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

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 astonomical 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 sprectra

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/Lo) - Brightness relative to the sun
- Radius (R/Ro) - Radius relative to the sun
- Absolute magnitude (Mv) - 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.

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

Sortie :

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

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

Output ;

```
Temperature (K)           0
Luminosity(L/Lo)          0
Radius(R/Ro)              0
Absolute magnitude(Mv)    0
Star type                 0
Star color                0
Spectral Class            0
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.

```
# Check for missing values
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.

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

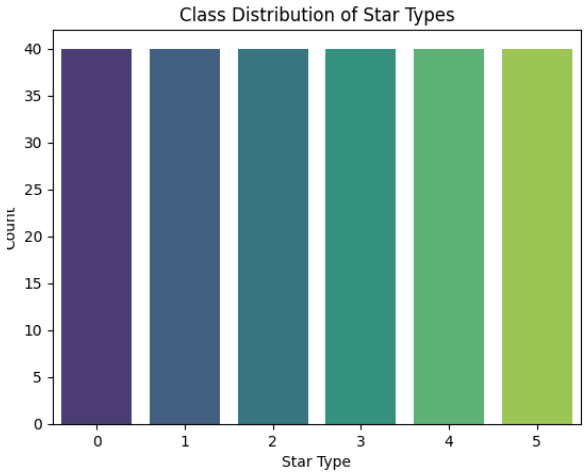

Figure 1: Classes Distribution

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.

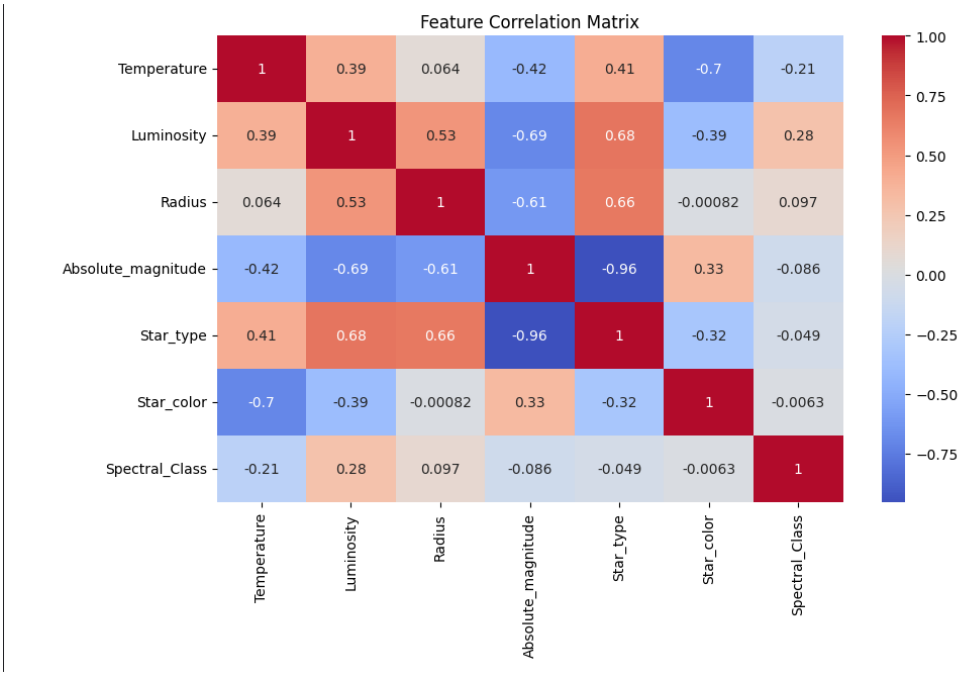

Figure 2: Correlation Matrix

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.

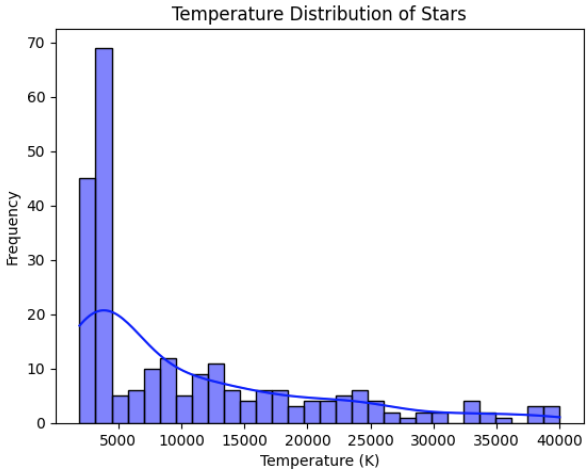

Figure 3: Distribution of temperatures

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.

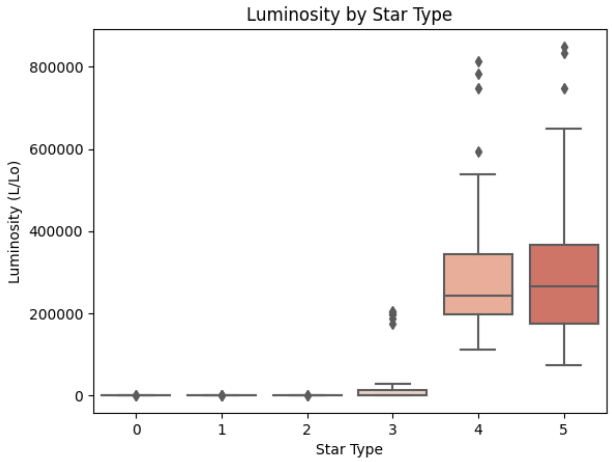

Figure 4: Luminosity vs. Star Type

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/Lo) 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

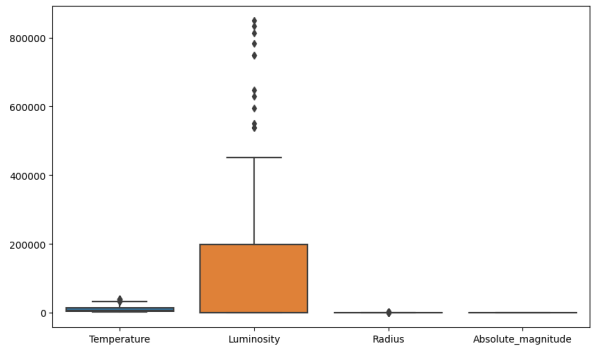

Figure 5: Outliers for the measures

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 :

```
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Apply outlier removal on Luminosity (or any other feature if needed)
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) :

```
# Log transform Luminosity
df['Luminosity_log'] = np.log1p(df['Luminosity'])
df_no_outliers['Luminosity_log'] = np.log1p(df_no_outliers['Luminosity'])

