Stellar Classification - SDSS17: Classifiers Comparison

Selin Dökmen, Kevin Garofalo, Francesco Renna, Vincenzo Siano, Inya Verstegen

Abstract
Stellar classification is a fundamental process in astronomy that is used to categorize stars based on their spectral characteristics. In this study, supervised machine learning algorithms are implemented to solve the stellar classification, which is a multiclass classification problem, using Knime Analytics Platform. Therefore, classification models were developed in order to decide whether an astronomical object is a Star, Galaxy or Quasar. To achieve this, the project deals with the class imbalance problem, and the understanding of the most important features, using feature selection approaches. The performance is then evaluated to find the optimal classification algorithm that would be used to classify stellar objects.

Contents

1 Introduction

The categorizations of astronomical objects are important in gaining insight into the universe. Classifying these objects accurately is essential for a wide range of astronomical studies and applications. Traditionally, classification models in astronomy differentiate between three primary categories: stars, galaxies, and quasars. However, the complexity of distinguishing among these classes can pose challenges for data-driven approaches, especially when computational efficiency and model interpretability are priorities.

This report evaluates how we can apply binary transformation approaches, such as One-vs-All, to adapt native binary classification algorithms for multi-class classification[1]. To address this, we first perform analysis and preprocessing activities, and then compare the performance of the different classification algorithms available in Knime Analytics Platform.

2 Dataset Description

The dataset selected to address our goal is the "Stellar Classification Dataset - SDSS17", available on Kaggle[2].

This dataset contains 100,000 observations of space taken by the *SDSS (Sloan Digital Sky Survey)*. Every observation is described by 17 *feature* columns and 1 *class* column which identifies it to be either a star, galaxy or quasar.

- `obj_ID`: Unique identifier for each astronomical object, used by the CAS.
- `alpha`: Right ascension angle (J2000 epoch).
- `delta`: Declination angle (J2000 epoch).
- `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.
- `run_ID`: Run number, which identifies the specific scan.
- `rerun_ID`: Rerun number, which specifies how the image was processed.
- `cam_col`: Camera column, which indicates the scanline within the run.
- `field_ID`: Field number, which identifies each field.
- `spec_obj_ID`: Unique identifier for optical spectroscopic objects. This means that 2 different observations with the same value of this ID must share the output class.
- `class`: Object class (galaxy, star or quasar).
- `redshift`: Redshift value, based on the increase in wavelength.
- `plate`: Identifier for each SDSS plate.
- `MJD`: Modified Julian Date, indicating when a piece of SDSS data was taken.
- `fiber_ID`: Specifies the fiber that pointed the light at the focal plane in each observation.

2.1 Class Distribution

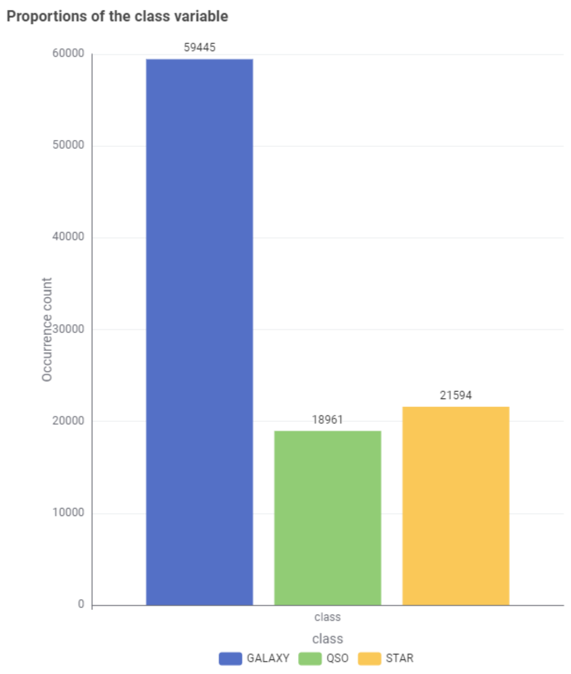

Figure 1: Proportions of the unbalanced dataset.

