MEMO-F524
MASTER THESIS

Automated classification of stars and systems using machine learning

Author :

TALHAOUI Yassin

Section :

Computer science

Promotor :

Defrance Matthieu

April 15, 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 Mise en pratique : Mode opératoire

Dans la suite, je vais mettre en pratique toutes les techniques de machine learning acquise durant mon master en science informatique sur des jeux de données contenant des données liées aux étoiles et objets celestes. L'objectif sera d'utiliser la classification via des modéles de achine learning choisis afin de classer les étoiles/objets celestes en différentes catégories. Pour cela, je vais suivre les étapes suivantes :

- Explorer l'ensemble de données (EDA : vérifier les valeurs manquantes, les distributions, les corrélations)
- Sélection des caractéristiques (matrice de corrélation, classement par importance)
- Classification binaire (commencer par deux classes, par exemple, Main Sequence vs. White Dwarf)
- Classification multi-classes (passer progressivement aux six classes) Tester plusieurs classificateurs (régression logistique, SVM, forêt aléatoire, réseaux neuronaux, etc.
- Évaluer les performances (précision, exactitude, rappel, score F1, matrice de confusion)
- Rédiger des explications (justifier les choix, comparer les résultats)

10 Ensemble de données stellaires pour prédire les types d'étoiles

L'objectif de ce projet est de construire un modèle de machine learning capable de prédire le type d'une étoile sur la base de ses propriétés physiques, telles que la température, la luminosité, le rayon, la magnitude absolue, la couleur et la classe spectrale. Cette tâche de classification suit le diagramme de Hertzsprung-Russell (HR), un outil fondamental en astrophysique qui catégorise les étoiles en fonction de leur température et de leur luminosité.

Nous commençons par un problème de classification binaire, en sélectionnant deux types d'étoiles distincts pour faciliter l'entraînement et l'évaluation des modèles. Ensuite, nous l'étendons progressivement à la classification multi-classes avec les six types d'étoiles.

10.1 À propos du jeu de données

Ce jeu de données nous vient de Kaggle et comprend 240 étoiles, classées en six types d'étoiles différents:

- 0 → Brown Dwarf
- 1 → Red Dwarf
- 2 → White Dwarf

- 3 → Main Sequence
- 4 → Supergiant
- 5 → Hypergiant

Les propriétés de chaque étoile sont mesurées par rapport au Soleil :

- Température (K) - Température de surface en kelvins
- Luminosité (L/Lo) - Luminosité par rapport au soleil
- Rayon (R/Ro) - Rayon par rapport au soleil
- Magnitude absolue (Mv) - Luminosité intrinsèque
- Couleur de l'étoile - Couleur observée après analyse spectrale
- Classe spectrale - Classification basée sur les raies spectrales (O, B, A, F, G, K, M)

10.2 Collecte et préparation des données

L'ensemble des données a été créé en utilisant des équations astrophysiques réelles et des sources de données telles que :

- la loi de Stefan-Boltzmann - pour calculer la luminosité
- Loi de déplacement de Wien - Pour estimer la température de surface
- Relations de magnitude absolue - Pour déterminer la luminosité intrinsèque
- Méthodes de parallaxe - Pour dériver les valeurs du rayon

L'ensemble des données a été compilé à partir de plusieurs sources en ligne, ce qui a nécessité environ trois semaines de collecte et de prétraitement, en veillant à ce que les données manquantes soient calculées à l'aide de formules astrophysiques.

10.3 Importance de l'étude

Comprendre comment les étoiles sont classées et comment elles évoluent dans le temps est fondamental pour l'astronomie et l'astrophysique. Cette étude aide à :

- Valider le diagramme HR à l'aide du machine learning.
- Développer des modèles prédictifs pour la classification des étoiles.
- Explorer l'importance des caractéristiques dans la différenciation des étoiles.

Ce projet s'appuie sur des classificateurs de machine learning pour analyser les performances de différents modèles dans la classification des étoiles, depuis la simple classification binaire jusqu'à la classification multi-classes à 6 classes. Les résultats peuvent fournir des indications précieuses sur les relations entre les propriétés stellaires et sur la manière dont les étoiles s'inscrivent dans le cadre de l'évolution stellaire.

10.4 Exploration et analyse du dataset

Avant de construire un modèle de classification, il est essentiel d'explorer et de comprendre l'ensemble de données afin d'identifier les modèles, les relations et les besoins potentiels de prétraitement. Cette section se concentre sur l'analyse de la structure de l'ensemble de données, la vérification des valeurs manquantes, la visualisation des distributions et la compréhension des corrélations entre les caractéristiques.

10.4.1 1. Aperçu de l'ensemble de données

Nous commençons par charger l'ensemble de données et par afficher ses premières lignes afin de comprendre sa structure et le type de données qu'il contient.

```
# 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
```

Nous remarquons que :

- Le nombre de variables est de 7 et que nous avons 240 échantillons dans l'ensemble de données.
- Les types de données sont numériques, sauf **Star color** et **Spectral Class** qui sont catégorielles.

10.4.2 2. Vérification des valeurs manquantes

Les données manquantes peuvent avoir un impact sur la performance du modèle. Nous vérifions les valeurs manquantes afin de déterminer si des étapes de prétraitement, telles que l'imputation ou la suppression, sont nécessaires.

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

Sortie ;

```
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
```

Nous n'avons pas de valeurs manquantes.

10.4.3 3. Résumé statistique

La génération de statistiques sommaires permet de connaître l'étendue, la moyenne et la distribution des variables numériques.

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

Sortie :

```
        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
```

```
4  std     9552.425037     179432.244940   517.155763      10.532512       1.711394
5  min     1939.000000     0.000080        0.008400        -11.920000      0.000000
6  25%     3344.250000     0.000865        0.102750        -6.232500       1.000000
7  50%     5776.000000     0.070500        0.762500        8.313000        2.500000
8  75%     15055.500000    198050.000000   42.750000       13.697500       4.000000
9  max     40000.000000    849420.000000   1948.500000     20.060000       5.000000
```

Cela permet de :

- détecter les valeurs aberrantes (par exemple, des températures ou des luminosités anormalement élevées).
- Comprendre l'échelle et la variance des caractéristiques.

10.4.4 4. Distribution des classes (vérification de l'équilibre)

Pour nous assurer que notre ensemble de données n'est pas fortement déséquilibré, nous visualisons la distribution des différents types d'étoiles.

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

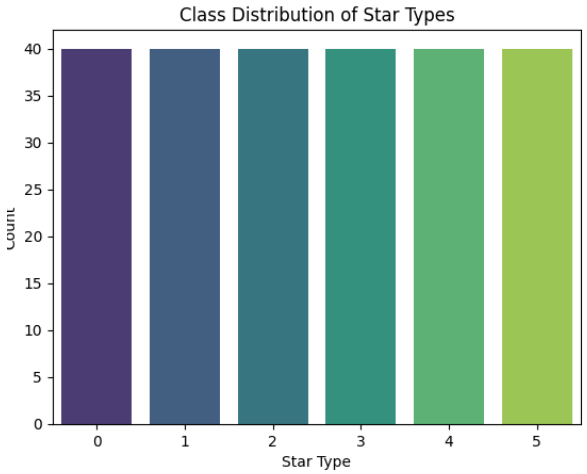

Figure 1: Distribution des classes

L'ensemble de données est parfaitement équilibré entre les différentes classes, avec 40 échantillons par classe.

10.4.5 5. Analyse de corrélation des variables

10.5 Matrice de Correlation

Cette matrice de corrélation donne un aperçu des relations entre les différentes caractéristiques de l'ensemble de données. Voici quelques observations clés :

10.5.1 Forte corrélation avec le type d'étoile :

La luminosité (0,68), le rayon (0,66) et la température (0,41) présentent une forte corrélation positive avec le type d'étoile.

La magnitude absolue (-0,96) est fortement corrélée négativement avec le type d'étoile, ce qui est logique puisque les étoiles plus brillantes (valeurs de magnitude plus faibles) ont tendance à se situer plus haut dans la hiérarchie de classification.

10.5.2 Dépendances des variables :

Luminosité et rayon (0.53) : Les étoiles plus grandes ont tendance à être plus lumineuses.

