Compare Machine Learning Algorithms With Astronomical Dataset

Puiwan Wang
Fettes College, Edinburgh EH4 1QX, UK

Abstract: Stellar classification is the most important task in astronomy, with the goal of classifying all celestial objects based on their spectral properties to infer their physical characteristics. This paper aims to classify stars, galaxies, and quasars using some of the most popular machine learning algorithms, namely Logistic Regression, SVM, Random Forest, and Naïve Bayes, on the Sloan Digital Sky Survey Data Release 17 dataset. This followed a strict methodology that included various preprocessing steps: cleaning, normalization, handling outliers, and feature engineering to improve model performance.
Each of the algorithms, during training, optimized its internal parameters to minimize any prediction errors and maximize reliability. Data were split into a training-test set; feature normalization was used for model stability. The metrics used for training algorithms included precision, recall, F1-score, and confusion matrices that provided a specific comparison of strengths and weaknesses for the algorithms. Random Forest emerged as the most effective classifier, achieving 98% accuracy due to its ability to handle complex patterns, while Logistic Regression and SVM delivered moderate performance with accuracies of 84% and 85%, respectively. Naïve Bayes, though computationally efficient, struggled with the dataset's complexity, achieving only 67% accuracy.
This state-of-the-art research emphasizes how important the selection of algorithms is, given dataset characteristics, and classification objectives. This study also points to feature engineering and comprehensive evaluation for enhancing predictive reliability. By showcasing an efficient usage of machine learning for stellar classification, this work contributes to the development of automated analysis in astronomy and opens perspectives toward further improvements by enhanced feature selection, ensemble techniques, and larger datasets.

Keywords: Classification Stellar Classification, Machine Learning models, Feature Engineering, Predictive Accuracy, Sloan Digital Sky Survey (SDSS)

1 Introduction

1.1 Context

In astronomy, stellar classification[1] plays a vital role in deciphering the nature of celestial objects by categorizing them based on spectral characteristics, such as light absorption and emission patterns. This classification spans stars, galaxies, and quasars, with distinctions grounded in their spectral signatures, which reveal underlying physical properties like temperature, chemical composition, and density. Historically, the early cataloguing of stars[2] within our galaxy fostered a deeper understanding of our place in the cosmos. Pioneering discoveries—such as the realization that the Andromeda Galaxy lay beyond the Milky Way—have expanded the boundaries of classification. With advancements in telescopes[3], astronomers now classify not only nearby stars but also a vast array of galaxies and quasars, enabling broader insights into the structural and evolutionary dynamics of the universe.

1.2 Scope of Research

For this study, I've chosen to implement a comparative analysis of four machine learning algorithms:
Logistic Regression models the probability that an object belongs to a particular class by fitting a logistic function to the data, making it suitable for binary or multiclass classification. Its straightforward approach makes it computationally efficient, although it assumes a linear decision boundary, which may limit its performance on non-linear data.
Support Vector Machine (SVM) seeks an optimal hyperplane to separate classes in a high-

dimensional space, leveraging the kernel trick for complex boundaries. This method is especially useful for cases where distinct class boundaries exist but is computationally intensive when applied to large datasets, which will be an important factor to evaluate in this study.

Random Forest builds an ensemble of decision trees, using a voting mechanism to aggregate their predictions. By randomly selecting features at each split, it reduces overfitting, capturing complex, non-linear patterns within the data. However, it may be less interpretable than other models, which can pose a challenge in understanding specific decision criteria.

Naïve Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence between features. While this assumption rarely holds true in complex datasets, Naïve Bayes can still perform surprisingly well, especially in cases with distinct class distributions. Its simplicity also makes it computationally efficient, providing a useful benchmark against more complex algorithms.

1.3 Methodology

1.3.1 Programming Language

Python is the leading choice in data science and machine learning because it has extensive, free libraries such as scikit-learn, Pandas, and NumPy, which ease many data processing and algorithm implementation tasks.

