Analysis of Various AI Models with and without Clustering for Stellar Classification

MSc Research Project

MSc Artificial Intelligence TopUp

Marcin Pelech

Student ID: 23201789

School of Computing
National College of Ireland

Supervisor: Dr. Devanshu Anand

August 2025

National College of Ireland

Project Submission Sheet

School of Computing

Student Name:	Marcin Pelech
Student ID:	23201789
Programme:	MSc Artificial Intelligence TopUp
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr. Devanshu Anand
Submission Due Date:	11/08/2025
Project Title:	Analysis of Various AI Models with and without Clustering for Stellar Classification
Word Count:	
Page Count:	33

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other authors' written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Marcin Pelech
Date:	11th August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

- Attach a completed copy of this sheet to each project (including multiple copies).
- Attach a Moodle submission receipt of the online project submission to each project (including multiple copies).
- You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.
- Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Abstract

Today, artificial intelligence is used in many fields and is showing great performance in a wide range of areas. It can analyse large data sets and provide accurate predictions in a short period of time. Implementing Machine Learning Algorithms in Astrophysics where Each Observatory or Satellite Generates an enormous number of data could help to interpret findings and provide better insides to data collected. AI Algorithms not only get accurate predictions but are also quite fast when working with large datasets. They are free of human error, saving time and they handle tasks that couldn't be done by humans in reasonable time. Stellar Classification is a fundamental part of astronomy and aims to classify stars by their spectral characteristics. This project will classify astronomical objects to: Star, Quasar or Galaxy by using and comparing various AI models like: Classification: Support Vector Machines, K-Nearest Neighbors, Random Forest, Decision Tree, Naive Bayes, Convolutional Neural Network (CNN); Regression: Support Vector Regression, Decision Tree Regression, Random Forest Regression, Linear Regression. A key component of this study is the comparison of model performance on datasets with and without clustering. Among all models, the Random Forest Classifier trained on non-clustered data achieved the highest accuracy of 98.32%, slightly outperforming its clustered counterpart by 0.01%. However, Naive Bayes showed the most significant improvement with clustering, increasing its accuracy from 91.47% to 94.03%. The CNN model also benefited from clustering, improving its accuracy from 97.36% to 97.43%. These findings highlight the effectiveness of ensemble learning techniques and suggest that unsupervised preprocessing like clustering can enhance performance for specific models. Overall, this study demonstrates the potential of AI in astrophysical data analysis and the impact of clustering on model accuracy.
Keywords: stellar classification, comparing various AI models, clustering, Sloan Digital Sky Survey (SSDS)

1 Introduction

This study explores a wide spectrum of machine learning techniques including traditional classifiers, regression models, and deep learning architectures for the task of stellar classification using Sloan Digital Sky Survey (SDSS) data. It also investigates the impact of clustering method, Gaussian Mixture Model (GMM), as a preprocessing step to enhance model performance. The dataset is getting preprocessing, including feature scaling, selection, and balancing, to ensure proper evaluation. Performance is assessed using metrics such as accuracy, precision, recall, F1-score, and R^2. By comparing models across different learning techniques and preprocessing strategies, this thesis aims to identify optimal approaches for automated stellar classification. Astrophysicists need to work with enormous datasets that are collected from many observatories and satellites. Rubin Observatory (Rubin Observatory 2025) collects 20 terabytes