Magnitude absolue et luminosité (-0,69) : Une luminosité plus élevée se traduit par des valeurs de magnitude absolue plus faibles (relation inverse par définition).

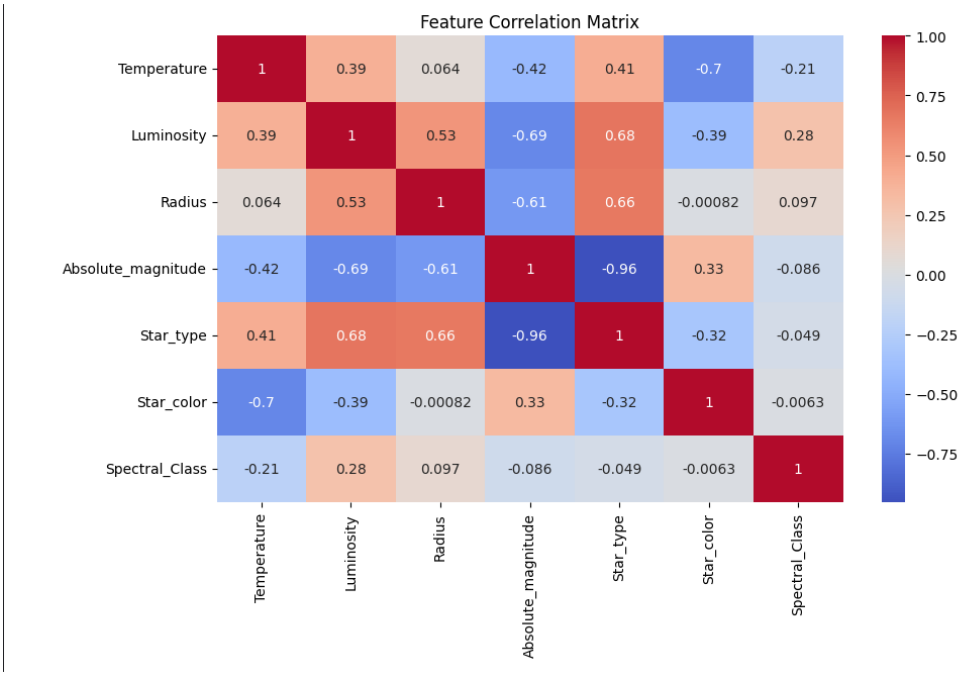

Figure 2: Matrice de Correlation

10.5.3 Couleur et classe spectrale des étoiles :

Couleur et température des étoiles (-0,7) : Cette corrélation négative est logique car les étoiles chaudes ont tendance à apparaître en bleu, tandis que les étoiles froides apparaissent en rouge.

Couleur de l'étoile et magnitude absolue (0,33) : Les étoiles les plus brillantes ont tendance à présenter des caractéristiques de couleur spécifiques.

10.5.4 La classe spectrale présente de faibles corrélations :

Il ne présente que des corrélations mineures avec d'autres variables, ce qui suggère que même s'il fournit des informations de classification, il n'est peut-être pas le prédicteur le plus puissant par rapport à des valeurs numériques telles que la température, la luminosité et la magnitude absolue.

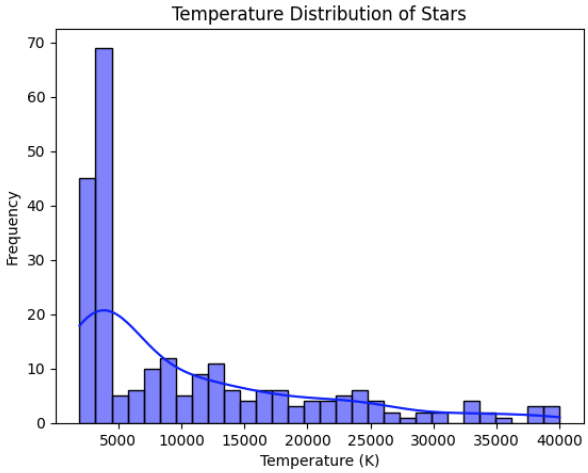

Figure 3: Distribution des températures

10.5.5 Implications :

- Prédicteurs clés : La luminosité, le rayon, la magnitude absolue et la température devraient être prioritaires pour les modèles de classification.
- Sélection des features : La classe spectrale ayant des corrélations plus faibles, sa contribution au pouvoir prédictif devrait être analysée plus en détail.
- Redondances éventuelles : La magnitude absolue et la luminosité ont une forte corrélation inverse, ce qui signifie qu'une seule pourrait suffire pour la modélisation.

10.6 6. Visualisation de la distribution des caractéristiques

Pour comprendre comment les caractéristiques varient selon les différents types d'étoiles, nous utilisons des histogrammes et des diagrammes boxplots.

10.6.1 Distribution des températures :

Distribution des températures : voir figure 3

Cet histogramme visualise la distribution des températures de surface des étoiles dans l'ensemble des données. Voici les principales observations :

10.6.2 Distribution asymétrique

- La majorité des étoiles ont des températures basses, avec un pic autour de 4000-5000 K.

- Au fur et à mesure que la température augmente, la fréquence diminue progressivement, ce qui montre que les étoiles plus chaudes sont plus rares.
- Peu d'étoiles ont une température supérieure à 30 000 K, ce qui indique que les étoiles extrêmement chaudes (par exemple, les étoiles de type O) sont moins courantes.

10.6.3 La plupart des étoiles sont froides

- Le grand nombre d'étoiles dont la température avoisine les 4 000 à 6 000 K suggère une forte proportion d'étoiles main sequence (telles que les étoiles de type G semblables au Soleil et les étoiles plus froides de type K/M).
- Ces étoiles plus froides (comme les Red Dwarfs) ont une longue durée de vie et sont plus abondantes dans l'univers.

10.6.4 Les étoiles chaudes sont moins fréquentes

L'histogramme montre une tendance à la baisse pour les étoiles de plus de 10 000 K, ce qui correspond aux attentes astrophysiques puisque les étoiles massives et plus chaudes (par exemple, les types O et B) ont une durée de vie plus courte et sont moins fréquemment observées.

10.6.5 Courbe de densité

La courbe KDE (Kernel Density Estimation) met encore plus en évidence la forme de la distribution, renforçant le fait que la plupart des étoiles se situent dans la gamme des températures les plus froides.

Conclusion

- L'ensemble des données est dominé par des étoiles plus froides, ce qui est cohérent avec la population stellaire réelle.
- Les étoiles chaudes et massives sont moins fréquentes, comme le prévoient les théories de l'évolution stellaire.
- Cette distribution s'aligne bien avec le diagramme de Hertzsprung-Russell, où la plupart des étoiles sont plus froides et tombent dans la catégorie de la séquence principale.

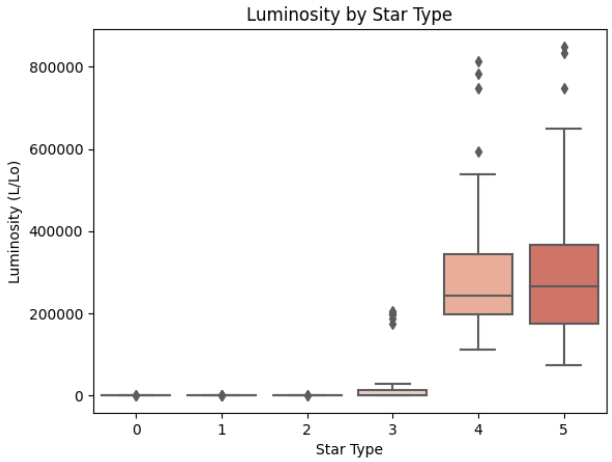

Figure 4: Enter Caption

Luminosity vs. Star Type :

10.6.6 Interprétation du graphique de la luminosité en fonction du type d'étoile

voir figure 4

Ce diagramme en boîte permet de visualiser comment la luminosité (L/Lo) varie en fonction des différents types d'étoiles (0-5). Voici les principales observations :

Plages de luminosité distinctes pour les types d'étoiles

- Les types 0, 1 et 2 (brown dwarfs, red dwarfs et white dwarfs) ont une luminosité très faible, proche de zéro, ce qui indique qu'il s'agit d'étoiles peu lumineuses.
- Les étoiles de type 3 (étoiles main sequence) ont une luminosité légèrement plus élevée, mais toujours relativement faible par rapport aux étoiles géantes.
- Les types 4 et 5 (supergéantes et hypergéantes) ont des luminosités nettement plus élevées, avec des valeurs extrêmes atteignant plus de 800 000 fois la luminosité du Soleil.