1.3.2 Dataset

The dataset I've selected, Stellar Classification Dataset – SDSS17, is sourced from Kaggle and originates from the Sloan Digital Sky Survey Data Release 17 (DR17)[4]. With a usability rating of 10/10, this dataset is ideal for comparative analysis, providing well-organized and accessible data that facilitates a range of classification tasks. The SDSS data is released under the public domain.

1.3.3 Data Processing

The methodology that was used throughout this study involved several key steps in data preparation, processing, and feature engineering that can effectively increase the quality of the dataset and, therefore, the model's predictive capability.

At the data preprocessing stage, the dataset was duly explored to comprehend its structure, character, and any potential aberrations that could distort results. Care was taken with the missing values; most large datasets usually have some, so that they do not introduce bias. Instead of an imputation strategy that drops the missing entries, filling gaps with the most representative values, such as the mean or median in a manner that maintains coherence, the data set was treated. Duplicate entries were removed to make each record unique, an important issue when each object in the dataset needed to be identified uniquely. This process also included preliminary feature range checks to ensure that data points fell within known physical limits-a very important basis for ensuring the integrity of the data in astronomy.

After cleaning, attention turned to the preprocessing tasks that would align and get the dataset ready for modelling. Normalization of the numeric features was performed using Min-Max Scaling. It let the model compare features on the same scale, thus not biasing it toward those features that had larger ranges. Another important step involved in the pipeline of processing was outlier handling, in which extreme values, by capping and interpolation, diminished their strong influence while maintaining the underlying trend of the data. Checks on the correlation of the data informed the preparation by indicating features that would provide strong predictive signals without redundant overlaps.

Feature engineering followed next as a means to enrich and define the dataset in great detail, allowing the model to capture the important patterns more substantively. It ranged from the creation of new features, including color indices such as u-g, g-r that further elaborate on the properties of each object, to spatial clustering, representative of the distribution and relational structure of the objects in the sky.

The positional attributes are binned, like the celestial coordinates, in cluster analyses. As an example, alpha and delta in this dataset are in sections representative of the arc of the sky, so the model can find spatial groupings and trends more effectively. Eventually, the dataset was divided into training, verification, and testing, such that there is no mutual dependence of subsets and enough size to properly allow their evaluation. This could be something like 80-10-10 or 70-15-15 divisions, such that it generalizes well but is not overfitted. At any rate, these latter steps rehashed the raw dataset into an extremely refined and structured format ready for machine learning. Thus, this prepared the model for the acquisition of meaningful insights into the phenomena of the heavens. Each stage in this

methodology has thus contributed to the preparedness of this dataset, in tune with an iterative systematics, to come up with quality data and hence accuracy and reliability in view of model and study goals.

1.4 Research Aim

Through this comparison, I aim to assess each algorithm's suitability for stellar classification based on spectral characteristics, specifically examining their accuracy, computational efficiency, and ability to capture nuanced patterns within the data. By analysing these models, I hope to gain insights into the predictive power of different algorithmic approaches for astronomical classification, ultimately contributing to the broader field of automated astrophysical analysis.

2. Feature and Dataset

2.1 Dataset Feature Introduction

The dataset consists of 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.
100,000 pieces of data is considered a large dataset, which prevents random error during training, thus reduce error of the model.

2.2 dataset Features

Table 1. List of Dataset Features, Including All Columns of Variables in Dataset

obj_ID	Object Identifier, the unique value that identifies the object in the image catalog used by the CAS
alpha	Right Ascension angle (at J2000 epoch)
delta	Declination angle (at 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 used to identify the specific scan
rereun_ID	Rerun Number to specify how the image was processed
cam_col	Camera column to identify the scanline within the run
field_ID	Field number to identify each field
spec_obj_ID	Unique ID used for optical spectroscopic objects (this means that 2 different observations with the same spec_obj_ID must share the output class)
class	object class (galaxy, star or quasar object)
redshift	redshift value based on the increase in wavelength
plate	plate ID, identifies each plate in SDSS
MJD	Modified Julian Date, used to indicate when a given piece of SDSS data was taken
fiber_ID	fiber ID that identifies the fiber that pointed the light at the focal plane in each observation

