
MEMO-F524
MASTER THESIS

AUTOMATED CLASSIFICATION OF STARS AND SYSTEMS USING
MACHINE LEARNING

Author :

TALHAOUI Yassin

Section :

COMPUTER SCIENCE

Promotor :

DEFRANCE MATTHIEU

April 21, 2025

Contents

| | | |
|-----------|---|-----------|
| 1 | Introduction | 5 |
| 2 | Integrating machine learning techniques with traditional astrophysical methods | 5 |
| 3 | Using spectral data | 7 |
| 3.1 | Introduction to spectra | 7 |
| 3.2 | The benefits of stellar spectra | 9 |
| 4 | Previous studies on stellar spectral analysis | 10 |
| 4.1 | Analysis of chemical composition | 10 |
| 4.2 | Temperature estimation | 10 |
| 4.3 | Brightness prediction | 10 |
| 4.4 | Methodologies and algorithms | 10 |
| 5 | The Payne: Self-consistent ab initio Fitting of Stellar Spectra | 11 |
| 5.1 | Main points and results | 11 |
| 5.2 | Methodologies and algorithms | 11 |
| 5.3 | Implications and contributions | 11 |
| 5.4 | Implementation | 12 |
| 5.5 | Tutorial | 14 |
| 6 | Challenges and limitations | 15 |
| 7 | Recent advances and innovations | 17 |
| 8 | Future directions and emerging trends | 18 |
| 9 | Implementation: Operating mode | 20 |
| 10 | Stellar dataset to predict star types | 20 |
| 10.1 | About the dataset | 20 |
| 10.2 | Data collection and preparation | 21 |
| 10.3 | Importance of the study | 21 |
| 10.4 | Exploration and analysis of the dataset | 22 |
| 10.4.1 | 1. Dataset Overview | 22 |
| 10.4.2 | 2. Checking for missing values | 23 |
| 10.4.3 | 3. Statistical summary | 23 |

| | | |
|---------|--|----|
| 10.4.4 | 4. Class distribution (balance check) | 24 |
| 10.4.5 | 5. Variables corrélation Analysis | 25 |
| 10.5 | Correlation Matrix | 25 |
| 10.5.1 | Strong correlation with star type: | 25 |
| 10.5.2 | Variables Dépendancies : | 25 |
| 10.5.3 | Color and spectral class of stars: | 25 |
| 10.5.4 | The spectral class shows weak correlations : | 26 |
| 10.5.5 | Implications : | 27 |
| 10.6 | 6. Visualization of feature distribution | 27 |
| 10.6.1 | Distribution of temperatures : | 27 |
| 10.6.2 | Asymmetric distribution | 27 |
| 10.6.3 | Most stars are cold | 28 |
| 10.6.4 | Hot stars are less common | 28 |
| 10.6.5 | Density curve | 28 |
| 10.6.6 | Interpretation of the brightness graph as a function of star type | 29 |
| 10.6.7 | Wide range of supergiants and hypergiants | 29 |
| 10.6.8 | High-luminosity aberrant stars | 30 |
| 10.6.9 | A clear classification based on brightness | 30 |
| 10.6.10 | Conclusion | 30 |
| 10.7 | Data cleansing | 31 |
| 10.7.1 | Rename columns | 31 |
| 10.8 | Outlier Treatment | 32 |
| 10.8.1 | Boxplot Interpretation (Outlier Analysis) | 32 |
| 10.8.2 | Brightness exhibits extreme outliers | 32 |
| 10.8.3 | Temperature, radius, and absolute magnitude have few or no extreme outliers | 32 |
| 10.8.4 | Brightness outliers should be analyzed carefully | 32 |
| 10.9 | How to manage these outliers instead of deleting them? | 33 |
| 10.9.1 | Using the interquartile range (IQR) method to filter out extreme outliers : | 33 |
| 10.9.2 | We log-transform the brightness to reduce skewness and scale all features using RobustScaler (outlier-resistant) : | 33 |
| 10.10 | Train models with/without outliers and compare performance : | 33 |
| 10.10.1 | Perfect Classification | 35 |
| 10.10.2 | Outliers have minimal impact | 36 |
| 10.10.3 | Possible Overfitting ? | 36 |

| | | |
|-----------|--|-----------|
| 10.11 | Logistic Regression : | 36 |
| 10.11.1 | Results of Logistic Regression : | 37 |
| 10.11.2 | Why a régression logistique ? | 38 |
| 10.11.3 | Interpretation of the Logistic Regression results | 38 |
| 10.12 | Comparison between models (with and without outliers) | 38 |
| 10.13 | Confusion Matrix | 40 |
| 10.13.1 | Interprétation of the Confusion Matrix | 40 |
| 10.13.2 | Model performance analysis | 41 |
| 10.13.3 | Possible reasons for classification errors | 41 |
| 10.13.4 | Recommendations for improvement | 41 |
| 10.14 | visualize the importance of features | 43 |
| 10.15 | More advanced analysis | 46 |
| 10.15.1 | Temperature distribution : | 46 |
| 10.15.2 | Luminosity distribution: | 46 |
| 10.15.3 | Radius distribution : | 47 |
| 10.15.4 | Absolute magnitude distribution : | 47 |
| 10.16 | Statistical data before and after removal of outliers | 49 |
| 10.17 | Binary classification: White Dwarf vs. Main Sequence | 50 |
| 10.17.1 | Key takeaways : | 56 |
| 10.18 | Interpreting the learning curve (see figure 14) : | 57 |
| 10.19 | Final thoughts | 57 |
| 11 | Stellar classification dataset - SDSS17 | 57 |
| 11.1 | Dataset Overview | 57 |
| 12 | Exploratory Data Analysis (EDA) | 59 |
| 12.1 | Data analysis | 59 |
| 12.2 | Visualization of class distribution | 59 |
| 12.3 | Distribution of the main features | 60 |
| 12.4 | Density distributions of the main features | 61 |
| 12.5 | Correlation heatmap of the main features | 62 |
| 12.6 | Interpretation of the redshift distribution by class (analysis of features relative to the target) | 64 |
| 13 | Feature Engineering | 66 |
| 14 | Model training | 66 |

| | |
|--|-----------|
| 14.1 Random Forest | 66 |
| 14.2 XGBoost | 69 |
| 14.3 Logistic Regression | 71 |
| 14.4 Neural Network (Multi-Layer Perceptron - MLP) | 71 |
| 15 Comparison of models | 74 |
| 16 Final recommendation | 74 |
| 17 Conclusion | 75 |

1 Introduction

The aim of this Master’s thesis is to understand and investigate how current machine learning techniques can help us study the stellar properties of single and binary stars. Initially, astrophysics was a very data-poor scientific field, but over time numerous space missions and large astronomical studies have enabled us to collect massive amounts of data from hundreds of millions of astronomical sources. In the GAIA era, the volume of data is set to increase even further, into the petabyte range. Astrophysics has thus become a data-intensive science. All this stellar data has led to the emergence of new algorithmic, computational and statistical challenges. It is in this context that machine learning techniques, used in various fields of scientific study, could prove invaluable in extracting information from observations. By taking advantage of machine learning algorithms, astrophysicists can tackle a wide range of research questions more effectively, enabling us to better understand the cosmos and make new discoveries.

In this work, I focused on applying machine learning models to two astronomical datasets: *sdss17* and *star-dataset*. The first dataset consists of a set of stellar spectral data, used to classify astronomical objects into different categories such as stars, galaxies and quasars. The second dataset contains information on the star systems of a 6-class stellar dataset for the classification of stars, whose key properties we have sought to identify and whose evolutionary stages we have sought to predict using advanced classification techniques. We used various machine learning models, including logistic regression, random forest, gradient boosting (XGBoost) and neural networks, to evaluate their performance in these tasks. Through systematic hyperparameter tuning and model evaluation, we identified the most effective approaches for each dataset, demonstrating the power of machine learning in modern astrophysics.

The results not only highlight the strengths and limitations of various algorithms in astronomical data processing, but also offer insight into how automated classification techniques can contribute to large-scale stellar population studies. By integrating data-driven methods into astrophysical research, we are paving the way for more efficient and accurate analyses of the vast datasets produced by current and future space missions.

2 Integrating machine learning techniques with traditional astrophysical methods

To explore the integration of machine learning techniques with traditional astrophysical methods, we will embark on a challenge where the combination of domain knowledge and data-driven approaches will illuminate the cosmos with great clarity. This collaborative effort harnesses the expertise of astronomers and data scientists, bringing together the knowledge of researchers and computational insights to explore stellar

properties. What follows is a presentation of the methodologies used in several studies in this field.

Understanding traditional astrophysical methods

We'll start by exploring traditional astrophysical methods, including spectroscopy, photometry and stellar modeling. Without going too far into the subtleties of stellar classification systems, evolutionary trajectories and chemical abundance analyses, in order to establish a solid knowledge base in the field.

Identify Challenges and Opportunities

By examining the field of astrophysical research, researchers identify areas where traditional methods run into limitations or inefficiencies. Identifying opportunities for improvement such as machine learning techniques can give us something in terms of automating labor-intensive tasks or discovering hidden patterns in the data, to guide our study towards improving the accuracy of predictions and gaining new insights from the data we hold.

Engage in interdisciplinary collaboration

Collaboration between astronomers and data scientists creates an environment where domain experts and practitioners in the field of machine learning converge to exchange ideas, methodologies and points of view. This interdisciplinary dialogue facilitates knowledge sharing and bridges the gap between theoretical astrophysics and computational data analysis.

Data acquisition and processing

Among the most important tasks is the management of diverse and representative data sets comprising stellar spectra, photometric measurements and ancillary information, including stellar classifications, ages and metallicities. Rigorous pre-processing techniques deal with data quality issues such as noise, calibration errors and missing values, guaranteeing the integrity of the analyses.

Feature Engineering and feature selection

Collaboration with astrophysicists enables us to identify relevant astrophysical features that include important stellar properties. Drawing on our domain knowledge, we design informative features that capture spectral line intensities, continuum shapes and other key characteristics. Dimensionality reduction techniques distill high-dimensional spectral data into interpretable feature spaces, improving computational efficiency and interpretability.

Model development and validation

Co-designing machine learning models tailored to specific astrophysical questions or tasks, such as stellar parameter estimation or classification, takes our exploration a step further. Adopting a variety of algorithms, including traditional regression and classification methods, as well as sophisticated deep learning architectures such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), enables us to extract information from complex data.

Interpretability and transparency

With an emphasis on model interpretability and transparency, researchers are collaborating with astronomers to develop techniques for a posteriori interpretability. Feature significance analysis, attention mechanisms and diagnostic explanations of patterns highlight the underlying logic of pattern predictions, facilitating confidence and understanding.

Refinement and iterative evaluation

By adopting an iterative approach to model refinement and evaluation, we solicit feedback from domain experts at every stage of the development process. By continually validating model performance against field observations, we refine algorithms and methodologies based on empirical observations and domain-specific considerations, ensuring robust and reliable analyses.

Added value

Thanks to this combination of traditional astrophysical and machine learning techniques, new perspectives are opening up on the complex field of stellar properties, paving the way for future advances in our understanding of the cosmos.

3 Using spectral data

To investigate the machine learning techniques that can help us, we will consider large samples of stellar spectra from surveys such as GALAH, Gaia-ESO survey.

3.1 Introduction to spectra

The importance of spectra

Our understanding of the physical properties of stars depends largely on the analysis of their spectra. By examining absorption lines, we can determine the mass, temperature and composition of stars. The shape

of the lines provides information about atmospheric processes.

Composition

Stellar spectra consist of a continuous spectrum on which narrow spectral lines are superimposed, mainly dark absorption lines, but sometimes also bright emission lines.

Continuous spectra

The continuous spectrum comes from the star's hot surface. The atmosphere absorbs specific wavelengths, creating dark zones in the spectrum, indicating different chemical compositions.

Classification

Stellar spectra are classified according to the intensity of these spectral lines. This classification system was initiated by Isaac Newton, then perfected by Joseph Fraunhofer and others.

Measurement methods

Stellar spectra are generally obtained using objective prisms or slit spectrographs. These methods enable detailed analysis of individual spectral lines.

Analysis

Spectra are converted into intensity plots, revealing flux density as a function of wavelength. The shape of the spectral lines provides valuable information on stellar atmospheres, while the intensity of the lines can be used to determine chemical compositions.

Harvard spectral classification

Developed at Harvard Observatory, this classification system ranks stars according to their spectral characteristics, mainly temperature. It includes letters for spectral types and numbers for subclasses.

Yerkes spectral classification

A more precise classification system introduced by the Yerkes Observatory takes into account both temperature and luminosity. It classifies stars into six luminosity categories, providing a better understanding of their properties.

Particular spectra

Some stars exhibit particular spectra due to factors such as strong stellar winds, rotation or binary interactions. Examples include Wolf-Rayet stars, Be stars and shell stars.

3.2 The benefits of stellar spectra

In addition, the analysis of stellar spectra enables us to understand the nature and evolution of celestial bodies like stars. They provide us with information such as :

- **Temperature:** The surface temperature of a star and that of its outer envelope can be deduced from the color of the light it emits. Indeed, a hotter star will appear bluer, as the higher temperature favors the emission of light at shorter wavelengths, in accordance with the law of thermal radiation. By analyzing its spectrum, it is possible to estimate an “effective” temperature for the star, taking into account the transfer of radiation through the different layers of its stellar atmosphere.
- **Chemical composition:** The particular frequencies of spectral lines provide distinct information on which elements absorb or emit photons. Spectroscopic databases have been developed from the study of laboratory-produced spectra, making it easier to identify the origin of observed absorption or emission lines in astronomical spectra. By analyzing the relative intensity of the characteristic lines of the elements detected, and based on theoretical models, it is possible to infer the chemical composition of each star’s atmosphere.
- **The velocity:** $\Delta\lambda$, the shift of observed spectral lines, is commonly used to measure velocities. It is used to calculate the radial velocity \mathbf{v} of a celestial object, which corresponds to its velocity component along the line of sight. This velocity is expressed as

$$v = c \frac{\Delta\lambda}{\lambda}.$$

In short, the analysis of stellar spectra is a powerful tool for probing the secrets of stars. By revealing their temperature, chemical composition and velocity, these spectra offer us a fascinating window into the nature and evolution of celestial stars. They are thus an essential pillar of astrophysical research, enabling us to better understand the mysteries of the universe.

4 Previous studies on stellar spectral analysis

4.1 Analysis of chemical composition

The researchers used machine learning techniques to analyze stellar spectra and infer their chemical composition. By training models on spectral data with known chemical abundances, the algorithms can predict the elemental composition of stars based on their spectra. Feature extraction methods such as **principal component analysis (PCA)** were used to reduce the dimensionality of the spectral data while retaining relevant information. Machine learning algorithms such as **support vector machines (SVM)** or **random forests** were employed to classify stars into different chemical abundance classes based on their spectra.