As we can see from Fig. 1, quasars appeared less frequently compared to galaxies and stars, aligning with the expectation that quasars are rarer and further celestial objects.
The dataset is *unbalanced* with respect to the target variable. This needs to be taken into account and addressed, and it will be further analyzed.

3 Data Preparation

In order to make the data more suitable for the classification task, some preprocessing steps have been applied, including dealing with outliers, checking for possible missing values, column filtering and normalizing the dataset.

3.1 Preprocessing and Transformation

After reading our dataset, we handled numeric *outliers* by using the suited KNIME node, which identifies outliers using the *Interquartile Range (IQR).*

After dealing with the outliers, a *missing value* node is used to manage this issue, replacing missing integers and doubles with a median value and dropping rows with missing string values in a column. Even though no missing values were observed in the dataset, this handling is still performed to enable an eventual scalability.

Afterwards, the important features that were determined by Feature Importance process were retained to reduce noise and computations. The `spec-obj-ID` feature is also maintained: even though it seems to have less importance, it is actually a critical element for differentiating each class, as described in the Section 2.

Then, we have applied Z-score normalization on the selected columns to scale our data such that the values are normally distributed. This ensured that no single feature affected the model disproportionally.

3.2 Exploratory Data Analysis

Exploratory data analysis methodologies are applied in order to visualize the data distributions and correlations, and improve understanding beforehand.

We mainly produced a linear correlation matrix (as visible in Fig. 2) between features, and it emerged that the features with the highest correlation values with `class` are `u`, `g` and `redshift`.

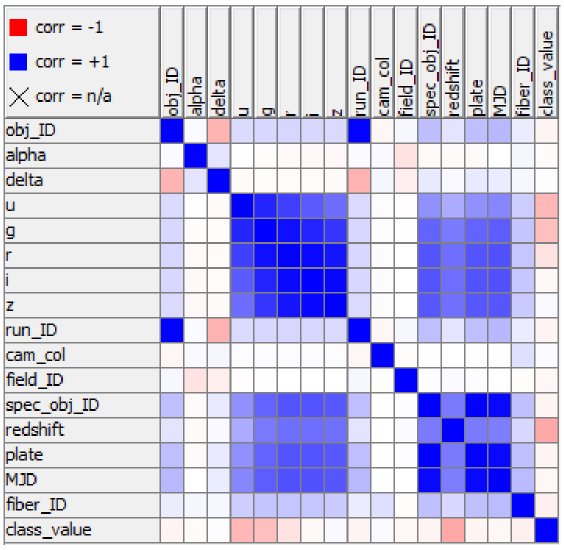

Figure 2: Feature correlation matrix.

The exploration of the relationship of `u` and `g` with other features displayed difficulties for determining the class of the observations. Despite this, we decided to further explore `redshift` against other features, and this led to the conclusion that `redshift` is the most significant feature to classify our records.

3.3 Feature Importance

Feature importance has been applied to detect the most important features and use them to reduce computations and noise, without losing relevant information. This has been implemented with Tree Ensemble, which learns an ensemble of decision trees, such as Random Forest variants. This method gave us an importance score for each feature, allowing us to understand their relative significance in the classification task.

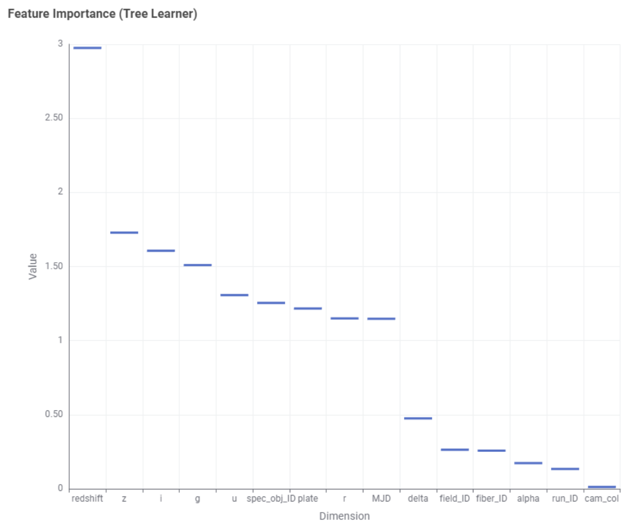

Figure 3: Feature Importance based on Tree Learner.

