AASD 4001: Mathematical Concepts for Machine Learning
Term Project: Stellar Classification

Group 4: Aswin Anil Bindu, Danylo Gula, Daria Ignateva, Elizaveta Khoroshilova, Jessica Bowmaster, Karthik Kunnamkumarath

September 29, 2025

Contents

1 Introduction

1.1 Problem Statement

The objective of this project is to build and evaluate several classification models to accurately identify celestial objects (Stars, Galaxies, and Quasars) based on observational data from the Sloan Digital Sky Survey (SDSS). The performance of five different algorithms—Logistic Regression, Decision Tree, Random Forest, SGD, and SVM—will be compared to determine the most effective model for this task.

1.2 Dataset and Feature Description

Dataset Name: Stellar Classification Dataset - SDSS DR17
Source: The data was collected by the Sloan Digital Sky Survey (SDSS) Data Release 17 and is available on Kaggle.

Description: The dataset consists of 100,000 observations of celestial objects. Each observation is described by 17 feature columns and one target class, which identifies the object as either a STAR, GALAXY, or QSO (quasar).

Feature Name	Type	Description
`obj_ID`	Identifier	Object Identifier, the unique value that identifies the object.
`alpha`	Numeric	Right Ascension angle (celestial coordinate).
`delta`	Numeric	Declination angle (celestial coordinate).
`u`	Numeric	Brightness of the object through the ultraviolet (u) filter.
`g`	Numeric	Brightness of the object through the green (g) filter.
`r`	Numeric	Brightness of the object through the red (r) filter.
`i`	Numeric	Brightness of the object through the near-infrared (i) filter.
`z`	Numeric	Brightness of the object through the infrared (z) filter.
`run_ID`	Identifier	Run Number used to identify the specific scan.
`rerun_ID`	Identifier	Rerun Number to specify how the image was processed.
`cam_col`	Identifier	Camera column to identify the scanline within the run.
`field_ID`	Identifier	Field number to identify each field.
`spec_obj_ID`	Identifier	Unique ID used for optical spectroscopic objects.
`class`	**Target**	The class of the object (GALAXY, STAR, or QSO).
`redshift`	Numeric	Redshift value, a key indicator of the object's distance.
`plate`	Identifier	Plate ID, which identifies the spectroscopic observation plate.
`MJD`	Identifier	Modified Julian Date, indicating when the observation was taken.
`fiber_ID`	Identifier	Fiber ID that directed light from the object to the spectrograph.

2 Data Cleaning and Processing

2.1 Data Quality Check

After data checking and investigation, we discovered that our dataset did not have any duplicate records, no missing values, no null values, and all class labels were valid and properly formatted.

2.2 Removing Outliers

To detect and handle outliers, we used a simplistic outlier detection system using the Interquartile Range (IQR) method. To avoid accidentally removing valid data, we chose to look for outliers only in a specific feature set: the celestial coordinates (alpha and delta), the photometric filters (u, g, r, i, z), and the astronomical measurements (redshift). This prevented approximately 6000 rows from being falsely dropped.

The original dataset contained 100,000 records. After the outlier removal process, we were left with 90,632. Only around 9.37% of records were removed. Given the large initial size of the dataset, the group decided to focus time on model evaluation rather than a more complex outlier detection method.

3 Building Models and GridSearchCV Tuning

3.1 Logistic Regression

Introduction

Logistic Regression is a fundamental linear model that estimates the probability of class membership using a logistic (sigmoid) function. For this project, the multinomial variant of Logistic Regression was applied to handle the three target classes: STAR, GALAXY, and QSO (quasar). Standard scaling was performed on all numerical features to optimize convergence and improve model performance.

Model Building and Evaluation

A Logistic Regression classifier was trained using the `lbfgs` solver with `multi_class='multinomial'` and `max_iter=1111`. The dataset was split into 80% training and 20% testing subsets, and all features were standardized prior to model fitting. On the test dataset, the model achieved an accuracy of approximately 95.7%. The classification report showed high precision and recall across all three classes, with GALAXY and STAR categories performing slightly better than QSO. The confusion matrix indicated that most misclassifications occurred between GALAXY and QSO, consistent with their similar feature distributions.