10.6.7 Large éventail de supergéantes et d'hypergéantes

- La boîte des types 4 et 5 est beaucoup plus grande, ce qui indique que la luminosité de ces étoiles est très variable.
- Ces types comprennent certaines des étoiles les plus lumineuses de l'univers, mais leur luminosité peut varier de modérée à extrêmement brillante.

10.6.8 Les étoiles à haute luminosité aberrantes

- Certaines étoiles extrêmement lumineuses des catégories des supergéantes et des hypergéantes apparaissent comme des valeurs aberrantes au-dessus des moustaches.
- Il s'agit d'étoiles rares et ultra-lumineuses, probablement des supergéantes ou des hypergéantes bleues massives.

10.6.9 Une classification claire basée sur la luminosité

- Le diagramme en boîte confirme que le type d'étoile est fortement corrélé à la luminosité.
- Les étoiles sequence main (type 3) servent de transition entre les naines de faible luminosité et les géantes très lumineuses.

10.6.10 Conclusion

- Cette distribution suit le modèle d'évolution stellaire attendu, où les naines sont peu lumineuses, les étoiles de la séquence principale ont une luminosité modérée, et les géantes/hypergéantes sont très lumineuses.
- La séparation nette de la luminosité suggère que cette caractéristique est très pertinente pour la classification des étoiles.

10.7 Nettoyage des données

10.7.1 Renommer les colonnes

Nous allons maintenant procéder au nettoyage des données en remplaçant d'abord space par _ et en renommant ces colonnes :

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

En supprimant les paranthèses, on obtient :

- Temperature
- Luminosity
- Radius
- Absolute_magnitude

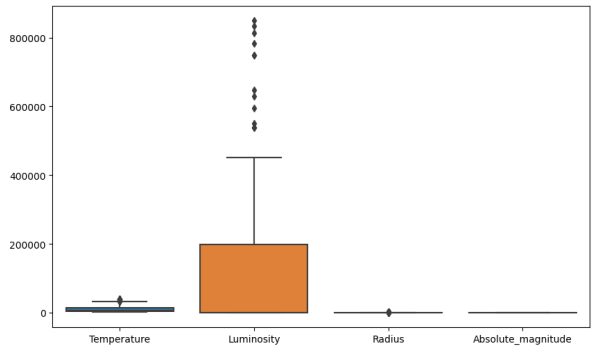

Figure 5: Outliers for the measures

10.8 Traitement des valeurs aberrantes

Les valeurs aberrantes peuvent affecter les performances des modèles de machine learning. Il est possible de les détecter à l'aide de boxplots (voir figure 5).

10.8.1 Interprétation du Boxplot (Analyse des valeurs aberrantes)

Le boxplot visualise la distribution de quatre variables numériques : Température, Luminosité, Rayon et Magnitude absolue. Voici ce que nous pouvons observer :

10.8.2 La luminosité présente des valeurs aberrantes extrêmes

La luminosité présente une large dispersion avec de nombreuses valeurs aberrantes extrêmes au-dessus de la limite supérieure. Cela suggère que certaines étoiles ont une luminosité extrêmement élevée, correspondant probablement aux supergéantes et hypergéantes. Cela correspond à l'astrophysique, où certaines étoiles rares sont beaucoup plus lumineuses que d'autres.

10.8.3 La température, le rayon et la magnitude absolue présentent peu ou pas de valeurs aberrantes extrêmes

Les distributions de ces paramètres sont relativement compactes, avec peu ou pas de points extrêmes. Cela suggère que la plupart des étoiles suivent un modèle plus régulier dans ces caractéristiques par rapport à la luminosité.

10.8.4 Les valeurs aberrantes de la luminosité doivent être analysées avec soin

Comme le boxplot montre que la Luminosité a des valeurs aberrantes extrêmes, nous allons les supprimer. Cependant, ces valeurs aberrantes peuvent représenter de véritables phénomènes astronomiques plutôt que des erreurs. Au lieu de les supprimer aveuglément, nous devrions analyser leur impact sur les modèles

de classification avant de décider d'une quelconque transformation (par exemple, mise à l'échelle logarithmique).

10.9 Comment gérer ces valeurs aberrantes au lieu de les supprimer ?

En astrophysique, les valeurs extrêmes de luminosité, de température et de rayon représentent souvent des phénomènes stellaires réels, tels que les supergéantes et les hypergéantes, plutôt que des erreurs. Le simple fait de les supprimer pourrait fausser l'ensemble des données et avoir un impact sur la capacité du modèle de classification à reconnaître ces types d'étoiles.

Au lieu de supprimer aveuglément les valeurs aberrantes, nous pouvons adopter d'autres approches : Expérimenter avec et sans valeurs aberrantes.

Voici une approche permettant de comparer les performances d'un modèle avec et sans valeurs aberrantes afin de déterminer leur impact sur la classification :

10.9.1 Utilisation de la méthode de l'intervalle interquartile (IQR) pour filtrer les valeurs aberrantes extrêmes :

```
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 Nous transformons la luminosité en logarithme pour réduire l'asymétrie et nous mettons à l'échelle toutes les caractéristiques à l'aide de RobustScaler (résistant aux valeurs aberrantes) :

```
# 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 Entraîner des modèles avec/sans valeurs aberrantes et Comparer les performances :

- Si le modèle avec les valeurs aberrantes est plus performant, nous les conservons.
- Si le modèle sans valeurs aberrantes améliore la classification, nous les supprimons.

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))
```

Résultats 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
```

Les modèles d'évaluation (avec et sans valeurs aberrantes) atteignent une précision parfaite de 100% pour toutes les mesures (précision, rappel et score F1). Ce que cela signifie :

10.10.1 Classification parfaite

Le modèle classe correctement tous les échantillons. Cela indique que les données sont bien séparées, ce qui facilite la classification.

10.10.2 Les valeurs aberrantes ont un impact minimal

Les résultats sont presque identiques entre les modèles avec et sans valeurs aberrantes.

Le support (nombre d'instances par classe) est légèrement différent, mais les performances restent parfaites. Cela suggère que les valeurs aberrantes n'ont pas eu d'impact négatif sur les performances du modèle.

10.10.3 Overfitting possible ?

Une précision de 100/100 peut indiquer que l'ensemble de données est trop facile à classer ou que le modèle a mémorisé les données d'apprentissage au lieu de bien généraliser.

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 Résultats 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 Pourquoi la régression logistique ?

- Interprétable – Nous pouvons analyser l'importance des variables (coefficients).
- Rapide et efficace – Fonctionne bien pour les petits ensembles de données.
- Modèle de base – Permet de comparer avec des modèles plus complexes.

10.11.3 Interpretation des résultats de la Logistic Regression

10.12 Comparaison entre les modèles (avec et sans valeurs aberrantes)

- Précision améliorée :
  - Avec valeurs aberrantes : 89,58%
  - Sans valeurs aberrantes : 93,48%
  - La suppression des valeurs aberrantes a entraîné une amélioration de la précision d'environ 4%.
- Score F1 amélioré:
  - Les scores macro et pondérés F1 ont augmenté après la suppression des valeurs aberrantes, indiquant une classification plus équilibrée.

Effet des valeurs aberrantes sur les performances du modèle

- Le modèle avec des valeurs aberrantes a montré un rappel légèrement inférieur pour certaines classes, ce qui signifie qu'il a mal classé certaines étoiles.
- Le modèle sans valeurs aberrantes a obtenu de meilleurs résultats dans la plupart des classes, ce qui suggère que les valeurs aberrantes ont eu un impact négatif sur le modèle en introduisant du bruit.

Observations spécifiques à la classe

- Classe 0 (par exemple, white dwarfs) :
  - Avec valeurs aberrantes : rappel à 100%, ce qui signifie que toutes les naines blanches réelles ont été correctement identifiées.
  - Sans valeurs aberrantes : toujours un rappel à 100
- Classes 2 et 5 (par exemple, étoiles géantes et supergéantes) :
  - Toujours une précision et un rappel à 100%, indiquant qu'il s'agit des types d'étoiles les plus reconnaissables.

