

# DATA WAREHOUSING AND MINING

## MINI-PROJECT

**TE CMPN A**

## **STAR TYPE CLASSIFICATION**

Submitted By

Sr No	Name	Roll No
1)	Maaroor Khan	22102A0020
2)	Shardul Palande	22102A0004

Under the Guidance of  
**Dr. Kavita Shirsat**

Department of  
Computer Engineering

Vidyalankar Institute of  
Technology Wadala(E),  
Mumbai 400 037



**University of Mumbai**

2024-25

## **ABSTRACT**

This project focuses on the classification of stars into various categories based on their physical properties, such as temperature, luminosity, radius, absolute magnitude, star color, and spectral class. A Decision Tree classifier is employed to perform the classification task. The project integrates a Flask-based RESTful API that allows users to submit star data and receive real-time classification predictions. Data preprocessing techniques, including feature factorization, are utilized to optimize model performance. Initial attempts using Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) resulted in lower accuracy, leading to the selection of the Decision Tree model. The results demonstrate effective classification, showcasing the interpretability and simplicity of the Decision Tree model.

# **1)Introduction**

Stars are classified into different types based on their physical properties, which can be quantified through various attributes. With advancements in machine learning, automating the star classification process using these attributes has become a viable approach. This project aims to develop a machine learning model that classifies stars into six categories: Brown Dwarf, Red Dwarf, White Dwarf, Main Sequence, Supergiant, and Hypergiant. The classification system is designed to be user-friendly by integrating a Flask-based RESTful API, enabling users to submit star data and receive predictions in real-time.

## **2) Objectives**

**\*Data Analysis:** To analyze and preprocess a dataset of star characteristics to ensure compatibility with machine learning models.

**\*Model Selection:** To evaluate different machine learning algorithms (SVM, KNN, and Decision Tree) for classifying stars and select the most effective model based on accuracy and interpretability.

**\*API Development:** To develop a RESTful API using Flask to facilitate user interaction with the star classification model, allowing for real-time predictions based on user-submitted data.

**\*GUI Development:** To create a graphical user interface (GUI) that enables users to easily input star characteristics and view classification results.

**\*Performance Evaluation:** To evaluate the selected model's performance using appropriate metrics and to ensure the API provides accurate and timely predictions.

### **3) Algorithms Used**

#### **A. Decision Tree Classifier**

The Decision Tree algorithm builds a model of decisions based on feature splits and outcomes. Each internal node of the tree represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (in this case, star type). The Decision Tree model is chosen for its intuitive nature, providing clear insights into the classification process, though it can be prone to overfitting without proper pruning techniques.

#### **B. Initial Attempts with SVM and KNN**

Initially, both SVM and KNN algorithms were tested for star classification. The accuracy calculated using the sklearn accuracy score was 93% for KNN, 95% for SVM, and 99.8% for the Decision Tree model. While SVM is powerful for high-dimensional spaces and KNN is straightforward and effective for certain datasets, both models showed lower accuracy in this particular classification task compared to the Decision Tree. This prompted a shift in focus toward the Decision Tree model, which provided better performance on the dataset.

## **4. Methodology**

### **A. Data Preprocessing**

The dataset used for this project contains physical attributes of stars, such as temperature, luminosity, radius, absolute magnitude, star color, and spectral class. The categorical columns (Star color and Spectral Class) were factorized to convert string labels into integer representations, ensuring compatibility with machine learning algorithms.

Additionally, the dataset was scaled using StandardScaler to normalize the features, preventing any feature from dominating the model due to differences in magnitude.

### **B. Training the Decision Tree Model**

The Decision Tree classifier was trained on the preprocessed dataset. The model was evaluated using various performance metrics, such as accuracy, precision, and recall, to assess its effectiveness in classifying stars into the predefined categories.

### **C. Flask API Integration**

A Flask-based RESTful API was developed to expose the Decision Tree model as a web service. The API accepts star data in JSON format via POST requests and returns the predicted star type. The API also supports cross-origin requests (CORS), enabling it to be consumed by various web or mobile applications.