# Select numerical features for scaling
features = ["Temperature", "Luminosity_log", "Radius", "Absolute_magnitude"]
scaler = RobustScaler()

df[features] = scaler.fit_transform(df[features])
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 :

```
# Define target variable & features
X = df[features]
y = df["Star_type"]

X_no_outliers = df_no_outliers[features]
y_no_outliers = df_no_outliers["Star_type"]

# Split datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
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)

# Train Random Forest Model
model = RandomForestClassifier(random_state=42)
model_no_outliers = RandomForestClassifier(random_state=42)

model.fit(X_train, y_train)
model_no_outliers.fit(X_train_no, y_train_no)

# Predictions
y_pred = model.predict(X_test)
y_pred_no_outliers = model_no_outliers.predict(X_test_no)

# Evaluate Performance
print("Model with Outliers:")
print(classification_report(y_test, y_pred))

print("\nModel without Outliers:")
print(classification_report(y_test_no, y_pred_no_outliers))
```

Results of Random Forest :

```
Model with Outliers:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         7
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00         8
           4       1.00      1.00      1.00         8
           5       1.00      1.00      1.00        11

    accuracy                           1.00        48
   macro avg       1.00      1.00      1.00        48
weighted avg       1.00      1.00      1.00        48


Model without Outliers:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00         8
           2       1.00      1.00      1.00         9
           3       1.00      1.00      1.00         7
           4       1.00      1.00      1.00         6
           5       1.00      1.00      1.00         5

    accuracy                           1.00        46
   macro avg       1.00      1.00      1.00        46
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 :

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Define target variable & features
X = df[features]
y = df["Star_type"]

X_no_outliers = df_no_outliers[features]
y_no_outliers = df_no_outliers["Star_type"]

# Split datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
    stratify=y)
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, stratify=y_no_outliers)

# Train Random Forest Model
model_log = LogisticRegression(max_iter=1000, multi_class="multinomial", solver="lbfgs")
model_no_outliers_log = LogisticRegression(max_iter=1000, multi_class="multinomial", solver="
    lbfgs")

model_log.fit(X_train, y_train)
model_no_outliers_log.fit(X_train_no, y_train_no)

# Predictions
y_pred = model_log.predict(X_test)
y_pred_no_outliers = model_no_outliers_log.predict(X_test_no)

```

```
# Evaluate Performance
print("Model with Outliers:")
print("Accuracy:", accuracy_score(y_test, y_pred))

print(classification_report(y_test, y_pred))

print("\nModel without Outliers:")
print("Accuracy:", accuracy_score(y_test_no, y_pred_no_outliers))

print(classification_report(y_test_no, y_pred_no_outliers))
```

10.11.1 Results of Logistic Regression :

```
Model with Outliers:
Accuracy: 0.8958333333333334
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         8
           1       0.86      0.75      0.80         8
           2       1.00      1.00      1.00         8
           3       0.86      0.75      0.80         8
           4       0.88      0.88      0.88         8
           5       1.00      1.00      1.00         8

    accuracy                           0.90        48
   macro avg       0.90      0.90      0.89        48
weighted avg       0.90      0.90      0.89        48


Model without Outliers:
Accuracy: 0.9347826086956522
              precision    recall  f1-score   support

           0       0.89      1.00      0.94         8
           1       0.88      0.88      0.88         8
           2       1.00      1.00      1.00         8
           3       0.88      0.88      0.88         8
           4       1.00      0.86      0.92         7
           5       1.00      1.00      1.00         7

    accuracy                           0.93        46
   macro avg       0.94      0.93      0.94        46
weighted avg       0.94      0.93      0.93        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.

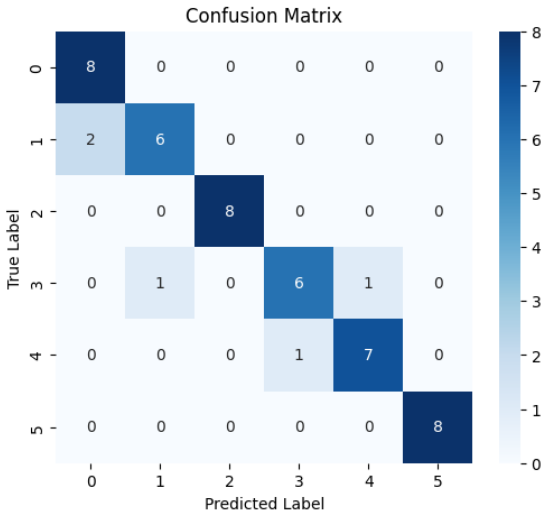

Figure 6: Confusion Matrix

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.

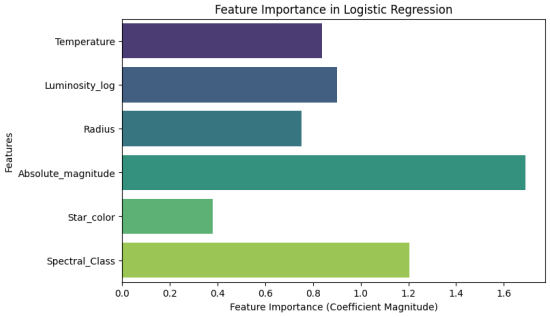

Figure 7: Features importance logistic regression with outliers

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.

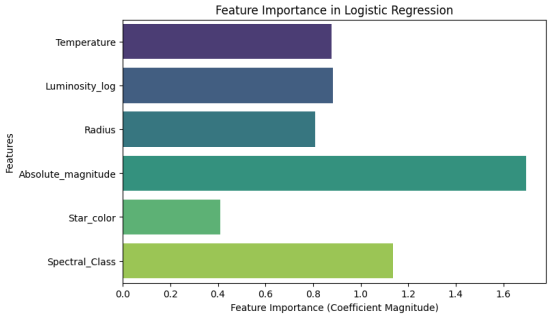

Figure 8: Features importance logistic regression without outliers

  - 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.

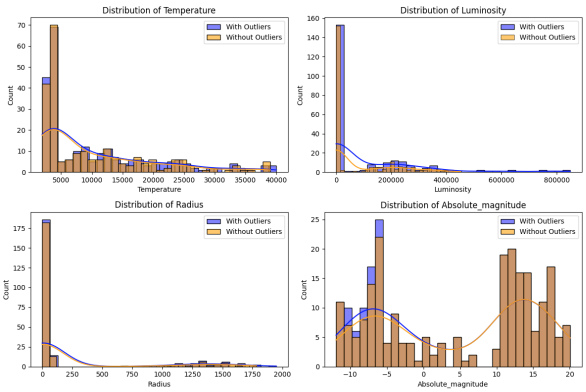

Figure 9: Overall distributions

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 Absolue 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.

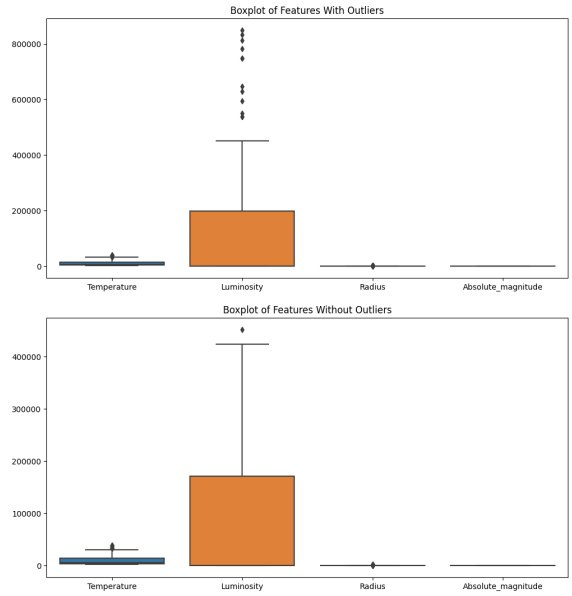

Figure 10: check the evolution of distributions

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.

Statistical sumary about the variables

With outliers :