We used every feature with a score value greater than 1, except for `MJD` and `plate`. We excluded these two because it turned out that they do not affect performance at all.

4 Classification Models

In this project, various classification techniques were used to identify the most suitable approach. Seven classifiers were applied throughout the Knime workflow, categorized into the following 4 groups:

- **Heuristic Models**: In this project, Decision Tree (J48), k-Nearest Neighbor (kNN), Random Forest and XGBoost (specific implementation of Gradient Boosting) are applied.
- **Regression based Models**: Logistic regression is used, which models a linear relationship between inputs and a categorical output, using a parametric conditional probability.
- **Separation Models**: Support Vector Machine (Weka's SMO implementation) and Multi-Layer Perceptron are used, which both partition the attributes' space.
- **Probabilistic Models**: Naive Bayes is applied, exploiting Bayes formula and computing posterior probability to classify records.

4.1 Performance Measures

To evaluate the performance of the models, a variety of measures were computed. These metrics provide a detailed evaluation of classification performance under different conditions. The measures applied are the following:

- **Accuracy**: The number of correct predictions on the total number of predictions. It is calculated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  Moreover, it measures the capability of the classification model to give reliable predictions on new records. However, the accuracy measure is not suitable for imbalanced datasets, as in our case. Therefore, the following measures are preferred.

- **Precision**: The fraction of correctly predicted positive records out of all positive predictions. It provides information on the ability of the model to minimize false positives.

$$p = \frac{TP}{TP + FP}$$

- **Recall**: The fraction of actual positive records correctly identified by the model. It provides information on the ability of the model to minimize false negatives.

$$r = \frac{TP}{TP + FN}$$

- F_1 **measure**: Combines precision (p) and recall (r) into a single metric by calculating their harmonic mean:

$$F_{measure} = \frac{2 \cdot r \cdot p}{r + p}$$

- **ROC and AUC**: The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds. The ROC curve depicts the performance of a classifier without regard to class distribution. Thus, it is useful for confronting different classification models, considering the Area Under the Curve (AUC) value. It ranges from 0 to 1 and measures the overall performance of the classifier. The closer it is to 1, the closer the classifier has TPR to 1 (ideal classifier).

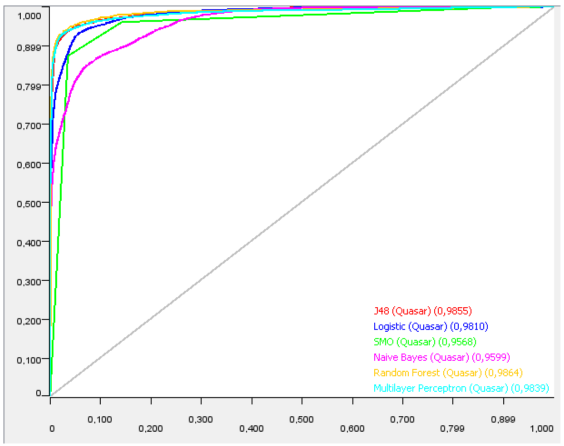

Figure 4: ROC of Quasar class (initially the least occurrent label) for balanced data from the MultiClassClassifier node.

5 Classification and Results

This section presents and compares the results of the classification methods to determine the optimal model. Before evaluating them, the impact of class imbalance is examined by comparing models for balanced and imbalanced dataset. Additionally, the performance of Holdout and k-fold Cross-Validation is assessed for both the Classification with the specific learner nodes, and the Classification employing a metaclassifier node. To effectively handle multi-class datasets using native binary classifiers, the MultiClass Classifier node was employed[4]. It decomposes multi-class problems into multiple binary classification tasks, enabling a structured comparison between binary classification models and those adapted for multi-class scenarios.