of data every 24 hours which is like listening Spotify for 50 years or watching Netflix for three years. (Astrostatistics 2025)Large Synoptic Survey Telescope (LSST) also produces 20 terabytes of data over one night and there are many more devices of this type around the Earth and space. Without additional tools like machine learning preprocesses, these are very large datasets to find meaningful information's, patterns, and predictions that could take enormous amount of human work power. Advancement of machine learning is that these AI algorithms can work quite fast with these datasets and preprocesses in a reasonable amount of time. AI can predict, and classify stellar objects with very good accuracy and precision, which provides for astrophysicists great tool that can speed up they work and save time, eliminating human error and allow to see patterns/anomalies/outliers across dataset which can hint at new astrophysical phenomena. Stellar classification is a fundamental part of astronomy, classifying astronomical objects based on their spectral characteristics, and the accurate classification of stellar objects is essential for understanding the structure and evolution of the universe. With the huge data from surveys like SDSS, machine learning offers scalable solutions to automate this process. In recent years, many studies have applied machine learning to stellar classification and using the SDSS dataset with promising results. (Deen Omat at al., (2022)) tested eight supervised learning models and found that Random Forest achieved 98% accuracy, correctly classifying all star instances. (Zhuliang Qi at al., (2022)) compared Decision Tree, Random Forest, and Support Vector Machine, reporting Random Forest again as the best performer with 98% accuracy. (Tanvi Mehta at al., (2022)) evaluated six models and concluded that Support Vector Classifier (SVC) yielded the highest accuracy 97.1%, outperforming others like KNN and Decision Tree. Mehmet Bilge at al., 2024 enhanced performance using graph-based feature selection, where XGBoost reached 98.02% accuracy and 99.81% ROC score, demonstrating the value of feature engineering. While these studies have made significant progress in applying machine learning to stellar classification, they often overlook the impact of clustering techniques. This thesis addresses that gap by evaluating both classification, and regression models, with and without clustering. It also aims to evaluate the performance of various AI models, both classification and regression for stellar classification, and to check the impact of clustering techniques on model accuracy, and finding out how the clustering techniques affect the performance of AI models. It will show which model type classification or regression produce higher accuracy for SDSS data. To ensure the consistency and objectivity of model performance, this study applies a series of preprocessing techniques to the SDSS dataset. Feature scaling is performed using StandardScaler, which normalises input data and prevents models from being biased toward features with larger numerical ranges. Categorical labels are encoded using LabelEncoder, allowing machine learning models to interpret non-numeric data effectively. To address class imbalance, a common issue in astronomical datasets SMOTE (Synthetic Minority Over-sampling Technique) is employed, generating synthetic samples for minority classes. Additionally, Local Outlier Factor (LOF) is used to detect and filter out anomalous data points, improving data quality and reducing noise.

These preprocessing steps together will improve the model, reduce bias, and ensure consistent performance across classification and regression tasks. The thesis is organized as follows: Section 2 (Data Description and Prepossessing), introduces the dataset and outlines the preprocessing steps applied. Section 3 (Models Overview), describes the machine learning models used for stellar classification. Section 4 (Methodology), details the experimental design, including clustering techniques and evaluation metrics. Section 5 (Results and Evaluation), presents the experimental results and discusses key findings. Section 6 (Conclusion and Future Work), summarises the study and proposes directions for future research. Section 7 (Appendices), provides supplementary materials, including the sample code and extended outputs. Section 8 (Acknowledgments), expresses gratitude to the contributors and supporters of the research.

2 Related Work

2.1 Classification Models

2.1.1 Support Vector Classifier (SVC)

Support Vector Machines (SVM) are supervised learning models used for classification tasks, particularly effective in high-dimensional spaces. In this project, the LinearSVC variant is employed, which uses a linear kernel to find the optimal hyperplane that separates classes with the maximum margin. In the study by (Omat et al. (2022)), Support Vector Classification (SVC) was used to classify stellar objects using data from SDSS DR17, their SVC got 97% accuracy. While their implementation of SVC contributed to high classification accuracy, the authors did not address key data preprocessing steps such as class balancing or outlier elimination. In the study by (Tanvi Mehta at al., (2022)) their implementation of SVC got 97,1% accuracy but as previous researchers they didn't address key data preprocessing steps such as class balancing or outlier elimination.

2.1.2 Decision Tree Classifier

Decision Tree is a supervised learning algorithm that builds a tree-like model of decisions based on feature values. Decision Tree is intuitive, interpretable, and capable of handling both numerical and categorical data. In the study by (Tanvi Mehta at al., (2022)), their implementation of Decision Tree Classifier got 86,2% accuracy, but didn't include any method like Synthetic Minority Over-sampling Technique (SMOTE) to mitigate class imbalance and applying statistical methods to detect and remove outliers. In the study by (Zhuliang Qi at al., (2022)), his implementation of Decision Tree Classifier got 97% accuracy as the author had used SMOTE to balance data, but didn't use any method to detect and remove outliers.

2.1.3 K Nearest Neighbors Classifier

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm that classifies data points based on the majority class among their nearest neighbours. It is simple yet effective, particularly in cases where decision boundaries are irregular or non-linear. The model predicts the class of each test sample by examining the labels of its k closest training samples, measured using Euclidean distance. In the study by (Mehmet Bilge at al., (2024)), their implementation got 89,77%, the author had used graph based features selection, but didn't implement any methods to amend data unbalance or outliers elimination.

2.1.4 Naive Bayes Classifier

Naive Bayes Classifier is a probabilistic classifier based on Bayes' Theorem, assuming independence among predictors. While being conceptually simple, it often performs well on high-dimensional datasets and is effective for text classification and categorical data. In the study of (Tanvi Mehta at al., (2022)), their implementation of Naive Bayes Classifier got 87,4% accuracy, the authors hadn't used any method to deal with unbalanced data, or to detect and remove outliers. In study by (Omat et al. (2022)), Naive Bayes Classifier got 91% accuracy as previous author had used no methods for unbalanced data or outlier elimination.

