Star Classification and Clustering: A Comparative Analysis

Machine Learning Project, Academic Year 2025/2026
University of Verona

Marco Equisetto
VR535007

February 17, 2026

Abstract

This report presents an in-depth analysis of the Star Classification dataset, comparing the performance of multiple supervised and unsupervised models: it investigates how data preprocessing heavily impacts model outcomes and explores how different techniques can be combined to leverage their respective strengths.
Three distinct supervised learning models are evaluated: **K-Nearest Neighbors** (**KNN**), **Random Forest Classifier** (**RFC**) and **Support Vector Machines** (**SVM**). Additionally, unsupervised learning techniques, specifically **K-Means** and **Gaussian Mixture Model** (**GMM**), are employed to explore the intrinsic topological nature of the astronomical data. Model performance is primarily evaluated using the F1-score, with the addition of accuracy, precision and recall for classification tasks, while Adjusted Rand Index (ARI), Normalized Mutual Info (NMI) and Silhouette Coefficient are utilized for clustering validation.

Contents

1 Introduction

The classification of celestial objects is a fundamental task in modern astrophysics. With the advent of large-scale sky surveys, and the constantly increasing amount of satellites and telescopes, the volume of spectral and photometric data has grown exponentially, creating the need to move from manual classification to *machine learning* and *deep learning* models [5]. Although machine learning models pale in comparison to deep learning ones in terms of ability to process high dimensionality and quantity data, they still offer a more than viable solution for rudimentary tasks such as the one tackled in this discussion.

2 Motivation and Rationale

This project addresses the problem of classifying objects from the Sloan Digital Sky Survey (SDSS) [5] into three distinct classes: **Galaxies**, **Stars**, and **Quasars**. The rationale behind this work is to compare the efficacy of distance-based and ensemble models in handling astronomical data, which is often characterized by high dimensionality, considerable size and high levels of noise. Most of the currently employed models rely on classifying bodies by redshift and spectral indexes. This project turned such a task into one that relies **solely on color and emitted light**.

3 State of the Art

Astronomical classification has traditionally relied on the *Morgan-Keenan* (MK) method, where each star has a spectral class and a luminosity class assigned to it. It has evolved from manual inspection to automatic pipelines capable of processing and handling data of orders of magnitude that are incomparable to those historically analyzed by hand. Current state-of-the-art methods generally fall into two categories depending on the input data format: ensemble methods for tabular photometric data and Deep Learning architectures for raw spectral or time-series data. For datasets consisting of extracted features (magnitudes, colors, redshift), much like the dataset used in this project, Gradient Boosted Decision Trees (GBDTs) are currently considered the accepted standard [4], which is one of the reasons why Random Forest Classifier (RFC) was considered as a viable choice and why it was expected to performed quite well in this task. While tabular data with few selected features is efficient, the highest level of accuracy is obtained through the use of "raw" data, these being mathematical data like 1D-Spectra (a combination of intensity and wavelength) or light curves (brightness over time). Such data types are of complicated distributions and huge quantities, rendering them effectively almost impossible to handle by simple Machine Learning models, and therefore more suitable for Deep Learning models, such as Neural Networks.
Despite all the technological advancements, the field of stellar classification still faces many hurdles, mainly:

- **Imbalance**: The nature of the universe renders some objects much rarer than others (e.g. highly redshifted Quasars) and therefore data about said objects can be extremely hard to obtain, making it difficult to train models with equal representation of all classes.
- **Domain Shift**: Due to how much celestial objects can vary both inside and outside classes, training a model on one dataset with labeled data from one survey and then applying it to another dataset might result in poor performance due to the presence of different levels of noise profiles and instrument sensitivity, which is and always will be inevitable since data come from different measuring instruments.

4 Objectives

The primary objective of this project is to develop a robust machine learning pipeline for celestial object classification and to find out which of the tested models best fits the task at hand. Specific objectives include:

1. **Data Preprocessing:** To implement effective outlier removal, balancing and feature engineering (calculating color indices like $u - g$) to improve model separability and to remove problematic noisy features such as position based features (namely *alpha* and *delta*).
2. **Supervised Comparison:** To evaluate and tune hyperparameters for KNN, Random Forest, and SVM to find the best performing one for each of them.
3. **Unsupervised Exploration:** To analyze the dataset using Clustering (K-Means, GMM) and assess the impact of Principal Component Analysis (PCA) on clustering performance.

5 Methodology

All experiments were conducted using Python, mainly utilizing the `scikit-learn` [2], `pandas`, and `seaborn` [6] libraries specifically, combined with other performance metric libraries. The single trainings of the various methods were developed separately and then brought together in one single train cycle, to standardize all of them and put all models in the same starting conditions, distribution and random seed generation.