- Classe 1, 3, 4 (par exemple, séquence principale et autres types) :
  - Ces classes présentaient des erreurs de classification avec des valeurs aberrantes, mais leur suppression a amélioré à la fois la précision et le recall.

Pourquoi la suppression des valeurs aberrantes a-t-elle été utile ?

- Les valeurs extrêmes de luminosité, de rayon et de grandeur absolue peuvent avoir faussé les limites de décision du modèle.
- En supprimant les valeurs aberrantes, le modèle s'est concentré sur la distribution majoritaire plutôt que d'être influencé par des cas extrêmes.

Points clés à retenir

- La suppression des valeurs aberrantes extrêmes a amélioré la précision du modèle.
- La plupart des classes étaient mieux classées après suppression des valeurs aberrantes.
- La régression logistique a bien fonctionné, mais un modèle plus complexe (par exemple, Random Forest) pourrait mieux capturer les relations non linéaires.

Prochaines étapes

- Vérifiez la matrice de confusion pour voir où les erreurs se produisent.
- Analysez l'importance des fonctionnalités à partir des coefficients de régression logistique.

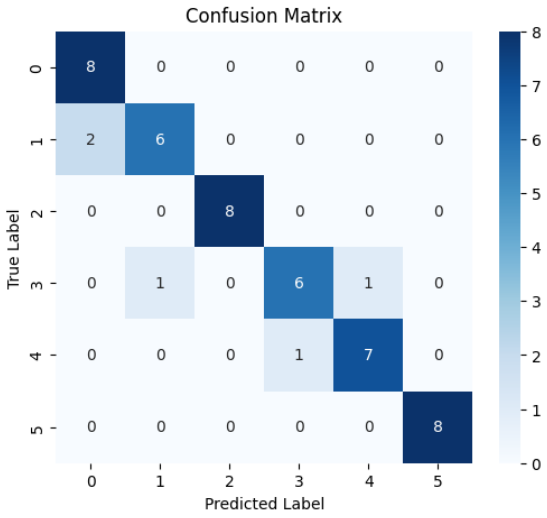

Figure 6: Matrice de confusion

10.13 Matrice de confusion

La matrice de confusion (voir figure 6) nous aide à comprendre quels types d'étoiles sont mal classés.

- Les valeurs diagonales représentent des classifications correctes.
- Les valeurs hors diagonale indiquent des erreurs de classification (quelles classes sont confondues).

10.13.1 Interprétation de la matrice de confusion

La matrice de confusion fournit une vue détaillée des performances de classification du modèle sur six types d'étoiles (0 à 5). Chaque ligne représente la classe réelle et chaque colonne représente la classe prédite. Les valeurs diagonales indiquent des classifications correctes, tandis que les valeurs hors diagonale représentent des classifications erronées.

Observations clés :

Classifications correctes (valeurs diagonales)

- Étoile Type 0 : 8 sur 8 correctement classés (précision à 100%).
- Étoile Type 1 : 6 sur 8 correctement classées (précision de 75%).
- Étoile Type 2 : 8 sur 8 correctement classés (précision à 100%).
- Étoile Type 3 : 6 sur 8 correctement classées (précision de 75%).
- Étoile Type 4 : 7 sur 8 correctement classés (précision de 87,5%).

- Étoile Type 5 : 8 sur 8 correctement classés (précision à 100%).

Erreurs de classification (valeurs hors diagonale)

- Étoile de type 1 : 2 échantillons mal classés comme type 0 (le modèle a confondu certaines étoiles de type 1 avec le type 0).
- Étoile de type 3 :
  - 1 échantillon classé à tort comme type 1.
  - 1 échantillon classé à tort comme type 4.
- Étoile de type 4 : 1 échantillon classé à tort comme type 3.

10.13.2 Analyse des performances du modèle

Haute précision pour la plupart des classes :

- Les types 0, 2 et 5 ont une classification parfaite (précision à 100%).
- Le type 4 n'a qu'une seule erreur de classification, conservant une précision élevée (87,5%).

Confusion entre types d'étoiles similaires :

- Le type 1 est confondu avec le type 0 → Cela pourrait indiquer un chevauchement de leurs caractéristiques.
- Le type 3 est confondu avec les types 1 et 4 → Suggère que le type 3 partage des caractéristiques avec les deux.
- Le type 4 est confondu avec le type 3, indiquant une similitude potentielle en termes de luminosité, de température ou de rayon.

10.13.3 Raisons possibles des erreurs de classification

Chevauchement des features :

- Certains types d'étoiles peuvent avoir des propriétés qui se chevauchent (par exemple, température, luminosité), ce qui les rend plus difficiles à distinguer.

Déséquilibre des données :

- Si certains types d'étoiles ont moins d'échantillons, le modèle peut avoir du mal à apprendre leurs modèles.

Limites du modèle :

- Si La régression logistique est un modèle linéaire ; si les limites de décision entre les types d'étoiles sont non linéaires, il se peut qu'elles ne soient pas parfaitement capturées.

10.13.4 Recommandations d'amélioration

Ingénierie des features :

- Explorez les interactions d'ordre supérieur ou les features dérivées pour mieux séparer les classes qui se chevauchent.

Essayez un modèle non linéaire :

- Les arbres de décision, les forêts aléatoires ou les réseaux de neurones pourraient mieux capturer les relations complexes.

Équilibrez l'ensemble de données :

- Utilisez des techniques de suréchantillonnage/sous-échantillonnage en cas de déséquilibre des données.

Réglage des hyperparamètres :

- Ajustez la force de régularisation dans la régression logistique pour affiner les performances.

Conclusion

- Le modèle de régression logistique fonctionne bien, avec une grande précision pour la plupart des classes.
- Certaines erreurs de classification se produisent, en particulier entre des types d'étoiles similaires, suggérant des caractéristiques qui se chevauchent.
- D'autres améliorations peuvent être apportées en affinant les fonctionnalités ou en utilisant des modèles plus complexes.

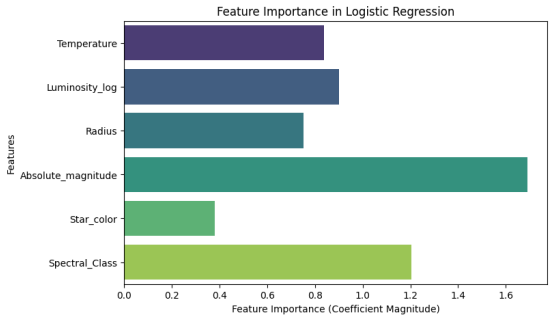

Figure 7: Features importance logistic regression avec outliers

10.14 visualiser l'importance des features

La régression logistique étant un modèle linéaire, nous pouvons analyser l'importance des caractéristiques en examinant les valeurs absolues des coefficients du modèle. Des valeurs absolues plus élevées indiquent une plus grande influence sur les décisions de classification.

Nous utiliserons un graphique à barres montrant l'importance des fonctionnalités dans les différentes variables d'entrée.

Modèle avec valeurs aberrantes (voir figure 7) :

Interprétation de l'importance des features pour la régression logistique (avec valeurs aberrantes)

Le graphique de l'importance des features pour la régression logistique avec valeurs aberrantes montre comment différentes caractéristiques influencent la classification des étoiles. Voici une analyse détaillée de l'impact de chaque fonctionnalité :

- Magnitude_absolue (plus haute importance)
  - Cette feature a la plus grande magnitude de coefficient, ce qui signifie qu'elle joue le rôle le plus important dans la distinction des types d'étoiles.
  - Puisque la magnitude absolue est une mesure de la luminosité intrinsèque, il est logique qu'elle affecte fortement la classification.
- Spectral_Class (deuxième feature la plus importante)
  - La classe spectrale définit le type d'une étoile en fonction de sa température et de sa couleur.
  - Sa grande importance suggère que la classification spectrale s'aligne bien avec la catégorisation des types d'étoiles de l'ensemble de données.