2.1.5 Random Forest Classifier

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes for classification tasks. It is particularly robust to over-fitting and performs well on high-dimensional datasets with complex feature interactions. In the study by (Mehmet Bilge et al. (2024)), the author implementation got 97.85% accuracy.

2.1.6 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are deep learning architectures primarily designed for spatial data, such as images and time series. They utilize convolutional layers to extract hierarchical features, making them highly effective for pattern recognition tasks. Although CNNs are traditionally applied to 2D image data, they can be adapted to 1D inputs for classification tasks involving structured tabular data. In the study by (Jing-Hang et al. (2022)), their implementation of CNN got 94.4% accuracy in the classification task on SSDS dataset.

2.2 Regression Models

2.2.1 Support Vector Regression

Support Vector Regression (SVR) is a powerful supervised learning algorithm obtained from Support Vector Machines (SVM). SVR is particularly effective in high-dimensional spaces and for datasets with complex relationships. Couldn't find any papers with SVR, probably as it hadn't performed well.

2.2.2 Random Forest Regressor

The Random Forest Regressor is an ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of the individual trees. This approach reduces over-fitting, and improves generalisation compared to a single decision tree. In the study by (Ahmed Taha et al. (2022)), the author didn't use Random Forest Regressor to check if objects are star, quasar, or galaxy. But they did use it for checking coordinates with high R^2, nearly perfect.

2.2.3 Decision Tree Regressor

Decision Tree Regressor is a non-parametric supervised learning algorithm that predicts continuous outcomes by recursively partitioning the feature space. It builds a tree structure, where each internal node represents a decision-based on feature values, and each leaf node corresponds to a predicted numerical value. In the study by (Krishna et al. (2024)), the author didn't use Decision Tree Regressor to check if objects are star, quasar, or galaxy. But used for checking redshift estimation with RMS value, which was above 0.16.

2.2.4 Linear Regression

Linear Regression is a fundamental statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data.

2.3 Overview

While reviewing existing literature on stellar classification using the SDSS dataset, it became evident that some techniques have been unexplored. Notably, clustering algorithms have not been applied in any of the reviewed studies, regardless of their strength in identifying underlying trends and patterns in unlabelled data. Additionally, only two papers (Ahmed Taha et al. (2022)), and (Zhuliang Qi at al., (2022)) were found to implement method for addressing class imbalance, suggesting that data balancing are rarely implemented in this domain. Furthermore, outlier elimination techniques were used in three papers (Ahmed Taha et al. (2022)), (Tanvi Mehta at al., (2022)), and (Sabeesh et al. (2022)), even such preprocessing steps can significantly improve model performance. These gaps

highlight opportunities for methodological innovation and justify the presence of these techniques in the present study.

3 Methodology

3.1 Dataset Description

This study uses the Stellar Classification Dataset - SDSS17, publicly available on Kaggle and originally sourced from the Sloan Digital Sky Survey (SDSS) Data Release 17. The dataset contains 100,000 astronomical observations, each described by 17 features and a class label indicating whether the object is a star, galaxy, or quasar. For this project, a subset of seven features and one class label was selected for training and evaluation: Photometric measurements: u, g, r, i, z (capturing brightness across five spectral bands) Spectroscopic redshift: redshift (indicating the object's velocity and distance) Instrument metadata: plate (identifying the spectroscopic plate used during observation). This feature set enables reliable classification and supports both supervised learning and preprocessing techniques such as clustering.

Table 1: Feature Descriptions in the SDSS Dataset

Feature	Details
obj_ID	Object Identifier, the unique value that identifies the object in the image catalog used by the CAS
alpha	Right Ascension angle (at J2000 epoch)
delta	Declination angle (at J2000 epoch)
u	Ultraviolet filter in the photometric system
g	Green filter in the photometric system
r	Red filter in the photometric system
i	Near Infrared filter in the photometric system
z	Infrared filter in the photometric system
run_ID	Run Number used to identify the specific scan
rerun_ID	Rerun Number to specify how the image was processed
cam_col	Camera column to identify the scanline within the run
field_ID	Field number to identify each field
spec_obj_ID	Unique ID used for optical spectroscopic objects (two observations with the same ID share the output class)
class	Object class (galaxy, star, or quasar)
redshift	Redshift value based on the increase in wavelength
plate	Plate ID, identifies each plate in SDSS
MJD	Modified Julian Date, used to indicate when a given piece of SDSS data was taken
fiber_ID	Fiber ID that identifies the fiber pointing light at the focal plane in each observation

3.2 Preprocessing Steps

3.2.1 Overview

To prepare the dataset for machine learning classification, the following preprocessing steps were applied to the selected features (u, g, r, i, z, redshift, plate, class) from the SDSS17 dataset:

3.2.2 Missing Value Handling

No missing values were observed in dataset.