3. Training Model and Result Analysis

3.1 Training Process

It is a major step in machine learning where the model learns the pattern[5] within the dataset for the right prediction on unseen data. During this stage, pre-processed and cleaned datasets will be fed as input for several algorithms, each with a diverse approach toward classification. Each of the various algorithms will iteratively optimize its internal parameters[6] with the data to minimize the error and turn out to be a better predictor. Logistic regression gives the probability of each class, while SVM gives the best hyperplane to separate each different class. Random Forest is an ensemble technique which forms a great number of decision trees. Naive Bayes is a probabilistic approach using conditional probabilities. This will mean that the step will benchmark the performance of each algorithm on training data and then cross-check for accuracy against another varied validation set so that we can compare the effectiveness of the models for choosing the best model that suits our classification problem. With this approach, I should obtain a generalized model with real data and minimal errors to support maximum predictive reliability[7].

3.2 Process of Training

The numpy, pandas, and skylearn libraries play a

crucial role during the training process. Much of the training is carried out through importing a major part of the libraries coming from skylearn. The following script prepares and trains four machine learning classifiers: Random Forest, Logistic Regression, Support Vector Machine-SVM, and Naive Bayes to classify the stellar objects within the dataset. It follows the import of the major libraries: numpy for numerical operations, pandas for the manipulation of data, including modules from sklearn as it would ease the workflow of machine learning. It then follows the loading of a dataset from a CSV file into a DataFrame called stellar. It contains several features explaining the stellar objects and a target column labeled as 'class', representing the classification labels.

The feature set X will be formed in the preparation of data, sans the target column, with a target variable y comprised of only values of the 'class' column. The data is split into its respective training and test sets using the train_test_split, where 80% is for training and the remaining 20% for testing, represented as X_train and y_train and X_test and y_test, respectively[8]. A random seed (random_state=42) is set to ensure the split remains consistent across different runs, enhancing reproducibility in model evaluation.

The next step will be feature normalization. Normalization is one of the most critical preprocessing steps to which many algorithms, including Logistic Regression and SVM, are sensitive. This function normalizes the data by scaling it so that each feature has an average of 0 and a standard deviation of 1. This may contribute to the models' stability and/or performance. It fits the scaler on the training data in this section, then commits the same transformation to the test set.

Due to their diversified approaches, a dictionary named classifiers is created that contains a list of four classification algorithms: Random Forest is an ensemble-based method that builds lots of decision trees and then combines the output by taking a vote on each; n_estimators=100 is the optimum number of trees which balances model performance and computational power. Logistic Regression is representative of the linear model, which calculates the probabilities of each class; hence, it is useful in binary and multi-class classification. The SVM algorithm tries to find such an optimum hyperplane that maximizes the margin between classes. In this implementation of the SVM, a linear kernel is specified. Lastly, Gaussian Naive Bayes is a Naive Bayes classifier that assumes independence in features using Bayes' theorem and is computationally so fast on large datasets.

The subsequent steps will fit the method to the training data and predict on the test set using the predict method. To evaluate the performance of each model, the functions to be used are classification_report and confusion_matrix. A classification report encompasses a comprehensive metric including precision, recall, F1-score, and support of each class in the data while giving an insight into the strengths and weaknesses of each model. The confusion matrix explains the count for true positives, true negatives, false positives, and false negatives. Aggregated together, these results will give a very good estimate of the generalization ability of each model to whatever other unseen data there is and hence the best choice of model on which classification can be performed with most confidence. This provides a structured manner in which the performances can be compared and thus offers an informed choice during classification.