```
[21]: df.describe()
```

	Temperature	Luminosity	Radius	Absolute_magnitude	Star_type	Star_color	Spectral_Class
count	240.000000	240.000000	240.000000	240.000000	240.000000	240.000000	240.000000
mean	10497.462500	107188.361635	237.157781	4.382396	2.500000	7.250000	3.758333
std	9552.425037	179432.244940	517.155763	10.532512	1.711394	4.926659	2.090007
min	1939.000000	0.000080	0.008400	-11.920000	0.000000	0.000000	0.000000
25%	3344.250000	0.000865	0.102750	-6.232500	1.000000	2.000000	1.000000
50%	5776.000000	0.070500	0.762500	8.313000	2.500000	10.000000	5.000000
75%	15055.500000	198050.000000	42.750000	13.697500	4.000000	10.000000	5.000000
max	40000.000000	849420.000000	1948.500000	20.060000	5.000000	18.000000	6.000000

Without outliers :

```
[22]: df_no_outliers.describe()
```

	Temperature	Luminosity	Radius	Absolute_magnitude	Star_type	Star_color	Spectral_Class
count	228.000000	228.000000	228.000000	228.000000	228.000000	228.000000	228.000000
mean	10015.592105	76535.885054	204.891963	5.041763	2.390351	7.469298	3.723684
std	9191.797309	119703.515899	484.511133	10.387936	1.682062	4.879511	2.075228
min	1939.000000	0.000080	0.008400	-11.920000	0.000000	0.000000	0.000000
25%	3336.000000	0.000802	0.098525	-5.992500	1.000000	2.000000	1.000000
50%	5046.000000	0.010500	0.501000	10.635000	2.000000	10.000000	5.000000
75%	14136.250000	170950.000000	31.250000	13.990000	4.000000	10.000000	5.000000
max	39000.000000	452000.000000	1783.000000	20.060000	5.000000	18.000000	6.000000

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:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Load dataset
df = pd.read_csv("6 class csv.csv")

# Display first few rows
display(df.head())

# Basic info and missing values
print(df.info())
```

```
print(df.isnull().sum())

# Encoding categorical variables
le_color = LabelEncoder()
df['Star color'] = le_color.fit_transform(df['Star color'])

le_spectral = LabelEncoder()
df['Spectral Class'] = le_spectral.fit_transform(df['Spectral Class'])

# Correlation matrix
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Matrix')
plt.show()

# Binary classification: Selecting two classes
df_binary = df[df['Star type'].isin([2, 3])]  # Example: White Dwarf vs. Main Sequence
X = df_binary.drop(columns=['Star type'])
y = df_binary['Star type']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train classifier (Random Forest as example)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Feature importance
importances = pd.Series(clf.feature_importances_, index=df_binary.drop(columns=['Star type']).
    columns)
importances.sort_values().plot(kind='barh', title='Feature Importances')
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).

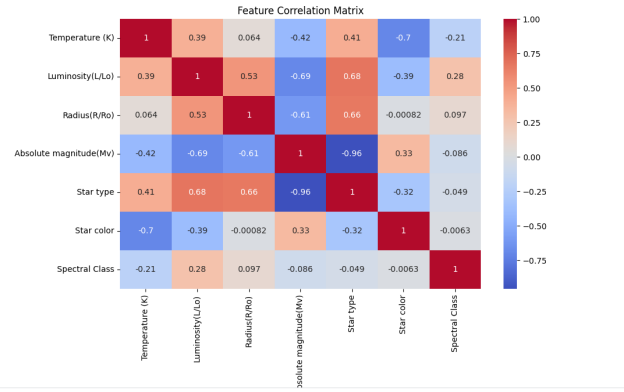

Figure 12: Binary correlation matrix

```
Accuracy: 1.0
[[8 0]
 [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      1.00      1.00        16
weighted avg       1.00      1.00      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 :

