Classification: The Immune Landscape of Cancer using RNA-seq Expression Data

Project Description

This project implements a supervised classification pipeline to predict tumour immune subtypes (C1–C6) from RNA-seq gene expression profiles. The data consists of expression levels for 440 immune-related genes across ~9,126 cancer samples from the TCGA Pan-Cancer Atlas (with labels defined by Thorsson et al. 2018). The goal is to build a high-performing and interpretable ML model to accurately classify each sample's immune subtype. Key steps include data preprocessing, dimensionality reduction (PCA), training multiple classifiers, and thorough performance evaluation.

Dataset Summary

The analysis uses a pan-cancer immune dataset derived from TCGA. After filtering and aligning sample IDs, the final modelling dataset contains 9,126 patient samples (rows) and 440 gene expression features plus a subtype label. The immune subtypes C1–C6 are unevenly represented (e.g. C2 and C1 have the most samples, C6 the fewest), motivating careful handling of class imbalance. The gene expression data were originally normalized (TPM) and have a wide dynamic range, necessitating scaling in preprocessing.

Environment and Execution

This project was developed in a Jupyter Notebook running on a local Anaconda Python environment (Python 3.x). All code is executed interactively in the notebook. To run the analysis, clone the repository, install the required libraries (see Dependencies or requirements.txt), then open the notebook (.ipynb) in Jupyter and execute each cell sequentially. The notebook is well-documented, with explanations and outputs for each step.

Notebook Structure and Tasks Overview

- **Dataset Loading** – Load the full RNA-seq expression matrix (Pan-Cancer Atlas) and immune subtype labels.
- **Data Structuring** – Parse and align sample identifiers, transpose data as needed, and merge expression values with subtype labels.
- **Balanced Sampling** – Perform stratified sampling to address class imbalance. 2009 samples are selected (approximately 385–462 per subtype) to create a balanced training set.
- **Preprocessing** – Conduct data quality checks and transformations: handle missing values using iterative imputation (MICE), treat outliers via IQR-based winsorization, apply SMOTE oversampling to equalize classes, and normalize features (Min–Max scaling) for modelling.
- **Visualization** – Generate exploratory plots (boxplots, violin plots, PCA scatter plots, heatmaps) to inspect feature distributions and sample patterns.
- **Train-Test Split** – Split the balanced dataset into training (80%) and test (20%) sets while preserving subtype proportions.

- **Model Training** – Train multiple classifiers on the training set, including Logistic Regression, Random Forest, XGBoost, MLP (neural network), and LightGBM. (Support Vector Machine is also used in comparisons/ensembles.)
- **Hyperparameter Tuning** – Optimize model hyperparameters using GridSearchCV (e.g. tuning LightGBM parameters) to improve validation performance.
- **Evaluation** – Assess each model on the test set using metrics such as accuracy, confusion matrices, precision/recall (class-wise F1-scores), and ROC-AUC. Multi-class ROC curves are plotted to visualize performance for each subtype (optional).
- **(Optional) Ensemble Stacking** – Combine top models (e.g. LightGBM, XGBoost, SVM) in a stacking classifier with a Logistic Regression meta-learner to further boost performance.

Models Used

The following algorithms are implemented and compared:

- **Logistic Regression** (baseline linear model; also used as the meta-learner in stacking).
- **Random Forest** (ensemble of decision trees for robust classification).
- **XGBoost** (gradient-boosted trees to capture complex gene interactions).
- **MLPClassifier** (multilayer perceptron neural network via scikit-learn).
- **LightGBM** (high-performance gradient boosting framework; tuned via GridSearchCV).
- **Support Vector Machine (SVM)** (kernel-based classifier, included in some comparisons and the ensemble).

Each model is trained on the pre-processed training data. We compare their performance and select the best models for ensembling.

Evaluation Summary

Model performance is summarized using accuracy, precision/recall, and AUC. Confusion matrices and ROC curves are used to examine how well each subtype is classified. For example, the stacking ensemble combining LightGBM, XGBoost, and SVM achieves high overall accuracy on the test set, outperforming individual models; the integrated pipeline of balancing, SMOTE, tuning, and stacking yields robust performance on rare subtypes. Exact metric values (accuracy and AUC by class) are reported in the notebook. Learning curves and bias–variance plots (LightGBM) are also generated to assess under/overfitting.

Visualizations

The notebook produces several illustrative plots:

- **Heatmaps** (e.g. gene expression correlation matrix or clustered subtype profiles).
- **Bar Charts** (e.g. distribution of samples per subtype or feature importance rankings).
- **Box/Violin Plots** (showing gene expression distributions by subtype).
- **PCA Scatter Plots** (2D and 3D projections of samples coloured by subtype).

- **ROC Curves** and **Confusion Matrix Heatmaps** (to visualize classifier performance for each immune subtype).

Dependencies

The project requires the following Python libraries (included in requirements.txt):

- numpy
- pandas
- scikit-learn
- matplotlib
- seaborn
- xgboost
- lightgbm
- imbalanced-learn
- jupyter (for running the notebook)

These can be installed via pip install -r requirements.txt or using Anaconda.

Author

Mohammad Arsalan