5.1 Dataset Description

The dataset of choice is the *Star Classification* dataset (sourced from Kaggle/SDSS) [1]. It initially contains $100,000$ observations, with features like spectral columns (u, g, r, i, z), redshift, various IDs, spatial coordinates, namely *alpha* and *delta*, and a target class, a categorical variable with three levels: `GALAXY`, `STAR`, `QSO`.

5.2 Data Preprocessing and Feature Extraction

To prepare the data for training, the following steps were taken:

5.2.1 Cleaning

The ID columns were dropped completely from the start since they do not provide real information and would very likely introduce biases or noise if kept.

5.2.2 Missing Values Check

Missing/zero values introduce sparsity and lower precision of models if many are present. In this specific dataset, the procedure did not detect any missing or 'zero' values, so no real removal action was needed and performed.

5.2.3 Outlier Detection

First, data coming from sensor malfunction was manually deleted (e.g., removing rows where $u = -9999$), then outliers were removed based on valid photometric ranges [4]. This detection was performed combining two different methods:

- **Interquartile Range (IQR):** A rule that statistically defines a "reasonable" range in which data points could fall, anything out of which is considered an outlier. In this case, the line was traced based on a **1.5IQR rule** and only the data above (to the right of) such threshold was considered valid and kept.

- **Gaussian Mixture Model (GMM):** Applying a simple instance of GMM to initial data can show outliers as points which have a very low percentage probability to belong to any and all of the classes that are present, indicating that they are very likely outliers.

These techniques were combined and applied to produce the outlier detection result seen in Figure [1].

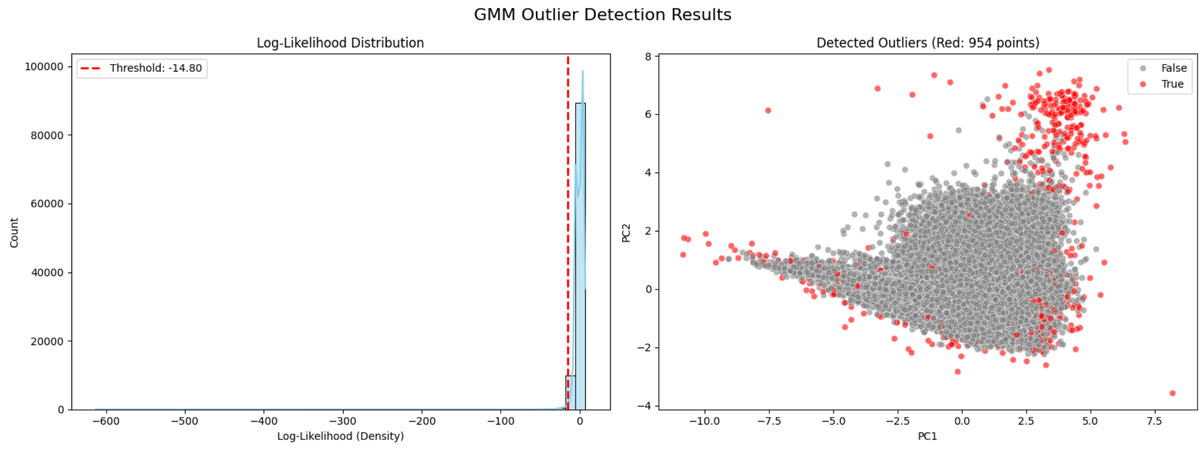


Figure 1: **Dataset Outliers**: On the left the distribution and where the threshold was placed by the 1.5IQR rule, effectively excluding all data to the left of it. On the right the GMM instance where outlier points were highlighted in red.

5.2.4 Label Encoding

Since the target label for classification was categorical (it being a string with the name of the object), a label encoder was used to convert it to a number. Keeping categorical, string-like features might have the model learn inexistent correlations based on the length of the word or specific letters. Encoding to a simple numeric ID removes the textual features entirely, leaving only the category identity.

5.2.5 Balancing

When considering whether the dataset needed balancing or not, the distribution of classes was analyzed: its result can be visualized in Figure [2]. These charts show that, even though there technically is an imbalance of representation, it is not strong enough to require direct action on the dataset itself. Utilizing the *class_weight* flag in models that contain it (such as RFC and SVM) will suffice to ensure each of them follows its own procedure to balance data. As for the KNN model, since it can handle slight imbalance in a rather optimal way, no invasive balancing of the dataset felt strictly necessary.

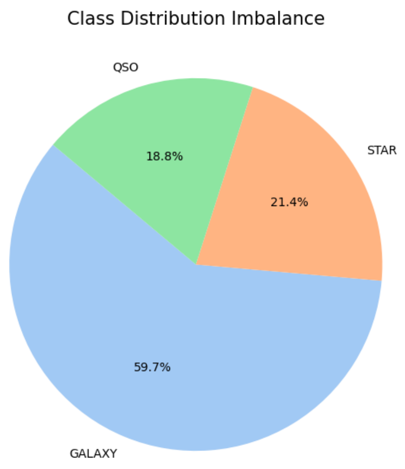


Figure 2: **Label Distribution**: How many entries are present inside each class represented as a percentage on the total amount of data.