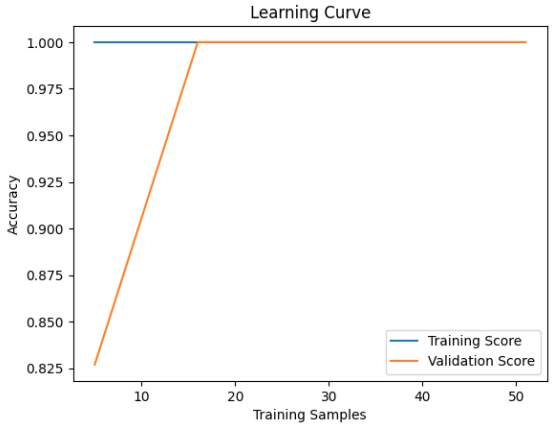

Figure 14: learning curve

- 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 :

```
Cross-Validation Scores: [1. 1. 1. 1. 1.]
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)

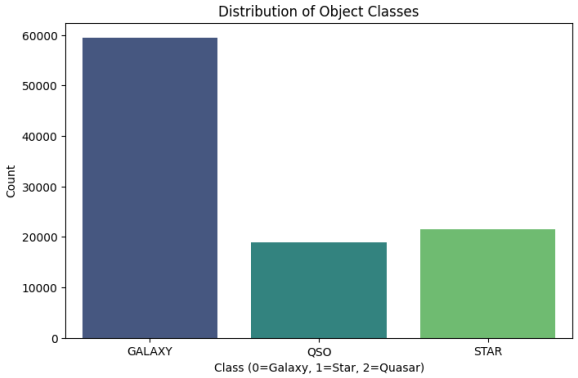

Figure 15: Distribution des classes

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 Visualisation de la distribution des classes

Diagramme à barres (voir figure 15) du nombre d'objets (étoiles, galaxies, quasars), afin de vérifier si les classes sont équilibrées.

Le graphique à barres illustre la distribution des classes d'objets dans l'ensemble de données, qui se compose de trois catégories : galaxies, étoiles et quasars (QSO).

Observations clés:

- Déséquilibre de classe :
  - Les galaxies sont la catégorie la plus courante, avec un nombre significativement plus élevé par rapport aux deux autres classes.
  - Les étoiles et les quasars (QSO) apparaissent dans des proportions relativement plus petites.
- Impact potentiel sur les performances du modèle :

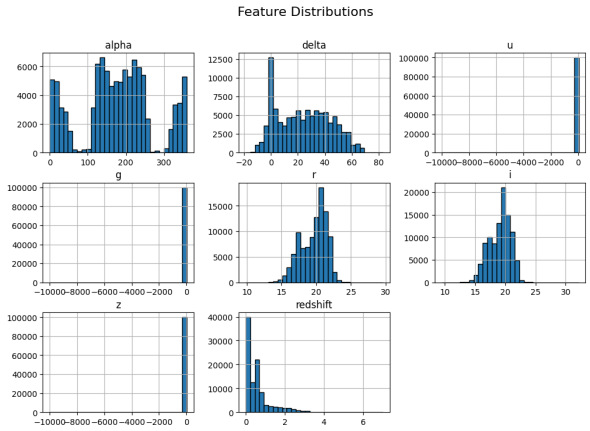

Figure 16: Distributions des features principales

  - Le déséquilibre peut affecter la formation du modèle, car les modèles de machine learning ont tendance à favoriser la classe majoritaire (Galaxies).
  - Si cette situation n'est pas résolue, cela pourrait conduire à une précision et à un rappel inférieurs pour les classes sous-représentées (Quasars et Étoiles).

- Stratégies d'atténuation :
  - Envisager d'utiliser la pondération des classes dans la formation du modèle pour équilibrer l'influence des classes.
  - Explorer l'augmentation des données ou le suréchantillonnage (par exemple, SMOTE) pour les quasars et les étoiles afin d'améliorer la généralisation du modèle.
  - Utilisez des mesures d'évaluation telles que le score F1 et le recall pour garantir des performances équitables dans toutes les classes.

Cette analyse est essentielle pour comprendre les biais des ensembles de données et concevoir un modèle qui fonctionne bien sur tous les objets astronomiques.

12.3 Distributions des features principales

Les histogrammes ci-dessus (voir figure 16) fournissent des informations sur la distribution des features clés dans l'ensemble de données, qui sont essentielles pour comprendre la nature des données.

Observations clés :

- Ascension droite (alpha) :
  - Les valeurs varient de 0 à 360 degrés, correspondant à la longitude céleste.

  - La distribution semble multimodale, suggérant différentes régions d'observation ou des modèles de regroupement dans le ciel.

- Déclinaison (delta) :
  - Les valeurs sont principalement concentrées entre -20 et 60 degrés, ce qui correspond à la région du ciel observable dans SDSS.
  - On observe un pic autour de 0 degré, indiquant une concentration plus dense d'objets dans certaines régions du ciel.

- Magnitudes photométriques (u, g, r, i, z) :
  - Ces features représentent la luminosité mesurée dans différents filtres de longueur d'onde.
  - Les distributions u, g et z semblent biaisées avec des valeurs extrêmement négatives, ce qui pourrait indiquer un problème de mise à l'échelle des données ou des valeurs aberrantes.
  - Les bandes r et i présentent une distribution normale, ce qui est attendu pour les mesures basées sur la magnitude en astronomie.

- Décalage vers le rouge (redshift) :
  - La distribution est fortement asymétrique vers la droite, ce qui signifie que la plupart des objets ont de faibles valeurs de décalage vers le rouge, avec une majorité concentrée près de 0.
  - Quelques objets présentent des décalages vers le rouge très élevés, correspondant probablement à des quasars ou à des galaxies lointaines s'éloignant à grande vitesse.

Problèmes potentiels et prochaines étapes :

- Les valeurs aberrantes dans les features photométriques (u, g, z) doivent être étudiées et éventuellement corrigées.
- La mise à l'échelle (scaling) des features (par exemple, la standardisation ou la normalisation) doit être envisagée pour améliorer les performances du modèle.
- Les distributions multimodales en alpha et delta pourraient suggèrer la nécessité d'une analyse de clustering plus poussée.

Cette analyse exploratoire permet de garantir que l'ensemble de données est bien préparé pour la classification tout en mettant en évidence les étapes de prétraitement potentielles pour améliorer la précision du modèle.

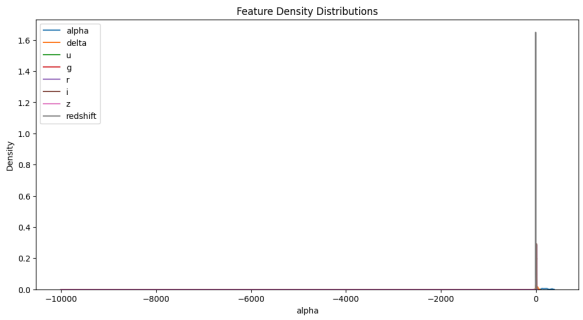

Figure 17: Distributions de densité

12.4 Distributions de densité des features principales

Le graphique de densité (voir figure 17) fournit un aperçu de la distribution des principales features de l'ensemble de données. Cependant, d'après la visualisation, il semble y avoir un problème avec l'échelle de certaines features, ce qui rend difficile la distinction de leurs modèles de densité.

Observations clés :

- Valeurs extrêmement négatives
  - Le graphique montre des valeurs extrêmement négatives pour certaines features (par exemple, g, u, z), ce qui est très inhabituel.
  - Cela suggère des anomalies potentielles dans les données, une mise à l'échelle incorrecte ou des valeurs aberrantes qui doivent être traitées.
- Très concentré, proche de zéro
  - La majeure partie de la densité est regroupée près de zéro, ce qui rend difficile l'observation de distributions significatives.
  - Cela indique que certaines features ont des échelles sensiblement différentes par rapport à d'autres, ce qui peut potentiellement affecter la formation du modèle.
- Problème de mise à l'échelle des features
  - Étant donné que les données astronomiques couvrent souvent plusieurs magnitudes, certaines features (par exemple, le décalage vers le rouge, les magnitudes) nécessitent probablement une transformation logarithmique ou une normalisation pour être correctement visualisées.
  - La visualisation actuelle suggère qu'une mise à l'échelle est nécessaire avant que des modèles significatifs puissent être observés.

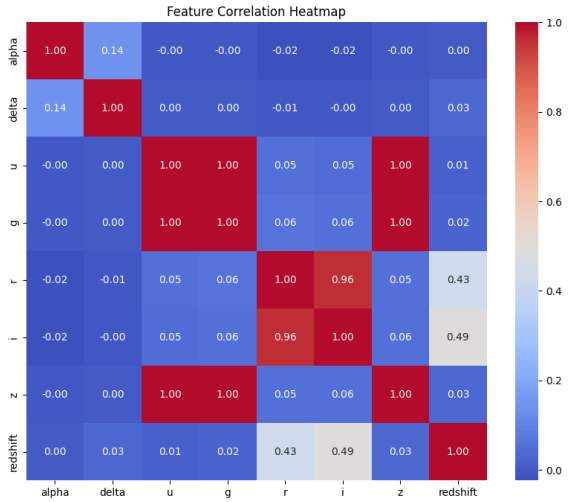

Figure 18: features correlation heatmap

Prochaines étapes :

- Recherchez les valeurs aberrantes potentielles dans l'ensemble de données, en particulier les valeurs négatives dans les entités où elles ne devraient pas se produire.
- Appliquer la normalisation ou la standardisation pour amener toutes les features à une même échelle.

12.5 Heatmap de corrélation des principales features