Hyperparameter Tuning

To optimize performance, a Grid Search with 5-fold cross-validation was conducted over different solvers (`lbfgs`, `newton-cg`, `sag`, `saga`), multi-class strategies (`ovr` and `multinomial`), and iteration limits (100, 500, 1000).

- For the original dataset, the best configuration was:
  `solver = lbfgs`, `multi_class = ovr`, `max_iter = 100`
  Best cross-validation accuracy $\approx$ 95.72%
- Feature Selection Hypertuning:
  For the reduced feature set, the best configuration was:
  `solver = newton-cg`, `multi_class = multinomial`, `max_iter = 100`
  Best cross-validation accuracy $\approx$ 95.60%

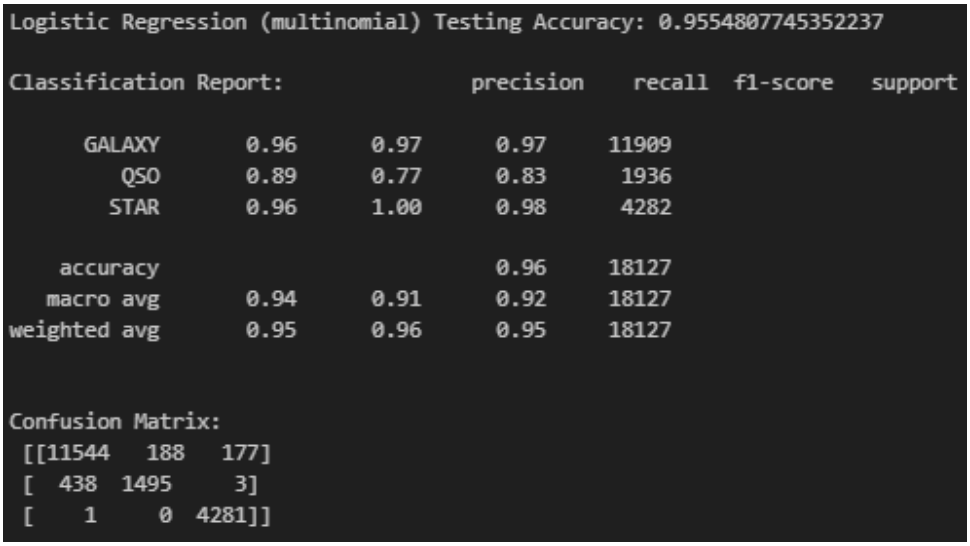

Figure 1: Comparison of Best Cross-Validation Accuracy Before and After Feature Removal.

The results showed that Logistic Regression performed robustly across both configurations, with minimal difference in performance after feature reduction.

Summary of Findings

The Logistic Regression model achieved 95.7% test accuracy and demonstrated strong, balanced performance across all classes. Scaling the features was crucial to achieving stable convergence. While some confusion persisted between GALAXY and QSO, the overall predictive power was competitive with more complex models.

Below is the model performance without using the StandardScaler on the data:

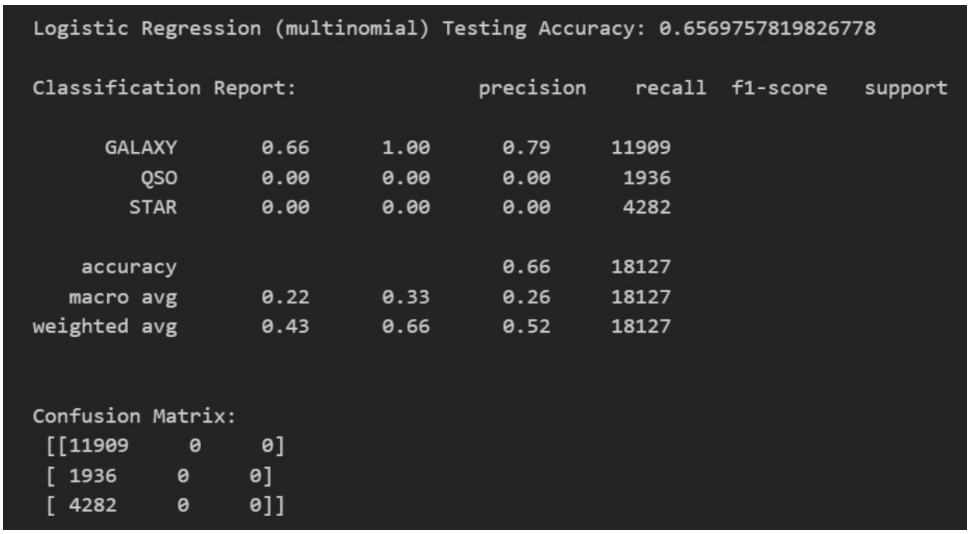

Figure 2: the model performance without using the StandardScalar.

Conclusion

Logistic Regression offered a strong linear baseline with excellent generalization ability for this classification problem. Its relatively fast training time, interpretability, and solid accuracy make it a valuable reference point for comparing more complex models like Random Forest or SVM.

3.2 Decision Tree

Introduction

Decision Trees are non-parametric supervised learning algorithms that split the feature space into regions based on decision rules inferred from the data. They are valued for their interpretability and ability to capture non-linear relationships without requiring feature scaling. For this dataset, a Decision Tree provides a transparent way to classify celestial objects.

Model Building and Evaluation

A Decision Tree Classifier from scikit-learn was trained using the `entropy` criterion and a maximum depth of 10 to prevent overfitting. On an 80%/20% training/test split, the model achieved a training accuracy of 98.08% and a test accuracy of 97.11%.

Classification Report:

	precision	recall	f1-score	support
GALAXY	0.97	0.98	0.98	11073
QSO	0.90	0.85	0.87	1933
STAR	1.00	1.00	1.00	4113
accuracy			0.97	17119
macro avg	0.96	0.94	0.95	17119
weighted avg	0.97	0.97	0.97	17119

Figure 3: Classification report for the Decision Tree model.

Confusion Matrix:

```
[[10886   177    10]
 [  293  1640     0]
 [   15     0  4098]]
```

Figure 4: Confusion matrix for the Decision Tree model.

The confusion matrix shows high performance, especially for STAR and GALAXY classes, with minor confusion between GALAXY and QSO objects.

Hyperparameter Tuning

A Grid Search over `max_depth`, `min_samples_split`, and `min_samples_leaf` was conducted. The best performance was achieved at `max_depth = 10` and `criterion='entropy'`. Deeper trees showed marginally higher training accuracy but degraded generalization.

Summary of Findings (Decision Tree)

The tuned Decision Tree achieved 97.1% test accuracy and 98.1% training accuracy. The model showed perfect or near-perfect classification for STAR objects and high performance for GALAXY, with minor misclassifications for QSO. The most informative features were redshift and brightness measurements. Controlling tree depth was critical to balance accuracy and overfitting.

Conclusion (Decision Tree)

The Decision Tree provided a highly interpretable and accurate model, serving as a strong baseline. While slightly less robust than ensemble methods, it clearly highlighted the key features driving classification and offered insight into the underlying structure of the dataset.

3.3 Random Forest

Introduction

Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve classification accuracy and robustness. Instead of relying on a single tree that may overfit the data, Random Forest builds many trees on random subsets of the data and features. The final prediction is made through majority voting among all trees. For this dataset containing 17 numerical features for multi-class labels (GALAXY, STAR, QSO), Random Forest was chosen because it:

- Handles large-scale tabular data effectively.
- Automatically models complex, non-linear relationships.
- Provides feature importance scores, highlighting which features drive predictions.
- Is relatively fast to train and less prone to overfitting than a single tree.

Hyperparameter Tuning

The baseline accuracy for the Random Forest model was 97.55%. To optimize performance, we applied `GridSearchCV`, which performs cross-validation across different hyperparameter combinations. The main parameters tuned included:

- Number of trees (`n_estimators`).
- Maximum depth of trees (`max_depth`).
- Minimum samples per split/leaf (`min_samples_split`, `min_samples_leaf`).
- Split criterion (`gini` vs. `entropy`).