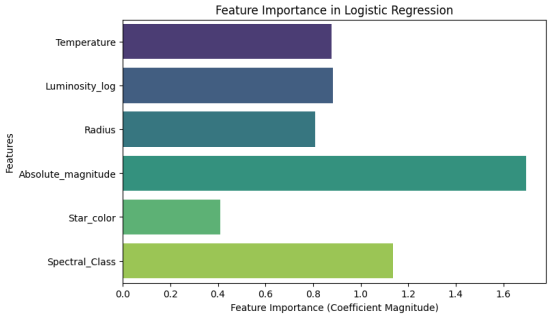

Figure 8: Features importance logistic regression sans outliers

- Luminosity_log, température et rayon (importance modérée)
  - Ces trois features ont des niveaux d'importance comparables.
  - Luminosity_log : Puisque nous avons logarithmiquement transformé la luminosité, son importance reflète à quel point la luminosité d'une étoile influence la classification.
  - Température : Un facteur clé dans la classification des étoiles, mais son importance est légèrement inférieure à la magnitude absolue et à la classe spectrale.
  - Rayon : affecte la classification, mais pas aussi fortement que les autres features.
- Star_color (fonctionnalité la moins importante)
  - La couleur des étoiles a le plus faible impact sur la classification.
  - Cela peut être dû à la redondance, car la température et la classe spectrale capturent déjà une grande partie des mêmes informations.

Modèle sans valeurs aberrantes (voir figure 8) :

Interprétation de l'importance des features pour la régression logistique (sans valeurs aberrantes)

La distribution de l'importance des features dans le modèle sans valeurs aberrantes reste similaire à celle avec valeurs aberrantes, mais avec quelques différences notables :

- Absolute_magnitude (toujours la fonctionnalité la plus importante)
  - Tout comme dans le modèle avec valeurs aberrantes, la magnitude absolue joue le rôle le plus important dans la classification.
  - Son importance reste élevée, ce qui renforce le fait que la luminosité intrinsèque d'une étoile est

un facteur dominant dans sa classification.

- Spectral_Class (deuxième fonctionnalité la plus importante)
  - L'importance de la classe spectrale reste forte, ce qui indique que la suppression des valeurs aberrantes ne réduit pas son pouvoir prédictif.
- Température, Luminosity_log et Rayon (importance modérée, mais légèrement décalée)
  - Température : conserve une importance similaire, mais son impact semble légèrement plus fort par rapport au modèle avec valeurs aberrantes.
  - Luminosity_log : reste une feature clé mais peut être devenue légèrement moins influente après la suppression des valeurs extrêmes.
  - Température : Un facteur clé dans la classification des étoiles, mais son importance est légèrement inférieure à la magnitude absolue et à la classe spectrale.
  - Rayon: a une importance légèrement inférieure à celle du modèle avec valeurs aberrantes, ce qui suggère que les valeurs extrêmes auraient pu exagérer son effet auparavant.
- Star_color (toujours la fonctionnalité la moins importante)
  - L'impact de la couleur des étoiles reste le plus faible, confirmant en outre qu'elle n'ajoute pas beaucoup d'informations uniques au-delà de la température et de la classe spectrale.

Comparaison : avec valeurs aberrantes et sans valeurs aberrantes

Feature	**valeurs aberrantes**	**sans valeurs aberrantes**	**Comparaison**
Absolute_magnitude	La plus haute	La plus haute	Reste la feature la plus dominante.
Spectral_Class	haute	haute	Garde une forte influence dans les deux cas.
Temperature	Modérée	Légèrement plus élevée	Légère augmentation en importance.
Luminosity_log	Modérée	Légèrement plus basse	Moins d'impact après suppression.
Radius	Modérée	Inférieure	Moins d'impact après suppression.
Star_color	La plus basse	La plus basse	C'est toujours la feature la moins utile.

Table 1: Comparaison de l'importance des features avec et sans valeurs aberrantes

Conclusion

- La suppression des valeurs aberrantes améliore la stabilité : l'importance de features telles que le rayon et le journal de luminosité devient plus équilibrée, évitant ainsi la suraccentuation causée par des valeurs extrêmes.
- L'ampleur absolue et la classe spectrale restent dominantes : peu importe si nous incluons ou excluons les valeurs aberrantes, ces deux features déterminent la classification des étoiles.

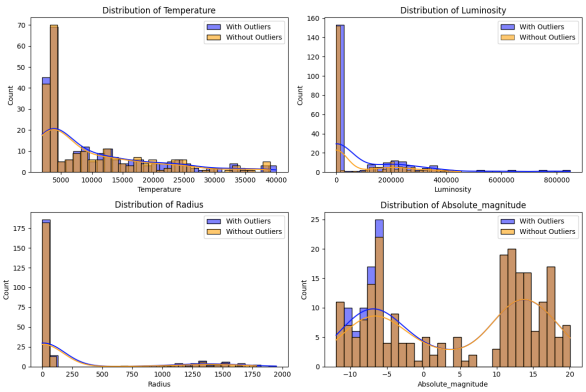

Figure 9: Distributions globales

- La pertinence de la température augmente légèrement : sans valeurs aberrantes, la température semble contribuer davantage à la classification, peut-être parce que les valeurs aberrantes ont déjà déformé sa relation.

Cette analyse fournit une justification solide pour supprimer les valeurs aberrantes, car elle rend le modèle plus stable et interprétable tout en maintenant des performances de classification élevées.

10.15 Analyse plus avancée

Vue systématique de la distribution des variables

Histogrammes pour voir les distributions globales :

Les histogrammes (voir figure 9) comparent la distribution de quatre caractéristiques clés (température, luminosité, rayon, grandeur absolue) avant et après suppression des valeurs aberrantes.

10.15.1 Répartition de la température :

- Avec valeurs aberrantes (bleu) : la distribution est fortement asymétrique vers la droite, avec une longue queue s'étendant au-delà de 30 000 K.
- Sans valeurs aberrantes (orange) : La majorité des valeurs restent inférieures à 10 000 K et la distribution devient plus concentrée.
- Interprétation : la suppression des valeurs aberrantes élimine les étoiles extrêmement chaudes (par exemple, les étoiles massives de type O), ce qui conduit à une plage de températures plus réaliste.

10.15.2 Répartition de la luminosité :

- Avec valeurs aberrantes (bleu) : les valeurs vont jusqu'à 800 000 luminosités solaires, avec une longue queue inclinée vers la droite.
- Sans valeurs aberrantes (orange) : La luminosité maximale est considérablement réduite, conduisant à une distribution plus compacte et moins asymétrique.
- Interprétation : la suppression des valeurs aberrantes supprime les supergéantes extrêmement brillantes, concentrant le modèle sur les étoiles géantes et de main sequence plus typiques.

10.15.3 Distribution de rayon :

- Avec valeurs aberrantes (bleu) : Il y a une longue queue s'étendant au-delà de 1000 rayons solaires.
- Sans valeurs aberrantes (orange) : La majorité des étoiles ont un rayon inférieur à 250.
- Interprétation : Les étoiles géantes et supergéantes avec des rayons énormes ont été supprimées, rendant l'ensemble de données plus équilibré.

10.15.4 Distribution de magnitude absolue :

- Avec valeurs aberrantes (bleu) : La distribution couvre une large plage, avec des valeurs négatives (étoiles très brillantes).
- Sans valeurs aberrantes (orange) : la plage est moins étalée et la distribution apparaît plus lisse.
- Interprétation : La suppression des valeurs extrêmes conduit à un ensemble de données qui représente mieux la majorité des étoiles.

Conclusion :

- L'ensemble de données était fortement asymétrique en raison de valeurs extrêmes de température, de luminosité et de rayon.
- Après la suppression des valeurs aberrantes, les distributions sont devenues plus normales et moins biaisées, conduisant à un modèle plus performant.
- Le nouvel ensemble de données se concentre sur les géantes de la séquence principale et modérées plutôt que sur les étoiles extrêmes.