3.2.3 Feature Selection

Seven features were selected based on exploratory analysis: five photometric bands (u, g, r, i, z), redshift, plate and class.

Table 2: Feature Selected from the Dataset.

Feature	Details
u	Ultraviolet filter in the photometric system
g	Green filter in the photometric system
r	Red filter in the photometric system
i	Near Infrared filter in the photometric system
z	Infrared filter in the photometric system
redshift	Redshift value based on the increase in wavelength
plate	Plate ID, identifies each plate in SDSS
class	Object class (galaxy, star, or quasar)

3.2.4 Outliers Handling

Outlier Detection and Removal were identified using the Local Outlier Factor (LOF) algorithm. A threshold at the 3rd percentile of the LOF scores was used to filter out around 3,000 anomalous samples.

Table 3: Number of Observations per Class

Class	Observations
Star	59,445
Galaxy	21,594
Quasar	18,961

Table 4: Updated Number of Observations per Class

Class	Observations
Star	58,036
Galaxy	20,460
Quasar	18,504

3.2.5 Resampling for Class Balance

To address class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) was applied with k_neighbors=5 and random_state = 42. This generated synthetic samples for the minority classes to balance the distribution:

Table 5: Observations per Class before SMOTE

Class	Observations
Star	58,036
Galaxy	20,460
Quasar	18,504

Table 6: Observations per Class after SMOTE

Class	Observations
Star	58,036
Galaxy	58,036
Quasar	58,036

3.2.6 Normalization

The features (u, g, r, i, z, redshift, and plate) were standardized using StandardScaler to ensure a mean of zero and a standard deviation of one across each feature.

3.2.7 Categorical Encoding

The class feature was encoded using Label Encoding to convert it into a numerical format suitable for models input.

3.2.8 Clustering Integration

To find patterns, and possibly improve classification and regression performance, clustering techniques were integrated into the machine learning pipeline. The Gaussian Mixture Model (GMM) algorithm was chosen as its best for continuous numerical data, and u, g, r, i, z, redshift $\rightarrow$ all continuous.

3.2.9 Train-Test Split

The balanced dataset was split into training (80%) and testing (20%) subsets, to train and evaluate models.

3.2.10 Evaluation Metrics

To evaluate model performance, a set of metrics were applied to both classification and regression tasks. For classification models, the metrics included Speed, Accuracy, Precision, Recall, and F1 Score, giving view of predictive quality and computational efficiency. In addition to these quantitative metrics, a confusion matrix was generated for each classifier to visually assess the distribution of correct and incorrect predictions across classes. This matrix provided deeper insight into model behaviour. For regression models, evaluation focused on Speed, $\mathbf{R^2}$ Score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). These metrics together measure how well the model fits the data, and how large the prediction errors are. To deepen the understanding of model behaviour, a residual plot was used to visualize the difference between actual and predicted values. This plot helped identify patterns or biases in the predictions, which might not be evident from metrics alone. Together, these evaluation tools ensured a good assessment of model performance across both classification and regression tasks.

3.2.11 Comparative Analysis

To evaluate the impact of clustering on model performance, both classification and regression algorithms, were tested on datasets with and without clustering. For classification tasks, models such as Random Forest, Decision Tree, K-Nearest Neighbors, CNN, Naive Bayes, and Support Vector Machines were assessed using key metrics: speed, accuracy, precision, recall, and F1 score. Some models trained on clustered data outperformed their non-clustered counterparts. Even algorithms traditionally sensitive to noise, like Naive Bayes, showed improvement in accuracy, precision, and recall when clustering was applied.

3.3 Model Implementation and Overview

3.3.1 Random Forest Classifier

The Random Forest Classifier was implemented using the scikit-learn library. To determine the optimal number of estimators, a loop was executed from 1 to 19, training the model on the non-clustered and clustered dataset and recording accuracy for each configuration. The best-performing setup was selected for both clustered and non-clustered data.

Table 7: Random Forest Classifier Setup Summary

Parameter	**Details**
Library	`scikit-learn`
Hyperparameter Tuned	`n_estimators`
Tuning Method	Manual iteration with accuracy tracking
Best Configuration (Without Clustering)	`n_estimators = 16`
Best Configuration (With Clustering)	`n_estimators = 19`
Random State	`30` (fixed for reproducibility)

For evaluation preparation confusion matrices were generated for both clustered and non-clustered cases to visualize classification performance. The model was timed during training and prediction to assess computational efficiency. Accuracy, precision, recall, and F1 score were calculated using weighted averages.