4.2 Temperature estimation

Machine learning techniques were used to estimate the effective temperature of stars from their spectral characteristics. Researchers used regression techniques such as linear regression or neural networks to predict the temperature of stars from their spectral characteristics. Feature engineering methods, including wavelength selection or continuum normalization, were applied to improve the predictive performance of temperature estimation models.

4.3 Brightness prediction

Machine learning algorithms have been used to predict stellar brightness, which is a measure of their intrinsic luminosity. Luminosity estimation is essential for understanding the properties of stars and their evolutionary stages. Regression algorithms such as Gaussian processes or deep learning models such as convolutional neural networks (CNNs) have been used to predict stellar brightness from spectral data. Feature extraction techniques, such as line intensity indices or flux ratios, have been used to capture relevant information from stellar spectra for brightness prediction.

4.4 Methodologies and algorithms

Feature extraction : Methods such as PCA have been used to extract relevant features from stellar spectra while reducing dimensionality. **Classification** : Algorithms such as SVM, Random Forests or k-nearest neighbors were used to classify stars into different categories based on their spectral characteristics. **Regression** : Techniques such as linear regression, Gaussian processes or neural networks have been used to predict continuous stellar parameters such as temperature or luminosity from spectral data.

5 The Payne: Self-consistent ab initio Fitting of Stellar Spectra

The Payne is a significant contribution to the field of stellar astrophysics, particularly in the area of « spectral » analysis using machine learning techniques. The study presents a new approach to fitting stellar spectra, known as "The Payne," which combines the principles of physical modeling with machine learning algorithms to achieve self-consistent and accurate spectral fitting.

5.1 Main points and results

The Payne uses a framework that incorporates fundamental physical principles, such as stellar atmosphere models and atomic physics, into the spectral interpolation process. Unlike traditional empirical methods, The Payne performs a self-consistent ab initio fitting of stellar spectra, meaning it derives physical parameters directly from observational data without relying on pre-existing models. By using advanced machine learning algorithms, such as artificial neural networks, The Payne is able to efficiently and accurately model the complex relationships between stellar parameters and spectral characteristics. The study demonstrates the effectiveness of The Payne in interpolating stellar spectra across a wide range of stellar types and evolutionary stages. The self-consistent nature of The Payne allows robust determination of key stellar parameters, including effective temperature, surface gravity, metallicity, and elemental abundances, directly from observed spectra.

5.2 Methodologies and algorithms

Payne uses artificial neural networks as a machine learning algorithm for spectral fitting. These neural networks are trained on a large dataset of synthetic spectra generated from stellar atmosphere models. Feature extraction techniques, such as wavelength binning or continuum normalization, can be used to preprocess the spectral data before feeding it into the neural network. The neural network architecture is designed to capture the complex nonlinear relationships between input spectral features and output stellar parameters. During the training process, the neural network learns to match the observed spectra to the corresponding stellar parameters, resulting in high accuracy and precision in spectral fitting.

5.3 Implications and contributions

The Payne represents a significant advancement in spectral analysis techniques, providing a more robust and physically motivated approach to fitting stellar spectra compared to traditional methods. By combining the principles of physical modeling with machine learning algorithms, The Payne allows astronomers to derive accurate and self-consistent stellar parameters directly from observational data. The self-adaptive nature of

The Payne makes it particularly suitable for analyzing large-scale spectroscopic surveys, for which automated and efficient methods of spectral analysis are essential. The study opens new avenues for the study of stellar populations, chemical abundances, and stellar evolution by providing a powerful tool for analyzing stellar spectra with unprecedented accuracy and precision.

5.4 Implementation

The Payne is a stellar spectra interpolation framework developed by Ting-Yuan Sen. It aims to perform self-consistent interpolation of stellar spectra using a combination of physical modeling and machine learning techniques. The source code is available on a public repository on GitHub at https://github.com/tingyuansen/The_Payne.

The Payne is intended to be a sophisticated neural network-based tool designed to analyze stellar spectra and infer stellar parameters such as effective temperature, surface gravity, and element abundance. Implementing The Payne involves several components, each serving a specific purpose in the overall workflow. Here is a detailed description of each component and how they collectively contribute to the construction and use of The Payne.

Neural network architecture

The neural network architecture is the core of "The Payne" and is responsible for learning the mapping between stellar spectra and stellar parameters. It comprises multiple layers, including input, hidden, and output layers, with various activation functions and regularization techniques to facilitate learning and prevent overfitting. "The Payne" uses a deep learning neural network architecture, often with multiple hidden layers, to capture the complex relationships inherent in stellar spectra.

Training data

The training data consists of a large collection of stellar spectra, each associated with the corresponding stellar parameters obtained from reference sources such as spectroscopic studies or theoretical models. These spectra are typically preprocessed to remove noise, normalize intensities, and handle missing values before being fed into the neural network for training. The quality and diversity of the training data have a significant impact on the performance and generalization ability of "The Payne."

Training phase

The training phase involves iteratively feeding sets of preprocessed spectra into the neural network and adjusting its weights and biases to minimize the discrepancy between predicted and actual stellar parameters. The training phase is typically performed using optimization algorithms such as Rectified Adam (RAdam),

which efficiently update the network parameters based on the gradients of the loss function. Hyperparameters such as the learning rate, dataset size, and number of epochs are tuned to optimize the convergence and performance of the neural network.

Rectified Adam Optimization (RAdam)

RAdam is an advanced optimization algorithm that improves upon the traditional Adam optimizer by rectifying its adaptive learning rate. It addresses the problem of poor convergence and overshoot in the early stages of the training phase by dynamically adjusting the learning rate based on the variance of past gradients. RAdam's implementation in "The Payne" ensures stable and efficient optimization of neural network parameters, leading to faster convergence and improved model performance.

Spectral model

The spectral model component encapsulates the mathematical formulation and physical principles underlying the relationship between stellar parameters and spectral characteristics. It provides functions for generating synthetic spectra based on given stellar parameters and wavelength grids, enabling the synthesis of spectra across a wide range of stellar atmospheres and compositions. The spectral model serves as the ground truth against which the Payne predictions are validated and calibrated.

Parameter inference

Once the neural network is trained and validated, it can be used to infer stellar parameters from the observed spectra. Given a new spectrum, "The Payne" uses the trained neural network to predict corresponding stellar parameters, such as effective temperature, surface gravity, and chemical abundances. These inferred parameters can then be compared to reference values or used for further analysis, such as stellar population studies or exoplanet characterization.

Post-processing and uncertainty estimation

Post-processing steps may involve refining the inferred stellar parameters, performing quality checks, and estimating uncertainties associated with the predictions. Uncertainty estimation is crucial for quantifying the reliability of parameter inference and assessing the robustness of the neural network model. Techniques such as bootstrap resampling or Bayesian inference can be used to characterize the uncertainty of the predicted stellar parameters.

Integration with spectral analysis pipelines

“The Payne” can be easily integrated into existing spectral analysis pipelines used by astronomers and astrophysicists to study stellar populations, galactic dynamics, and exoplanet characterization. It provides a powerful tool to automate and accelerate the analysis of large-scale spectral datasets, allowing researchers to extract valuable information about the properties and evolution of stars and galaxies.

Contribution to research

In summary, "The Payne" represents a cutting-edge approach to stellar parameter inference using neural networks and advanced optimization techniques. Its modular design, combined with a robust training process and spectral model, allows astronomers and researchers to unlock the full potential of stellar spectra to understand the complexities and mysteries of the universe.

5.5 Tutorial

A Jupyter notebook written in Python serves as a comprehensive guide to understanding and using the features of “The Payne” code to fit stellar spectra. Here’s a detailed breakdown of the information provided in the notebook:

Introduction and overview

The notebook begins by introducing “The Payne” code and its main features, highlighting its role in fitting stellar spectra using a combination of physical modeling and machine learning techniques. It describes the notebook’s main objectives, including generating model spectra, interpolating observed spectra, and training custom neural networks.

Configuration and dependencies

The setup process is also described, including importing the libraries and modules needed to run "The Payne" code. It also defines essential parameters such as the wavelength grating and APOGEE mask, which are crucial for processing and analyzing stellar spectra.

Generation of model spectra

The notebook demonstrates how to generate model spectra for individual stars based on input labels such as effective temperature, surface gravity, and element abundance. It explains the process of scaling the labels and using a neural network to predict the spectrum corresponding to the given parameters.

Spectral Interpolation

Here, the notebook simulates an observed spectrum by adding noise to the generated spectrum. It then shows how "The Payne" code adapts to the noisy spectrum using its fitting algorithms, eventually recovering the input labels from the fitted spectrum.

Downloading and Installing Real Spectra

The notebook provides practical examples by downloading real APOGEE spectra and interpolating them using "The Payne" code. By applying the code to real observational data, it demonstrates its effectiveness and applicability to astrophysical research.

Training custom neural networks

The notebook provides instructions for training custom neural networks using user-defined training data. It explains how to specify parameters such as the number of neurons, learning rate, and ensemble size, and provides visualizations of the training and validation errors to monitor the training process.

Practicals Notes

The notebook concludes with practical tips and considerations for effectively using "The Payne" code. Aspects such as computational efficiency, training parameter optimization, and potential challenges users may encounter when performing spectral fitting and neural network training are discussed.

By following the examples and instructions provided in the Jupyter Notebook, users can gain a comprehensive understanding of "The Payne" code and leverage its stellar spectra fitting and analysis capabilities in their astrophysics research efforts. Additionally, the notebook provides insights into customizing neural networks and optimizing training parameters for specific research objectives and datasets.

6 Challenges and limitations

When applying machine learning techniques to stellar astrophysics, several challenges and limitations must be considered.

Data quality

Stellar spectra data samples may contain artifacts, instrumental effects, or calibration errors that can affect data quality. In addition, variations in data quality between different observational sources or instruments can introduce biases or inconsistencies into the analysis.

Samples size

Obtaining large and diverse data samples of stellar spectra for training machine learning models can be challenging, especially for rare or exotic stellar objects. Limited sample sizes can lead to insufficient coverage of the parameter space, affecting the model's ability to generalize to unseen data.

Noise Reduction

Stellar spectra are often subject to noise from various sources, including photon noise, background noise, and instrumental effects. Developing robust noise reduction techniques that effectively filter out noise while preserving the underlying signal is crucial for accurate spectral analysis.

Interpretation of the models

Machine learning models, especially complex ones such as deep learning models, can lack interpretability, making it difficult to understand how they arrive at their predictions. Interpretable models are essential for understanding the physical processes underlying stellar phenomena and for validating the reliability of model predictions.

Generalisation Performance

Overfitting occurs when a model learns to capture noise or irrelevant patterns in the training data, leading to poor generalization performance on unseen data. Regularization techniques, cross-validation, and model complexity control are essential to mitigate overfitting and ensure model robustness.

Introduction of Bias

This can occur when the training dataset is not representative of the underlying population of interest, leading to biased model predictions. Care must be taken to ensure that the training dataset adequately covers the full diversity of stellar properties and avoids biases introduced by observational or sampling methods.

Generalization to unseen data

Machine learning models trained on one observational dataset may not generalize well to unseen data from different telescopes, instruments, or observing conditions. Transfer learning techniques, which leverage knowledge gained from one dataset to improve performance on another, can help address generalization issues.

Motivation

Addressing these challenges and limitations is essential to successfully applying machine learning techniques to stellar astrophysics. By developing robust methodologies, incorporating domain knowledge, and carefully evaluating model performance, researchers can overcome these obstacles and unlock the full potential of machine learning techniques to advance our understanding of the cosmos.

7 Recent advances and innovations

Recent advances in machine learning methodologies have significantly improved the analysis of stellar spectra, offering innovative approaches for feature engineering, dimensionality reduction, and model optimization. Here are some examples:

Dimensionality reduction

Variational autoencoders (VAEs) and generative adversarial networks (GANs) have been used for unsupervised dimensionality reduction of spectral data. These techniques learn low-dimensional representations of spectra while preserving essential information, facilitating more efficient processing and analysis.

Models Optimization

Bayesian optimization methods, such as Gaussian processes and Bayesian neural networks, have been used for hyperparameter tuning and model optimization. These techniques enable more efficient exploration of the hyperparameter space and better convergence of machine learning models.

Deep learning Architectures

- Convolutional neural networks (CNNs) have been applied to spectral data for tasks such as stellar object classification, spectral feature identification, and stellar parameter estimation. Convolutional neural networks can automatically learn spatial patterns in spectral data, making them well-suited for tasks that involve analyzing spatially structured information.
- Recurrent neural networks (RNNs) have been used to model temporal dependencies in time series of spectral data, enabling prediction of stellar variability and transient events.
- Long Short-Term Memory (LSTM) networks, a type of RNN, have shown promise in capturing long-term dependencies in sequential spectral data, enabling more accurate modeling of stellar dynamics over time.

These recent advances in machine learning methodologies have revolutionized the analysis of stellar spectra, enabling more accurate and efficient processing of observational data. By leveraging deep learning architectures and innovative feature engineering and model optimization techniques, researchers can gain new insights into the complex physical processes occurring in stars and galaxies.

8 Future directions and emerging trends

As the possibilities of machine learning continue to expand and our understanding of stellar astrophysics deepens, future directions in research promise to open new perspectives and advance our knowledge of the cosmos. Emerging trends in both fields offer exciting opportunities for innovation and discovery.

Future prospects

In the future, research in machine learning and stellar astrophysics is expected to explore increasingly complex and interdisciplinary questions. Integrating advanced machine learning techniques with traditional astrophysical methods will enable researchers to tackle the fundamental mysteries of astrophysics with greater precision and efficiency.

Identify emerging trends

- **Multimodal data analysis** : With the advent of multi-wavelength, multi-messenger astronomy, future research will focus on integrating data from diverse sources, such as optical, infrared, radio, and gravitational-wave observations. Machine learning algorithms capable of analyzing multimodal data streams will play a critical role in uncovering synergies and correlations between different wavelengths and cosmic messengers.
- **Transfer learning** : Transfer learning techniques, which leverage knowledge gained in one domain to improve performance in another, will become increasingly common in stellar astrophysics. By transferring learned representations of well-studied stellar populations to underexplored regions of parameter space, transfer learning enables more efficient exploration and characterization of diverse stellar populations.
- **Ensemble methods** : Ensemble learning approaches, which combine predictions from multiple models to improve accuracy and robustness, will be leveraged to address the uncertainties and complexities inherent in astrophysical phenomena. Ensemble methods provide a powerful framework for integrating diverse models, data sources, and observational uncertainties, enabling more reliable predictions and inferences.