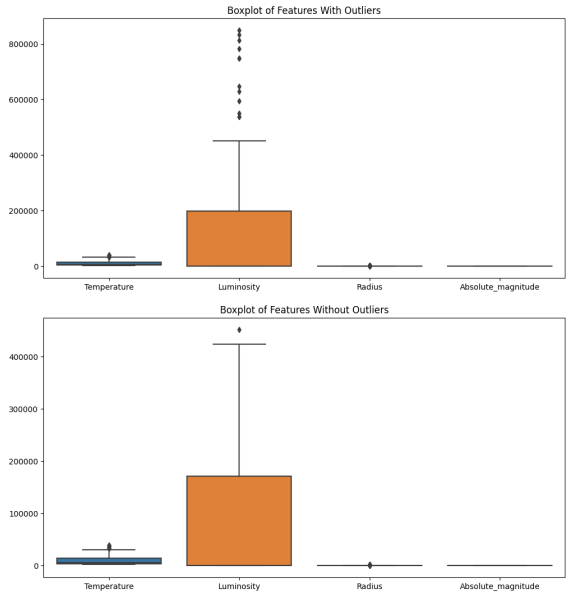

Figure 10: vérifier l'évolution des distributions

Boxplots pour vérifier comment les distributions changent :

Interprétation des boxplots :

Les boxplots (voir figure 10) comparent la distribution de la température, de la luminosité, du rayon et de la magnitude absolue avant et après la suppression des valeurs aberrantes.

Boxplot avec valeurs aberrantes (graphique du haut) :

- La luminosité présente des valeurs aberrantes extrêmes supérieures à 800 000, créant un boxplot très étendu.
- La température et le rayon affichent également des valeurs extrêmes.
- La magnitude absolue est plus compacte avec moins de valeurs extrêmes.

Problème clé : la présence de distributions très asymétriques et de valeurs aberrantes extrêmes, en particulier dans Luminosity, affecte l'équilibre de l'ensemble de données.

Boxplot sans valeurs aberrantes (graphique du bas) :

- Les valeurs aberrantes ont été supprimées, Luminosity présente moins de valeurs extrêmes.
- La température et le rayon affichent une plage plus compacte.

L'ensemble de données est désormais plus équilibré et la répartition des valeurs est réduite.

Principaux points à retenir :

- La suppression des valeurs aberrantes a considérablement réduit les valeurs extrêmes, en particulier pour la luminosité.
- L'ensemble de données est désormais mieux adapté aux modèles de machine learning, réduisant ainsi les biais.
- Même si la luminosité présente encore des valeurs élevées, elle est beaucoup plus contrôlée.

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: résumés statistiques (avec ou sans valeurs aberrantes)

10.16 Données statistiques avant et aprés suppression des valeur aberrantes

Interprétation des résumés statistiques (avec ou sans valeurs aberrantes)

Ce tableau (voir figure 11 fournit des statistiques descriptives (moyenne, écart type, valeurs min/max, percentiles) pour l'ensemble de données avant et après suppression des valeurs aberrantes.

Avec les valeurs aberrantes

- La luminosité a une valeur maximale extrêmement élevée (849 420) et un écart type très élevé (179 432), confirmant des valeurs aberrantes extrêmes.
- La température a également une très large plage (1 939 à 40 000).
- Le rayon a un maximum de 1 948,5, bien supérieur au 75e centile (42,75).
- La magnitude absolue semble moins affectée par les valeurs aberrantes (plage : -11,92 à 20,06).

Problème principal : l'ensemble de données comporte des valeurs extrêmes, en particulier en termes de luminosité, de température et de rayon, qui peuvent biaiser les modèles.

Sans valeurs aberrantes

- La luminosité maximale diminue considérablement (de 849 420 à 452 000), réduisant ainsi l'impact des valeurs extrêmes.
- La température maximale diminue légèrement (40 000 à 39 000), mais l'écart reste important.
- Le rayon maximum passe de 1 948,5 à 1 783, ce qui indique que certaines valeurs élevées ont été supprimées.
- Les écarts types diminuent pour la luminosité, la température et le rayon, indiquant un ensemble de données plus équilibré.

Problème principal : l'ensemble de données comporte des valeurs extrêmes, en particulier en termes de luminosité, de température et de rayon, qui peuvent biaiser les modèles.

Améliorations clés :

- L'ensemble de données est désormais plus stable, avec des variations moins extrêmes.
- Les réductions de l'écart type suggèrent une distribution plus normale, améliorant ainsi la fiabilité du modèle.

10.17 Classification binaire : White Dwarf vs. Main Sequence

l'exploration, encodage des variables catégoriques, analyse des corrélations et une première classification binaire à l'aide du modéle Random Forest :

```
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')
```

```
59 plt.show()
```

Explications

Encodage des variables catégoriques

L'encodage des variables catégoriques est nécessaire car la plupart des algorithmes de machine learning requièrent des entrées numériques plutôt que des labels textuels.

Les algorithmes de machine learning nécessitent des données numériques

De nombreux modèles (par exemple, régression logistique, Random Forest, réseaux neuronaux) fonctionnent sur des données numériques et ne peuvent pas traiter des chaînes de caractères catégorielles comme "Rouge", "Bleu", ou "O", "B", "A".

Garantir la comparabilité

Le codage transforme les catégories dans un format qui permet aux algorithmes d'interpréter les différences et les relations entre les classes.

Codage des labels pour les données de type ordinal

La couleur des étoiles et la classe spectrale présentent un ordre ou un regroupement inhérent, c'est pourquoi nous utilisons le codage des labels, qui attribue des nombres entiers uniques aux différentes catégories. Bien que les classes spectrales (O, B, A, F, G, K, M) suivent un ordre connu en astrophysique (O étant la plus chaude, M la plus froide), le codage par étiquette permet de préserver cette structure pour les modèles.

Alternative : One-Hot-Encoding

Si les catégories étaient vraiment nominales (sans ordre), le One-Hot Encoding (OHE) pourrait être une alternative. Cependant, l'OHE augmente la dimensionnalité, ce qui n'est pas idéal pour les petits ensembles de données.

En codant les caractéristiques catégorielles, nous permettons au modèle de traiter et d'apprendre des modèles de manière efficace sans introduire d'incohérences.