3.3 analysis of Result

Precision, recall, F1-score, and confusion matrix are some key parameters that define the performance of machine learning algorithms in classification. Precision is the accuracy of the positive predictions, which quantifies the number of correctly predicted positives out of the total positives predicted, something very useful when the cost associated with false positives is too high. Recall, or sensitivity, refers to the number of actual positives the model captures. This is highly important where a positive case has a severe consequence when missing. The F1-score encompasses both precision and recall in a single score and represents the balance between the two best. This is highly useful in an imbalanced dataset since it considers both false positives and false negatives.

These predictions are given by the confusion matrix as the counts of true positives, false positives, true negatives, and false negatives for the different classes. This enables, further, the finding of very specific areas where the model is going wrong, showing where it finds it difficult to choose between classes. Taken together, this set of metrics overview model performance and

enable deep comparisons of the relative strengths and weaknesses. The following discussion then describes how the performance metrics will go about calculating the classifiers: random forest, logistic regression, SVM, and Naive Bayes to classify the stellar objects.

3.3.1 Analysis of random forest classifier

Classifier: Random Forest

	precision	recall	f1-score	support
0	0.98	0.99	0.98	11860
1	0.98	0.99	0.99	4343
2	0.97	0.93	0.95	3797
accuracy			0.98	20000
macro avg	0.98	0.97	0.97	20000
weighted avg	0.98	0.98	0.98	20000

```
[[11689    60   111]
 [   36  4306     1]
 [  249    18  3530]]
```

Figure 1. Result of Random Forest Classifier

According to Figure 1 (result of random forest classifier), The Random Forest classifier showed the best performance: 98% accuracy and most generalized. For all classes, it showed very high overall precision: 0.98 for classes 0 and 1, and 0.97 for class 2; and recalls of 0.99 for classes 0 and 1 and 0.93 for class 2. Its F1-scores lay between 0.95 and 0.98, showing balanced performance. The confusion matrix presented a few misclassifications-meaning this model was quite robust in terms of complicated data.

3.3.2 Analysis of Logistic Regression Classifier

Classifier: Logistic Regression

	precision	recall	f1-score	support
0	0.83	0.94	0.89	11860
1	0.79	0.55	0.64	4343
2	0.91	0.85	0.88	3797
accuracy			0.84	20000
macro avg	0.84	0.78	0.80	20000
weighted avg	0.84	0.84	0.83	20000

```
[[11206   393   261]
 [ 1932  2367    44]
 [  321   248  3228]]
```

Figure 2. Result of Logistic Regression Classifier

According to Figure 2 (result of logistic regression classifier), Logistic Regression did pretty mid-stream, with an accuracy of 84%. Precision for classes 0, 1, and 2 was 0.83, 0.79, and 0.91, respectively. Recall was significantly imbalanced between the classes, whereas class 1 had the worst recall, performing at only 0.55, class 0 had a recall of 0.94, with class 2 posting 0.85. F1 scores came in at 0.89, 0.64, and 0.88 for classes 0, 1, and 2, respectively. The confusion matrix illustrates that the number of false negatives is rather high in class 1, reflecting the difficulty of distinguishing it from other classes.

3.3.3 Analysis of Support Vector Machine(SVM) Classifier

Classifier: SVM

	precision	recall	f1-score	support
0	0.83	0.97	0.90	11860
1	0.85	0.53	0.66	4343
2	0.95	0.85	0.90	3797
accuracy			0.85	20000
macro avg	0.88	0.79	0.82	20000
weighted avg	0.86	0.85	0.84	20000

```
[[11517   187   156]
 [ 2012  2321    10]
 [  343   227  3227]]
```

Figure 3. Result of Support Vector Machine Training Classifier

According to Figure 3 (result of SVM classifier), Accuracy for the SVM classifier stood at 85%, a bit better than that achieved from Logistic Regression. Precision values are 0.83 for class 0, 0.85 for class 1, and 0.95 for class 2. For recall, class 0 is extremely high at 0.97 while class 1 is pretty low at 0.53, with class 2 at 0.85. The F1-scores are 0.90, 0.66, and 0.90. Just like in Logistic Regression, class 1 suffers from misclassifications, which indicates some difficulties in model training for that class.