Discussion of potential applications

Machine learning offers immense potential to revolutionize future astronomical surveys and missions, by proposing new approaches for data analysis, interpretation and discovery:

- **Automated analysis of observations** : Machine learning algorithms will streamline the analysis of large-scale astronomical observations, enabling the automated detection and characterization of celestial objects, transient events, and astrophysical phenomena. Real-time data processing and event classification will improve our ability to identify rare and elusive cosmic phenomena.
- **Precision cosmology** : Machine learning techniques will facilitate precision cosmological analyses by extracting subtle signals from cosmological datasets, such as maps of the cosmic microwave background, observations of large-scale structures, and gravitational wave observations. Advanced statistical methods and model selection techniques will enable more accurate parameter estimation and hypothesis testing in cosmological models.
- **Characterization of exoplanets** : Machine learning algorithms will advance the field of exoplanet characterization by enabling the detection and classification of exoplanetary systems from stellar spectra and photometric observations. New feature extraction methods and data-driven models will improve our ability to identify exoplanets, characterize their atmospheres, and assess their habitability potential.

In summary, future research directions in machine learning techniques and stellar astrophysics will explore emerging trends such as multimodal data analysis, transfer learning, and ensemble methods, paving the way for transformative advances in our understanding of the universe. By harnessing the power of machine learning techniques, astronomers will unlock new perspectives on the cosmos, unravel its mysteries, and push the boundaries of human knowledge.

9 Implementation: Operating mode

In the following, I will apply all the machine learning techniques acquired during my master's degree in computer science to datasets containing data related to stars and celestial objects. The objective will be to use classification via selected machine learning models in order to classify stars/celestial objects into different categories. To do this, I will follow the following steps:

- Explore the dataset (EDA: check missing values, distributions, correlations)
- Feature selection (correlation matrix, importance ranking)
- Binary classification (start with two classes, e.g., Main Sequence vs. White Dwarf)
- Multi-class classification (gradually moving to six classes). Test multiple classifiers (logistic regression, SVM, random forest, neural networks, etc.
- Evaluate performance (precision, accuracy, recall, F1 score, confusion matrix)
- Write explanations (justify choices, compare results)

10 Stellar dataset to predict star types

The goal of this project is to build a machine learning model capable of predicting the type of a star based on its physical properties, such as temperature, luminosity, radius, absolute magnitude, color, and spectral class. This classification task follows the Hertzsprung-Russell (HR) diagram, a fundamental tool in astrophysics that categorizes stars based on their temperature and luminosity.

We start with a binary classification problem, selecting two distinct star types to facilitate model training and evaluation. We then gradually extend it to multi-class classification with all six star types.

10.1 About the dataset

This dataset comes to us from Kaggle and includes 240 stars, classified into six different star types:

- 0 → Brown Dwarf
- 1 → Red Dwarf
- 2 → White Dwarf
- 3 → Main Sequence
- 4 → Supergiant

- 5 → Hypergiant

The properties of each star are measured relative to the Sun:

- Temperature (K) - Surface temperature in kelvins
- Luminosity (L/L_o) - Brightness relative to the sun
- Radius (R/R_o) - Radius relative to the sun
- Absolute magnitude (M_v) - Intrinsic luminosity
- Star color - Color observed after spectral analysis
- Spectral class - Classification based on spectral lines (O, B, A, F, G, K, M)

10.2 Data collection and preparation

The dataset was created using real astrophysical equations and data sources such as:

- Stefan-Boltzmann law - to calculate brightness
- Wien's Displacement Law - To estimate surface temperature
- Absolute Magnitude Relationships - To Determine Intrinsic Brightness
- Parallax Methods - To Derive Radius Values

The dataset was compiled from multiple online sources, taking about three weeks to collect and preprocess, ensuring that missing data were calculated using astrophysical formulas.

10.3 Importance of the study

Understanding how stars are classified and how they evolve over time is fundamental to astronomy and astrophysics. This study helps to:

- Validate the HR diagram using machine learning.
- Developing predictive models for star classification.
- Explore the importance of features in differentiating stars.

This project uses machine learning classifiers to analyze the performance of different models in stellar classification, from simple binary classification to 6-class multi-class classification. The results can provide valuable insights into the relationships between stellar properties and how stars fit into stellar evolution.

10.4 Exploration and analysis of the dataset

Before building a classification model, it is essential to explore and understand the dataset to identify patterns, relationships, and potential preprocessing needs. This section focuses on analyzing the structure of the dataset, checking for missing values, visualizing distributions, and understanding correlations between features.

10.4.1 1. Dataset Overview

We start by loading the dataset and displaying its first few rows to understand its structure and the type of data it contains.

```
1 # Display basic information about the dataset
2 df.info()
3 df.head()
```

Sortie :

```
1 <class 'pandas.core.frame.DataFrame'>
2 RangeIndex: 240 entries, 0 to 239
3 Data columns (total 7 columns):
4  #   Column                                Non-Null Count  Dtype
5  ---  ---
6  0   Temperature (K)                       240 non-null    int64
7  1   Luminosity(L/Lo)                       240 non-null    float64
8  2   Radius(R/Ro)                           240 non-null    float64
9  3   Absolute magnitude(Mv)                 240 non-null    float64
10  4   Star type                             240 non-null    int64
11  5   Star color                             240 non-null    object
12  6   Spectral Class                         240 non-null    object
13 dtypes: float64(3), int64(2), object(2)
14 memory usage: 13.2+ KB
15 Temperature (K) Luminosity(L/Lo)      Radius(R/Ro)  Absolute magnitude(Mv)  Star type
    Star color      Spectral Class
16 0      3068      0.002400      0.1700  16.12  0      Red      M
17 1      3042      0.000500      0.1542  16.60  0      Red      M
18 2      2600      0.000300      0.1020  18.70  0      Red      M
19 3      2800      0.000200      0.1600  16.65  0      Red      M
20 4      1939      0.000138      0.1030  20.06  0      Red      M
```

We note that:

- The number of variables is 7 and we have 240 samples in the dataset.

- Data types are numeric, except **Star color** and **Spectral Class** which are categorical.

10.4.2 2. Checking for missing values

Missing data can impact model performance. We check for missing values to determine if preprocessing steps, such as imputation or deletion, are necessary.

```
1 # Check for missing values
2 df.isnull().sum()
```

Output ;

```
1 Temperature (K)          0
2 Luminosity(L/Lo)         0
3 Radius(R/Ro)             0
4 Absolute magnitude(Mv)    0
5 Star type                0
6 Star color               0
7 Spectral Class           0
8 dtype: int64
```

No missing values

10.4.3 3. Statistical summary

Generating summary statistics allows you to know the range, mean and distribution of numeric variables.

```
1 # Check for missing values
2 df.isnull().sum()
```

Output :

| | Temperature (K) | Luminosity(L/Lo) | Radius(R/Ro) | Absolute magnitude(Mv) | Star type |
|-------|-----------------|------------------|--------------|------------------------|------------|
| count | 240.000000 | 240.000000 | 240.000000 | 240.000000 | 240.000000 |
| mean | 10497.462500 | 107188.361635 | 237.157781 | 4.382396 | 2.500000 |
| std | 9552.425037 | 179432.244940 | 517.155763 | 10.532512 | 1.711394 |
| min | 1939.000000 | 0.000080 | 0.008400 | -11.920000 | 0.000000 |
| 25% | 3344.250000 | 0.000865 | 0.102750 | -6.232500 | 1.000000 |
| 50% | 5776.000000 | 0.070500 | 0.762500 | 8.313000 | 2.500000 |
| 75% | 15055.500000 | 198050.000000 | 42.750000 | 13.697500 | 4.000000 |
| max | 40000.000000 | 849420.000000 | 1948.500000 | 20.060000 | 5.000000 |

It allows to :

- detect outliers (e.g., abnormally high temperatures or brightness).
- Understanding the scale and variance of features.

10.4.4 4. Class distribution (balance check)

To ensure that our dataset is not severely imbalanced, we visualize the distribution of different star types.

```
1 # Count the occurrences of each star type
2 sns.countplot(x=df['Star type'], palette='viridis')
3 plt.xlabel("Star Type")
4 plt.ylabel("Count")
5 plt.title("Class Distribution of Star Types")
6 plt.show()
```

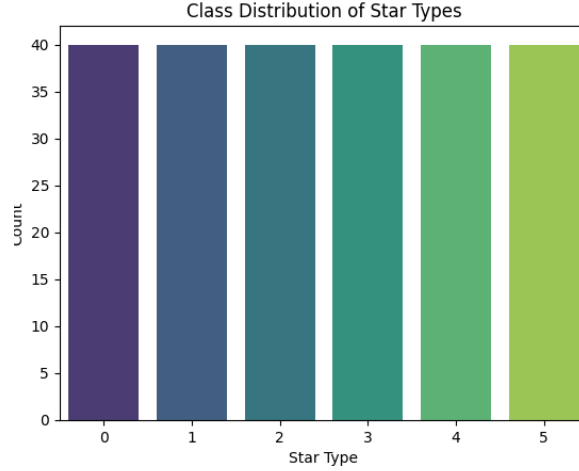


Figure 1: Classes Distribution

The dataset is perfectly balanced between the different classes, with 40 samples per class.

10.4.5 5. Variables corrélation Analysis

10.5 Correlation Matrix

This correlation matrix provides insight into the relationships between different features of the dataset. Here are some key observations:

10.5.1 Strong correlation with star type:

Luminosity (0.68), radius (0.66) and temperature (0.41) show a strong positive correlation with star type.

Absolute magnitude (-0.96) is strongly negatively correlated with star type, which makes sense since brighter stars (lower magnitude values) tend to be higher in the classification hierarchy.

10.5.2 Variables Dépendancies :

Luminosity and Radius (0.53): Larger stars tend to be brighter.

Absolute magnitude and luminosity (-0.69): Higher luminosity results in lower absolute magnitude values (inverse relationship by definition).

10.5.3 Color and spectral class of stars:

Star Color and Temperature (-0.7): This negative correlation makes sense because hot stars tend to appear blue, while cool stars appear red.

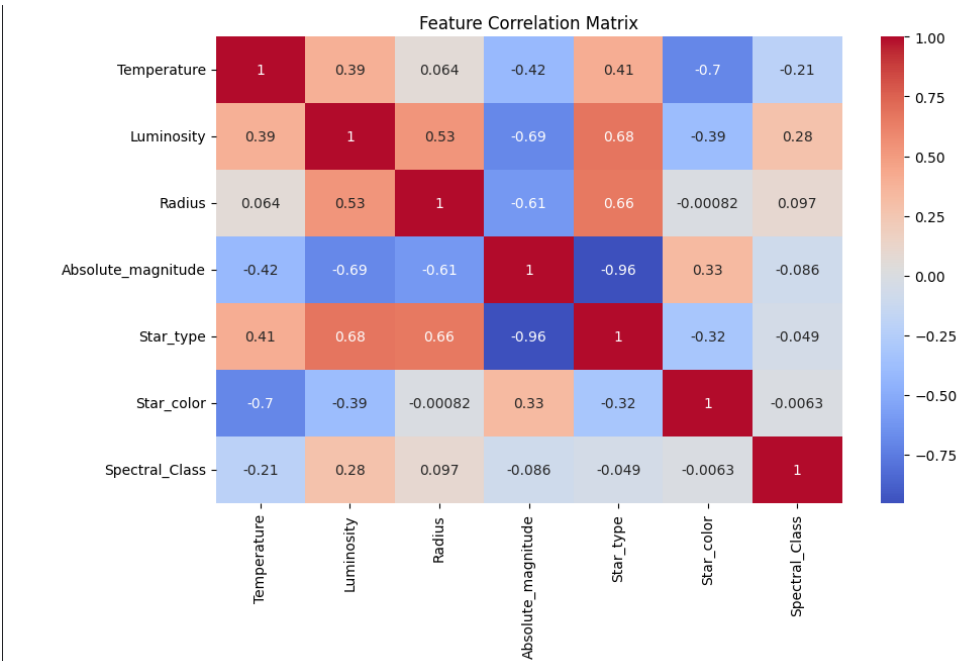


Figure 2: Correlation Matrix

Star color and absolute magnitude (0.33): The brightest stars tend to exhibit specific color characteristics.

10.5.4 The spectral class shows weak correlations :

It shows only minor correlations with other variables, suggesting that although it provides some classification information, it may not be the most powerful predictor compared to numerical values such as temperature, brightness, and absolute magnitude.

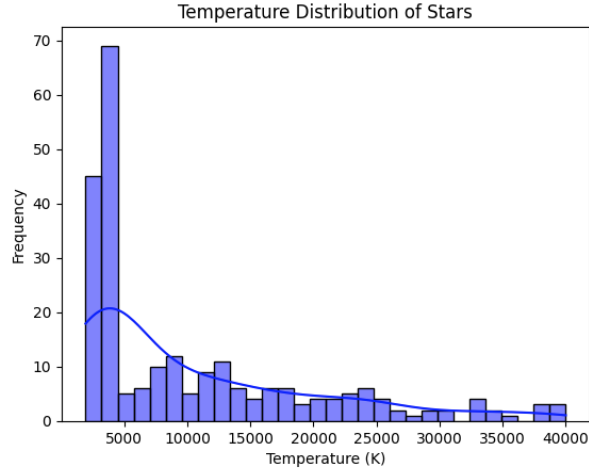


Figure 3: Distribution of temperatures

10.5.5 Implications :

- Key predictors: Brightness, radius, absolute magnitude, and temperature should be prioritized for classification models.
- Feature selection: Since the spectral class has weaker correlations, its contribution to predictive power should be analyzed in more detail.
- Possible redundancies: Absolute magnitude and brightness have a strong inverse correlation, meaning that one might be sufficient for modeling.

10.6 6. Visualization of feature distribution

To understand how characteristics vary across different types of stars, we use histograms and boxplots.

10.6.1 Distribution of temperatures :

Temperature distribution: see figure 3

This histogram visualizes the distribution of stellar surface temperatures across the data set. Here are the key observations:

10.6.2 Asymmetric distribution

- Most stars have low temperatures, peaking around 4000-5000 K.
- As the temperature increases, the frequency gradually decreases, showing that hotter stars are rarer.

- Few stars have a temperature above 30,000 K, indicating that extremely hot stars (e.g., O-type stars) are less common.

10.6.3 Most stars are cold

- The large number of stars with temperatures around 4000–6000 K suggests a high proportion of main sequence stars (such as Sun-like G-type stars and cooler K/M-type stars).
- These cooler stars (like the Red Dwarfs) are long-lived and more abundant in the universe.

10.6.4 Hot stars are less common

The histogram shows a decreasing trend for stars above 10,000 K, which is consistent with astrophysical expectations since massive, hotter stars (e.g., O and B types) have shorter lifetimes and are less frequently observed.

10.6.5 Density curve

The Kernel Density Estimation (KDE) curve further highlights the shape of the distribution, reinforcing the fact that most stars are in the colder temperature range.