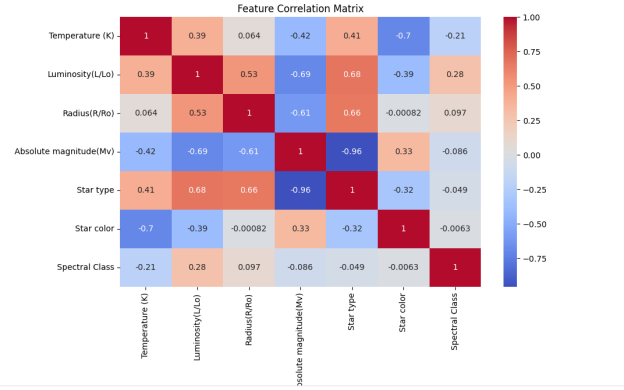

Figure 12: Binary correlation matrix

```
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

Interprétation de la matrice de corrélation des featrues pour la classification binaire (voir figure 12)

La matrice de corrélation permet de comprendre les relations entre les features avant d'appliquer du machine learning. Analysons-la dans le contexte de la classification binaire (type d'étoile : 2 contre 3).

Observations clés de la matrice de corrélation

Forte corrélation négative (-0,96) entre la magnitude absolue et le type d'étoile

- Cela signifie qu'à mesure que la magnitude absolue augmente (c'est-à-dire que l'étoile apparaît plus pâle), le type d'étoile passe d'une classe à une autre.
- La magnitude absolue étant logarithmique, une valeur plus faible signifie que l'étoile est plus brillante. Cela suggère qu'une classe pourrait contenir des étoiles plus brillantes, tandis que l'autre en a des plus pâles.

Corrélations positives fortes (supérieures à 0,6)

- Luminosité et type d'étoile (0,68) : une classe a tendance à avoir des étoiles plus brillantes.
- Rayon et type d'étoile (0,66) : une classe contient probablement des étoiles avec des rayons plus grands.

- Température et type d'étoile (0,41) : la température joue également un rôle mais est moins influente que la luminosité/le rayon.

Corrélation négative entre la couleur et la température des étoiles (-0,70)

- Une étoile plus bleu-blanc (valeur plus faible de « couleur de l'étoile ») est associée à des températures plus élevées.
- Cela a du sens dans la classification stellaire, où les étoiles bleues sont plus chaudes que les étoiles rouges.

Faibles corrélations avec la classe spectrale

- La classe spectrale a une faible corrélation avec la plupart des features, ce qui suggère qu'elle n'est pas le prédicteur le plus fort pour votre tâche de classification.

Comment cela aide-t-il la classification ?

- La magnitude absolue, la luminosité et le rayon semblent être les prédicteurs les plus puissants pour distinguer les deux types d'étoiles.
- La température et la couleur de l'étoile fournissent également des informations utiles, mais elles sont légèrement moins corrélées avec la cible.
- La classe spectrale n'apporte pas beaucoup, donc la supprimer pourrait simplifier le modèle sans avoir un impact important sur la précision.

Interprétation des résultats de la classification avec random forest

Le modèle a obtenu des performances parfaites avec une précision de 1,0 (100 %) sur la tâche de classification binaire. Décomposons les résultats.

Interprétation de la matrice de confusion

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

- Les lignes représentent les classes réelles (type d'étoile 2 et type d'étoile 3).
- Les colonnes représentent les classes prédites.
- Les valeurs indiquent combien d'instances ont été classées correctement ou incorrectement.

Informations clés :

- 8 cas de classe 2 ont été correctement classés (vrais positifs, TP).
- 8 cas de classe 3 ont été correctement classés (vrais positifs, TP).
- Aucun faux positif (FP) ou faux négatif (FN), ce qui signifie qu'il n'y a eu aucune erreur de classification.
- Performance de classification parfaite.

Analyse de la précision, du recall et du score F1

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

- Précision (TP / (TP + FP)) : Combien de classes 2 (ou 3) prédites étaient réellement correctes ?
  - Ici, c'est 1,00 (100%), ce qui signifie qu'il n'y a pas d'erreur de classification.
- Rappel (TP / (TP + FN)) : Parmi les classes 2 (ou 3) réelles, combien en avons-nous prédites correctement ?
  - Encore une fois, 1,00 (100%), ce qui signifie que le modèle a correctement identifié toutes les instances.
- Score F1 (moyenne harmonique de précision et de rappel) :
  - Également 1,00, ce qui montre que le modèle a parfaitement équilibré les deux mesures.
- Moyenne macro et moyenne pondérée :
  - Étant donné que les deux classes sont équilibrées (8 instances chacune), les deux moyennes sont également de 1,00.

Points clés à retenir

- Précision parfaite (100%) : le modèle a correctement classé tous les exemples de test.
- Aucune erreur de classification : aucun faux positif (FP) ou faux négatif (FN).
- Classes bien séparées : les caractéristiques (comme la grandeur absolue, la luminosité, le rayon) sont probablement très discriminantes.

Préoccupations potentielles :

- Overfitting ? Si l'ensemble de données est petit, le modèle a peut-être mémorisé des modèles au lieu de généraliser.
- Taille de l'ensemble de test ? Il n'y avait que 16 échantillons de test, les résultats pourraient donc ne pas être généralisables à un ensemble de données plus vaste.
- Essayer la validation croisée : pour confirmer la robustesse, il faudrait tester le modèle sur différentes divisions.

Vérification du surapprentissage (overfitting) :

Vérifions les performances avec une cross-validation: :

- La K-Fold cross-validation permet d'évaluer les performances sur différents sous-ensembles de données.
- Si le modèle fonctionne bien sur certains plis mais mal sur d'autres, cela peut indiquer un surajustement (overfitting).

Résultats :

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

Les scores de la cross-validation sont [1. 1. 1. 1. 1.], ce qui signifie que pour chaque pli, le modèle a atteint une précision de 100%. La précision moyenne du CV est également de 1,0 (100 %), ce qui suggère que le modèle est systématiquement parfait dans différents sous-ensembles des données d'entraînement.

10.17.1 Points clés à retenir :

Généralisation parfaite sur les données d'entraînement

- le modèle fonctionne de manière identique sur tous les plis de validation croisée, ce qui signifie qu'il n'y a aucune variation de précision entre les différentes divisions de données.
- Cela peut indiquer que les données sont trop faciles à classer ou que le modèle a appris des modèles très distincts pour les features sélectionnées.

