Automated Stellar Classification Using Advanced Algorithms

Name: Siva Amirthalingam
Student ID: S24008073
Course Title: Advanced Data Structures and Algorithms
Date: 9/1/2025

Problem Narration and Illustration

In modern astronomy, classifying celestial objects is essential for understanding the universe. This project automates stellar classification, distinguishing between **GALAXY**, **QSO**, and **STAR** categories using spectroscopic data. Stellar classification supports scientific pursuits, such as studying galaxy evolution and cataloging stars. With increasing data from telescopes like the Sloan Digital Sky Survey (SDSS), manual classification is impractical, necessitating machine learning for efficiency and accuracy.

The chosen dataset [1] includes attributes such as **alpha, delta, u, g, r, i, z**, and **redshift**, with the target variable, class, indicating the category. Below is the class distribution of the dataset:

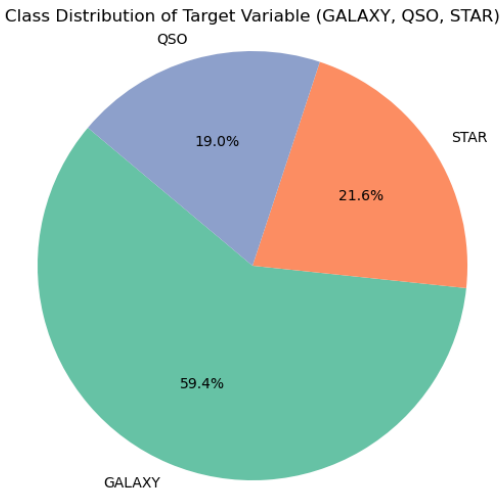

Figure 1: Class Distribution of the dataset

The objective is to build and evaluate machine learning models to classify these objects accurately, addressing the challenges of large-scale data processing while reducing human errors.

Algorithm Design and Justification

The project employs a machine learning pipeline comprising data preprocessing, feature engineering, and model training. Missing values were handled with mean imputation, and features were standardized for uniform scaling. Polynomial features were generated to capture interaction terms, enhancing the model's ability to learn complex patterns. These features allow the models to account for non-linear relationships between variables, such as interactions between different spectroscopic measurements, improving classification accuracy. The processed data was used to train

Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, Gradient Boosting, XGBoost, and Decision Trees.

Logistic Regression served as a baseline model for its simplicity and interpretability. KNN offered a non-parametric comparison, while Random Forest provided robustness and handled non-linear relationships effectively. Gradient Boosting and XGBoost iteratively improved performance by focusing on misclassified samples, and Decision Trees offered interpretability. XGBoost emerged as the best model due to its high accuracy, computational efficiency, and ability to handle large datasets.

Implementation and Demonstration of Proficiency

The project was implemented using Python, leveraging libraries such as Pandas, Scikit-learn, XGBoost, and Seaborn. Data preprocessing included handling missing values, standardizing features, and generating polynomial terms. The data was split into 70% for training and 30% for testing, ensuring that the models were evaluated on unseen data to validate their generalizability.

Each model was trained and tested on the processed data. Logistic Regression required adjustments to ensure convergence. KNN provided a benchmark for non-parametric methods. Random Forest, Gradient Boosting, and XGBoost were fine-tuned for optimal performance, and Decision Trees served as a baseline for interpretability. Performance was evaluated using accuracy and ROC-AUC metrics. Visualizations, such as feature importance charts and performance comparisons, provided insights into model behavior.

Thorough testing ensured the implementation's correctness, including verifying preprocessing steps and resolving compatibility issues with polynomial features. The project demonstrated proficiency in advanced data structures, algorithms, and object-oriented programming, handling real-world computational challenges effectively.

Visualizations and Outputs

The visualizations in this project provide crucial insights into model performance and feature significance. These visual aids guided the selection of XGBoost as the best-performing model by highlighting its superior accuracy and ROC-AUC compared to others.

They also revealed areas where certain models underperformed, prompting refinements such as hyperparameter tuning and feature selection to optimize overall performance.

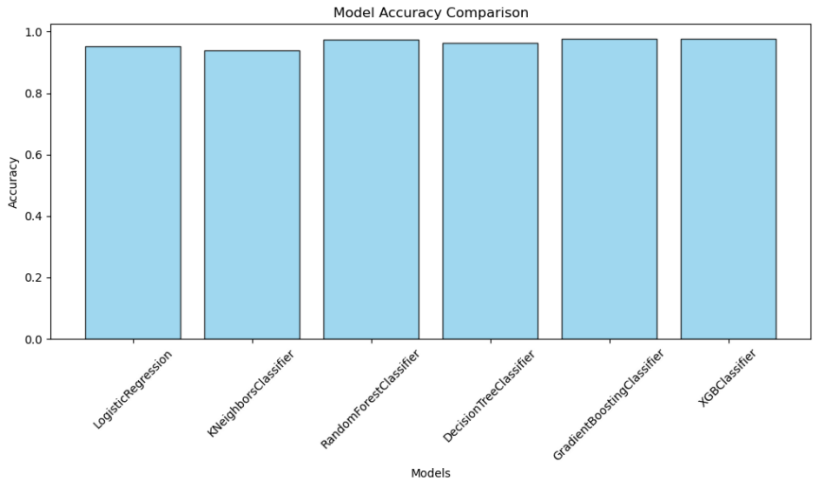

Figure 2: This chart highlights the relative accuracy of each model, showing XGBoost's superior performance.