5.2.6 Feature Engineering

The features of this dataset followed distributions that can be seen in Figure [3]. Based on correlation analysis, which was prompted by some of the implemented method's dire need of correlation-free environments to work at their best, said features produced the correlation matrix shown in Figure [4a], which implied some action needed to be taken to fix the high correlation between them.
Instead of deleting all highly correlated features right away, a different approach was considered, which was to duplicate the dataset and create the two following versions:

- **Tree-Based Set (X_{tree}):** Retains both raw magnitudes (u, g, r, i, z) and synthetic color indices (used for RFC)
- **Linear-Model Set (X_{linear}):** Drops the raw magnitudes and retains **only** the synthetic color indices to reduce multicollinearity (used for KNN, SVM and clustering).

This decision was taken because while models like KNN, SVM and clustering approaches see their performance hindered by correlated features, and in their case those need to be removed, RFC is effectively immune to such an even and therefore can benefit from both keeping the original features and introducing new information on top of it. Correlation between the newly added features was evaluated and its result is seen in Figure [4b].

5.2.7 Scaling

A `StandardScaler` was applied to normalize features to zero mean and unit variance.

5.2.8 Dimensionality Reduction

For clustering analysis, **PCA** was applied to reduce the feature space to 2 principal components [3].

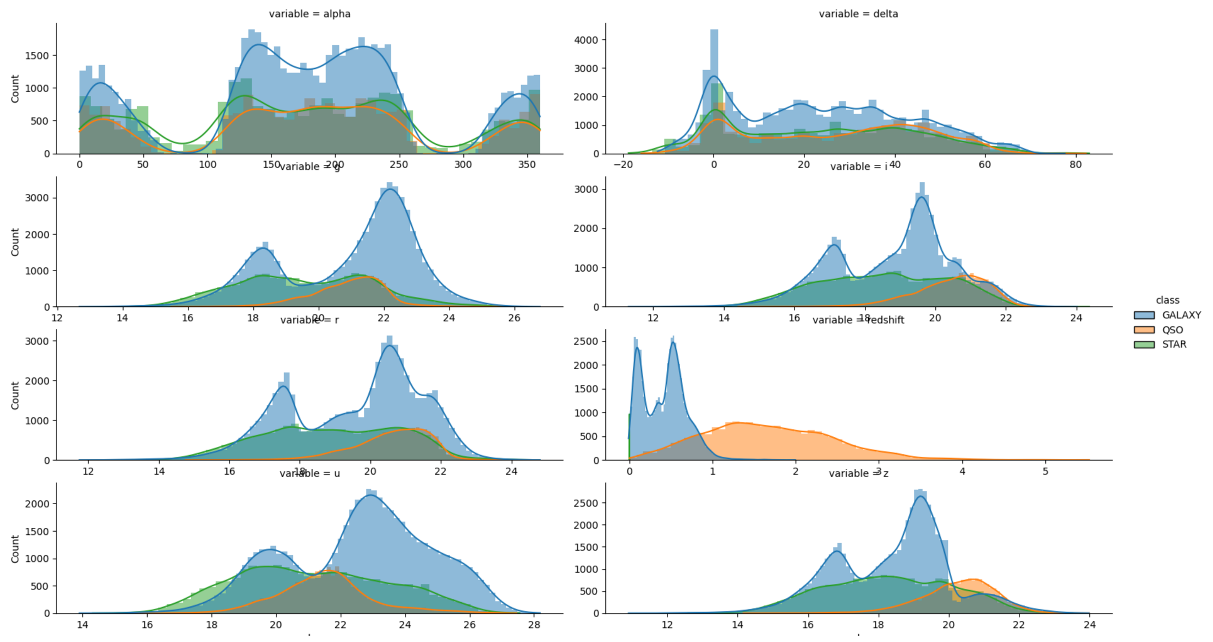
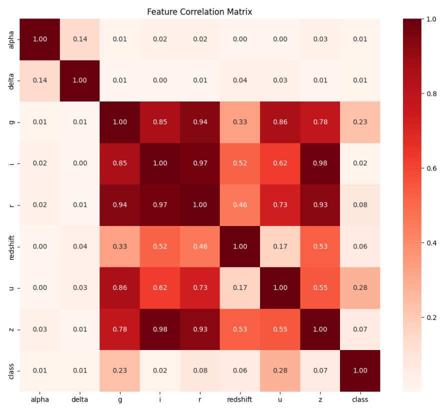
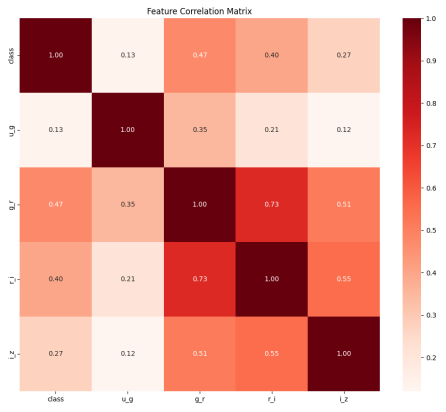


Figure 3: **Feature Distribution**: This plot visualizes how each of the post-preprocessing features is distributed inside each class. The most interesting feature is *redshift*, as can be seen by its peculiar distribution with respect to the other ones.