Conclusion

- The data set is dominated by cooler stars, which is consistent with the actual stellar population.
- Hot, massive stars are less common, as predicted by theories of stellar evolution.
- This distribution aligns well with the Hertzsprung-Russell diagram, where most stars are cooler and fall into the main sequence category.

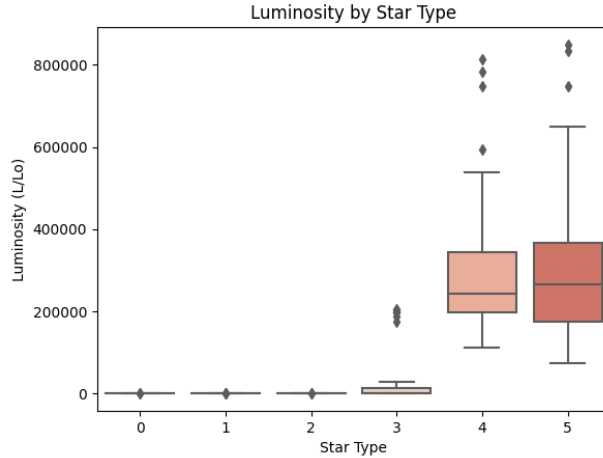


Figure 4: Luminosity vs. Star Type

Luminosity vs. Star Type :

10.6.6 Interpretation of the brightness graph as a function of star type

see figure 4

This box plot helps visualize how brightness (L/L_{\odot}) varies across different star types (0-5). Here are the key observations:

Distinct brightness ranges for star types

- Types 0, 1 and 2 (brown dwarfs, red dwarfs and white dwarfs) have a very low luminosity, close to zero, which indicates that they are faint stars.
- Type 3 stars (main sequence stars) have a slightly higher luminosity, but still relatively low compared to giant stars.
- Types 4 and 5 (supergiants and hypergiants) have significantly higher luminosities, with extreme values reaching more than 800,000 times the luminosity of the Sun.

10.6.7 Wide range of supergiants and hypergiants

- The box for types 4 and 5 is much larger, indicating that the brightness of these stars is very variable.
- These types include some of the most luminous stars in the universe, but their brightness can range from moderate to extremely bright.

10.6.8 High-luminosity aberrant stars

- Some extremely luminous stars in the supergiant and hypergiant categories appear as outliers above the whiskers.
- These are rare, ultra-luminous stars, probably supergiants or massive blue hypergiants.

10.6.9 A clear classification based on brightness

- The box plot confirms that star type is strongly correlated with luminosity.
- Main sequence stars (type 3) serve as a transition between low-luminosity dwarfs and very luminous giants.

10.6.10 Conclusion

- This distribution follows the expected stellar evolution pattern, where dwarfs are faint, main sequence stars have moderate luminosity, and giants/hypergiants are very luminous.
- The sharp separation in brightness suggests that this feature is highly relevant for stellar classification.

10.7 Data cleansing

10.7.1 Rename columns

Now we'll clean up the data by first replacing space with `_` and renaming these columns:

- Temperature (K)
- Luminosity(L/Lo)
- Radius(R/Ro)
- Absolute magnitude(Mv)

Removing the parentheses, we get:

- Temperature
- Luminosity
- Radius
- Absolute_magnitude

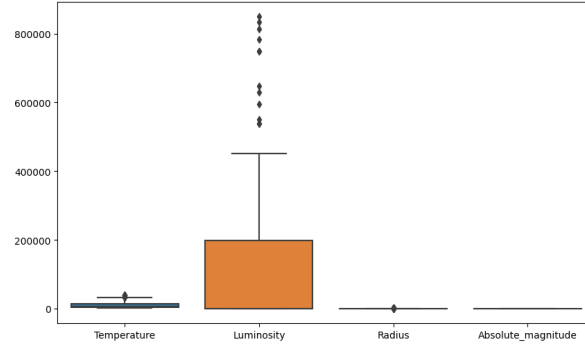


Figure 5: Outliers for the measures

10.8 Outlier Treatment

Outliers can affect the performance of machine learning models. They can be detected using boxplots. (see figure 5).

10.8.1 Boxplot Interpretation (Outlier Analysis)

The boxplot visualizes the distribution of four numerical variables: Temperature, Luminosity, Radius, and Absolute Magnitude. Here's what we can observe:

10.8.2 Brightness exhibits extreme outliers

The luminosity shows a wide spread with many extreme outliers above the upper limit. This suggests that some stars have extremely high luminosity, likely corresponding to supergiants and hypergiants. This is consistent with astrophysics, where some rare stars are much more luminous than others.

10.8.3 Temperature, radius, and absolute magnitude have few or no extreme outliers

The distributions of these parameters are relatively compact, with few or no outliers. This suggests that most stars follow a more regular pattern in these characteristics relative to luminosity.

10.8.4 Brightness outliers should be analyzed carefully

Since the boxplot shows that the Brightness has extreme outliers, we will remove them. However, these outliers may represent real astronomical phenomena rather than errors. Instead of blindly removing them, we should analyze their impact on the classification models before deciding on any transformation (e.g., logarithmic scaling).

10.9 How to manage these outliers instead of deleting them?

In astrophysics, extreme values of luminosity, temperature, and radius often represent actual stellar phenomena, such as supergiants and hypergiants, rather than errors. Simply removing them could skew the entire dataset and impact the classification model's ability to recognize these types of stars.

Instead of blindly removing outliers, we can take other approaches: Experiment with and without outliers.

Here is an approach to compare the performance of a model with and without outliers to determine their impact on classification:

10.9.1 Using the interquartile range (IQR) method to filter out extreme outliers :

```
1 def remove_outliers(df, column):
2     Q1 = df[column].quantile(0.25)
3     Q3 = df[column].quantile(0.75)
4     IQR = Q3 - Q1
5     lower_bound = Q1 - 1.5 * IQR
6     upper_bound = Q3 + 1.5 * IQR
7     return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
8
9 # Apply outlier removal on Luminosity (or any other feature if needed)
10 df_no_outliers = remove_outliers(df, "Luminosity")
```

10.9.2 We log-transform the brightness to reduce skewness and scale all features using RobustScaler (outlier-resistant) :

```
1 # Log transform Luminosity
2 df['Luminosity_log'] = np.log1p(df['Luminosity'])
3 df_no_outliers['Luminosity_log'] = np.log1p(df_no_outliers['Luminosity'])
4
5 # Select numerical features for scaling
6 features = ["Temperature", "Luminosity_log", "Radius", "Absolute_magnitude"]
7 scaler = RobustScaler()
8
9 df[features] = scaler.fit_transform(df[features])
10 df_no_outliers[features] = scaler.fit_transform(df_no_outliers[features])
```

10.10 Train models with/without outliers and compare performance :

- If the model with the outliers performs better, we keep them.

- If the model without outliers improves the classification, we remove them.

Random Forest :

```
1 # Define target variable & features
2 X = df[features]
3 y = df["Star_type"]
4
5 X_no_outliers = df_no_outliers[features]
6 y_no_outliers = df_no_outliers["Star_type"]
7
8 # Split datasets
9 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
10 X_train_no, X_test_no, y_train_no, y_test_no = train_test_split(X_no_outliers, y_no_outliers,
    test_size=0.2, random_state=42)
11
12 # Train Random Forest Model
13 model = RandomForestClassifier(random_state=42)
14 model_no_outliers = RandomForestClassifier(random_state=42)
15
16 model.fit(X_train, y_train)
17 model_no_outliers.fit(X_train_no, y_train_no)
18
19 # Predictions
20 y_pred = model.predict(X_test)
21 y_pred_no_outliers = model_no_outliers.predict(X_test_no)
22
23 # Evaluate Performance
24 print("Model with Outliers:")
25 print(classification_report(y_test, y_pred))
26
27 print("\nModel without Outliers:")
28 print(classification_report(y_test_no, y_pred_no_outliers))
```

Results of Random Forest :

| | | | | | |
|----|-------------------------|-----------|--------|----------|---------|
| 1 | Model with Outliers: | | | | |
| 2 | | precision | recall | f1-score | support |
| 3 | | | | | |
| 4 | 0 | 1.00 | 1.00 | 1.00 | 8 |
| 5 | 1 | 1.00 | 1.00 | 1.00 | 7 |
| 6 | 2 | 1.00 | 1.00 | 1.00 | 6 |
| 7 | 3 | 1.00 | 1.00 | 1.00 | 8 |
| 8 | 4 | 1.00 | 1.00 | 1.00 | 8 |
| 9 | 5 | 1.00 | 1.00 | 1.00 | 11 |
| 10 | | | | | |
| 11 | accuracy | | | 1.00 | 48 |
| 12 | macro avg | 1.00 | 1.00 | 1.00 | 48 |
| 13 | weighted avg | 1.00 | 1.00 | 1.00 | 48 |
| 14 | | | | | |
| 15 | | | | | |
| 16 | Model without Outliers: | | | | |
| 17 | | precision | recall | f1-score | support |
| 18 | | | | | |
| 19 | 0 | 1.00 | 1.00 | 1.00 | 11 |
| 20 | 1 | 1.00 | 1.00 | 1.00 | 8 |
| 21 | 2 | 1.00 | 1.00 | 1.00 | 9 |
| 22 | 3 | 1.00 | 1.00 | 1.00 | 7 |
| 23 | 4 | 1.00 | 1.00 | 1.00 | 6 |
| 24 | 5 | 1.00 | 1.00 | 1.00 | 5 |
| 25 | | | | | |
| 26 | accuracy | | | 1.00 | 46 |
| 27 | macro avg | 1.00 | 1.00 | 1.00 | 46 |
| 28 | weighted avg | 1.00 | 1.00 | 1.00 | 46 |

The evaluation models (with and without outliers) achieve perfect 100% accuracy for all metrics (precision, recall, and F1 score). This could mean :

10.10.1 Perfect Classification

The model correctly classifies all samples. This indicates that the data is well separated, making classification easier.

10.10.2 Outliers have minimal impact

The results are almost identical between models with and without outliers.

The support (number of instances per class) is slightly different, but the performance remains perfect. This suggests that the outliers did not negatively impact the model's performance.

10.10.3 Possible Overfitting ?

An accuracy of 100/100 may indicate that the dataset is too easy to classify or that the model has memorized the training data instead of generalizing well.

10.11 Logistic Regression :

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.metrics import classification_report, accuracy_score
5
6 # Define target variable & features
7 X = df[features]
8 y = df["Star_type"]
9
10 X_no_outliers = df_no_outliers[features]
11 y_no_outliers = df_no_outliers["Star_type"]
12
13 # Split datasets
14 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
15                                                    stratify=y)
16
17 X_train_no, X_test_no, y_train_no, y_test_no = train_test_split(X_no_outliers, y_no_outliers,
18                                                                    test_size=0.2, random_state=42, stratify=y_no_outliers)
19
20 # Train Random Forest Model
21 model_log = LogisticRegression(max_iter=1000, multi_class="multinomial", solver="lbfgs")
22 model_no_outliers_log = LogisticRegression(max_iter=1000, multi_class="multinomial", solver="
23                                           lbfgs")
24
25 # Predictions
26 y_pred = model_log.predict(X_test)
27 y_pred_no_outliers = model_no_outliers_log.predict(X_test_no)
```

```

28 # Evaluate Performance
29 print("Model with Outliers:")
30 print("Accuracy:", accuracy_score(y_test, y_pred))
31
32 print(classification_report(y_test, y_pred))
33
34 print("\nModel without Outliers:")
35 print("Accuracy:", accuracy_score(y_test_no, y_pred_no_outliers))
36
37 print(classification_report(y_test_no, y_pred_no_outliers))

```

10.11.1 Results of Logistic Regression :