La heatmap de corrélation (voir figure 18) représente visuellement les relations entre les différentes entités de l'ensemble de données, avec des valeurs allant de -1 à 1 :

- 1 (rouge) → Corrélation positive parfaite : lorsqu'une feature augmente, l'autre augmente également.
- -1 (bleu) → Corrélation négative parfaite : lorsqu'une feature augmente, l'autre diminue.
- 0 (bleu foncé) → Aucune corrélation : les features sont indépendantes.

Observations clés :

- Fortes corrélations positives :
  - r et i (0,96) : Ces deux features sont fortement corrélées, ce qui indique qu'elles fournissent presque les mêmes informations. L'une d'elles pourrait être redondante.
  - u, g et z (corrélation 1) : ces trois caractéristiques semblent être presque identiques, ce qui suggère une duplication potentielle ou une forte dépendance.

  - r et redshift (0,43) et i et redshift (0,49) : Il existe une corrélation modérée entre ces features et le redshift, ce qui signifie qu'elles pourraient être utiles pour le prédire.

- Corrélation faible ou inexistante :
  - alpha et delta (0,14) : Faible corrélation, ce qui signifie que les coordonnées célestes n'influencent pas fortement les autres features mesurées.
  - La plupart des autres paires de features ont une corrélation proche de zéro, ce qui indique qu'elles sont indépendantes et contribuent de manière unique à l'ensemble de données.

- Implications pour le machine learning :
  - Sélection des features : Étant donné que u, g et z sont presque identiques, il peut être inutile de conserver les trois. Une technique de réduction de dimensionnalité (par exemple, PCA) pourrait s'avérer utile.
  - Multicolinéarité : une forte corrélation entre r et i suggère que l'un d'eux pourrait être supprimé pour éviter la redondance dans certains modèles (par exemple, la régression linéaire).
  - Pouvoir prédictif : r et i ont une corrélation modérée avec redshift, ce qui signifie qu'ils pourraient être utiles pour le prédire.

Etapes envisageables :

- Étudiez la redondance des features et envisagez de supprimer ou de transformer les features hautement corrélées.
- Analysez l'importance de chaque feature pour déterminer celles qui contribuent le plus à la classification.
- Si nécessaire, appliquez des techniques de réduction de dimensionnalité pour éviter les problèmes de multicolinéarité.

Cette heatmap permet d'affiner le processus d'ingénierie des features, en garantissant que le modèle est formé avec les variables les plus pertinentes et les plus indépendantes.

12.6 Interprétation de la distribution du redshift par classe (analyse des features par rapport a la cible)

Ce boxplot (voir figure 19) montre comment le redshift varie selon les différentes classes d'objets astronomiques : galaxie, quasar (QSO) et étoile. Les principaux points à retenir sont les suivants :

Distribution du redshift par classe :

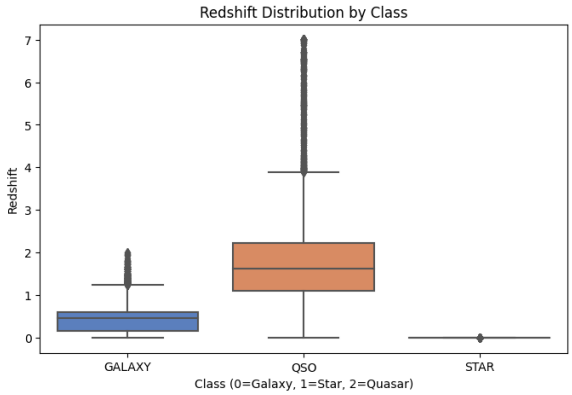

Figure 19: Distribution de redshift en fonction de la target

- Galaxies (case de gauche, bleue) :
  - Ils ont un redshift relativement faible, généralement compris entre 0 et 1.
  - Certaines valeurs aberrantes existent au-delà de 1, mais elles ne sont pas extrêmes.
- Quasars (QSO) (case du milieu, orange) :
  - Affiche une gamme beaucoup plus large de valeurs de redshift, de près de 0 jusqu'à plus de 4.
  - Ils ont un redshift médian le plus élevé par rapport aux galaxies et aux étoiles.
  - Il existe un grand nombre de valeurs aberrantes au-dessus de 4, indiquant un sous-ensemble de quasars très éloignés.
- Étoiles (case de droite, bleu foncé) :
  - Ils ont un redshift presque nul, ce qui signifie qu'ils sont beaucoup plus proches de nous que les galaxies et les quasars.
  - Il n'y a pas de dispersion significative dans les valeurs de redshift, ce qui confirme que les étoiles ne présentent pas de redshift élevé.

Relation entre le redshift et la classe :

- Le redshift est une feature distinctive forte :
  - Les étoiles ont un redshift presque nul.
  - Les galaxies ont un faible redshift.

- Les quasars ont le redshift le plus élevé, ce qui signifie qu'ils sont les objets les plus éloignés observés.

Implications pour les modèles de classification :

- Le redshift est une feature clé pour distinguer les quasars des galaxies et des étoiles.
- Les étoiles peuvent être facilement identifiées en raison de leur redshift proche de zéro.
- Les galaxies et les quasars présentent un certain chevauchement, mais les quasars ont généralement un redshift plus élevé.
- Un modèle de classification peut exploiter le redshift pour améliorer la précision, en particulier pour séparer les quasars des galaxies.

Conclusion :

Ce graphique confirme que le redshift joue un rôle crucial dans la distinction des objets astronomiques. Il est particulièrement utile pour identifier les quasars, qui sont beaucoup plus éloignés que les galaxies et les étoiles.

13 Feature Engineering

Pour préparer l'ensemble de données à la modélisation, les étapes de prétraitement suivantes ont été appliquées :

- Suppression des colonnes non informatives: plusieurs fonctionnalités liées à l'ID (obj_ID, spec_obj_ID, run_ID, etc.) ont été exclues car elles ne contribuent pas à la tâche de classification.
- Encodage de la variable cible : la colonne de classe, contenant des étiquettes catégorielles (Galaxy, Star, Quasar), a été codée en valeurs numériques pour les modèles de machine learning.
- Création de features d'index de couleur pour modéliser les différences de magnitude :
  - 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"]
- Mise à l'échelle des fonctionnalités : les features numériques ont été standardisées à l'aide de StandardScaler pour garantir l'uniformité et améliorer les performances du modèle.

- Transformation logarithmique pour le redshift (réduction de l'asymétrie) et suppression de la colonne redshift d'origine (puisque nous utilisons maintenant log_redshift)
- Répartition train-test : l'ensemble de données a été divisé en 80% de training set et 20% de test set, garantissant une représentation équilibrée des classes.

14 Entraînement de modèles

14.1 Random Forest

Dans un premier temps, une RandomForest a été utilisée sur l'ensemble de données prétraitées. Ce modèle a été choisi en raison de sa capacité à gérer des modèles complexes et de sa robustesse face à l'overfitting. Le modèle a été évalué à l'aide de matrices de précision, de confusion et de rapports de classification pour évaluer ses performances initiales.

Résultats de la randomForest :