3.3.2 Decision Tree Classifier

The Decision Tree Classifier was implemented using the scikit-learn library with default parameters. No hyperparameter tuning was performed, as the model was intended to serve as a baseline for comparison. The classifier was trained and tested on both clustered and non-clustered datasets to evaluate the impact of clustering on performance.

Table 8: Decision Tree Classifier Setup Summary

Parameter	Details
Library	`scikit-learn`
Hyperparameters Tuned	None (default configuration)
Tuning Method	Not applicable
Best Configuration (Without Clustering)	Default
Best Configuration (With Clustering)	Default
Random State	`30` (fixed for reproducibility)

For evaluation preparation confusion matrices were generated to visualize classification accuracy. Execution time was recorded to assess computational efficiency. Accuracy, precision, recall, and F1 score were calculated using weighted averages.

3.3.3 K Nearest Neighbors Classifier

The K Nearest Neighbors (KNN) Classifier was implemented using the scikit-learn library. To determine the optimal number of neighbors (n_neighbors), a loop was executed from 1 to 19, and accuracy was recorded for each configuration. The best-performing value was selected for both clustered and non-clustered datasets.

Table 9: K Nearest Neighbors Classifier Setup Summary

Parameter	Details
Library	`scikit-learn`
Hyperparameters Tuned	`n_neighbors`
Tuning Method	Manual loop from 1 to 19
Best Configuration (Without Clustering)	`n_neighbors = 1`
Best Configuration (With Clustering)	`n_neighbors = 1`
Distance Metric	`minkowski` with `p = 2` (equivalent to Euclidean)(default)
Weighting	`uniform` (default)

For evaluation preparation confusion matrix was plotted for visual inspection of classification performance. Execution time recorded to assess computational efficiency. Accuracy, precision, recall, and F1 score calculated using weighted averages.

3.3.4 Support Vector Machines

Support Vector Machines were implemented using LinearSVC from scikit-learn, wrapped in a pipeline with StandardScaler for feature normalization. Hyperparameter tuning was performed using grid search with 5-fold cross-validation to identify the best configuration for both clustered and non-clustered datasets.

Table 10: Support Vector Machines Setup Summary

Parameter	**Details**
Library	`scikit-learn`
Model Used	`LinearSVC`
Preprocessing	`StandardScaler` (via pipeline)
Hyperparameters Tuned	`C`, `penalty`, `loss`
Tuning Method	GridSearchCV with 5-fold cross-validation
Best Configuration (Without Clustering)	`C = 150`, `penalty = 'l2'`, `loss = 'squared_hinge'`
Best Configuration (With Clustering)	`C = 180`, `penalty = 'l2'`, `loss = 'squared_hinge'`
Max Iterations	`10000`

For evaluation preparation confusion matrix was plotted to visualize classification performance. Execution time recorded to assess computational efficiency. Accuracy, precision, recall, and F1 score calculated using weighted averages.

3.3.5 Naive Bayes

The Naive Bayes Classifier was implemented using the GaussianNB model from the scikit-learn library. No hyperparameter tuning was performed, as the algorithm is inherently simple and relies on probabilistic assumptions. The model was trained and evaluated on both clustered and non-clustered datasets to assess the impact of clustering on classification performance.

Table 11: Naive Bayes Classifier Setup Summary

Parameter	Details
Library	`scikit-learn`
Model Used	`GaussianNB`
Hyperparameters Tuned	None (default configuration)
Tuning Method	Not applicable
Best Configuration (Without Clustering)	Default
Best Configuration (With Clustering)	Default
Random State	Not applicable

For evaluation preparation confusion matrices were generated to visualize classification performance. Execution time was recorded to assess computational efficiency. Accuracy, precision, recall, and F1 score were calculated using weighted averages.

3.3.6 Convolutional Neural Network (CNN)

In this study, a 1D CNN was implemented using TensorFlow/Keras to classify stellar objects from the SDSS dataset. The input features were reshaped to match the expected format for convolutional layers. The architecture consisted of a two Conv1D layer followed by a Flatten layer and a Dense output layer with softmax activation for multiclass classification. To optimize performance, a grid search was conducted using KerasClassifier to explore hyperparameters. The best-performing configuration for data with clustering was selected based on validation accuracy.

Table 12: CNN Classifier Setup Summary

Parameter	With Clustering	Without Clustering
Library	`Keras + TensorFlow`	`Keras + TensorFlow`
Model Type	`Conv1D`	`Conv1D`
Conv1D Activation	`relu`	`relu`
Output Layer Activation	`softmax`	`softmax`
Filters	64	64
Kernel Size	2	2
Optimizer	`adam`	`adam`
Epochs	100	80
Best Accuracy	97.43%	97.36%

For evaluation preparation confusion matrix and classification report were generated. Execution time was recorded. Accuracy, precision, recall, and F1 score were computed using weighted averages.