(a) **Initial Correlation**: Noticeable high correlation (> 0.9) between several features.



(b) **Final Correlation**: Correlation between engineered features is much lower (this is effectively the C.M. for the X_{linear} dataset.)

Figure 4: **Correlation Matrix Comparison**: The effect of feature engineering and deletion of highly correlated features

6 Models Implemented

The following supervised models were implemented using **5-Fold Cross-Validation** [3], with a `StandardScaler` applied within each pipeline to prevent data leakage.

- **K-Nearest Neighbors (KNN):** Trained on the X_{linear} dataset. Tuned for $k \in [1, 100]$.
- **Random Forest Classifier (RFC):** Trained on the X_{tree} dataset. To optimize performance while handling class imbalance, the model utilized 'class_weight='balanced'' and a **GridSearchCV** was performed over the following parameter space:
  - $n\_estimators \in [50, 150]$
  - $max\_depth \in [None, 10, 20]$
  - $min\_samples\_split \in [2, 5, 10]$
  - $min\_samples\_leaf \in [1, 3, 5]$
- **Support Vector Machine (SVM):** Trained on the X_{linear} dataset with $class\_weight =' balanced'$. Due to the high computational cost of SVMs ($O(n^3)$), hyperparameter tuning was performed on a random **10% subset** of the training data. The model was tuned for regularization $C \in [0.001, 0.01, 0.1, 1, 10, 100]$ and kernel type $kernel \in ['rbf',' linear',' sigmoid']$. The best configuration was then retrained on the full dataset.

For unsupervised learning, **K-Means** and **Gaussian Mixture Models (GMM)** were utilized with $k = 3$ (or $n\_components = 3$) to match the known number of classes. To test the hypothesis that dimensionality reduction aids cluster separation, both models were trained on two distinct versions of the data:

1. **Raw Scaled Data:** The full X_{linear} feature set.
2. **PCA Reduced Data:** The X_{linear} set projected onto 2 Principal Components ($n\_components = 2$).

7 Experiments and Results

Each dataset was split into 80% training and 20% testing sets. The primary metric for model selection was the **Macro F1-Score** to account for potential class imbalances. The following are the most important ones to discuss.

7.1 The Redshift Feature

An important observation needs to be made regarding the *redshift* feature, that is the influence that it has on the performance of models. *Redshift* is a phenomenon where the light (or other electromagnetic radiation) emitted by a celestial object is shifted toward the red end of the electromagnetic spectrum. This feature is heavily tied to the **Doppler Effect** and how light gets stretched by gravity, cosmological stretch and distance. By nature, celestial objects that emit light are highly characterized by this metric, and it is, in fact, a very telling feature: during development, it was discovered that even with very minimal corrective actions on the dataset, the performance of trained models that included these features were already on a degree of accuracy well above 0.97, effectively meaning that the logical link between the class of the object and the redshift feature was so high that it was almost as if the data remained labeled even after removing the class feature during training.
For the purpose of this project, and in order to create a bigger challenge, it was decided to also drop *redshift* completely from the features included in the training and testing dataset. By doing

so, classification became less trivial and was solely based on color, which was also a nice deviation from the SOTA.

7.2 Synthetic Features

Removing the *redshift* feature and *alpha* and *delta* features (position based features which have no meaningful information at all as to what object they refer to) meant that the model now only had very few color-based features to work with, namely:

- **u**: Ultraviolet filter in the photometric system
- **g**: Green filter in the photometric system
- **r**: Red filter in the photometric system
- **i**: Near Infrared filter in the photometric system
- **z**: Infrared filter in the photometric system

With some of these being highly correlated features (as shown in [4a]), the new challenge was to find a way to work with the few remaining and extrapolate meaningful data from them.
This is when it was decided to create the following **synthetic features**:

- **u_g:** $u - g$ (**Ultraviolet-Green Index**: Sensitive to hot, young stars and Quasar UV-excess)
- **g_r:** $g - r$ (**Green-Red Index**: A standard measure of surface temperature for main-sequence stars)
- **r_i:** $r - i$ (**Red-Near Infrared Index**: Useful for detecting cooler, redder objects like M-dwarfs)
- **i_z:** $i - z$ (**Near Infrared-Infrared Index**: Helps distinguish high-redshift galaxies)

Introducing synthetic features means that models now have compound attributes with real-world meaning they can use during training and evaluation. As mentioned previously, the engineered features were added to the already present ones for RFC and substituted for all the other implemented classification and clustering methods.