3.3.4 Analysis of Naïve Bayes Classifier

Classifier: Naive Bayes

	precision	recall	f1-score	support
0	0.85	0.66	0.74	11860
1	0.42	0.53	0.47	4343
2	0.63	0.88	0.73	3797
accuracy			0.67	20000
macro avg	0.63	0.69	0.65	20000
weighted avg	0.71	0.67	0.68	20000

```
[[7809 2920 1131]
 [1183 2296  864]
 [ 182  282 3333]]
```

Figure 4. Result of Naïve Bayes Classifier

According to Figure 4 (result of Naïve Bayes classifier), The worst performance came from Naive Bayes, which scored an accuracy of 67%. Precision values were quite low, with the high of 0.42 being in class 1, whereas class 0 was 0.85 and class 2 was 0.63. Recall metrics indicated an inability to recognize TPs, especially in classes 0 and 1 at 0.66 and 0.53, correspondingly, while class 2 had a relatively better 0.88. The F1 scores-0.74 for class 0, 0.47 for class 1, and 0.73 for class 2-painted a picture of imbalance in performance. The confusion matrix indicated considerable misclassifications, hence showing Naive Bayes was struggling with the complexity of the dataset.

4. Evaluation

The methodology followed in this work provides a sure-shot path toward systematically exploring which machine learning algorithms work better in classifying stellar objects. Strong points of the proposed method are related to data preprocessing, model selection, and a rigorous evaluation framework. Cleaning, standardization, and splitting into training and test sets are highly important initial steps that any algorithm can take to get off to the very best possible start with the goal of learning from data. Among the included algorithms were Random Forest, Logistic Regression, SVM, and Naive Bayes-all contending in finding a full-scale approach toward which methods best fit the characteristics of the dataset. Besides, metrics such as precision, recall, F1-score, and the confusion matrix have made it possible to compare models in detail, thus drawing informed conclusions.

Of course, there are a few areas where this methodology might be further improved. The dataset was normalized, but additional feature engineering would probably make models perform even better. These are advanced feature selection or dimensionality reduction techniques, such as PCA[9], which are effective in determining whether better feature representations may yield improved classification tasks, especially if employing some models that don't go so well, like Naive Bayes.

This approach was also based on one split for training and testing, which is very biased within the performance evaluation. Techniques of cross-validation, such as k-fold[10], would make much more strength in the test of every model over several splits of the data. This will reduce the probability of having erroneous results because of a poor split in the division between training and testing. Therefore, the findings become more reliable.

Another possible weakness may include making use of the same set of measures in evaluating different models. While the precision, recall, F1-score, and accuracy are very important features, other algorithms such as Random Forest, which can be optimized for better ensembles by providing parameters like n_estimators[11] and tree depth that could help to avoid overfitting, were not considered. The extra steps of optimization will fine-tune the performance of each model and make the result more accurate.

In all, the methodology adopted in this study generally succeeds in yielding the results of the stated research goals by giving good insight into how the machine learning algorithms classify stellar objects. However, feature engineering would be further improved with the inclusion of cross-validation and tuning model-specific parameters, which better increases the robustness and reliability of future investigations[12].

5. Conclusion

Stellar object classification studies performance dependence on various machine learning algorithms considering actual realization and basic assumptions made. The Random Forest classifier performed the best, with up to 98% accuracy for unseen data, since RF generalizes that well to unseen data. This underlines the advantage of ensembles, whereby avoidance of overfitting enables many trees to work together in modelling complex relationships of data while keeping the model predictive.

Logistic regression and the support vector machine yielded moderated results with an accuracy of 84% and 85%, respectively. These models yielded suitable results on some classes and, particularly, class 0, where they completely failed for class 1. Such huge differences in class performance may indicate that these models require more feature engineering or modification in training for capturing the fine patterns.