```
Random Forest Accuracy: 0.9788
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.99      0.98     11889
           1       0.97      0.93      0.95      3792
           2       0.99      1.00      1.00      4319

    accuracy                           0.98     20000
   macro avg       0.98      0.97      0.98     20000
weighted avg       0.98      0.98      0.98     20000
```

Le modèle Random Forest a atteint une précision impressionnante de 97,88 %, ce qui signifie qu'il a correctement classé près de 98 % des objets de l'ensemble de tests. Analysons plus en détail les résultats :

- Précision (valeur prédictive positive) : proportion de prédictions correctes parmi toutes les instances prédites pour chaque classe.
  - 0,98 pour les galaxies → Lorsque le modèle prédit « Galaxie », il est correct à 98 %.
  - 0,97 pour les étoiles → 97 % des « étoiles » prédites sont en réalité des étoiles.
  - 0,99 pour les quasars → Extrêmement précis dans l'identification des quasars.
- Recall (sensibilité) : la proportion d'instances réelles qui ont été correctement prédites.

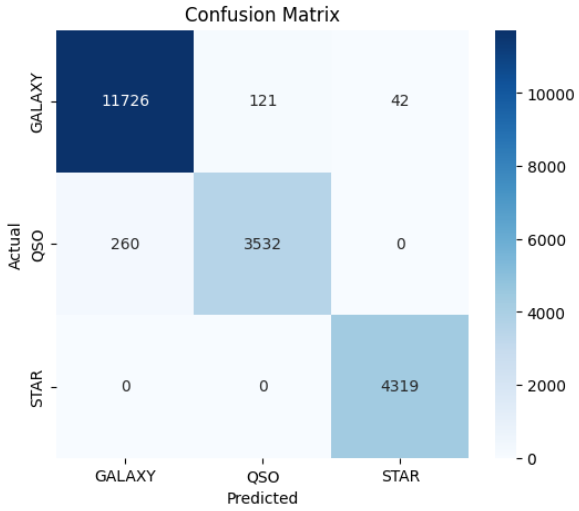

Figure 20: Matrice de confusion

- 0,99 pour les galaxies → Le modèle a identifié correctement 99 % des galaxies réelles.
- 0,93 pour les étoiles → Certaines étoiles ont été mal classées (confusion potentielle avec les galaxies).
- 1,00 pour les quasars → Le modèle a correctement identifié tous les quasars (aucun faux négatif).

- Score F1 : la moyenne harmonique de la précision et du recall (équilibre les deux mesures).
  - Idéal pour les quasars (1,00), suivi des galaxies (0,98) et des étoiles (0,95).
  - Les étoiles (1) ont le rappel le plus faible (0,93), ce qui suggère une certaine erreur de classification.
- Support : le nombre d'instances réelles par classe dans l'ensemble de données.
  - La plupart des échantillons appartiennent à la classe des Galaxies (11 889), suivie des Quasars (4 319) et des Étoiles (3 792).

Observations clés :

- Précision globale élevée (97,88 %) → Le modèle fonctionne exceptionnellement bien.
- Excellentes performances sur les quasars (classe 2) → Un rappel de 100 % signifie qu'aucun quasar n'a été mal classé.
- Légère faiblesse dans les étoiles (classe 1) → Un rappel de 0,93 suggère que certaines étoiles sont confondues avec des galaxies ou des quasars.

Matrice de confusion (figrure 20) :

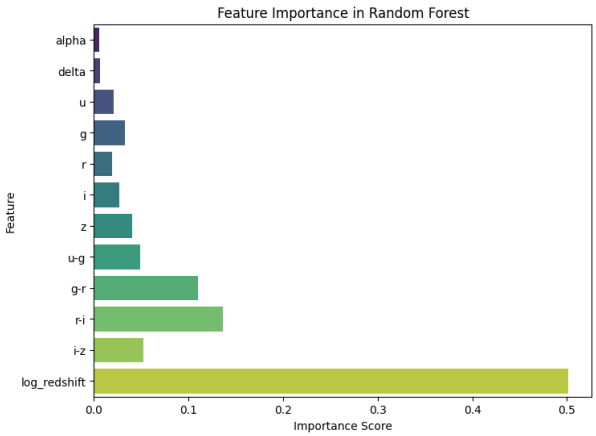

Figure 21: Features importance

- Les galaxies sont très bien classées → Seulement 1,37 % sont mal classées (163/11889).
- Les quasars présentent une légère erreur de classification (260 mal classés comme galaxies, 121 comme QSO) → 93 % de recall (3532/3792).
- Les étoiles sont parfaitement classées → 100 % de précision pour les étoiles (4319/4319) !

Analyse de l'imortance des features (figure 21) :

- Le redshift logarithmique est la caractéristique la plus importante (score d'importance 0,5) : Cela signifie que le log_redshift joue un rôle dominant dans la distinction entre galaxies, quasars et étoiles. Le redshift est un facteur crucial en astronomie, car il indique la vitesse et la distance d'un objet par rapport à la Terre, ce qui le rend très pertinent pour la classification.
- Les indices de couleur sont importants :Les indices r-i, g-r, i-z ont une importance modérée. Ces indices de couleur (différences de magnitudes dans différents filtres) aident à distinguer les types d'objets en fonction de leurs caractéristiques spectrales.
- Les magnitudes individuelles ont une importance plus faible : Les magnitudes u, g, r, i, z sont moins importantes que les indices de couleur. Cela suggère que les différences relatives de magnitude (indices de couleur) fournissent une information plus significative que les magnitudes absolues.
- Les features de position (alpha, delta) sont les moins importantes : L'ascension droite (alpha) et la déclinaison (delta) ne contribuent presque pas à la classification. Cela est logique, car la classification des objets repose principalement sur leurs propriétés spectrales plutôt que sur leur position spatiale.

Features Sélection avec forêt aléatoire

Pour cela, nous utilisons un seuil d'importance des fonctionnalités (à l'aide de l'importance de la random

forest précedentes) Nous supprimons les feature ayant une importance très faible : Inférieure à 0.02.

Résultats :