7.3 Pipeline Training

Each model was implemented inside its own pipeline, which handled training, fitting and evaluation. This solution made it possible to give each model the best chance to outperform the other ones, implementing further case-specific preprocessing and tuning on top of the commonly shared one.

7.4 Supervised Learning Performance: Hyperparameter Tuning and Evaluation

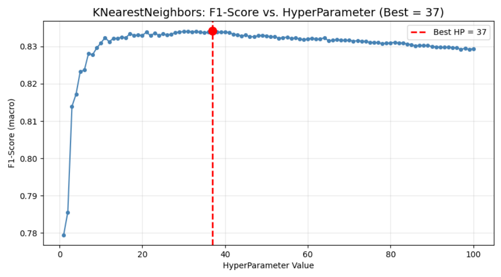
Cross-validation was used to find the optimal hyperparameters by training loops: each classification model was given incrementally higher hyperparameters and different combinations of them to work on the same dataset as all its predecessors. This method made it possible to investigate performance based solely on hyperparameter and not on change of data. Every model will be discussed separately.

7.4.1 K-Nearest Neighbor Classifier

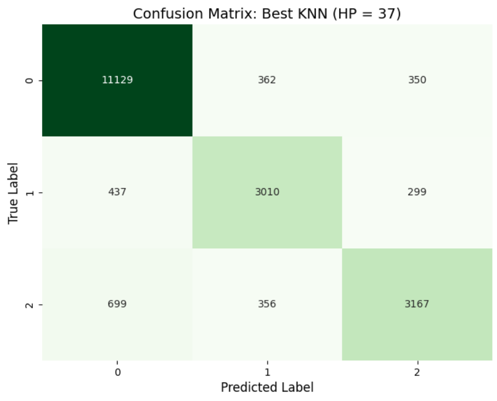
The choice to explore the implementation of the KNN model came from how simple and straight forward it is and from how well it scales, training wise, with iteration on hyperparameter values: since it only requires one, iteration to decide its best value is efficient [4]. It also works well with data that is assumed to be distributed in a way that renders similar data points in close proximity to each other, which looked to be the case when performing the analysis of Figure [1]. Training loop result can be seen in Figure [5a]. Having obtained the best value for K, a new KNN model was trained specifically with this hyperparameter value to show evaluation metrics on the best version of KNN obtainable on this dataset. The obtained performance metrics were the ones in Table [1].The confusion matrix of KNN with $n\_neighbors = 37$ can be seen in Figure [5b].

Table 1: KNN Performance Scores

F1-Score	**Accuracy**	**Precision**	**Recall**
0.8341	0.8736	0.8482	0.8312



(a) **K in KNN**: F1-score grows to a highpoint at $BestK = 37$, where it starts decreasing. In this case, there is a clear winner as for best hyperparameter value.



(b) **Confusion Matrix for KNN**

Figure 5: **KNN Performance:** A look into how well KNN performed at its best.

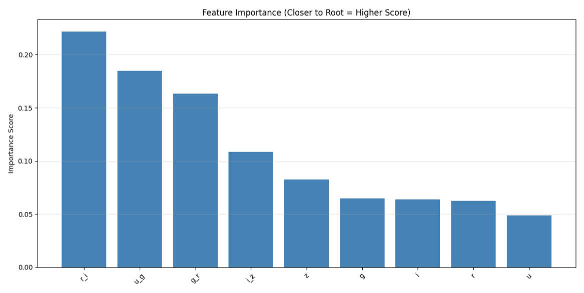
7.4.2 Random Forest Classifier

RFC models avoid overfitting data extremely well by taking the majority vote from multiple individual trees. It also is extremely robus when presented with correlated features, hence the decision of feeding it the X_{tree} dataset. To optimize the model, a **GridSearchCV** was performed to find the best combination of tree depth and split criteria. The grid search identified the optimal hyperparameters, favoring a balance between complexity and generalization. This approach ensured that the trees were not only sufficient in number but also individually optimized to handle the raw and synthetic feature mix. The graph at Figure [6a] shows the feature importance, that is how much information gain was provided by each feature: the higher the bar is, the more that feature helped RFC separate data (these are the features that are higher up in the trees, closer to the root). The features on the lower end are those who were used primarily for tie-braking deep down the trees, mostly helping to refine special edge cases. An in-depth visualization of this concept was provided in Figure [7].

