCR), defined as:

$$\text{BCR} = \frac{1}{C} \sum_{i=1}^{C} \frac{TP_i}{n_i} \tag{1}$$

where TP_i denotes correctly classified examples of class i, and n_i represents total examples from class i. Cross-validation yielded a mean BCR of 0.8794 ($\pm$0.0273).

IV Prediction and Submission

After validation, the final stacked model was trained on the entire training set, and predictions were made for the provided test set. Predictions were mapped back to original class labels ('active', 'inactive') and saved accordingly.

V Handling Missing Values and Dimensionality

The methodology explicitly addressed missing data by median and mode imputation, chosen for their simplicity and effectiveness. PCA was critical in handling the high dimensionality, ensuring efficient model training without significant loss of information.

VI Comparison of Approaches

The ensemble method outperformed single classifiers, confirmed by higher cross-validation BCR scores. Base models individually provided solid results, but their combination via stacking effectively improved overall robustness and accuracy.

VII External Resources and Tools

The solution extensively utilized Python libraries:

- **scikit-learn**: preprocessing, PCA, and model implementations (`https://scikit-learn.org`).
- **XGBoost and LightGBM**: Efficient gradient boosting implementations (`https://xgboost.readthedocs.io`, `https://lightgbm.readthedocs.io`).

This project was developed with assistance from OpenAI's ChatGPT (`https://chat.openai.com/`).

Conclusion

The stacking ensemble provided strong predictive accuracy, validated through rigorous cross-validation, achieving a competitive BCR. Future improvements could explore deeper hyperparameter tuning or additional feature engineering.