```

1  Model with Outliers:
2  Accuracy: 0.8958333333333334
3
4      precision    recall  f1-score   support
5
6      0          0.80      1.00      0.89         8
7      1          0.86      0.75      0.80         8
8      2          1.00      1.00      1.00         8
9      3          0.86      0.75      0.80         8
10     4          0.88      0.88      0.88         8
11     5          1.00      1.00      1.00         8
12
13     accuracy          0.90         48
14     macro avg          0.90         48
15     weighted avg          0.90         48
16
17  Model without Outliers:
18  Accuracy: 0.9347826086956522
19
20     precision    recall  f1-score   support
21
22     0          0.89      1.00      0.94         8
23     1          0.88      0.88      0.88         8
24     2          1.00      1.00      1.00         8
25     3          0.88      0.88      0.88         8
26     4          1.00      0.86      0.92         7
27     5          1.00      1.00      1.00         7
28
29     accuracy          0.93         46
30     macro avg          0.94         46
31     weighted avg          0.94         46

```

10.11.2 Why a régression logistique ?

- Interpretable – We can analyze the importance of variables (coefficients).
- Fast and efficient – Works well for small datasets.
- Basic model – Allows comparison with more complex models.

10.11.3 Interpretation of the Logistic Regression results

10.12 Comparison between models (with and without outliers)

- Improved accuracy :
 - With outliers : 89,58%
 - Without outliers : 93,48%
 - Removing outliers resulted in an improvement in accuracy of approximately 4%.
- Improved F1-score:
 - Both macro and weighted F1 scores increased after outlier removal, indicating a more balanced classification.

Effect of outliers on model performance

- The model with outliers showed slightly lower recall for some classes, meaning it misclassified some stars.
- The model without outliers performed better in most classes, suggesting that outliers negatively impacted the model by introducing noise.

Class-specific observations

- Class 0 (for example, white dwarfs) :
 - With outliers : recall of 100%, which means that all real white dwarfs have been correctly identified.
 - Without outliers : still a recall of 100%, showing robustness in the classification of this type.
- Classes 2 and 5 (e.g., giant and supergiant stars):
 - Still 100% precision and recall, indicating that these are the most recognizable star types.
- Class 1, 3, 4 (e.g., main sequence and other types):

- These classes had misclassifications with outliers, but removing them improved both accuracy and recall.

Why was outlier removal useful ?

- Extreme values of brightness, radius, and absolute magnitude may have distorted the model's decision limits.
- By removing outliers, the model focused on the majority distribution rather than being influenced by extreme cases.

Key points

- Removing extreme outliers improved the model's accuracy.
- Most classes were ranked higher after removing outliers.
- Logistic regression worked well, but a more complex model (e.g., Random Forest) might better capture nonlinear relationships.

Next steps

- Check the confusion matrix to see where errors occur.
- Analyze feature importance from logistic regression coefficients.

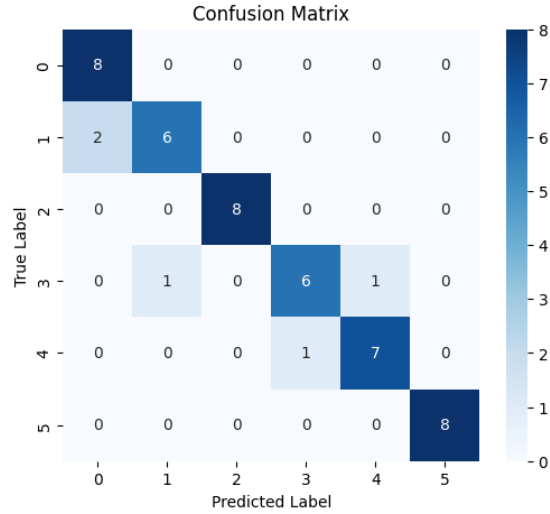


Figure 6: Confusion Matrix

10.13 Confusion Matrix

The Confusion Matrix (see figure 6) helps us understand which types of stars are misclassified.

- Diagonal values represent correct classifications.
- Off-diagonal values indicate classification errors (which classes are confused).

10.13.1 Interprétation of the Confusion Matrix

The confusion matrix provides a detailed view of the model's classification performance across six star types (0 to 5). Each row represents the actual class and each column represents the predicted class. Diagonal values indicate correct classifications, while off-diagonal values represent misclassifications.

Key Observations :

Correct classifications (diagonal values)

- Star Type 0: 8 out of 8 correctly classified (accuracy of 100%).
- Star Type 1: 6 out of 8 correctly classified (accuracy of 75%).
- Star Type 2: 8 out of 8 correctly classified (accuracy of 100%).
- Star Type 3: 6 out of 8 correctly classified (accuracy of 75%).
- Star Type 4: 7 out of 8 correctly classified (accuracy of 87,5%).
- Star Type 5: 8 out of 8 correctly classified (accuracy of 100%).

Classification errors (off-diagonal values)

- Type 1 star: 2 samples misclassified as type 0 (the model mistook some type 1 stars for type 0).
- Type 3 star:
 - 1 sample wrongly classified as type 1.
 - 1 sample wrongly classified as type 4.
- Type 4 star: 1 sample wrongly classified as type 3.

10.13.2 Model performance analysis

High accuracy for most classes:

- Types 0, 2 and 5 have perfect classification (accuracy of 100%).
- Type 4 has only one classification error, maintaining high accuracy (87,5%).

Confusion between similar star types:

- Type 1 is confused with type 0 → This could indicate an overlap of their characteristics.
- Type 3 is confused with types 1 and 4 → Suggests that type 3 shares characteristics with both.
- Type 4 is confused with type 3, indicating a potential similarity in brightness, temperature, or radius.

10.13.3 Possible reasons for classification errors

Overlapping features :

- Some types of stars may have overlapping properties (e.g., temperature, luminosity), making them more difficult to distinguish.

Data Imbalance :

- If some types of stars have fewer samples, the model may have difficulty learning their patterns.

Model limitations :

- If logistic regression is a linear model; if the decision boundaries between star types are nonlinear, they may not be perfectly captured.

10.13.4 Recommendations for improvement

Feature Engineering :

- Explore higher-order interactions or derived features to better separate overlapping classes.

Try a nonlinear model :

- Decision trees, random forests, or neural networks might better capture complex relationships.

Balance the dataset :

- Use oversampling/undersampling techniques when data imbalance occurs.

Hyperparameter tuning :

- Adjust the regularization strength in logistic regression to fine-tune performance.

Conclusion

- The logistic regression model performs well, with high accuracy for most classes.
- Some misclassification occurs, particularly between similar star types, suggesting overlapping characteristics.
- Further improvements can be made by refining features or using more complex models.

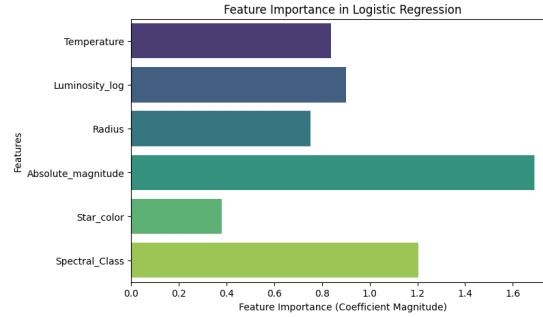


Figure 7: Features importance logistic regression with outliers

10.14 visualize the importance of features

Since logistic regression is a linear model, we can analyze the importance of features by examining the absolute values of the model coefficients. Higher absolute values indicate greater influence on classification decisions.

We will use a bar chart showing the importance of features in different input variables.

Model with outliers (see figure 7) :

Interpreting feature importance for logistic regression (with outliers)

The feature importance plot for logistic regression with outliers shows how different features influence star classification. Here's a detailed analysis of each feature's impact:

- Magnitude_absolute (highest importance)
 - This feature has the largest coefficient magnitude, meaning it plays the most important role in distinguishing star types.
 - Since absolute magnitude is a measure of intrinsic brightness, it makes sense that it would strongly affect classification.
- Spectral_Class (second most important feature)
 - Spectral class defines the type of a star based on its temperature and color.
 - Its high significance suggests that the spectral classification aligns well with the star type categorization of the dataset.
- Luminosity_log, temperature and radius (moderate importance)
 - These three features have comparable levels of importance.

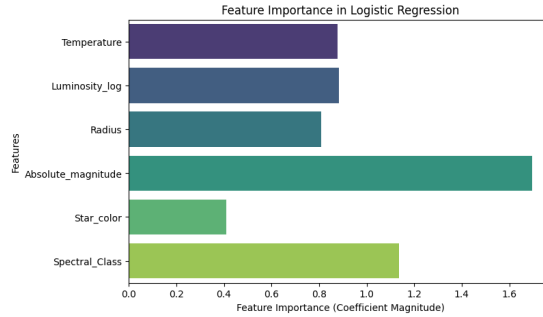


Figure 8: Features importance logistic regression without outliers

- Luminosity_log: Since we logarithmically transformed the luminosity, its magnitude reflects how much a star’s brightness influences the classification.
- Temperature: A key factor in classifying stars, but its importance is slightly less than absolute magnitude and spectral class.
- Radius: Affects classification, but not as strongly as other features.
- Star_color (least important feature)
 - Star color has the smallest impact on classification.
 - This may be due to redundancy, as temperature and spectral class already capture much of the same information.

Model without outliers (see figure 8) :

Interpretation of feature importance for logistic regression (without outliers)

The distribution of feature importance in the model without outliers remains similar to that with outliers, but with some notable differences:

- Absolute_magnitude (still the most important feature)
 - Just as in the outlier model, absolute magnitude plays the most important role in classification.
 - Its importance remains high, reinforcing the fact that the intrinsic luminosity of a star is a dominant factor in its classification.
- Spectral_Class (second most important feature)
 - The importance of the spectral class remains strong, indicating that removing outliers does not reduce its predictive power.

- Temperature, Luminosity_log and Radius (moderate importance, but slightly offset)
 - Temperature: Retains similar importance, but its impact appears slightly stronger compared to the outlier model.
 - Luminosity_log: remains a key feature but may have become slightly less influential after removing outliers.
 - Temperature: A key factor in classifying stars, but its importance is slightly less than absolute magnitude and spectral class.
 - Radius: has slightly lower importance than the outlier model, suggesting that extreme values might have exaggerated its effect previously.
- Star_color (still the least important feature)
 - The impact of star color remains the weakest, further confirming that it does not add much unique information beyond temperature and spectral class.

Comparison: with outliers and without outliers

| Feature | outliers | without outliers | Comparison |
|--------------------|----------|------------------|---|
| Absolute_magnitude | Highest | Highest | Remains the most dominant feature. |
| Spectral_Class | High | High | Maintains a strong influence in both cases. |
| Temperature | Moderate | Slightly higher | Slight increase in importance. |
| Luminosity_log | Moderate | Slightly lower | Less impact after deletion. |
| Radius | Moderate | Lower | Less impact after deletion |
| Star_color | Lowest | Lowest | It's still the least useful feature. |

Table 1: Comparison of feature importance with and without outliers

Conclusion

- Outlier removal improves stability: the importance of features such as radius and brightness log becomes more balanced, avoiding overemphasis caused by extreme values.
- Absolute magnitude and spectral class remain dominant: regardless of whether we include or exclude outliers, these two features determine the classification of stars.
- The relevance of temperature increases slightly: without outliers, temperature appears to contribute more to classification, perhaps because outliers have already distorted its relationship.

This analysis provides a strong rationale for removing outliers, as it makes the model more stable and interpretable while maintaining high classification performance.

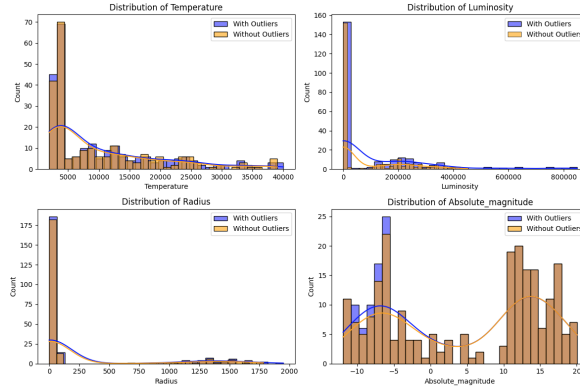


Figure 9: Overall distributions

10.15 More advanced analysis

Systematic view of the distribution of variables

Histograms to see overall distributions :

Histograms (see figure 9) compare the distribution of four key characteristics (temperature, brightness, radius, absolute magnitude) before and after removal of outliers.

10.15.1 Temperature distribution :

- With outliers (blue): The distribution is strongly skewed to the right, with a long tail extending beyond 30,000 K.
- No outliers (orange): The majority of values remain below 10,000 K and the distribution becomes more concentrated.
- Interpretation: Outlier removal eliminates extremely hot stars (e.g., massive O-type stars), leading to a more realistic temperature range.

10.15.2 Luminosity distribution:

- With outliers (blue): Values go up to 800,000 solar luminosities, with a long tail tilted to the right.
- No outliers (orange): The peak brightness is significantly reduced, leading to a more compact and less skewed distribution.
- Interpretation: Outlier removal removes extremely bright supergiants, focusing the model on more typical giant and main sequence stars.

10.15.3 Radius distribution :

- With outliers (blue): There is a long tail extending beyond 1000 solar radii.
- No outliers (orange): The majority of stars have a radius less than 250.
- Interpretation: Giant and supergiant stars with huge radii have been removed, making the dataset more balanced.

10.15.4 Absolute magnitude distribution :

- With outliers (blue): The distribution covers a wide range, with negative values (very bright stars).
- No outliers (orange): The range is less spread out and the distribution appears smoother.
- Interpretation: Removing outliers leads to a data set that better represents the majority of stars.

Conclusion :

- The dataset was highly skewed due to extreme values of temperature, brightness, and radius.
- After removing outliers, the distributions became more normal and less skewed, leading to a better performing model.
- The new dataset focuses on main-sequence and moderate giants rather than extreme stars.

Boxplots to check how distributions change :

Interpreting boxplots :

Boxplots (see figure 10) compare the distribution of temperature, brightness, radius, and absolute magnitude before and after outlier removal.

Boxplot with outliers (top graph) :

- The brightness has extreme outliers above 800,000, creating a very extended boxplot.
- Temperature and radius also show extreme values.
- The absolute magnitude is more compact with fewer extreme values.

Key issue: The presence of highly skewed distributions and extreme outliers, especially in Luminosity, affects the balance of the dataset.

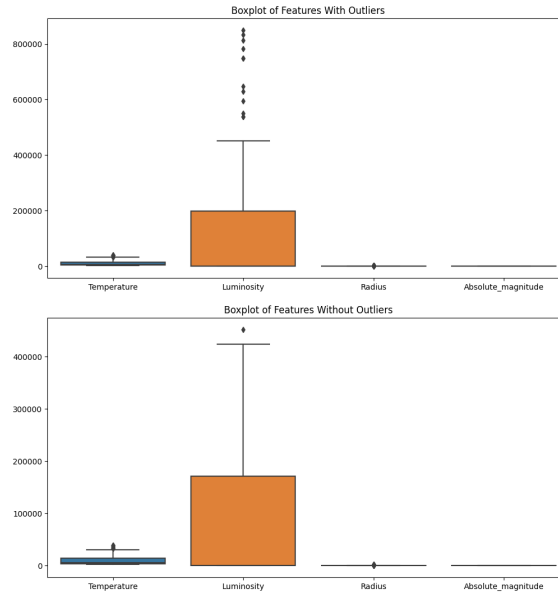


Figure 10: check the evolution of distributions

Boxplot without outliers (bottom graph) :

- Boxplot without outliers (bottom graph)
- Temperature and radius show a more compact range.

The dataset is now more balanced and the spread of values is reduced.

Key takeaways :

- Outlier removal significantly reduced extreme values, especially for brightness.
- The dataset is now better suited to machine learning models, reducing bias.
- Although the luminosity still has high values, it is much more controlled.

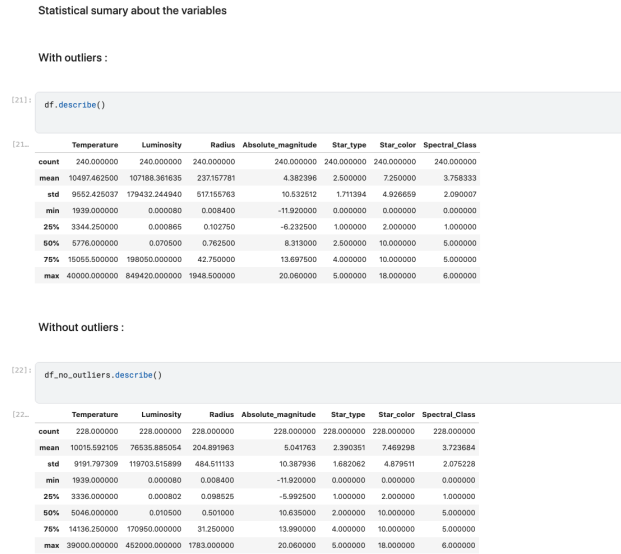


Figure 11: statistical summaries (with or without outliers)

10.16 Statistical data before and after removal of outliers

Interpretation of statistical summaries (with or without outliers)

This table (see figure 11 provides descriptive statistics (mean, standard deviation, min/max values, percentiles) for the dataset before and after outlier removal.

With the outliers

- The luminosity has an extremely high maximum value (849,420) and a very high standard deviation (179,432), confirming extreme outliers.
- The temperature also has a very wide range (1939 to 40,000).
- The radius has a maximum of 1948.5, well above the 75th percentile (42.75).
- The absolute magnitude appears less affected by outliers (range: -11.92 to 20.06).

Main issue : The dataset contains extreme values, particularly in terms of brightness, temperature and radius, which can bias the models.

No outliers

- The maximum luminosity decreases considerably (from 849,420 to 452,000), thus reducing the impact of extreme values.
- The maximum temperature decreases slightly (40,000 to 39,000), but the difference remains significant.
- The maximum radius changes from 1948.5 to 1783, indicating that some high values have been removed.
- Standard deviations decrease for brightness, temperature, and radius, indicating a more balanced data set.

Main problem: The dataset has extreme values, especially in terms of brightness, temperature, and radius, which can bias the models.

Key improvements :

- The dataset is now more stable, with less extreme variations.
- Reductions in standard deviation suggest a more normal distribution, thus improving the reliability of the model.

10.17 Binary classification: White Dwarf vs. Main Sequence

exploration, encoding of categorical variables, analysis of correlations and a first binary classification using the Random Forest model:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.model_selection import train_test_split
6 from sklearn.preprocessing import LabelEncoder, StandardScaler
7 from sklearn.ensemble import RandomForestClassifier
8 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
9
10 # Load dataset
11 df = pd.read_csv("6 class csv.csv")
12
13 # Display first few rows
14 display(df.head())
15
16 # Basic info and missing values
17 print(df.info())
```

```

18 print(df.isnull().sum())
19
20 # Encoding categorical variables
21 le_color = LabelEncoder()
22 df['Star color'] = le_color.fit_transform(df['Star color'])
23
24 le_spectral = LabelEncoder()
25 df['Spectral Class'] = le_spectral.fit_transform(df['Spectral Class'])
26
27 # Correlation matrix
28 plt.figure(figsize=(10,6))
29 sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
30 plt.title('Feature Correlation Matrix')
31 plt.show()
32
33 # Binary classification: Selecting two classes
34 df_binary = df[df['Star type'].isin([2, 3])] # Example: White Dwarf vs. Main Sequence
35 X = df_binary.drop(columns=['Star type'])
36 y = df_binary['Star type']
37
38 # Split data
39 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
40
41 # Standardizing numerical features
42 scaler = StandardScaler()
43 X_train = scaler.fit_transform(X_train)
44 X_test = scaler.transform(X_test)
45
46 # Train classifier (Random Forest as example)
47 clf = RandomForestClassifier(n_estimators=100, random_state=42)
48 clf.fit(X_train, y_train)
49 y_pred = clf.predict(X_test)
50
51 # Evaluation
52 print("Accuracy:", accuracy_score(y_test, y_pred))
53 print(confusion_matrix(y_test, y_pred))
54 print(classification_report(y_test, y_pred))
55
56 # Feature importance
57 importances = pd.Series(clf.feature_importances_, index=df_binary.drop(columns=['Star type']).
    columns)
58 importances.sort_values().plot(kind='barh', title='Feature Importances')
59 plt.show()