```
Random Forest Accuracy with Feature Selection: 0.98015
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.99      0.98     11889
           1       0.97      0.93      0.95      3792
           2       1.00      1.00      1.00      4319

    accuracy                           0.98     20000
   macro avg       0.98      0.97      0.98     20000
weighted avg       0.98      0.98      0.98     20000
```

Nous avons une légère amélioration au niveau de la précision qui passe à 0.98015. Mais le modéle de base était déjà assez performant.

14.2 XGBoost

Résultats :

```
XGBoost Accuracy: 0.9781
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.99      0.98     11889
           1       0.97      0.94      0.95      3792
           2       0.99      1.00      0.99      4319

    accuracy                           0.98     20000
   macro avg       0.98      0.97      0.97     20000
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: Comparison de performance de classification entre Random Forest et XGBoost

Observations clés :

- Les deux modèles fonctionnent de manière similaire (Random Forest est légèrement meilleur).

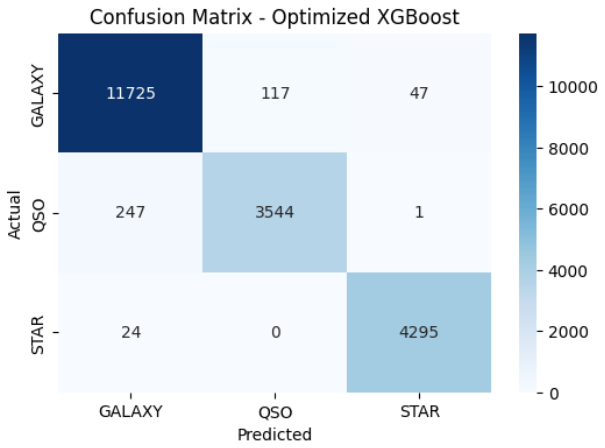

Figure 22: conf-mat-xgb

- XGBoost peut être plus efficace pour les grands ensembles de données, tandis que Random Forest est plus interprétable.

Matrice de confusion (figure 22):

- Très grande précision dans la détection des galaxies, avec une classification erronée minimale.
- La plupart des erreurs se produisent lorsque les quasars sont classés comme des galaxies, mais le recall global reste élevé.
- les étoiles sont classées avec une précision presque parfaite, avec seulement 24 classées à tort comme galaxies.

14.3 Logistic Regression

Entraînons un modèle de régression logistique et évaluons ses performances. Étant donné que la régression logistique est un modèle linéaire plus simple, elle risque de ne pas être aussi performante que Random Forest ou XGBoost.

Résultats :

```
Logistic Regression Accuracy: 0.9608
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.97      0.97     11889
           1       0.95      0.89      0.92      3792
           2       0.96      1.00      0.98      4319

    accuracy                           0.96     20000
   macro avg       0.96      0.95      0.95     20000
weighted avg       0.96      0.96      0.96     20000
```

- Galaxie (Classe 0) :Haute précision (0,96) et rappel (0,97), ce qui signifie que la plupart des galaxies sont correctement classées.
- Quasar (QSO, Classe 1) : Le recall est plus faible (0,89) par rapport aux autres classes. Cela signifie que certains quasars sont mal classés (peut-être comme des galaxies). La précision reste élevée (0,95), donc lorsque le modèle prédit un QSO, il est généralement correct.
- Étoile (Classe 2) : Le recall le plus élevé (1,00), ce qui signifie que presque toutes les étoiles ont été correctement identifiées. Cela suggère que les étoiles sont bien séparées dans l'espace des features.

14.4 Réseau neuronal (Perceptron multicouche - MLP)

Essayons un réseau neuronal (perceptron multicouche - MLP) pour la classification. J'utiliserai un réseau simple à propagation directe avec des couches entièrement connectées. Je l'entraînerai et évaluerai sa précision.

Résulats :

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

```
Epoch 3/20
2500/2500                                                              4s 1ms/step - accuracy:
    0.9641 - loss: 0.1180 - val_accuracy: 0.9629 - val_loss: 0.1242
Epoch 4/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9662 - loss: 0.1121 - val_accuracy: 0.9674 - val_loss: 0.1073
Epoch 5/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9666 - loss: 0.1054 - val_accuracy: 0.9690 - val_loss: 0.1028
Epoch 6/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9674 - loss: 0.1050 - val_accuracy: 0.9636 - val_loss: 0.1256
Epoch 7/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9700 - loss: 0.0999 - val_accuracy: 0.9673 - val_loss: 0.1085
Epoch 8/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9696 - loss: 0.0994 - val_accuracy: 0.9706 - val_loss: 0.1003
Epoch 9/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9699 - loss: 0.0964 - val_accuracy: 0.9718 - val_loss: 0.0976
Epoch 10/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9701 - loss: 0.0977 - val_accuracy: 0.9699 - val_loss: 0.1022
Epoch 11/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9704 - loss: 0.0961 - val_accuracy: 0.9714 - val_loss: 0.0981
Epoch 12/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9716 - loss: 0.0932 - val_accuracy: 0.9712 - val_loss: 0.0954
Epoch 13/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9719 - loss: 0.0919 - val_accuracy: 0.9720 - val_loss: 0.0941
Epoch 14/20
2500/2500                                                              4s 1ms/step - accuracy:
    0.9726 - loss: 0.0905 - val_accuracy: 0.9726 - val_loss: 0.0918
Epoch 15/20
2500/2500                                                              4s 1ms/step - accuracy:
    0.9718 - loss: 0.0932 - val_accuracy: 0.9735 - val_loss: 0.0917
Epoch 16/20
2500/2500                                                              4s 2ms/step - accuracy:
    0.9723 - loss: 0.0898 - val_accuracy: 0.9718 - val_loss: 0.0976
Epoch 17/20
```

```
2500/2500                                                    4s 2ms/step - accuracy:
    0.9726 - loss: 0.0890 - val_accuracy: 0.9728 - val_loss: 0.0960
Epoch 18/20
2500/2500                                                    4s 2ms/step - accuracy:
    0.9731 - loss: 0.0872 - val_accuracy: 0.9717 - val_loss: 0.0980
Epoch 19/20
2500/2500                                                    4s 2ms/step - accuracy:
    0.9736 - loss: 0.0887 - val_accuracy: 0.9740 - val_loss: 0.0905
Epoch 20/20
2500/2500                                                    4s 2ms/step - accuracy:
    0.9731 - loss: 0.0880 - val_accuracy: 0.9740 - val_loss: 0.0890

Final Accuracy : 0.9739500284194946
```