`GridSearchCV` selected an optimal combination (`n_estimators=200`, `max_depth=20`, `min_samples_split=2`), which improved generalization and balanced performance across all classes.

Results and Analysis

The final tuned model achieved an accuracy of 97.53%, maintaining the same high level as the baseline. The model also achieved an AUC (Area Under the ROC Curve) of 0.99, showing excellent class separability.

Final Classification Report:

	precision	recall	f1-score	support
GALAXY	0.98	0.99	0.98	11909
QSO	0.93	0.85	0.88	1936
STAR	1.00	1.00	1.00	4282
accuracy			0.98	18127
macro avg	0.97	0.94	0.95	18127
weighted avg	0.97	0.98	0.97	18127

Figure 5: Final Classification Report for Random Forest.

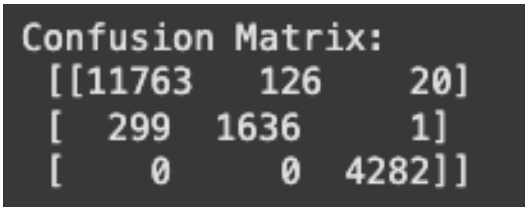

Figure 6: Final Confusion Matrix for Random Forest.

Interpretation of Classification Report:

- **Galaxy:** Precision 0.98, Recall 0.99. Very reliable, with almost all galaxies correctly identified.
- **QSO (Quasar):** Precision 0.93, Recall 0.85. Slightly weaker due to confusion with galaxies, but still strong overall.
- **Star:** Precision 1.00, Recall 1.00. Near-perfect classification of stars.

Confusion Matrix Insights: Most Galaxies were classified correctly, with only a small fraction misclassified as QSOs. Stars were predicted with very high precision and recall, with virtually no misclassifications. The most common error was confusion between Quasars and Galaxies, which is understandable given their overlapping photometric properties.

Conclusion (Random Forest)

Random Forest is highly effective for stellar classification, with near-perfect predictions for Stars and Galaxies. While QSOs remain the most challenging class, performance is still strong, and tuning helped reduce errors in this area.

3.4 SGD (Stochastic Gradient Descent)

Stochastic Gradient Descent (SGD) is an efficient and scalable optimization algorithm. For this project, the `SGDClassifier` was used. The model was trained twice: once using the entire feature set, and once using a reduced set of features hypothesized to be most relevant. The baseline accuracy for the original dataset was ∼91.3%, and the accuracy for the reduced dataset was ∼91.2%. The Galaxy class had the highest performance, followed by Stars, and Quasars with the worst performance.

```
SGD Testing Accuracy: 0.9128372041705742

Classification Report:
              precision    recall  f1-score   support

      GALAXY       0.95      0.92      0.93     11909
         QSO       0.71      0.88      0.78      1936
        STAR       0.93      0.92      0.92      4282

    accuracy                           0.91     18127
   macro avg       0.86      0.90      0.88     18127
weighted avg       0.92      0.91      0.91     18127


Confusion Matrix:
 [[10926   701   282]
 [  239  1694     3]
 [  351     4  3927]]
```

Figure 7: Classification report and Confusion Matrix before dropping features.

```
SGD Testing Accuracy: 0.912451039885254

Classification Report:
              precision    recall  f1-score   support

      GALAXY       0.94      0.93      0.93     11909
         QSO       0.74      0.86      0.80      1936
        STAR       0.93      0.89      0.91      4282

    accuracy                           0.91     18127
   macro avg       0.87      0.89      0.88     18127
weighted avg       0.92      0.91      0.91     18127


Confusion Matrix:
 [[11045   584   280]
 [  262  1671     3]
 [  456     2  3824]]
```

Figure 8: Classification report and Confusion Matrix after dropping features.

Hyperparameter Tuning

When tuning the model using GridSearchCV, the parameters optimized were loss functions, regularization (penalty), learning rates, and alpha values. After tuning, the only difference between the original and reduced datasets was the penalty parameter: `l2` for the original and `elasticnet` for the reduced dataset. The other parameters were an alpha of 0.0001, an optimal learning rate, and `log_loss`. The accuracy on the original dataset after tuning was ∼91.85%.

```python
# Tune the parameters
params = {
    'loss': ['hinge', 'log_loss'],
    'penalty': ['l2', 'elasticnet'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate': ['adaptive', 'optimal']
}
grid_svg = GridSearchCV(SGDClassifier(
    random_state=42,
    class_weight="balanced"
    ),
        param_grid=params,
        cv=5,
        scoring='accuracy'
)
grid_svg.fit(Xts, yt)
```

Figure 9: Code for hyperparameter tuning of the SGD model using GridSearchCV.