```

Explanations

Encoding categorical variables

Encoding categorical variables is necessary because most machine learning algorithms require numeric inputs rather than text labels.

Machine learning algorithms require digital data

Many models (e.g., logistic regression, Random Forest, neural networks) operate on numeric data and cannot handle categorical strings like "Red", "Blue", or "O", "B", "A".

Ensure comparability

Coding transforms categories into a format that allows algorithms to interpret differences and relationships between classes.

Encoding labels for ordinal data

Star color and spectral class exhibit an inherent order or grouping, which is why we use label coding, which assigns unique integers to the different categories. Although the spectral classes (O, B, A, F, G, K, M) follow a known order in astrophysics (O being the hottest, M the coolest), label coding preserves this structure for the models.

Alternative: One-Hot-Encoding

If the categories were truly nominal (without order), One-Hot Encoding (OHE) could be an alternative. However, OHE increases dimensionality, which is not ideal for small datasets.

By encoding categorical features, we allow the model to process and learn patterns efficiently without introducing inconsistencies.

Interpretation of the feature correlation matrix for binary classification (see figure 12)

The correlation matrix helps us understand the relationships between features before applying machine learning. Let's analyze it in the context of binary classification (star type: 2 vs. 3).

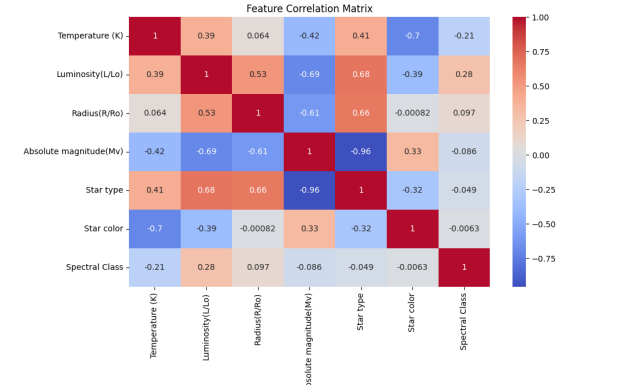


Figure 12: Binary correlation matrix

```

Accuracy: 1.0
[[8 8]
 [0 8]]
precision    recall  f1-score   support

      2       1.00      1.00      1.00         8
      3       1.00      1.00      1.00         8

 accuracy          1.00         16
  macro avg          1.00         16
 weighted avg          1.00         16

```

Figure 13: binary results : random forest

Key observations from the correlation matrix

Strong negative correlation (-0.96) between absolute magnitude and star type

- This means that as the absolute magnitude increases (i.e. the star appears fainter), the star type changes from one class to another.
- Since absolute magnitude is logarithmic, a lower value means the star is brighter. This suggests that one class might contain brighter stars, while the other has fainter ones.

Strong positive correlations (greater than 0.6)

- Luminosity and star type (0.68): One class tends to have brighter stars.
- Radius and star type (0.66): A class probably contains stars with larger radii.
- Temperature and star type (0.41): Temperature also plays a role but is less influential than luminosity/radius.

Negative correlation between color and temperature of stars (-0.70)

- A more blue-white star (lower "star color" value) is associated with higher temperatures.
- This makes sense in stellar classification, where blue stars are hotter than red stars.

Weak correlations with spectral class

- Spectral class has a low correlation with most features, suggesting that it is not the strongest predictor for your classification task.

How does this help classification ?

- Absolute magnitude, luminosity, and radius appear to be the most powerful predictors for distinguishing between the two types of stars.
- The star's temperature and color also provide useful information, but they are slightly less correlated with the target.
- The spectral class doesn't add much, so removing it could simplify the model without having a big impact on accuracy.

Interpretation of classification results with random forest

The model achieved perfect performance with an accuracy of 1.0 (100%) on the binary classification task. Let's break down the results.

Interpretation of the confusion matrix

```
1
2 [[8 0]
3  [0 8]]
```

- The lines represent the actual classes (star type 2 and star type 3).
- The columns represent the predicted classes.
- The values indicate how many instances were classified correctly or incorrectly.

Key information :

- 8 class 2 cases were correctly classified (true positives, TP).
- 8 class 3 cases were correctly classified (true positives, TP).
- No false positives (FP) or false negatives (FN), meaning there were no misclassifications.
- Perfect classification performance.

| Class | Precision | Recall | F1-Score | Support |
|-------------------------|--------------------|--------|----------|---------------------|
| 2 | 1.00 | 1.00 | 1.00 | 8 |
| 3 | 1.00 | 1.00 | 1.00 | 8 |
| Overall Accuracy | 1.00 (100%) | | | 16 instances |

Table 2: Classification Report for Binary Classification

Analysis of precision, recall and F1 score

- Accuracy ($TP / (TP + FP)$): How many predicted class 2 (or 3) were actually correct ?
 - Here it is 1.00 (100%), which means there is no classification error.
- Recall ($TP / (TP + FN)$): Among the real 2 (or 3) classes, how many did we predict correctly?
 - Again, 1.00 (100%), which means the model correctly identified all instances.
- F1 score (harmonic mean of precision and recall) :
 - Also 1.00, which shows that the model has perfectly balanced the two measures.
- Macro average and weighted average:
 - Since both classes are balanced (8 instances each), both averages are also 1.00.

Key takeaways

- Perfect accuracy (100%): The model correctly classified all test examples.
- No classification errors: no false positives (FP) or false negatives (FN).
- Well-separated classes: Features (such as absolute magnitude, brightness, radius) are likely to be highly discriminating.

Potential concerns :

- Overfitting ? If the dataset is small, the model may have memorized patterns instead of generalizing.
- Test set size? There were only 16 test samples, so the results might not be generalizable to a larger dataset.
- Try cross-validation: To confirm robustness, the model should be tested on different splits.

Checking for overfitting :

Let's check the performance with cross-validation :

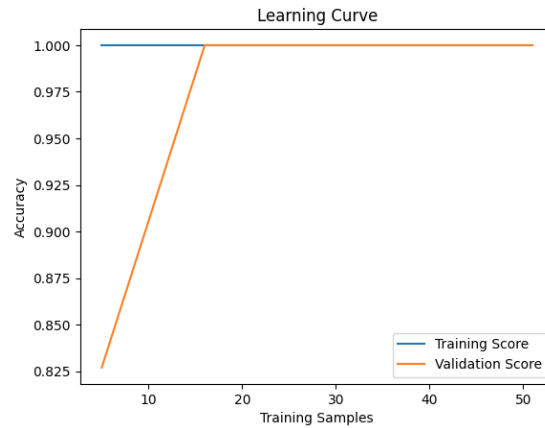


Figure 14: learning curve

- K-Fold cross-validation allows performance to be evaluated on different subsets of data.
- If the model works well on some folds but poorly on others, this may indicate overfitting.

Results :

```
1 Cross-Validation Scores: [1. 1. 1. 1. 1.]
2 Mean CV Accuracy: 1.0
```

The cross-validation scores are [1. 1. 1. 1. 1.], meaning that for each fold, the model achieved 100% accuracy. The average CV accuracy is also 1.0 (100%), suggesting that the model is consistently perfect in different subsets of the training data.

10.17.1 Key takeaways :

Perfect generalization on training data

- The model performs identically across all cross-validation folds, meaning there is no variation in accuracy between different data splits.
- This may indicate that the data is too easy to classify or that the model has learned very distinct patterns for the selected features.

Strong Indications of Overfitting

- It is very rare to achieve 100% accuracy in all cases, unless the dataset is very simple or has strong feature separability.
- If the test set (unseen data) also shows 100% accuracy, it may mean that the model is memorizing rather than generalizing.

10.18 Interpreting the learning curve (see figure 14) :

The learning curve shows that training and validation accuracy very quickly reaches 100

- Signs of overlearning
 - The learning accuracy is constantly at 100% → the model is probably memorizing the data instead of generalizing it.
 - No gap between training and validation curves → Generally, a small gap is expected due to generalization issues. Here, the two curves converge perfectly, which is rare in real-world problems.
- Possible Causes
 - Too few training samples: The model may see the same patterns repeatedly, making it easier to remember.

10.19 Final thoughts

Given that cross-validation and the learning curve show an accuracy of 100%, the model is probably too perfect for a real-world scenario. It would be worth checking whether the dataset is too simple, because even considering the entire dataset (with all classes), we obtain an accuracy of 100% (see previous sections) and an accuracy around 90% with logistic regression. In any case, the perfect accuracy obtained with a random forest is most likely due to:

- A dataset too simple to be classified.
- The classification power of ensemble learners like random forest.

11 Stellar classification dataset - SDSS17

11.1 Dataset Overview

The dataset comes to us from Kaggle eThe data set used in this study is the Stellar Classification Dataset - SDSS17, from the Sloan Digital Sky Survey (SDSS). It contains 100,000 astronomical observations, each classified as a galaxy, star, or quasar based on its spectral characteristics. The objective of this classification task is to develop a machine learning model capable of distinguishing these three categories based on the provided features.

Each observation includes 17 input features and 1 target variable (class), which indicates the object type. Features include photometric data (u, g, r, i, z), spatial coordinates (alpha, delta), and redshift measure-

ments, among others. In addition, the dataset contains several identification columns, such as `obj_ID` and `spec_obj_ID`, which are not relevant for classification and will be removed during preprocessing.

Columns :

- `obj_ID` = Object ID, the unique value that identifies the object in the image catalog used by 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 analysis
- `rereun_ID` = Replay number to specify how the image was processed
- `cam_col` = Camera column to identify the scan line in the race
- `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 increasing wavelength
- `plate` = Plate ID, identifies each plate in SDSS
- `MJD` = Modified Julian date, used to indicate when an SDSS data was taken
- `fiber_ID` = Fiber ID that identifies the fiber that pointed the light toward the focal plane in each observation

Key information

- Target variable: `class` (galaxy, star or quasar))
- `u`, `g`, `r`, `i`, `z` (photometric data)
- `alpha`, `delta` (position information, may be useful)

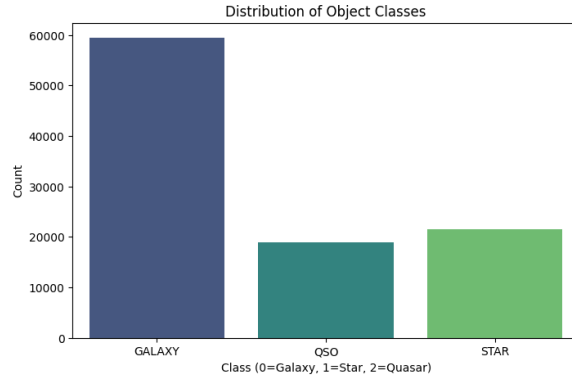


Figure 15: Class distribution

12 Exploratory Data Analysis (EDA)

12.1 Data analysis

- All features are digital, except **class** which is categorical
- No missing values
- No duplicates
- The dataset consists of 100,000 rows and 18 features

12.2 Visualization of class distribution

Bar chart (see figure 15) the number of objects (stars, galaxies, quasars), in order to check whether the classes are balanced.

The bar chart illustrates the distribution of object classes in the dataset, which consists of three categories: galaxies, stars, and quasars (QSOs).

Key observations :

- Classes imbalance :
 - Galaxies are the most common category, with a significantly higher number compared to the other two classes.
 - Stars and quasars (QSOs) appear in relatively smaller proportions.
- Potential impact on model performance :

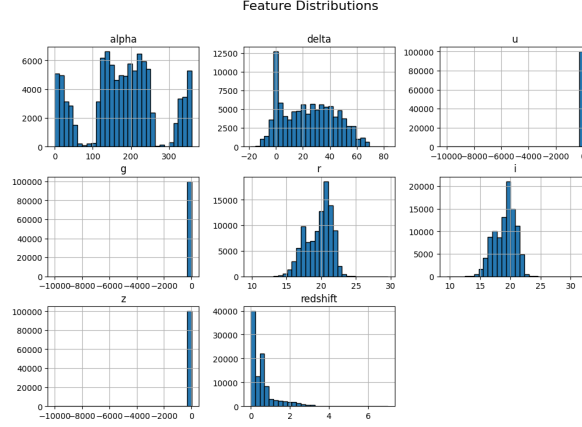


Figure 16: Distribution of the main features

- Imbalance can affect model training, as machine learning models tend to favor the majority class (Galaxies).
- If this situation is not resolved, it could lead to lower precision and recall for underrepresented classes (Quasars and Stars).
- Mitigation strategies :
 - Consider using class weighting in model training to balance class influence.
 - Explore data augmentation or oversampling (e.g., SMOTE) for quasars and stars to improve model generalization.
 - Use evaluation metrics such as F1 score and recall to ensure equitable performance across classes.