The performance was evaluated using multiple measures, including Accuracy, F1-Measure, Recall, Precision, and AUC. The analysis aimed to compare the effectiveness of various classifiers using different approaches. Furthermore, to facilitate a comprehensive analysis of model performance, the results obtained from different classification methods were combined, providing a clearer and more structured evaluation framework.

5.1 Comparison with Imbalanced Dataset

5.1.1 Holdout

The holdout method is a widely used validation technique for assessing the performance of classification models by dividing the dataset into training and testing subsets. The dataset was partitioned using an 80%-20% split, where 80% of the data was allocated for training and 20% for testing.

5.1.2 k-folds Cross Validation

K-fold Cross-Validation is a robust technique for evaluating the performance of classification models by partitioning the dataset into multiple subsets, ensuring a more reliable assessment of generalization error. In this study, a 5-fold cross-validation procedure was implemented for every model, dividing the dataset into five equal subsets. Each subset was used once as a test set while the remaining four subsets were used for training, allowing every data point to contribute to both training and testing. This process provides a more stable estimate of model performance. The cross-validation procedure was carried out using the X-Partitioner node.

5.2 Comparison with Balanced Dataset

5.2.1 Equal Size Sampling

As already mentioned, class imbalance is a critical issue in classification tasks, where an unequal distribution of class instances can lead to biased models that favour the majority class, thereby reducing the predictive performance for minority classes. Different techniques can be employed to correct class imbalance[3], including downsampling the majority class, or oversampling the minority class through methods such as the *Synthetic Minority Over-sampling Technique (SMOTE)*[5]. For our case, *Equal Size Sampling* was applied the node of the same name, which systematically removes instances from the dataset to ensure an equal distribution of classes within a nominal variable. By ensuring balanced class representation, classification models can achieve improved generalization error and more reliable performance across all categories.

5.2.2 Holdout

As before, the holdout method was also applied to the balanced dataset using an 80%-20% split. Balancing the dataset ensures that models do not favour the majority class, leading to a more representative evaluation of classification performance.

5.2.3 k-folds Cross Validation

The same considerations made in the corresponding section of the imbalanced case also apply here.

5.3 Feature Selection

Feature selection is a crucial step in classification, aiming to identify the most relevant variables while reducing redundancy and improving model efficiency. When applied to a balanced dataset, it ensures that the selected features are not biased towards the majority class, leading to more reliable predictions.

To enhance analysis, both Filter approaches and Wrapper method were implemented, allowing a clearer assessment of feature selection and classification performance.

5.3.1 Univariate Filter

Univariate Filter was performed using the *AttributeSelectedClassifier* node in KNIME. Using this method, attributes are selected before learning the classifier through an objective function. The `Ranker` search method was applied, evaluating each feature individually based on statistical criteria, such as *Information Gain Ratio*, which evaluates the worth of an attribute by measuring the information gain with respect to the class. This approach ranks attributes selecting the top five most relevant features for classification. Univariate filtering is computationally efficient and straightforward, making it effective for reducing dimensionality. However, it does not take into account for relationships between attributes, which may result in the selection of redundant variables.

5.3.2 Multivariate Filter

Multivariate Filter was conducted with the same node but with the `BestFirst` search method. Unlike univariate approaches, this one evaluates feature subsets rather than individual attributes, allowing for the identification of the most informative combinations. This method explores different subsets by considering interactions between features, ensuring that selected attributes contribute unique information to the model. Although more computationally intensive, multivariate filters typically lead to improved classification performance by reducing redundancy and capturing complex relationships within the data. In this study, this solution selects only two features, compared to the 5 features selected by the univariate filter, but with similar performance.