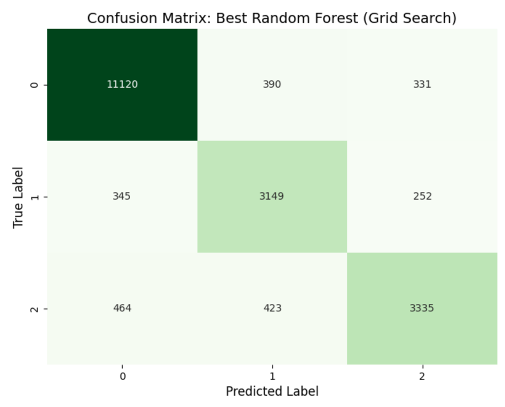
With the best hyperparameter configuration found by the grid search being: $max\_depth = 20$, $min\_samples\_leaf = 3$, $min\_samples\_split = 2$, $n\_estimators = 150$. The specific optimally configurated model, when trained in isolation, produced the following evaluation metrics can be seen in Table [2]. The confusion matrix of RFC with $n\_estimators = 150$ can be seen in Figure [6b].

Table 2: RFC Performance Scores

F1-Score	**Accuracy**	**Precision**	**Recall**
0.8583	0.8887	0.8594	0.8565



(a) **Feature Importance**: How much the features given to RFC helped classify.



(b) **Confusion Matrix for RFC**

Figure 6: **RFC Performance:** A look into how well RFC performed at its best.

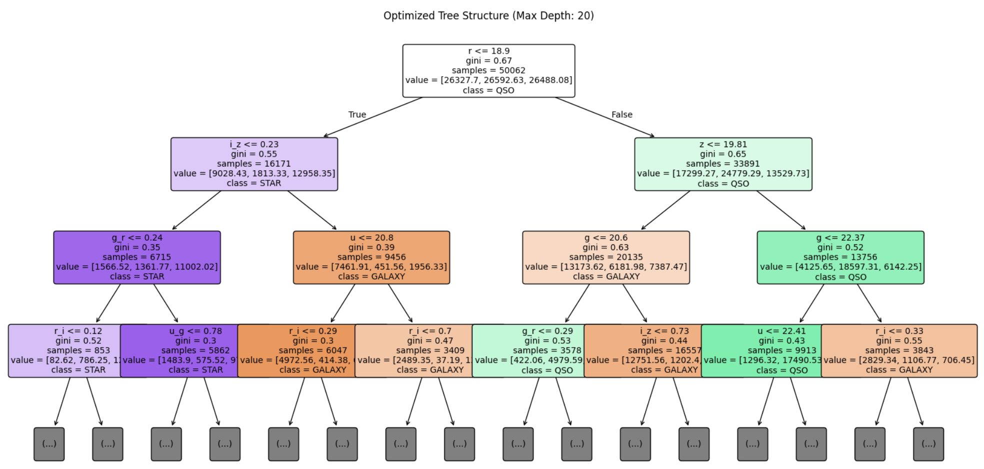

Optimized Tree Structure (Max Depth: 20)
r <= 18.9 gini = 0.67 samples = 50062 value = [26327.7, 26592.63, 26488.08] class = QSO
True
False
i_z <= 0.23 gini = 0.55 samples = 16171 value = [9028.43, 1813.33, 12958.35] class = STAR
z <= 19.81 gini = 0.65 samples = 33891 value = [17299.27, 24779.29, 13529.73] class = QSO
g_r <= 0.24 gini = 0.35 samples = 6715 value = [1566.52, 1361.77, 11002.02] class = STAR
u <= 20.8 gini = 0.39 samples = 9456 value = [7461.91, 451.56, 1956.33] class = GALAXY
g <= 20.6 gini = 0.63 samples = 20135 value = [13173.62, 6181.98, 7387.47] class = GALAXY
g <= 22.37 gini = 0.52 samples = 13756 value = [4125.65, 18597.31, 6142.25] class = QSO
r_i <= 0.12 gini = 0.52 samples = 853 value = [82.62, 786.25, 1 class = STAR
u_g <= 0.78 gini = 0.3 samples = 5862 value = [1483.9, 575.52, 9 class = STAR
r_i <= 0.29 gini = 0.3 samples = 6047 value = [4972.56, 414.38, class = GALAXY
r_i <= 0.7 gini = 0.47 samples = 3409 value = [2489.35, 37.19, 1 class = GALAXY
g_r <= 0.29 gini = 0.53 samples = 3578 value = [422.06, 4979.59 class = QSO
i_z <= 0.73 gini = 0.44 samples = 16557 value = [12751.56, 1202.4, class = GALAXY
u <= 22.41 gini = 0.43 samples = 9913 value = [1296.32, 17490.53 class = QSO
r_i <= 0.33 gini = 0.55 samples = 3843 value = [2829.34, 1106.77, 706.45] class = GALAXY
(...)

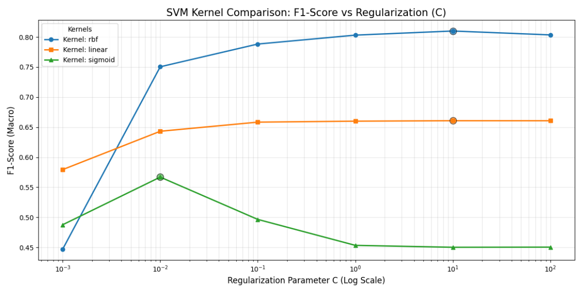
Figure 7: **Optimized Tree**: An in-depth view of the feature importance. The information is the same as the histogram, but here it is more apparent and easier to see how tree-like the organization of most important features really is: the closer to the root, the more discriminative a feature is.