It is also clear that the Naive Bayes classifier performed poorest among those, although computationally efficient, with an overall accuracy of 67%; its reliance on the independence assumption was a limitation since interdependencies within the features of the dataset undermined its precision and recall, especially for class 1. In fact, this underlines the difficulty in applying simple probabilistic models to such complex, real-world datasets and reinforces the demand for algorithms much more able to consider complex feature interactions.

This hence brings into focus the need to determine an appropriate classification algorithm based on any characteristic of the dataset and the classification objective[13]. The somewhat in-depth look into the metrics of performance through precision, recall, F1-score, and confusion matrix derived an appropriate understanding of the effectiveness of each model for decision-making in effective classification. The results depict clearly that techniques of machine learning can still be explored and

optimized in the classification of stellar objects, as the proper prediction of the proposed classes could really enhance astronomical research and understanding. Future directions may be made toward enhancement in feature selection, the exploration of advanced ensemble techniques, and integration of more data for better optimization of classifier performance across all classes[14].

References

[1]Stellar classification, based on spectral absorption and emission patterns, reveals fundamental properties like temperature and chemical composition. (Source: Carroll, Bradley W., and Dale A. Ostlie. An Introduction to Modern Astrophysics, 2nd ed., Pearson, 2006.)

[2]Historical cataloguing, along with landmark discoveries like Andromeda's distance from the Milky Way, has shaped our understanding of the universe's scale. (Source: Hubble, Edwin P., "A relation between distance and radial velocity among extra-galactic nebulae," Proceedings of the National Academy of Sciences, 1929.)

[3]Advances in telescope technology have allowed astronomers to classify distant celestial objects, providing insights into cosmic structure and evolution." (Source: Tyson, Neil DeGrasse.

[4]www.kaggle.com. (n.d.). Stellar Classification Dataset - SDSS17. [online] Available at: https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17/data.

[5]The concept of model generalization and the ability to predict on unseen data is a foundational aspect of supervised learning. See: Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016, pp. 98-102.

[6]Optimization techniques, like gradient descent, are used to adjust model parameters to minimize prediction error iteratively. For details, refer to: Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016, pp. 199-201.

[7]Ensuring a model is generalized and robust is essential for reliable predictions in real-world applications. Refer to: Domingos, Pedro. "A few useful things to know about machine learning." Communications of the ACM, vol. 55, no. 10, 2012, pp. 78-87.

[8]Proper dataset splitting is essential for model evaluation and preventing overfitting. See: Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." International Joint Conference on Artificial Intelligence, 1995, pp. 1137-1145.

[9]Feature engineering and dimensionality reduction methods like PCA (Principal Component Analysis) can enhance model performance, particularly for high-dimensional datasets. For a detailed look at PCA and feature selection, refer to: Jolliffe, I. T. Principal Component Analysis. Springer Series in Statistics, 2002.

[10]Cross-validation, specifically k-fold, is widely used to mitigate the variance due to specific data splits, enhancing the robustness and generalization of model evaluations. See: Kohavi, Ron. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection." IJCAI, 1995, pp. 1137-1143.

[11]Tuning parameters such as n_estimators and tree depth in Random Forests helps to prevent overfitting and improves model robustness. For further reading, see: Breiman, Leo. "Random forests." Machine Learning, vol. 45, no. 1, 2001, pp. 5-32.

[12]Incorporating cross-validation and hyperparameter tuning into feature engineering can significantly strengthen the reliability of machine learning models. See: Murphy, Kevin P. Machine Learning: A Probabilistic Perspective. MIT Press, 2012, pp. 250-255.

[13]Choosing the appropriate classification algorithm is crucial for achieving optimal performance based on the dataset characteristics. For a comprehensive review of classification algorithms, see: Hodge, V. J., and J. H. Austin. "A survey of outlier detection methodologies." Artificial Intelligence Review, vol. 22, no. 2, 2004, pp. 85-126.

[14]Improving feature selection can significantly enhance model performance, especially in complex datasets. See: Guyon, Isabelle, and André Elisseeff. "An introduction to variable and feature selection." Journal of Machine Learning Research, vol. 3, 2003, pp. 1157-1182.