5.3.3 Wrapper

In this project, the Wrapper method was employed for feature selection using the same node used for the Filter method, although configured with the `WrapperSubsetEval` evaluator. Wrappers assess the usefulness of feature subsets by evaluating their impact on a specific classifier's performance, effectively tailoring the feature selection process to the model in use. In our case, the evaluator conducts a 5-fold Cross-Validation to estimate the accuracy of the learning scheme for a set of attributes. This involves partitioning the dataset into five subsets, sequentially using each subset for testing while training on the remaining data.

The wrapper method is significantly computationally intensive due to the iterative nature of the process. This approach is particularly advantageous when interactions between features significantly influence model performance. It is worth noting that kNN, MLP, and SMO models were excluded due to their high computational cost, which made the feature selection process using wrapper excessively time-consuming.

5.4 Results

The performance of the classification models was evaluated using multiple metrics, including Accuracy, F1-Measure, Recall, Precision, and Area Under the Curve (AUC). The analysis aimed to compare the effectiveness of various classifiers. As already discussed, Accuracy is less robust and informative when applied to imbalanced datasets, because it may yield misleading results due to the disproportionate class distribution. For this reason, other measures (such as F1-measure, Recall, Precision) are preferred for the evaluation of the imbalanced configuration. Therefore, the Accuracy and AUC values of the balanced dataset for the metaclassifier node are highlighted in this report. In fact, the tables below show the accuracy performance evaluation of the metaclassifier learner for the holdout and cross-validation methods.

Model	**Accuracy**
Random Forest	0.964
Random Forest_CV	0.961
Multilayer Perceptron	0.962
Multilayer Perceptron_CV	0.957
J48	0.960
J48_CV	0.956
Logistic Regression	0.936
Logistic Regression_CV	0.934
Naïve Bayes	0.884
Naïve Bayes_CV	0.886
SMO	0.839
SMO_CV	0.839

Table 1: Model Accuracy for MultiClass-Classifier node (with Holdout and CV=Cross-Validation)

The analysis reveals that the top-performing models for both methodologies are consistent, with Random Forest, Multilayer Perceptron, and J48 showing nearly identical performance metrics.

The tables below present the AUC values achieved by each model, categorized by class value and methodology (holdout and CV). Notably, Random Forest demonstrates the highest AUC values across the evaluated conditions.

Classifier	**Area Under Curve**
Random Forest (Star)_CV	**1**
Random Forest (Star)	**1**
SMO (Star)_CV	0.988
SMO (Star)	0.99
Naïve Bayes (Star)_CV	0.998
Naïve Bayes (Star)	0.998
Multilayer Perceptron (Star)_CV	0.998
Multilayer Perceptron (Star)	0.999
Logistic Regression (Star)_CV	0.998
Logistic Regression (Star)	0.999
J48 (Star)_CV	0.998
J48 (Star)	0.999

Table 2: Area Under Curve for Star class (MultiClassClassifier node)

Classifier	Area Under Curve
Random Forest (Galaxy)_CV	**0.987**
Random Forest (Galaxy)	**0.988**
Multilayer Perceptron (Galaxy)_CV	0.981
Multilayer Perceptron (Galaxy)	0.985
J48 (Galaxy)_CV	0.979
J48 (Galaxy)	0.982
Logistic Regression (Galaxy)_CV	0.947
Logistic Regression (Galaxy)	0.952
Naive Bayes (Galaxy)_CV	0.923
Naive Bayes (Galaxy)	0.921
SMO (Galaxy)_CV	0.894
SMO (Galaxy)	0.899

Table 3: Area Under Curve for Galaxy class (MultiClassClassifier node)

Classifier	Area Under Curve
Random Forest (Quasar)_CV	**0.986**
Random Forest (Quasar)	**0.988**
Multilayer Perceptron (Quasar)_CV	0.984
Multilayer Perceptron (Quasar)	0.986
J48 (Quasar)_CV	0.985
J48 (Quasar)	0.988
Logistic Regression (Quasar)_CV	0.981
Logistic Regression (Quasar)	0.983
Naive Bayes (Quasar)_CV	0.96
Naive Bayes (Quasar)	0.96
SMO (Quasar)_CV	0.957
SMO (Quasar)	0.959

Table 4: Area Under Curve for Quasar class (MultiClassClassifier node)

Moreover, the table below shows the Accuracy values for the specific Learner nodes which perform also XGBoost and kNN, in addition to Random Forest and the other classifiers that have already been evaluated before. In this case, we observe a high accuracy for XGBoost, both for holdout and CV, comparable to the Random Forest.