3.3.7 Support Vector Regression

Support Vector Regression was implemented using LinearSVR from scikit-learn, wrapped in a pipeline with StandardScaler to normalize input features. Hyperparameter tuning was conducted using GridSearchCV to optimize performance for both clustered and non-clustered datasets.

Table 13: Support Vector Regression Setup Summary

Parameter	With Clustering	Without Clustering
Library	`scikit-learn`	`scikit-learn`
Model Used	`LinearSVR`	`LinearSVR`
C	0.1	0.1
Epsilon	0.0	0.0
Loss Function	`squared_epsilon_insensitive`	`squared_epsilon_insensitive`
Scaler	`StandardScaler`	`StandardScaler`
Max Iterations	10000	10000
Best R^2 Score	0.2434	0.2434

For evaluation preparation residual plots were generated to visualize prediction errors. Execution time was recorded. Regression metrics included R^2, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

3.3.8 Random Forest Regression

Random Forest Regression was implemented using RandomForestRegressor from scikit-learn. Hyperparameter tuning was performed using GridSearchCV to find best configuration for both clustered and non-clustered datasets.

Table 14: Random Forest Regression Setup Summary

Parameter	**With Clustering**	**Without Clustering**
Library	`scikit-learn`	`scikit-learn`
Model Used	`RandomForest Regressor`	`RandomForest Regressor`
n_estimators	500	500
max_depth	None	None
min_samples_split	2	2
min_samples_leaf	1	1
Scoring Metric	R^2	R^2
Best R^2 Score	0.9761	0.9759

For evaluation preparation residual plots were generated to visualize prediction errors. Execution time was recorded. Regression metrics included R^2, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

3.3.9 Decision Tree Regression

Decision Tree Regression was implemented using DecisionTreeRegressor from scikit-learn. Hyperparameter tuning was performed using GridSearchCV to find best configuration for both clustered and non-clustered datasets.

Table 15: Decision Tree Regression Setup Summary

Parameter	With Clustering	Without Clustering
Library	`scikit-learn`	`scikit-learn`
Model Used	`DecisionTree Regressor`	`DecisionTree Regressor`
max_depth	20	20
min_samples_split	15	10
min_samples_leaf	6	6
max_features	None	None
Scoring Metric	R^2	R^2
Best R^2 Score	0.9641	0.9634

For evaluation preparation residual plots were generated to visualize prediction errors. Execution time was recorded. Regression metrics included R^2, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

3.3.10 Linear Regression

Linear Regression was implemented using LinearRegression from scikit-learn. Hyperparameter tuning was performed using GridSearchCV to find best configuration for both clustered and non-clustered datasets.

Table 16: Linear Regression Setup Summary

Parameter	With Clustering	Without Clustering
Library	`scikit-learn`	`scikit-learn`
Model Used	`Linear Regression`	`Linear Regression`
fit_intercept	True	True
positive	False	False
Scoring Metric	R^2	R^2
Best R^2 Score	0.2434	0.2434

For evaluation preparation residual plots were generated to visualize prediction errors. Execution time was recorded. Regression metrics included R^2, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

4 Design Specification

4.1 Clustering

A Gaussian Mixture Model (GMM), is a probabilistic model that assumes your data is generated from a mixture of several Gaussian distributions, each representing a cluster. Unlike hard clustering methods like k-means, GMMs perform soft clustering, assigning probabilities to each data point for belonging to each cluster. Before using GMM, Principal Component Analysis (PCA) was used to help clustering algorithms like GMM find better groupings. Principal Component Analysis (PCA) is a dimensionality reduction method. It transforms your original features (like u, g, r, i, z, redshift, plate) into a new set of features called principal components. These components capture the most important patterns in the data while reducing noise and redundancy.

4.2 Hyper parameters

To optimize model performance, a combination of hyper-parameter tuning strategies was employed across different classifiers. For models such as Support Vector Machines (SVM), Convolutional Neural Network (CNN), and Support Vector Regression (SVR), a systematic approach using GridSearchCV with 5-fold cross-validation was applied. This allowed for an exhaustive search over predefined parameter grids. For other models, including the Random Forest Classifier and K Nearest Neighbors Classifier a manual loop was executed to iterate over a range of n_estimators values (from 1 to 19), recording accuracy for each configuration and selecting the best-performing setup. Some models, such as the Decision Tree Classifier and Naive Bayes, were implemented using default parameters to serve as baselines. This mixed approach balanced computational efficiency with thorough exploration, ensuring each model was tuned appropriately for both clustered and non-clustered datasets.

5 Implementation

5.1 Transformed Data

The raw SDSS dataset was cleaned and preprocessed, including handling missing values, normalization of features, outliers elimination, reassembling for class balance, categorical encoding, and selection of attributes (u, g, r, i, z, redshift, plate). Later for comparison second dataset was created with integrated clustering.