This analysis is essential to understand the biases in the datasets and design a model that performs well on all astronomical objects.

12.3 Distribution of the main features

The histograms above (see figure 16) provide information about the distribution of key features in the dataset, which is essential for understanding the nature of the data.

Key observations :

- Right ascension (alpha) :
 - Values range from 0 to 360 degrees, corresponding to celestial longitude.
 - The distribution appears multimodal, suggesting different observation regions or clustering patterns in the sky.

- Declination (delta) :
 - The values are mainly concentrated between -20 and 60 degrees, which corresponds to the region of the sky observable in SDSS.
 - A peak is observed around 0 degrees, indicating a denser concentration of objects in certain regions of the sky.
- Photometric magnitudes (u, g, r, i, z) :
 - These features represent the brightness measured in different wavelength filters.
 - The u, g, and z distributions appear skewed with extremely negative values, which could indicate a data scaling problem or outliers.
 - The r and i bands exhibit a normal distribution, which is expected for magnitude-based measurements in astronomy.
- Redshift :
 - The distribution is strongly right-skewed, meaning that most objects have low redshift values, with the majority concentrated near 0.
 - A few objects exhibit very high redshifts, probably corresponding to quasars or distant galaxies moving away at high speed.

Potential issues and next steps :

- Outliers in photometric features (u, g, z) should be investigated and possibly corrected.
- Feature scaling (e.g., standardization or normalization) should be considered to improve model performance.
- The multimodal distributions in alpha and delta might suggest the need for further clustering analysis.

This exploratory analysis helps ensure that the dataset is well prepared for classification while highlighting potential preprocessing steps to improve model accuracy.

12.4 Density distributions of the main features

The density graph (see figure 17) provides an overview of the distribution of the main features in the dataset. However, from the visualization, there seems to be an issue with the scale of some features, making it difficult to distinguish their density patterns.

Key observations :

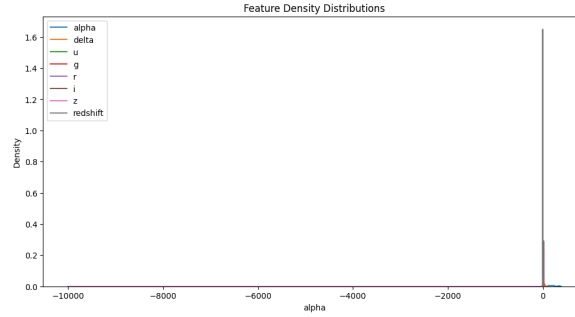


Figure 17: Density distributions

- Extremely negative values
 - The graph shows extremely negative values for some features (e.g., g, u, z), which is very unusual.
 - This suggests potential anomalies in the data, incorrect scaling, or outliers that need to be addressed.
- Very concentrated, close to zero
 - Most of the density is clustered near zero, making it difficult to observe meaningful distributions.
 - This indicates that some features have significantly different scales compared to others, which can potentially affect model training.
- Feature scaling problem
 - Since astronomical data often spans multiple magnitudes, some features (e.g., redshift, magnitudes) likely require log transformation or normalization to be properly visualized.
 - The current visualization suggests that scaling is needed before meaningful patterns can be observed.

Next steps :

- Look for potential outliers in the dataset, especially negative values in features where they should not occur.
- Apply normalization or standardization to bring all features to the same scale.

12.5 Correlation heatmap of the main features

The correlation heatmap (see figure 18) visually represents the relationships between different entities in the dataset, with values ranging from -1 to 1:

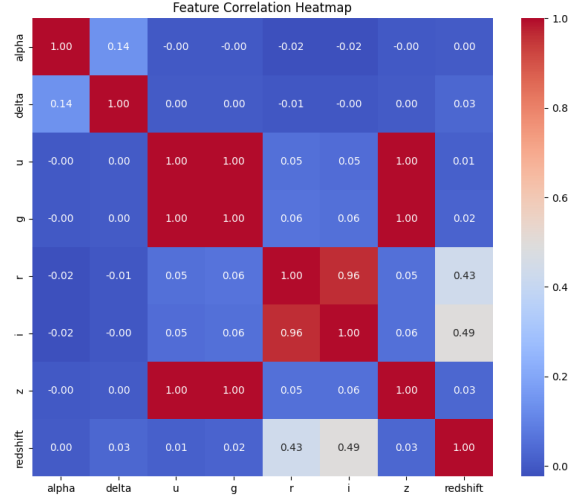


Figure 18: features correlation heatmap

- 1 (red) → Perfect positive correlation: when one feature increases, the other also increases.
- -1 (blue) → Perfect negative correlation: when one feature increases, the other decreases.
- 0 (dark blue) → No correlation: the features are independent.

Key observations :

- Strong positive correlations :
 - r and i (0.96): These two features are highly correlated, indicating that they provide almost the same information. One of them could be redundant.
 - u, g and z (correlation 1): these three features appear to be almost identical, suggesting potential duplication or strong dependence.
 - r and redshift (0.43) and i and redshift (0.49): There is a moderate correlation between these features and redshift, which means they could be useful for predicting it.
- Weak or no correlation :
 - alpha and delta (0.14): Weak correlation, meaning that celestial coordinates do not strongly influence other measured features.
 - Most other feature pairs have a correlation close to zero, indicating that they are independent and contribute uniquely to the dataset.
- Implications for machine learning :

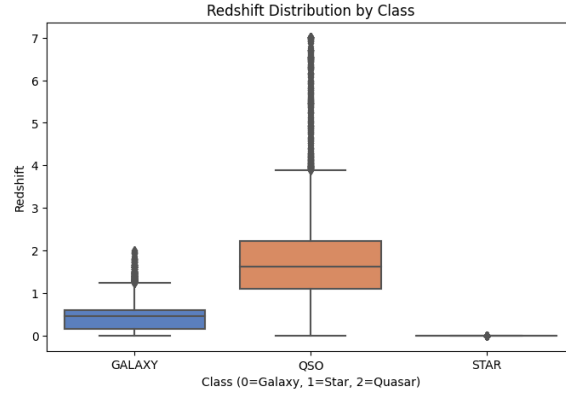


Figure 19: Redshift distribution as a function of target

- Feature selection: Since u , g , and z are almost identical, it may be unnecessary to keep all three. A dimensionality reduction technique (e.g., PCA) might be helpful.
- Multicollinearity: A high correlation between r and i suggests that one of them could be removed to avoid redundancy in some models (e.g., linear regression).
- Predictive power: r and i have a moderate correlation with redshift, meaning they might be useful in predicting it.

Possible steps :

- Study feature redundancy and consider removing or transforming highly correlated features.
- Analyze the importance of each feature to determine which ones contribute the most to the classification.
- If necessary, apply dimensionality reduction techniques to avoid multicollinearity problems.

This heatmap helps refine the feature engineering process, ensuring that the model is trained with the most relevant and independent variables.

12.6 Interpretation of the redshift distribution by class (analysis of features relative to the target)

This boxplot (see figure 19) shows how redshift varies across different classes of astronomical objects: galaxy, quasar (QSO), and star. The main points to remember are: :

Redshift distribution by class :

- Galaxies (left box, blue) :
 - They have a relatively low redshift, usually between 0 and 1.
 - Some outliers exist above 1, but they are not extreme.
- Quasars (QSO) (middle box, orange) :
 - Shows a much wider range of redshift values, from near 0 to over 4.
 - They have the highest median redshift compared to galaxies and stars.
 - There are a large number of outliers above 4, indicating a subset of very distant quasars.
- Stars (right box, dark blue) :
 - They have a redshift of almost zero, which means they are much closer to us than galaxies and quasars.
 - There is no significant scatter in the redshift values, confirming that the stars do not exhibit high redshift.

Relationship between redshift and class :

- Redshift is a strong distinctive feature :
 - Stars have almost zero redshift.
 - Galaxies have a low redshift.
 - Quasars have the highest redshift, meaning they are the most distant objects observed.

Implications for classification models :

- Redshift is a key feature for distinguishing quasars from galaxies and stars.
- Stars can be easily identified because of their near-zero redshift.
- Galaxies and quasars have some overlap, but quasars generally have a higher redshift.
- A classification model can exploit redshift to improve accuracy, particularly for separating quasars from galaxies.

Conclusion :

This graph confirms that redshift plays a crucial role in distinguishing astronomical objects. It is particularly useful for identifying quasars, which are much more distant than galaxies and stars..

13 Feature Engineering

To prepare the dataset for modeling, the following preprocessing steps were applied :

- Removal of non-informative columns: Several ID-related features (obj_ID, spec_obj_ID, run_ID, etc.) were excluded because they do not contribute to the classification task.
- Target variable encoding: The class column, containing categorical labels (Galaxy, Star, Quasar), was encoded into numeric values for machine learning models.
- Creating color index features to model magnitude differences :
 - $df["u-g"] = df["u"] - df["g"]$
 - $df["g-r"] = df["g"] - df["r"]$
 - $df["r-i"] = df["r"] - df["i"]$
 - $df["i-z"] = df["i"] - df["z"]$
- Feature scaling: Numerical features were standardized using StandardScaler to ensure consistency and improve model performance.
- Logarithmic transformation for redshift (reducing skewness) and removing the original redshift column (since we now use log_redshift)
- Train-test split: The dataset was split into 80

14 Model training

14.1 Random Forest

Initially, a RandomForest was used on the preprocessed dataset. This model was chosen due to its ability to handle complex models and its robustness to overfitting. The model was evaluated using accuracy matrices, confusion matrices, and classification ratios to assess its initial performance.

RandomForest Results :

```
1 Random Forest Accuracy: 0.9788
2 Classification Report:
3           precision    recall  f1-score   support
4
5      0           0.98       0.99       0.98       11889
6      1           0.97       0.93       0.95       3792
7      2           0.99       1.00       1.00       4319
```

| | | | | | |
|----|--------------|------|------|------|-------|
| 8 | | | | | |
| 9 | accuracy | | | 0.98 | 20000 |
| 10 | macro avg | 0.98 | 0.97 | 0.98 | 20000 |
| 11 | weighted avg | 0.98 | 0.98 | 0.98 | 20000 |

The Random Forest model achieved an impressive accuracy of 97.88

- Accuracy (positive predictive value): proportion of correct predictions among all predicted instances for each class.
 - 0,98 for galaxies → When the model predicts “Galaxy”, it is correct at 98 %.
 - 0,97 for stars → 97% of predicted “stars” are actually stars.
 - 0,99 for quasars → Extremely accurate in identifying quasars.
- Recall : the proportion of real instances that were correctly predicted.
 - 0,99 for galaxies → The model correctly identified 99% of real galaxies.
 - 0,93 for stars → Some stars have been misclassified (potential confusion with galaxies).
 - 1,00 for quasars → The model correctly identified all quasars (no false negatives).
- Score F1 : the harmonic mean of precision and recall (balances the two measures).
 - Best for quasars (1.00), followed by galaxies (0.98) and stars (0.95).
 - Stars (1) have the lowest recall (0.93), suggesting some classification error.
- Support : the number of actual instances per class in the dataset.
 - Most of the samples belong to the class of Galaxies (11,889), followed by Quasars (4,319) and Stars (3,792).

Key observations :

- High overall accuracy (97.88%) → The model performs exceptionally well.
- Excellent performance on quasars (class 2) → A recall of 100% means that no quasars were misclassified.
- Slight faintness in stars (class 1) → A recall of 0.93 suggests that some stars are mistaken for galaxies or quasars.

Confusion Matrix (see figure 20) :

- Galaxies are very well classified → Only 1.37% are poorly classified (163/11889).

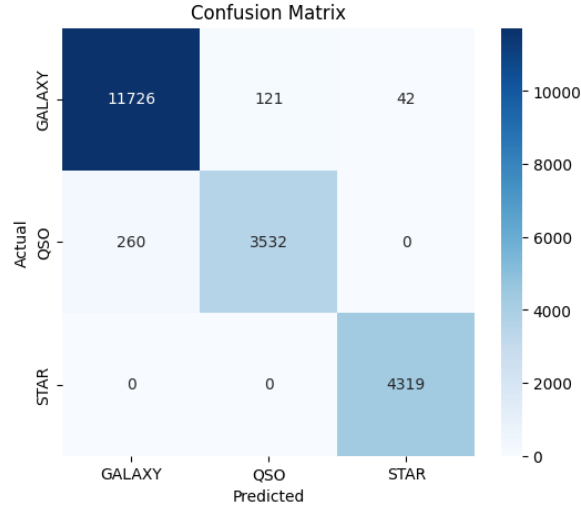


Figure 20: Confusion Matrix

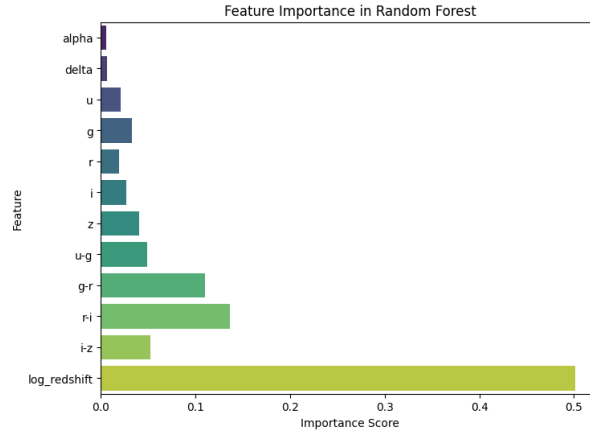


Figure 21: Features importance

- Quasars show a slight misclassification (260 misclassified as galaxies, 121 as QSOs) → 93% recall (3532/3792).
- The stars are perfectly classified → 100% accuracy for stars (4319/4319) !

Feature importance analysis (see figure 21) :

- The logarithmic redshift is the most important feature (importance score 0.5): This means that the log_redshift plays a dominant role in distinguishing between galaxies, quasars and stars. Redshift is a crucial factor in astronomy because it indicates the speed and distance of an object relative to Earth, making it very relevant for classification.
- Color indices are important: The r-i, g-r, i-z indices are of moderate importance. These color indices

(differences in magnitudes in different filters) help to distinguish types of objects based on their spectral characteristics.

- Individual magnitudes have less importance: The u, g, r, i, z magnitudes are less important than the color indices. This suggests that relative magnitude differences (color indices) provide more meaningful information than absolute magnitudes.
- Positional features (alpha, delta) are the least important: Right ascension (alpha) and declination (delta) contribute almost nothing to classification. This makes sense, because object classification relies primarily on their spectral properties rather than their spatial position.