7.4.3 Support Vector Machines

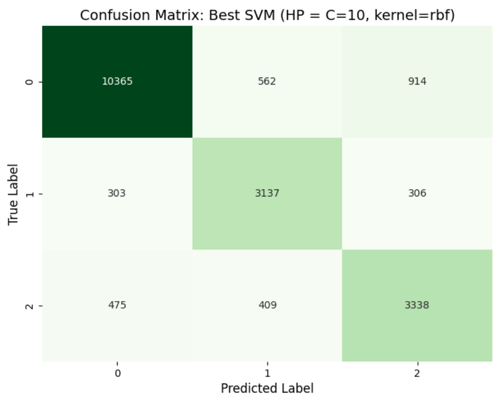
SVM models have a double degree of freedom: different kernel modes and hyperparameter value. Performance and how well it can find an optimal hyperplane to separate classes in an n-dimensional space hugely vary based on them. Due to the high computational cost of training SVMs on large datasets, hyperparameter tuning was performed on a random 10% subset of the training data. The best performing configuration found on the subset was then used to retrain the model on the full training dataset. The train loop for this model is displayed in Figure [8a]. It is apparent the best performing mode was $'rbf'$, which is the expected result: this indirectly confirms that the data is not linearly separable and therefore a linear kernel mode could have never performed well enough to surpass its competitors. RBF has the highest performance because it can sort of bend the separation line around data, better fitting the separation and achieving a higher precision and accuracy.
The statistics that follow are the ones obtained by training a SVM with kernel mode 'rbf' and with $C = 10$ and be seen in Table [3]. The confusion matrix of said configuration can be seen in Figure [8b].

Table 3: SVM Performance Scores

F1-Score	Accuracy	Precision	Recall
0.8092	0.8501	0.8087	0.8345



(a) **Performance of SVM**: Comparison between kernel modes and how well they performed cycling through the C values.



(b) **Confusion Matrix for SVM**

Figure 8: **SVM Performance:** A look into how well SVM performed at its best.

7.4.4 Final Metrics

From the analysis that was conducted, out of all the models that were trained and evaluated, the winner, that is the model that achieved the best results in terms of performance metrics, is the **Random Forest Classifier** model. A small table summarizing the results of all models was provided in Table [4].

Table 4: Model Performance Comparison (Best Hyperparameter Versions)

Model	F1 (Macro)	Accuracy	Precision (Macro)	Recall (Macro)
KNN	0.8341	0.8736	0.8482	0.8312
Random Forest	**0.8583**	**0.8887**	**0.8594**	**0.8565**
SVM	0.8092	0.8501	0.8087	0.8345

7.5 Clustering Analysis

Clustering performance was evaluated by comparing models trained on high-dimensional scaled data against those trained on a two-dimensional PCA projection. This comparison highlights a significant trade-off: while dimensionality reduction simplifies the feature space and does, in some way, help models produce a higher quality result, it does not inherently make the data more clusterable.

A final table was compiled, listing all the obtained evaluation metric results, which is available at Table [5].