5.2 Evaluation Results

Each regression model was evaluated using metrics such as Speed, R^2, MSE, RMSE, MAE. Each classification model was evaluated using metrics such as Speed, Accuracy, Precision, Recall, F1 Score.

5.3 Documentation

Detailed report was compiled. Including methodology, results, and comparative analysis of model performance.

5.4 Training Setup

5.4.1 Programming Environment

For this study Jupiter Notebook was used.

5.4.2 Programming Language

For this study Python 3.x was used as programming language.

5.4.3 Libraries Imported

Table 17: Imported Libraries and Tools Used in Thesis

Category	Libraries / Tools
Data Handling	numpy, pandas
Model Selection	train_test_split, GridSearchCV
Clustering	Gaussian Mixture Model (GMM) and Principal Component Analysis (PCA)
Preprocessing	StandardScaler, LabelEncoder, SMOTE, make_pipeline
Visualization	matplotlib.pyplot, seaborn
Classification Models	SVC, LinearSVC, KNeighborsClassifier, RandomForestClassifier, DecisionTreeClassifier, GaussianNB
Regression Models	SVR, LinearSVR, DecisionTreeRegressor, RandomForestRegressor, LinearRegression
Deep Learning	tensorflow, Keras layers (Dense, Dropout, Conv1D, etc.), Sequential, to_categorical
Model Wrapping	scikeras (KerasClassifier)
Evaluation Metrics	accuracy_score, confusion_matrix, precision_score, recall_score, f1_score, classification_report, r2_score
Outlier Detection	LocalOutlierFactor
Warnings Handling	ConvergenceWarning, warnings
Utilities	datetime, sklearn.pipeline, sklearn.exceptions

5.4.4 Hardware Specification

Processor: AMD Ryzen 7 7800X3D 8-Core Processor, 4201 Mhz, 8 Core(s), 16 Logical Processor(s)
OS Name: Microsoft Windows 11 Home
Total Physical Memory: 31.1 GB
Hard Drive: SSD

6 Evaluation

This section presents a detailed evaluation of the machine learning models applied to both classification and regression tasks on the Sloan Digital Sky Survey (SDSS17) dataset. The performance of each model was assessed using a range of metrics. For classification, metrics such as accuracy, precision, recall, F1 score, and execution time were computed using weighted averages to account for class imbalance and confusion matrix was generated for each model. For regression, evaluation included R^2 score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), execution time, and the residual plot was created. Additionally, the impact of clustering as a preprocessing step was analysed by comparing model performance on clustered versus non-clustered datasets. The results are presented in tabular form to provide clear comparison and interpretation.

6.1 Classification Models

Table 18: Classification Model Performance Comparison (Without Clustering)

Algorithm	Speed	Accuracy	Precision	Recall	F1 Score
Random Forest Classifier	00:00:04.44	98.322%	0.983273	0.983229	0.983220
K Nearest Neighbors Classifier	00:00:00.85	97.648%	0.976705	0.976480	0.976355
CNN	00:06:34.25	97.361%	0.973568	0.973609	0.973563
Decision Tree Classifier	00:00:01.36	97.275%	0.972760	0.972747	0.972741
Support Vector Machines	00:00:00.34	94.664%	0.946934	0.946643	0.946549
Naive Bayes	00:00:00.02	91.465%	0.915836	0.914652	0.914548

Table 19: Classification Model Performance Comparison (With Clustering)

Algorithm	Speed	Accuracy	Precision	Recall	F1 Score
Random Forest Classifier	00:00:04.85	98.314%	0.983173	0.983143	0.983132
K Nearest Neighbors Classifier	00:00:00.92	97.605%	0.976277	0.976050	0.975921
CNN	00:08:05.78	97.427%	0.974240	0.974269	0.974232
Decision Tree Classifier	00:00:01.39	97.344%	0.973443	0.973436	0.973433
Support Vector Machines	00:00:00.39	95.095%	0.950908	0.950951	0.950884
Naive Bayes	00:00:00.02	94.033%	0.941266	0.940325	0.940391

Table 19 and Table 18 present a performance comparison of classification models trained on clustered and non-clustered datasets. The integration of clustering techniques into the classification pipeline provide improvements across several models. While overall accuracy remained relatively stable, slight increase in precision, recall, and F1 score were observed in models such as CNN and Support Vector Machines, suggesting enhanced sensitivity to underlying data structures. For instance, CNN F1 score increased from 0.973563 to 0.974232, and SVM accuracy increased from 94.664% to 95.095%. These change indicate that clustering may help models better capture latent patterns in complex or imbalanced datasets. Interestingly, Naive Bayes and KNN also showed slight improvements in recall and precision, showing that clustering can enhance performance. However, the Random Forest Classifier maintained nearly identical performance, demonstrating that its ensemble nature already captures much of the data variability. Overall, the results support the presence of clustering as a valuable preprocessing step, especially for models sensitive to data distribution.