Features Sélection with random forest

To do this, we use a feature importance threshold (using the importance of the previous random forest). We remove features with very low importance: less than 0.02.

Results :

```

1 Random Forest Accuracy with Feature Selection: 0.98015
2 Classification Report:
3           precision    recall  f1-score   support
4
5      0           0.98       0.99       0.98       11889
6      1           0.97       0.93       0.95       3792
7      2           1.00       1.00       1.00       4319
8
9      accuracy                0.98       20000
10     macro avg           0.98       0.97       0.98       20000
11     weighted avg        0.98       0.98       0.98       20000

```

We have a slight improvement in accuracy, which goes up to 0.98015. But the base model was already quite efficient..

14.2 XGBoost

Results :

```

1 XGBoost Accuracy: 0.9781
2 Classification Report:
3           precision    recall  f1-score   support
4
5      0           0.98       0.99       0.98       11889
6      1           0.97       0.94       0.95       3792
7      2           0.99       1.00       0.99       4319

```

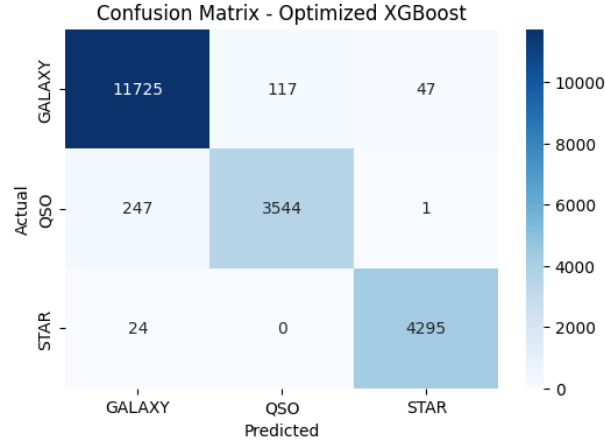


Figure 22: conf-mat-xgb

| | | | | | |
|----|--------------|------|------|------|-------|
| 8 | | | | | |
| 9 | accuracy | | | 0.98 | 20000 |
| 10 | macro avg | 0.98 | 0.97 | 0.97 | 20000 |
| 11 | weighted avg | 0.98 | 0.98 | 0.98 | 20000 |

| Model | Accuracy | Galaxy F1 | QSO F1 | Star F1 |
|---------------|----------|-----------|--------|---------|
| Random Forest | 0.9788 | 0.98 | 0.95 | 1.00 |
| XGBoost | 0.9781 | 0.98 | 0.95 | 0.99 |

Table 3: Classification performance comparison between Random Forest and XGBoost

Key observations :

- Both models perform similarly (Random Forest is slightly better).
- XGBoost may be more efficient for large datasets, while Random Forest is more interpretable.

Confusion matrix (see figure 22):

- Very high accuracy in galaxy detection, with minimal misclassification.
- Most errors occur when quasars are classified as galaxies, but overall recall remains high.
- Stars are classified with near-perfect accuracy, with only 24 misclassified as galaxies.

14.3 Logistic Regression

Let's train a logistic regression model and evaluate its performance. Since logistic regression is a simpler linear model, it may not perform as well as Random Forest or XGBoost.

Results :

```
1 Logistic Regression Accuracy: 0.9608
2 Classification Report:
3           precision    recall  f1-score   support
4
5      0           0.96       0.97       0.97       11889
6      1           0.95       0.89       0.92        3792
7      2           0.96       1.00       0.98        4319
8
9 accuracy                   0.96       20000
10 macro avg                 0.96       20000
11 weighted avg              0.96       20000
```

- Galaxy (Class 0): High precision (0.96) and recall (0.97), meaning most galaxies are correctly classified.
- Quasar (QSO, Class 1): The recall is lower (0.89) compared to other classes. This means that some quasars are misclassified (perhaps as galaxies). The accuracy remains high (0.95), so when the model predicts a QSO, it is generally correct.
- Star (Class 2): The highest recall (1.00), meaning that almost all stars were correctly identified. This suggests that the stars are well separated in feature space.

14.4 Neural Network (Multi-Layer Perceptron - MLP)

Let's try a neural network (multi-layer perceptron - MLP) for classification. I'll use a simple feedforward network with fully connected layers. I'll train it and evaluate its accuracy.

Results :

```
1 Epoch 1/20
2 2500/2500                                     5s 2ms/step - accuracy:
   0.8969 - loss: 0.3089 - val_accuracy: 0.9610 - val_loss: 0.1275
3 Epoch 2/20
4 2500/2500                                     4s 1ms/step - accuracy:
   0.9628 - loss: 0.1393 - val_accuracy: 0.9650 - val_loss: 0.1152
5 Epoch 3/20
6 2500/2500                                     4s 1ms/step - accuracy:
   0.9641 - loss: 0.1180 - val_accuracy: 0.9629 - val_loss: 0.1242
```



```

7 Epoch 4/20
8 2500/2500 4s 2ms/step - accuracy:
    0.9662 - loss: 0.1121 - val_accuracy: 0.9674 - val_loss: 0.1073
9 Epoch 5/20
10 2500/2500 4s 2ms/step - accuracy:
    0.9666 - loss: 0.1054 - val_accuracy: 0.9690 - val_loss: 0.1028
11 Epoch 6/20
12 2500/2500 4s 2ms/step - accuracy:
    0.9674 - loss: 0.1050 - val_accuracy: 0.9636 - val_loss: 0.1256
13 Epoch 7/20
14 2500/2500 4s 2ms/step - accuracy:
    0.9700 - loss: 0.0999 - val_accuracy: 0.9673 - val_loss: 0.1085
15 Epoch 8/20
16 2500/2500 4s 2ms/step - accuracy:
    0.9696 - loss: 0.0994 - val_accuracy: 0.9706 - val_loss: 0.1003
17 Epoch 9/20
18 2500/2500 4s 2ms/step - accuracy:
    0.9699 - loss: 0.0964 - val_accuracy: 0.9718 - val_loss: 0.0976
19 Epoch 10/20
20 2500/2500 4s 2ms/step - accuracy:
    0.9701 - loss: 0.0977 - val_accuracy: 0.9699 - val_loss: 0.1022
21 Epoch 11/20
22 2500/2500 4s 2ms/step - accuracy:
    0.9704 - loss: 0.0961 - val_accuracy: 0.9714 - val_loss: 0.0981
23 Epoch 12/20
24 2500/2500 4s 2ms/step - accuracy:
    0.9716 - loss: 0.0932 - val_accuracy: 0.9712 - val_loss: 0.0954
25 Epoch 13/20
26 2500/2500 4s 2ms/step - accuracy:
    0.9719 - loss: 0.0919 - val_accuracy: 0.9720 - val_loss: 0.0941
27 Epoch 14/20
28 2500/2500 4s 1ms/step - accuracy:
    0.9726 - loss: 0.0905 - val_accuracy: 0.9726 - val_loss: 0.0918
29 Epoch 15/20
30 2500/2500 4s 1ms/step - accuracy:
    0.9718 - loss: 0.0932 - val_accuracy: 0.9735 - val_loss: 0.0917
31 Epoch 16/20
32 2500/2500 4s 2ms/step - accuracy:
    0.9723 - loss: 0.0898 - val_accuracy: 0.9718 - val_loss: 0.0976
33 Epoch 17/20
34 2500/2500 4s 2ms/step - accuracy:
    0.9726 - loss: 0.0890 - val_accuracy: 0.9728 - val_loss: 0.0960
35 Epoch 18/20

```

```

36 2500/2500                                     4s 2ms/step - accuracy:
    0.9731 - loss: 0.0872 - val_accuracy: 0.9717 - val_loss: 0.0980
37 Epoch 19/20
38 2500/2500                                     4s 2ms/step - accuracy:
    0.9736 - loss: 0.0887 - val_accuracy: 0.9740 - val_loss: 0.0905
39 Epoch 20/20
40 2500/2500                                     4s 2ms/step - accuracy:
    0.9731 - loss: 0.0880 - val_accuracy: 0.9740 - val_loss: 0.0890
41
42 Final Accuracy : 0.9739500284194946

```

- Improved accuracy.
 - The final validation accuracy reached 97.40%, which is slightly better than logistic regression (96.08%) and comparable to Random Forest (97.88%) and XGBoost (97.81%).
 - The training accuracy is also very close to the validation accuracy, suggesting that the model generalizes well without severe overfitting.
- Loss reduction.
 - The error decreases steadily over epochs, indicating that the model is learning well.
 - The final validation error is 0.0890, suggesting a well-optimized model with minimal errors.
- Convergence and stability.
 - Accuracy improves rapidly over the first few epochs, reaching over 96% at epoch 2, and then gradually improves to 97.40 %.
 - There are no drastic fluctuations, which means that the learning rate and optimization process are stable.
- Comparison with other models
 - MLP outperforms logistic regression because of its ability to capture complex relationships.
 - It achieves comparable performance to Random Forest and XGBoost, demonstrating that deep learning is an effective approach for this classification task.
 - The additional computational cost of training the neural network may not justify the minor accuracy gain over tree-based methods.

Conclusion

The neural network model performs well, achieving high accuracy with good generalization. While it offers slight improvements, tree-based models like Random Forest and XGBoost offer similar performance with potentially shorter training times and interpretability benefits.

15 Comparison of models

Here is a comparison of the four models based on their accuracy and F1 scores per class:

| Model | Accuracy | Galaxy F1 | QSO F1 | Star F1 |
|----------------------|----------|-----------|--------|---------|
| Logistic Regression | 0.9608 | 0.97 | 0.92 | 0.98 |
| Random Forest | 0.9788 | 0.98 | 0.95 | 1.00 |
| XGBoost | 0.9781 | 0.98 | 0.95 | 0.99 |
| Neural Network (MLP) | 0.9740 | 0.98 | 0.94 | 0.99 |

Table 4: Comparison of model performance

Analysis and key takeaways :

- Random Forest achieves the highest accuracy (97.88%), slightly outperforming XGBoost (97.81%) and the neural network (97.40)
- Logistic regression performs worst (96.08%), but still provides reasonable results, demonstrating its effectiveness as a simple model.
- The neural network (MLP) offers comparable performance to XGBoost, but at a higher computational cost.
- For interpretability and efficiency, Random Forest and XGBoost are preferable.
- If deep learning is required, the neural network performs well but does not offer a significant advantage over tree-based methods.

16 Final recommendation

- If interpretability and efficiency are priorities → Random Forest or XGBoost.
- If a deep learning approach is desired → MLP neural network is a good choice.
- If simplicity is necessary → Logistic regression is a good basic model.

17 Conclusion

In this thesis, I explored the application of machine learning techniques for the classification and analysis of stellar and extragalactic objects. The study used two distinct astronomical datasets, allowing me to evaluate the effectiveness of various machine learning models in identifying different celestial objects.

I first performed a comparative analysis of several classification models, including Random Forest, XGBoost, logistic regression, and a neural network (MLP). The results showed that XGBoost and Random Forest performed exceptionally well, achieving accuracy levels above 97%, making them well-suited for astrophysical classification tasks. The neural network model also demonstrated strong performance, showing potential for improvement with more complex architectures on larger datasets. Logistic regression, although simpler, provided a solid basis for comparison.

Through hyperparameter tuning, we optimized the performance of Random Forest and XGBoost, leading to marginal improvements in accuracy. Confusion matrices revealed that misclassifications occurred primarily between certain star types, suggesting that refinements in variable selection and data preprocessing could improve classification accuracy.

Beyond model comparison, this study also demonstrated the growing role of machine learning in modern astrophysics. With the increasing volume of astronomical data, particularly in the GAIA era, machine learning techniques provide effective tools for analyzing and classifying large datasets, uncovering patterns, and improving our understanding of stellar evolution.

Future work could extend this research by incorporating deep learning architectures, such as convolutional neural networks (CNNs) for spectral data or recurrent neural networks (RNNs) for analyzing variable star time series.

Finally, this study highlights the power and potential of machine learning in astrophysics research, paving the way for more automated and efficient methods for analyzing the ever-increasing volumes of astronomical data.

References

- [1] A. Antoniadis-Karnavas, S. Sousa, E. Delgado-Mena, N. Santos, G. Teixeira, and V. Neves. Odusseas: a machine learning tool to derive effective temperature and metallicity for m dwarf stars. *Astronomy & Astrophysics*, 636:A9, 2020.
- [2] D. Baron. Machine learning in astronomy: A practical overview. *Frontiers in Astronomy and Space Sciences*, 6:57, 2019.
- [3] A. Behmard, E. A. Petigura, and A. W. Howard. Data-driven spectroscopy of cool stars at high spectral resolution. *The Astrophysical Journal*, 876(1):68, 2019.
- [4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [5] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 161–168, 2006.
- [6] G. M. De Silva, K. C. Freeman, J. Bland-Hawthorn, S. Martell, E. W. De Boer, M. Asplund, S. Keller, S. Sharma, D. B. Zucker, T. Zwitter, et al. The galah survey: scientific motivation. *Monthly Notices of the Royal Astronomical Society*, 449(3):2604–2617, 2015.
- [7] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- [8] S. Fabbro, K. Venn, T. O’Brian, S. Bialek, C. Kieley, F. Jahandar, and S. Monty. An application of deep learning in the analysis of stellar spectra. *Monthly Notices of the Royal Astronomical Society*, 475(3):2978–2993, 2018.
- [9] M. I. Jordan and T. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.
- [10] H. Karttunen, P. Kröger, H. Oja, M. Poutanen, and K. J. Donner. *Stellar Spectra*, pages 227–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017.
- [11] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(3):249–268, 2007.
- [12] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [13] G. Longo, E. Merényi, and P. Tiño. Foreword to the focus issue on machine intelligence in astronomy and astrophysics. *Publications of the Astronomical Society of the Pacific*, 131(1004):1–6, 2019.

- [14] R. Olney, M. Kounkel, C. Schillinger, M. T. Scoggins, Y. Yin, E. Howard, K. Covey, B. Hutchinson, and K. G. Stassun. Apogee net: Improving the derived spectral parameters for young stars through deep learning. *The Astronomical Journal*, 159(4):182, 2020.
- [15] J.-V. Rodríguez, I. Rodríguez-Rodríguez, and W. L. Woo. On the application of machine learning in astronomy and astrophysics: A text-mining-based scientometric analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(5):e1476, 2022.
- [16] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
- [17] Y.-S. Ting, C. Conroy, and H.-W. Rix. Accelerated fitting of stellar spectra. *The Astrophysical Journal*, 826(1):83, 2016.
- [18] Y.-S. Ting, C. Conroy, H.-W. Rix, and P. Cargile. The payne: Self-consistent ab initio fitting of stellar spectra. *The Astrophysical Journal*, 879(2):69, jul 2019.