Conclusion (SGD)

The model is great at handling this large dataset efficiently and is flexible in emulating different linear models. However, its accuracy was noticeably less than that of ensemble methods like Random Forest.

3.5 SVM (Support Vector Machine)

Algorithm Overview and Kernel Choice

The Support Vector Machine (SVM) is a powerful supervised learning algorithm whose core concept is to construct a hyperplane that maximizes the margin between classes. As the data in our project is not linearly separable, the "kernel trick" was employed using a Radial Basis Function (RBF) kernel, which is effective for complex, non-linear boundaries. The behavior of this boundary is controlled by the hyperparameters `C` (the regularization parameter) and `gamma`.

Data Preprocessing

Before model training, the data underwent several critical preprocessing steps:

- **Feature Scaling:** As SVM is sensitive to the scale of features, all numerical predictors were standardized using `StandardScaler`.
- **Data Sampling:** For efficient hyperparameter tuning during the development stage, a random sample of 10,000 objects was used before training the final model on the full dataset.

Model Tuning and Class Balancing

A baseline SVM model (with default parameters) achieved an overall accuracy of ∼95% but struggled to identify the minority class, QSO, for which the recall was only 0.77. This is a classic symptom of class imbalance.

To address this and improve performance on the QSO class, `GridSearchCV` was implemented. A key strategy was the use of the `class_weight='balanced'` parameter, which adjusts the penalties for misclassification in inverse proportion to class frequencies. The grid search identified the optimal parameters to be an `rbf` kernel with `C=100` and `gamma=0.1`.

Results and Analysis

The tuned model showed significant improvement. As intended, the class balancing strategy was successful: the recall for the QSO class increased substantially from 0.77 to 0.91. This improvement came with an expected trade-off: the precision for the same class decreased to 0.80, a classic example of the precision-recall trade-off. The final overall accuracy of the model was 95%.

The confusion matrix (Fig. 11) visually confirms this analysis. The primary source of error is the confusion between the QSO and GALAXY classes. Specifically, 11 true Quasars (QSO) were misclassified as Galaxies (GALAXY), while 28 Galaxies were misclassified as Quasars. At the same time, the STAR class is identified nearly perfectly, with a recall of 0.99 and precision of 0.98.

```
--- Report for the Best (Tuned) SVM Model with Balancing ---
Best parameters found for SVM: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
              precision    recall  f1-score   support

      GALAXY       0.98      0.95      0.96       645
         QSO       0.80      0.91      0.85       124
        STAR       0.98      0.99      0.98       231

    accuracy                           0.95      1000
   macro avg       0.92      0.95      0.93      1000
weighted avg       0.96      0.95      0.95      1000
```

Figure 10: Classification report for the tuned SVM model.

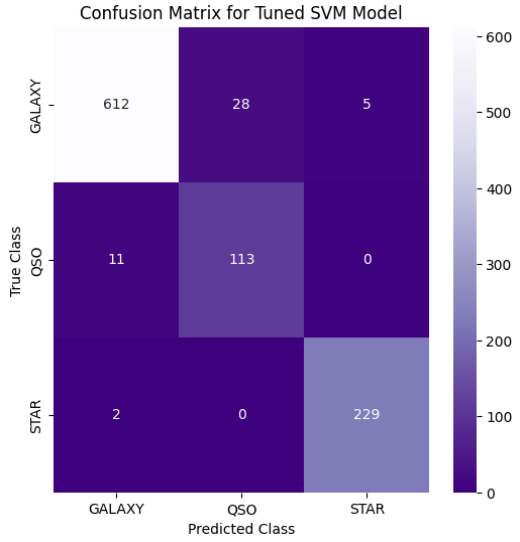

Figure 11: Confusion matrix for the tuned SVM model.