Model	Accuracy
XGBoost	**0.962**
XGBoost_CV	**0.961**
Random Forest	**0.962**
Random Forest_CV	**0.959**
kNN	0.95
kNN_CV	0.945

Table 5: Model Accuracy for specific Learner nodes

6 Conclusions

The objective of this project consisted in determining one or more multiclass classification models that allow to accurately predict the category of astronomical objects. It emerged that the important aspects are the redshift and the photometric filters. From these initial discoveries, we then performed the effective classification task and compared the results using different methods, like comparing imbalanced and balanced datasets, techniques such as k-folds Cross-Validation and hold-out. Finally, we also applied filter and wrapper approaches for feature selection process in order to retain the most crucial attributes and reduce noise and computational load.

Then, we evaluated the performance of the following models with hold-out and k-fold Cross-Validation: XGBoost, Random Forest, kNN, J48 (Decision Tree), Logistic Regression, Naive Bayes, Multi-Layer Perceptron, SMO (Support Vector Machine). Analyzing all these classifiers for multiclass classification task, XGBoost and Random Forest turned out to be the optimal solution for specific learners approach, while Random Forest emerged to be the best one also for metaclassifier.
To be more precise, both XGBoost and Random Forest achieve approximately an accuracy of 96%. Regarding the class imbalance problem, Equal Size Sampling was discovered to be the best approach, considering performances and computational time, also avoiding favouring the majority class, as it would happen with class imbalance.

Finally, feature selection played a pivotal role in enhancing classification performance by identifying the most relevant variables among the 15 columns available - removing earlier `obj_id` and `run_id` from the original 17 - while also reducing redundancy. Both Filter and Wrapper approaches were implemented, each offering distinct advantages. The Univariate Filter, utilizing the Ranker method and Information Gain Ratio, efficiently selected 5 features, but this method doesn't take into account inter-attribute relationships. In contrast, the Multivariate Filter with CFS and BestFirst search method identified only 2 features while maintaining similar performance, demonstrating the benefits of evaluating feature subsets for capturing complex relationships. Then, the Wrapper method, though computationally intensive, provided tailored feature selection through 5-fold Cross-Validation. Despite its potential for higher F-measure and accuracy, models such as kNN, XGBoost, MLP, and SMO were excluded due to excessive computational demands. It emerged that with this method, J48 and Random Forest achieve the highest values for F-measure and accuracy, respectively selecting 4 and 6 features. However, we have to keep in mind of the trade-off between performance and computational efficiency for feature selection strategies.

From this study, it is clear that the chosen stellar dataset is natively effective and optimized for the classification of stars, galaxies and quasars. Therefore, it allows getting high performance and clear understanding without requiring additional features and overall preprocessing.

References

[1] Datacamp. *Classification Algorithms compatible with Multiclass Classification.* URL: `https://www.datacamp.com/blog/classification-machine-learning#multi-class-classification-themu`.

[2] fedesoriano. *Stellar Classification Dataset - SDSS17.* URL: `https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17/`.

[3] Knime. *Learn to Deal with Imbalanced Dataset Classification.* URL: `https://www.knime.com/blog/correcting-predicted-class-probabilities-in-imbalanced-datasets`.

[4] Wikipedia. *Multiclass Classification.* URL: `https://en.wikipedia.org/wiki/Multiclass_classification`.

[5] Wikipedia. *Oversampling and undersampling in data analysis.* URL: `https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis`.