# 5. GUI

Temperature (K)

Luminosity(L/L<sub>o</sub>)

Radius(R/R<sub>o</sub>)

Absolute magnitude(M<sub>v</sub>)


Star color  

Select star color

Spectral Class  

Select spectral class

Predict Star



Welcome to Starry, a sophisticated platform designed to predict star types based on essential attributes. Simply input your star's characteristics, and let our advanced algorithms reveal its classification—whether it's a radiant blue giant or a tranquil red dwarf. Discover the mysteries of the cosmos with precision.

Temperature (K)

Luminosity(L/L<sub>o</sub>)

Radius(R/R<sub>o</sub>)

Absolute magnitude(M<sub>v</sub>)


Star color  

Blue-White

Spectral Class  

A

Predict Star



Welcome to Starry, a sophisticated platform designed to predict star types based on essential attributes. Simply input your star's characteristics, and let our advanced algorithms reveal its classification—whether it's a radiant blue giant or a tranquil red dwarf. Discover the mysteries of the cosmos with precision.

Temperature (K)

Luminosity(L/L<sub>o</sub>)

Radius(R/R<sub>o</sub>)

Absolute magnitude(M<sub>v</sub>)

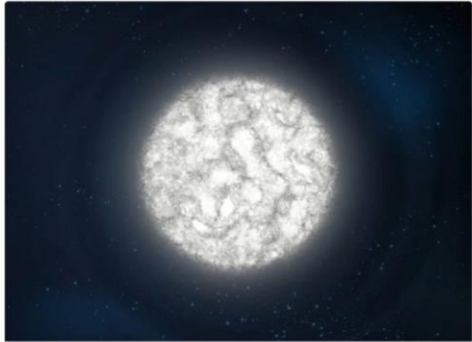
Star color  

Blue-White

Spectral Class  

A

Predict Star



A white dwarf is the remnant core of a star that has exhausted its nuclear fuel and shed its outer layers. Typically about the size of Earth but with a mass comparable to that of the Sun, white dwarfs are incredibly dense, with strong gravitational forces. They emit a faint, white glow due to residual heat, gradually cooling over billions of years. As they evolve, they will eventually fade into black dwarfs.

The GUI is designed for ease of use, featuring input fields for star attributes such as temperature, luminosity, radius, absolute magnitude, star color, and spectral class. Once the user inputs the required information, they can click a "Predict Star" button to receive predictions based on the provided data.

## **6. Analysis and Results**

The Decision Tree model was evaluated on the dataset after preprocessing.

### **A. Decision Tree Classifier Performance**

The Decision Tree model demonstrated a good balance between interpretability and performance. Although it was easier to interpret, it showed a tendency to overfit the training data. To mitigate this issue, the tree was pruned to enhance its generalization capability. The evaluation metrics indicated satisfactory accuracy in classifying the stars into their respective categories, with the Decision Tree achieving an accuracy of 99.8%.

### **B. Comparison of Algorithms**

The KNN and SVM algorithms were initially tested but achieved lower accuracy rates of 93% and 95%, respectively. This highlighted the superior performance of the Decision Tree classifier for this particular classification task.

### **C. API and GUI Performance**

The Flask API was tested for functionality and performance, successfully providing accurate predictions based on the submitted star data. The GUI was also evaluated, ensuring that user inputs were correctly processed and displayed, enhancing the overall user experience.



## **7. Conclusion**

This project successfully built and deployed a machine learning model to classify stars into different types based on their physical properties using a Decision Tree classifier. Initial attempts with SVM and KNN were made; however, they did not achieve satisfactory accuracy. The Decision Tree model provided a suitable balance of performance and interpretability, achieving an accuracy of 99.8%. The Flask-based RESTful API allowed the model to be served as a web service, enabling real-time predictions from external applications, and the GUI enhanced user interaction by simplifying the input process.

Future work could explore hyperparameter tuning, cross-validation, and adding more robust models to improve classification accuracy further. Additionally, deploying the API in a production-ready environment with security and scalability features could be the next step for practical use.