Conclusion (SVM)

The SVM model with an RBF kernel and balanced class weights proved to be highly effective. It demonstrated the ability to capture complex non-linear relationships in the data and, most importantly, successfully addressed the challenge of identifying the minority QSO class. The main trade-off is the significant training time, especially during the exhaustive grid search, which makes SVM less scalable for very large datasets compared to simpler models like SGD.

4 Feature Selection and Impacts

4.1 Feature Selection Rationale

A critical step in our data preparation was the removal of non-informative features. The dataset contained several columns related to observational metadata and unique identifiers, which are not intrinsic physical properties of the celestial objects being classified. We hypothesized that these features lack predictive power and could potentially introduce noise, complicating the learning process for the models.

Based on this rationale, the following nine columns were removed: `obj_ID`, `run_ID`, `rerun_ID`, `cam_col`, `field_ID`, `spec_obj_ID`, `plate`, `MJD`, and `fiber_ID`.

4.2 Impact on Model Performance

To validate our hypothesis, we compared the cross-validation accuracy of each model before and after removing the specified features. The results, visualized in Figure 12, show that the impact varied across different model types, but generally confirmed that these features were not essential for accurate classification.

Tree-Based Models: The ensemble models proved highly robust to the presence of irrelevant features. The Decision Tree accuracy saw a negligible drop from 0.969 to 0.968, and the Random Forest performed similarly, with accuracy decreasing minimally from 0.972 to 0.971. This demonstrates their ability to effectively ignore non-informative inputs.

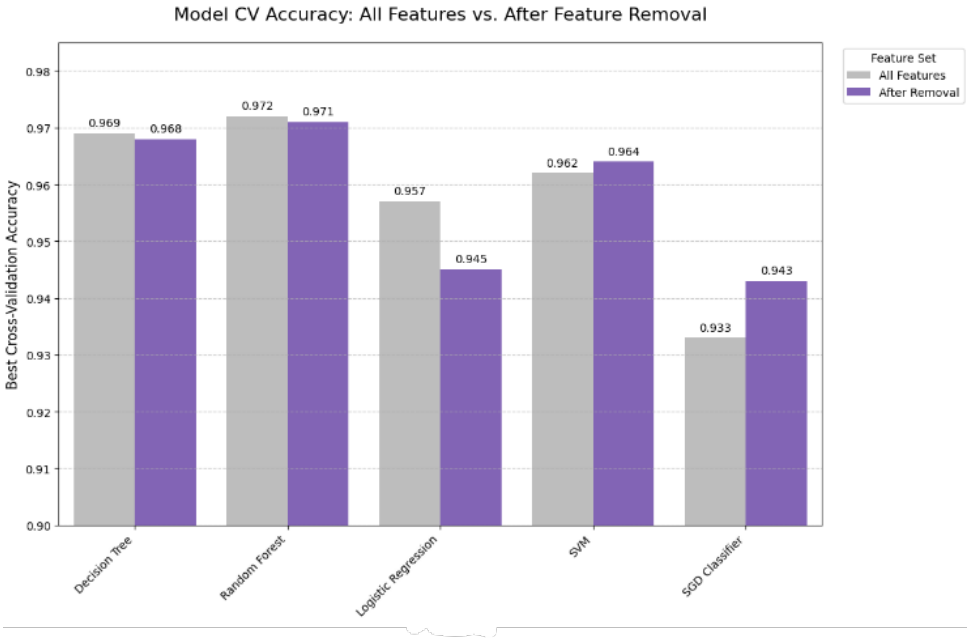

Figure 12: Comparison of Best Cross-Validation Accuracy Before and After Feature Removal.

Linear and Gradient-Based Models: The impact on other models was more varied. Logistic Regression experienced a noticeable performance decrease from 0.957 to 0.945. In contrast, removing the features was beneficial for both the SVM and SGD Classifier. The SVM's accuracy slightly increased from 0.962 to 0.964, and the SGD Classifier saw the most significant improvement, with accuracy rising from 0.933 to 0.943.