Fortes Indications d'Overfitting

- Il est très rare d'obtenir une précision de 100 % dans tous les cas, à moins que l'ensemble de données soit très simple ou qu'il présente une forte séparabilité des features.

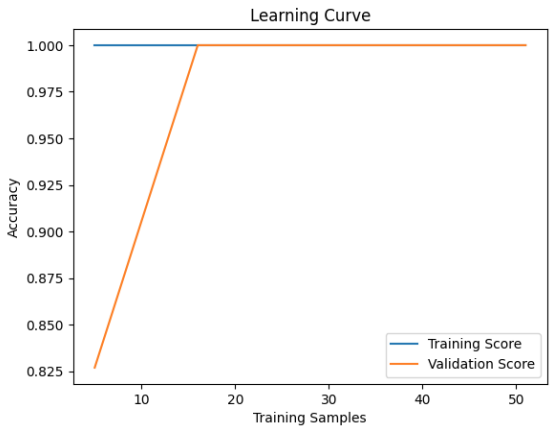

Figure 14: learning curve

- Si l'ensemble de test (données invisibles) affiche également une précision de 100 %, cela peut signifier que le modèle mémorise plutôt que généralise.

10.18 Interprétation de la courbe d'apprentissage (voir figure 14) :

La courbe d'apprentissage montre que la précision de la formation et de la validation atteint très rapidement 100 % et y reste. Voici ce que cela suggère :

- Signes de surapprentissage
  - La précision de l'apprentissage est constamment à 100% → le modèle mémorise probablement les données au lieu de les généraliser.
  - Aucun écart entre les courbes d'apprentissage et de validation → En général, un petit écart est attendu en raison des problèmes de généralisation. Ici, les deux courbes convergent parfaitement, ce qui est rare dans les problèmes du monde réel.
- Causes Possibles
  - Trop peu d'échantillons d'entraînement : le modèle peut voir les mêmes modèles de manière répétée, ce qui facilite la mémorisation.

10.19 Réflexions finales

Étant donné que la cross-validation et la courbe d'apprentissage affichent une précision de 100%, le modèle est probablement trop parfait pour un scénario réel. Il faudrait vérifier si le ensemble de données est trop simple, car mmême en condsidérant le dataset entier (avec toutes les classes), nous obtenons une accuraccy de 100% (voir sections précedentes) et une accurracy autour des 90% avec une régressionlogistique. En tout

cas, l'accurraccy parfait obtenu avec une random forest est trés probablement dû a :

- Un dataset trop simple pour être classifié.
- La puissance de classification des ensemble learners comme la random forest.

11 Ensemble de données de classification stellaire - SDSS17

11.1 Aperçu de l'ensemble de données

L'ensemble de données nous vient de Kaggle et est utilisé dans cette étude est le Stellar Classification Dataset - SDSS17, provenant du Sloan Digital Sky Survey (SDSS). Il contient 100 000 observations astronomiques, chacune classée comme galaxie, étoile ou quasar en fonction de ses caractéristiques spectrales. L'objectif de cette tâche de classification est de développer un modèle de machine learning capable de distinguer ces trois catégories en fonction des features fournies.

Chaque observation comprend 17 features d'entrée et 1 variable cible (classe), qui indique le type de l'objet. Les features englobent les données photométriques (u, g, r, i, z), les coordonnées spatiales (alpha, delta) et les mesures de décalage vers le rouge (redshift), entre autres. De plus, l'ensemble de données contient plusieurs colonnes d'identification, telles que obj_ID et spec_obj_ID, qui ne sont pas pertinentes pour la classification et seront supprimées lors du prétraitement.

Colonnes :

- obj_ID = Identifiant d'objet, la valeur unique qui identifie l'objet dans le catalogue d'images utilisé par le CAS
- alpha = Angle d'ascension droite (a J2000 epoch)
- delta = Angle de déclinaison (a J2000 epoch)
- u = Filtre ultraviolet dans le système photométrique
- g = Filtre vert dans le système photométrique
- r = Filtre rouge dans le système photométrique
- i = Filtre proche infrarouge dans le système photométrique
- z = Filtre infrarouge dans le système photométrique
- run_ID = Numéro d'exécution utilisé pour identifier l'analyse spécifique
- rereun_ID = Numéro de réexécution pour spécifier comment l'image a été traitée
- cam_col = Colonne de caméra pour identifier la ligne de balayage dans la course
- field_ID = Numéro de champ pour identifier chaque champ
- spec_obj_ID = ID unique utilisé pour les objets spectroscopiques optiques (cela signifie que 2 observations différentes avec le même spec_obj_ID doivent partager la classe de sortie)
- class = classe d'objet (galaxie, étoile ou objet quasar)
- redshift = valeur de décalage vers le rouge basée sur l'augmentation de la longueur d'onde
- plate = ID de plaque, identifie chaque plaque dans SDSS
- MJD = Date julienne modifiée, utilisée pour indiquer quand une donnée SDSS a été prise
- fiber_ID = ID de fibre qui identifie la fibre qui a pointé la lumière vers le plan focal dans chaque observation

Informations clés

- Variable cible : class (galaxie, étoile ou quasar).
- u, g, r, i, z (données photométriques)
- alpha, delta (informations de position, peuvent être utiles)

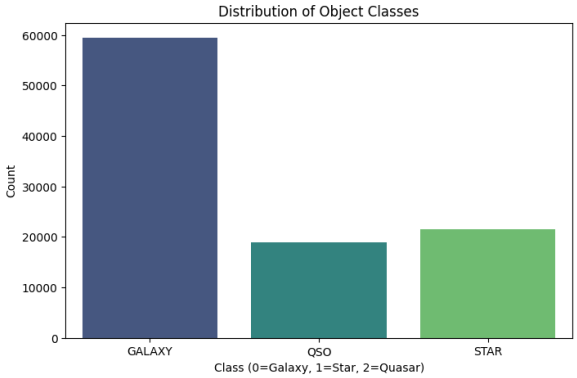

Figure 15: Distribution des classes

12 Analyse exploratoire des données (EDA)Analyse exploratoire des données (EDA)

12.1 Analyse des données

- Toutes les features sont numériques, sauf **class** qui catégorique.
- Pas de valeur manquantes.
- Pas de dupliquées.
- Le jeu de données se composent de 100000 lignes et 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 :

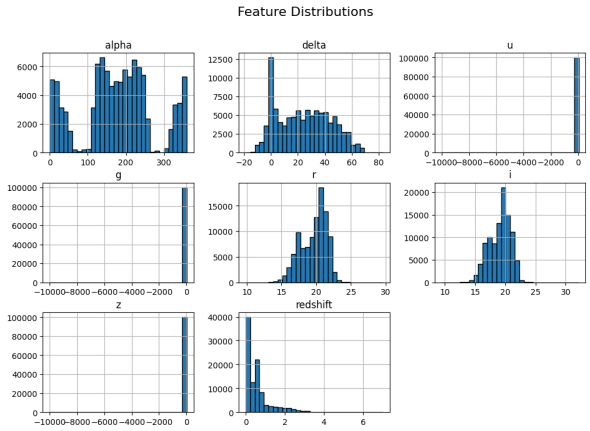

Figure 16: Distributions des features principales

  - 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.

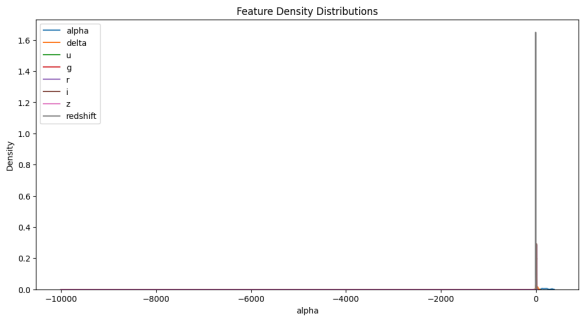

Figure 17: Distributions de densité

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.

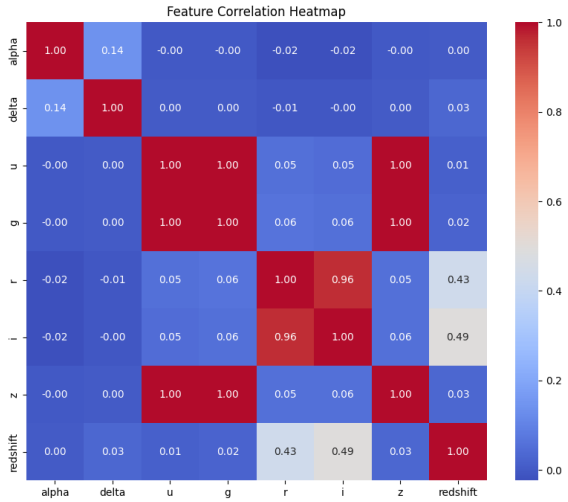

Figure 18: features correlation heatmap

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 :

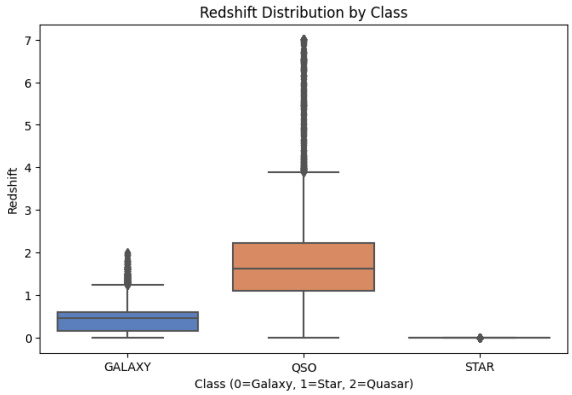

Figure 19: Distribution de redshift en fonction de la target

- 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.

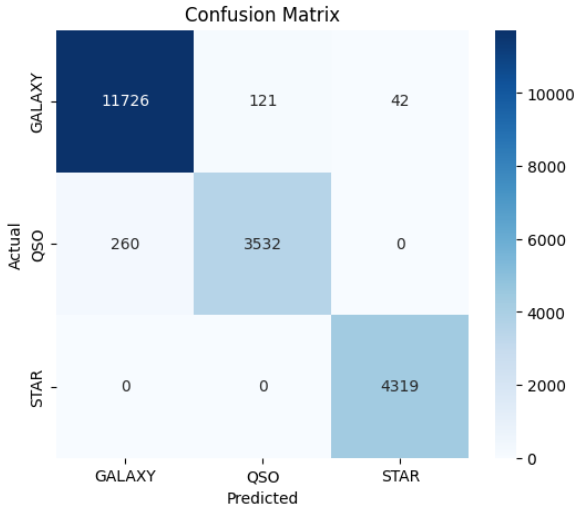

Figure 20: Matrice de confusion

- 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) :

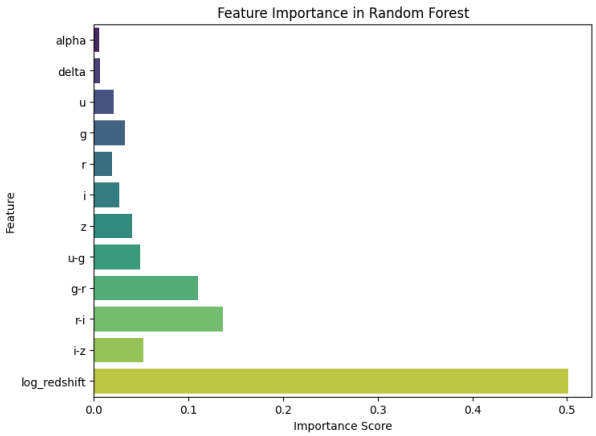

Figure 21: Features importance

- 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).

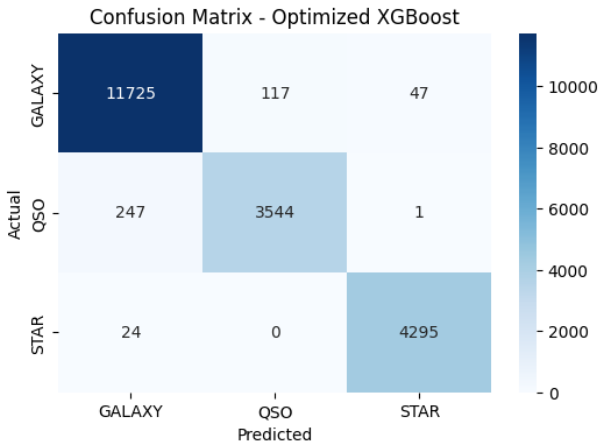

Figure 22: conf-mat-xgb

- 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.