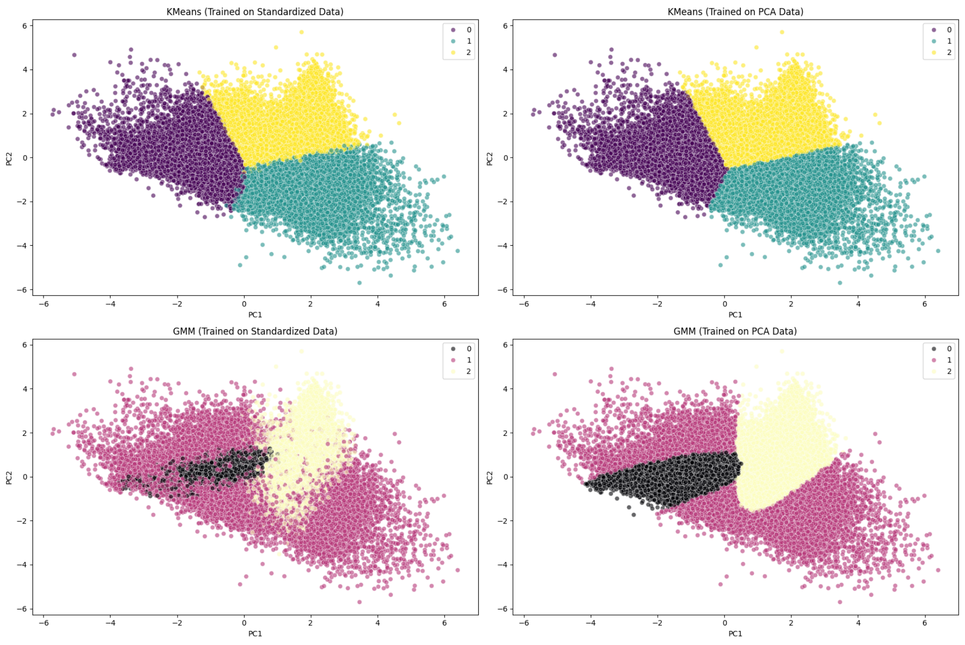


Figure 9: **Clustering Visualization**: Comparison between K-Means and GMM trained on raw scaled features (left) versus PCA-reduced features (right). The PCA projection visually collapses the variance into distinct regions, though significant overlap remains.

Table 5: Raw vs PCA Clustering: comparison of K-Means and GMM

Model	ARI	NMI	Silhouette
K-Means (raw)	0.1068	0.1904	0.3551
K-Means (PCA)	0.1129	0.1919	0.4338
GMM (raw)	0.0990	0.1466	0.1517
GMM (PCA)	0.0652	0.1302	0.3894

As illustrated in Figure [9], PCA significantly improved the Silhouette Score for both models clustering models. This indicates that PCA successfully filtered out noise, creating more defined and separable spatial groupings. However, quantitative metrics reveal that this spatial clarity did not translate to meaningful classification:

- **Weak Ground-Truth Alignment**: The Adjusted Rand Index (ARI) remained extremely low across all configurations, peaking at only 0.1129 for K-Means on PCA data. This suggests that the natural clusters formed by the algorithms have very little overlap with the actual *GALAXY, STAR,* and *QSO* labels.
- **Information Loss in GMM**: Interestingly, while GMM's Silhouette score improved with PCA, its ARI actually dropped from 0.0990 to 0.0652. This indicates that the Gaussian components in the reduced 2D space lost useful density information present in the original feature set.
- **Conclusion on Viability**: With Normalized Mutual Info (NMI) values hovering below 0.20, these features, even when optimized via PCA, are insufficient for unsupervised discovery of star classes. The data appears to not exist in discrete, well-separated density clusters.

8 Conclusion

This project set out to evaluate the performance of supervised and unsupervised machine learning models on the task of celestial object classification. A critical component of this study was the deliberate exclusion of the *redshift* feature, forcing the models to rely solely on photometric color indices and reference magnitudes.

8.1 Supervised Learning Synthesis

The results demonstrate that while *redshift* is the strongest predictor for cosmological classification, it is not strictly necessary for achieving reasonable accuracy in a controlled, machine learning context.

- **Non-Linearity:** The fact that **K-Nearest Neighbors (KNN)** classifier and **Random Forest** outperformed the linear configurations of the SVM model confirms that the decision boundaries between Stars, Galaxies, and Quasars in color space ($u - g$, $g - r$, etc.) are highly non-linear and complex.
- **Robustness:** Although RFC does require more training time and computational expenses, it is by far the most reliable, robust and rewarding model out of the ones that were considered in this project: it is able to efficiently reach satisfactory solutions in terms of accuracy and does not overfit data.

8.2 Unsupervised Learning Considerations

The clustering analysis provided a crucial negative result. Despite the application of PCA to reduce noise, neither **K-Means** nor **GMM** performed well. This suggests that in the photometric feature space, Stars, Galaxies, and Quasars do not form distinct, separated blobs. Instead, they likely exist on a space where classes overlap significantly. This finding highlights the necessity of labeled data (Supervised Learning) for this specific astronomical task.

8.3 Possible Future Enhancements

While the removal of redshift provided a challenging testbed for these algorithms, modern astronomical surveys often require high precision. Future iterations of this work could explore the use of **Hierarchical Classification**, which separate Stars from Extragalactic objects first, then classifying Galaxies and Quasars, or **Deep Learning on Raw Spectra**, which utilizes Neural Networks on the raw spectral data rather than extracted tabular features to capture feature subtleties invisible to standard photometry. In conclusion, while it is possible to apply simple classification, the distribution of the data makes it impossible to employ rudimentary clustering methods expecting a very precise outcome.

References

[1] Sloan Digital Sky Survey (SDSS), *Star Classification Dataset*, Kaggle Repository.

[2] Pedregosa, F., et al., ”Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, 2011.

[3] Beyan, C., ”Machine Learning Course Slides,” University of Verona, A.Y. 2025/2026.

[4] Géron, A., ”Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow,” *O'Reilly Media*, 2019.

[5] York, D. G., et al., ”The Sloan Digital Sky Survey: Technical Summary,” *The Astronomical Journal*, 2000.

[6] VanderPlas, J., ”Python Data Science Handbook: Essential Tools for Working with Data,” *O'Reilly Media*, 2016.