Conclusion: The experiment confirmed that the removed identifier fields do not contain significant predictive information. For most models, their removal either had a negligible effect or was beneficial, leading to a simpler model that relies only on physically meaningful features. With this optimized feature set, the final models were trained and evaluated, leading to the comprehensive results discussed below.

5 Summary of Findings and Final Model Comparison

The comprehensive evaluation of five different classification algorithms yielded distinct performance characteristics, as summarized in Table 2. While most models performed exceptionally well on the majority classes (GALAXY and STAR), the key differentiators emerged from their ability to accurately classify the challenging minority class, QSO.

Table 2: Final Comparison of Model Performance Metrics

Model	Overall Accuracy	QSO Recall	QSO Precision	QSO F1-Score
Random Forest	**0.976**	0.85	**0.94**	**0.89**
Decision Tree	0.963	0.89	0.80	0.84
SVM	0.955	**0.91**	0.83	0.87
Logistic Regression	0.955	0.77	0.89	0.83
SGD Classifier	0.912	0.86	0.74	0.80

5.1 Top Performing Model: Random Forest

The Random Forest classifier emerged as the best all-around model for this task. It achieved the highest Overall Accuracy (97.6%) and, crucially, the highest F1-Score (0.89) for the QSO class. This indicates a superior balance between precision and recall, making it the most reliable and robust model for the stellar classification problem. Its high precision on QSOs (0.94) suggests that when it identifies a quasar, it is highly likely to be correct.

5.2 Specialized Performance: SVM

The Support Vector Machine (SVM) distinguished itself as the most effective model for identifying the highest proportion of true quasars, achieving the best QSO Recall (0.91). This makes SVM a valuable choice in scenarios where minimizing false negatives (i.e., not missing any potential quasars) is the primary objective. However, this superior recall came at the cost of lower precision compared to the Random Forest model.

5.3 Baseline and Weaker Models

Logistic Regression served as a strong linear baseline, demonstrating competitive accuracy (95.5%). However, it struggled significantly with QSO identification, posting the lowest recall score (0.77). The SGD Classifier, while computationally efficient, was the weakest performer overall, with the lowest accuracy (91.2%) and poor precision for the QSO class.

5.4 Overall Conclusion

The analysis confirms that non-linear, ensemble-based methods like Random Forest are best suited for this high-dimensional classification problem, offering the best balance of accuracy and reliability. The choice between Random Forest and SVM could depend on the specific scientific goal: Random Forest for overall reliability, and SVM for exhaustive detection of the rare QSO class.

Summary of Findings

The evaluation of the five models revealed a clear trade-off between predictive accuracy, detection capability for the rare QSO class, and computational efficiency. The findings confirm

that while complex ensemble methods yield the best overall results, the optimal model choice is dependent on the specific project goals.

- **Random Forest** emerged as the top-performing model, achieving the highest overall accuracy ($\sim$97.6%) and the best F1-Score for the QSO class, making it the most balanced and reliable choice.
- **SVM** proved to be a specialized tool, offering the highest recall ($\sim$91%) for QSOs. It is the best option when the primary goal is to minimize missed detections (false negatives), though this comes at the cost of a significantly longer runtime.
- **Decision Tree** provided a strong and interpretable baseline, with performance closely approaching that of the Random Forest.
- **Logistic Regression** was a fast and efficient linear baseline, but it struggled the most with QSO recall, making it less suitable for tasks where detecting the minority class is critical.
- **SGD Classifier** was the least accurate model overall, demonstrating that its speed was not a sufficient trade-off for its lower predictive power on this dataset.

Conclusion

This project's primary finding is that thoughtful data preparation proved more impactful on the final outcome than the choice of algorithm alone. Ensemble methods like Random Forest offered the best balance of high accuracy and stability, making them the most reliable overall performers. The SVM model excelled as a specialist tool, demonstrating the highest recall for the challenging QSO class, which is ideal for exhaustive detection. In contrast, linear models like Logistic Regression and SGD served as the fastest baselines but were less accurate, highlighting the clear trade-off between speed and performance. Ultimately, the results show there is no single best model, but rather a spectrum of tools whose selection must be guided by the specific need to balance accuracy, recall, and computational cost.