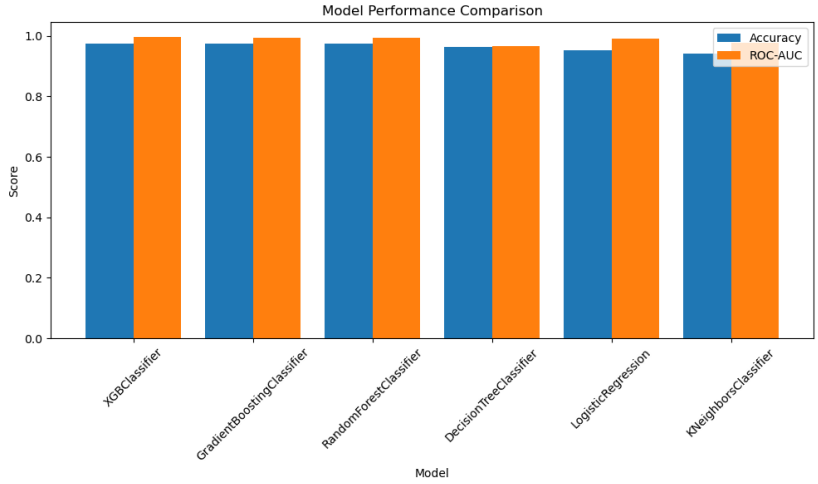

Figure 3: The bar chart provides a comparison of both accuracy and ROC-AUC scores for all models.

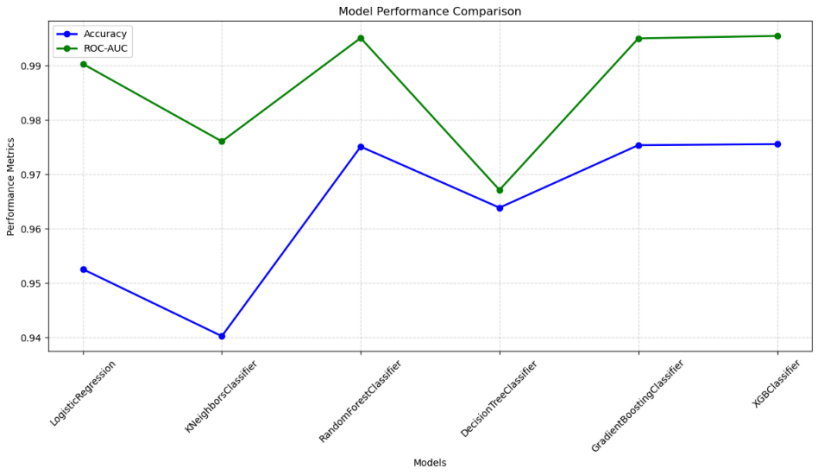

Figure 4: This line plot offers a detailed view of how performance metrics vary across models. Together, these visualizations enhance the interpretability and practical relevance of the models.

Computational Complexity Analysis

The computational complexity of this project was analyzed to ensure its scalability and efficiency. During data preprocessing, label encoding had a time complexity of O(n), where n is the number of samples, with a space complexity of O(n) to store the encoded labels. Feature scaling using StandardScaler required O(nd) time and O(d) space, where d is the number of features, to compute and store the mean and standard deviation. Polynomial feature generation was the most computationally intensive step, with both time and space complexities of O(nd^k), where k is the polynomial degree, due to the exponential increase in feature interactions.

The data splitting step (train_test_split) had a time and space complexity of O(n), as it involved a single iteration over the dataset to create two subsets. For model training and evaluation, the computational complexity varied by algorithm. Logistic Regression required O(nd + d^2) time, with O(d^2) space for storing the covariance matrix. K-Nearest Neighbors (KNN) had a time complexity of O(n_train * d) during prediction and required O(n_train * d) space to store the training data. Random Forest involved O(T * m * log(n_train)) time and O(T * n_train) space, where T is the number of trees and m is the number of features per split. Decision Trees had O(n_train * log(n_train)) time complexity and O(n_train) space for tree storage. Gradient Boosting and XGBoost, known for their iterative boosting processes, required O(T * n_train * log(n_train)) and O(T * n_train * log(n_train) + T * m * d) time, respectively, with both using O(T * n_train) space to store the trees.

For model evaluation, calculating accuracy and generating confusion matrices had a time complexity of O(n_test), where n_test is the size of the test set. ROC-AUC computation, which evaluates the area under the curve for each class, required O(n_test * n_classes) time. The overall computational complexity was primarily influenced by the polynomial feature expansion and the training of ensemble models like Random Forest, Gradient Boosting, and XGBoost. Efficient preprocessing and simpler models can significantly reduce the time and space requirements, especially for large datasets.

Overall Summary of Results

The performance of each model was evaluated using accuracy and ROC-AUC. XGBoost emerged as the best model with an accuracy of 0.9756 and a ROC-AUC of 0.9955, attributed to its advanced gradient boosting, regularization techniques, and efficient handling of missing data. Gradient Boosting and Random Forest followed closely, with accuracies of 0.9754 and 0.9751, respectively, and ROC-AUC scores above 0.995. Logistic Regression, though simpler, achieved a commendable accuracy of 0.9525 and a ROC-AUC of 0.9903. Decision Tree and KNN, while less accurate, provided valuable benchmarks for comparison.

The top five features contributing to XGBoost's success were **redshift** (0.8612), **g** (0.0322), **z** (0.0222), **u** (0.0150), and the interaction term **g z** (0.0100). These features were instrumental in achieving high classification accuracy.

This project highlights the potential of machine learning in automating stellar classification, paving the way for more efficient astronomical research. By enabling rapid and accurate classification, this approach can support large-scale studies of stellar populations, contribute to identifying rare celestial phenomena, and improve the allocation of telescope time for targeted observations.

References

[1] "SDSS," Sloan Digital Sky Survey, [Online]. Available: https://www.sdss.org. [Accessed 23 December 2024].