- Amélioration de la précision.
  - La précision de validation finale a atteint 97,40 %, ce qui est légèrement meilleur que la régression logistique (96,08 %) et comparable à Random Forest (97,88 %) et XGBoost (97,81 %).
  - La précision de d'entrainement est également très proche de la précision de la validation, ce qui suggère que le modèle se généralise bien sans overfitting sévère.
- Réduction de l'erreur (Loss).
  - L'erreur diminue constamment au fil des epoch, ce qui indique que le modèle apprend bien.
  - L'erreur de validation finale est de 0,0890, ce qui suggère un modèle bien optimisé avec des erreurs minimales.
- Convergence et stabilité.
  - La précision s'améliore rapidement au cours des premières epoch, atteignant plus de 96 % à l'époque 2, puis s'améliore progressivement jusqu'à 97,40 %.
  - Il n'y a pas de fluctuations drastiques, ce qui signifie que le taux d'apprentissage et le processus d'optimisation sont stables.
- Comparaison avec les autres modèles
  - Le MLP est plus performant que la régression logistique en raison de sa capacité à capturer des relations complexes.
  - Il atteint des performances comparables à Random Forest et XGBoost, démontrant que le deep learning est une approche efficace pour cette tâche de classification.

– Le coût de calcul supplémentaire lié à la formation du réseau neuronal ne justifie peut-être pas le gain de précision mineur par rapport aux méthodes basées sur les arbres.

Conclusion

Le modèle de réseau neuronal fonctionne bien, atteignant une grande précision avec une bonne généralisation. Bien qu'il offre de légères améliorations, les modèles basés sur des arbres comme Random Forest et XGBoost offrent des performances similaires avec des temps de formation potentiellement plus courts et des avantages en termes d'interprétabilité.

15 Comparaison des modèles

Voici une comparaison des quatre modèles en fonction de leur précision et de leurs scores F1 par classe:

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: Comparaison des performances des modèles

Analyse et principaux points à retenir :

- Random Forest atteint la précision la plus élevée (97,88 %), surpassant légèrement XGBoost (97,81 %) et le réseau neuronal (97,40 %).
- La régression logistique obtient les moins bons résultats (96,08 %), mais fournit néanmoins des résultats raisonnables, démontrant son efficacité en tant que modèle simple.
- Le réseau neuronaux (MLP) offre des performances comparables à XGBoost, mais à un coût de calcul plus élevé.
- Pour l'interprétabilité et l'efficacité, Random Forest et XGBoost sont préférables.
- Si du deep learning est nécessaire, le réseau neuronal fonctionne bien mais n'offre pas d'avantage significatif par rapport aux méthodes basées sur les arbres.

16 Recommandation finale

- Si l'interprétabilité et l'efficacité sont des priorités → Random Forest ou XGBoost.
- Si une approche de deep learning est souhaitée → le réseau de neurones MLP est un bon choix.

- Si la simplicité est nécessaire → La régression logistique est un bon modèle de base.

17 Conclusion

Dans cette thèse, j'ai exploré l'application des techniques de machine learning pour la classification et à l'analyse des objets stellaires et extragalactiques. L'étude s'est appuyée sur deux ensembles de données astronomiques distincts, ce qui a permis d'évaluer l'efficacité de divers modèles de machine learning dans l'identification de différents objets célestes.

J'ai d'abord effectué une analyse comparative de plusieurs modèles de classification, dont Random Forest, XGBoost, la régression logistique et un réseau neuronal (MLP). Les résultats ont montré que XGBoost et Random Forest étaient exceptionnellement performants, atteignant des niveaux de précision supérieurs à 97%, ce qui les rend tout à fait adaptés aux tâches de classification astrophysique. Le modèle de réseau neuronal a également fait preuve d'une grande performance, montrant un potentiel d'amélioration avec des architectures plus complexe sur des ensembles de données plus importants. La régression logistique, bien que plus simple, a fourni une base de comparaison solide.

Grâce au réglage des hyperparamètres, nous avons optimisé les performances de Random Forest et de XGBoost, ce qui a conduit à des améliorations marginales de la précision. Les matrices de confusion ont révélé que les erreurs de classification se produisaient principalement entre certains types d'étoiles, ce qui suggère que des perfectionnements dans la sélection des variables et le prétraitement des données pourraient améliorer la précision de la classification.

Au-delà de la comparaison des modèles, cette étude a également démontré le rôle croissant du machine learning dans l'astrophysique moderne. Avec l'augmentation du volume de données astronomiques, en particulier à l'ère de GAIA, les techniques de machine learning fournissent des outils efficaces pour analyser et classer de vastes ensembles de données, découvrir des modèles et améliorer notre compréhension de l'évolution stellaire.

Les travaux futurs pourraient étendre cette recherche en incorporant des architectures de deep learning, telles que les réseaux neuronaux convolutifs (CNN) pour les données spectrales ou les réseaux neuronaux récurrents (RNN) pour l'analyse des séries temporelles d'étoiles variables.

Enfin, cette étude met en évidence la puissance et le potentiel de l'apprentissage automatique dans la recherche astrophysique, ouvrant la voie à des méthodes plus automatisées et plus efficaces pour analyser les volumes toujours croissants de données astronomiques.

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'Briain, S. Bialek, C. Kielty, 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. Tiňo. 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.