Table 20: Precision Comparison of Classification Models

Algorithm	With Clustering	Without Clustering
Random Forest	0.983173	0.983273
Naive Bayes	0.941266	0.915836
Decision Tree	0.973443	0.972760
CNN	0.974240	0.973568
KNN	0.976277	0.976705
SVM	0.950908	0.946934

Table 21: Accuracy Comparison of Classification Models

Algorithm	With Clustering	Without Clustering
Random Forest	98.314%	98.322%
Naive Bayes	94.033%	91.465%
Decision Tree	97.344%	97.275%
CNN	97.427%	97.361%
KNN	97.605%	97.648%
SVM	95.095%	94.664%

Table 22: Speed Comparison of Classification Models

Algorithm	With Clustering	Without Clustering
Random Forest	00:00:04.854	00:00:04.448
Naive Bayes	00:00:00.024	00:00:00.020
Decision Tree	00:00:01.394	00:00:01.364
CNN	00:08:05.782	00:06:34.250
KNN	00:00:00.926	00:00:00.852
SVM	00:00:00.392	00:00:00.343

Table 23: Recall Comparison of Classification Models

Algorithm	With Clustering	Without Clustering
Random Forest	0.983143	0.983229
Naive Bayes	0.940325	0.914652
Decision Tree	0.973436	0.972747
CNN	0.974269	0.973609
KNN	0.976050	0.976480
SVM	0.950951	0.946643

Table 24: F1 Score Comparison of Classification Models

Algorithm	With Clustering	Without Clustering
Random Forest	0.983132	0.983220
Naive Bayes	0.940391	0.914548
Decision Tree	0.973433	0.972741
CNN	0.974232	0.973563
KNN	0.975921	0.976355
SVM	0.950884	0.946549

Table 25: Confusion Matrix of Random Forest Classifier with Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11547	193	39
Quasar	355	11127	0
Star	0	0	11561

Table 26: Confusion Matrix of Random Forest Classifier without Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11558	188	33
Quasar	362	11120	0
Star	1	0	11560

Table 27: Confusion Matrix of Decision Tree Classifier with clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11280	468	31
Quasar	397	11085	0
Star	28	1	11532

Table 28: Confusion Matrix of Decision Tree Classifier without Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11251	497	31
Quasar	397	11085	0
Star	24	0	11537

Table 29: Confusion Matrix of K Nearest Neighbors Classifier with Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11109	397	273
Quasar	121	11355	6
Star	34	3	11524

Table 30: Confusion Matrix of K Nearest Neighbors Classifier without Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11119	388	272
Quasar	117	11359	6
Star	33	3	11525

Table 31: Confusion Matrix of Support Vector Machines Classifier with Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	10816	831	132
Quasar	663	10819	0
Star	82	0	11479

Table 32: Confusion Matrix of Support Vector Machines Classifier without Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	10617	1029	133
Quasar	583	10899	0
Star	112	1	11448

Table 33: Confusion Matrix of Naive Bayes Classifier with Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11105	573	101
Quasar	1220	10262	0
Star	91	93	11377

Table 34: Confusion Matrix of Naive Bayes Classifier without Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	9923	1752	104
Quasar	1012	10470	0
Star	43	61	11457

Table 35: Confusion Matrix of Convolutional Neural Network (CNN) Classifier with Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11360	311	108
Quasar	464	11017	1
Star	12	0	11549

Table 36: Confusion Matrix of Convolutional Neural Network (CNN) Classifier without Clustering

Actual / Predicted	Galaxy	Quasar	Star
Galaxy	11342	313	124
Quasar	472	11008	2
Star	8	0	11553

6.2 Regression Models

Table 37: Regression Model Performance Comparison (With Clustering)

Algorithm	Speed	R^2	MSE	RMSE	MAE
Random Forest Regressor	00:06:14.88	0.976096	0.016021	0.126576	0.032079
Decision Tree Regressor	00:00:01.15	0.964105	0.024058	0.155106	0.032549
Support Vector Regression	00:00:00.08	0.243453	0.507058	0.712080	0.578549
Linear Regression	00:00:00.01	0.243407	0.507089	0.712102	0.578461

Table 38: Regression Model Performance Comparison (Without Clustering)

Algorithm	Speed	R^2	MSE	RMSE	MAE
Random Forest Regressor	00:05:59.63	0.975925	0.016136	0.127026	0.032488
Decision Tree Regressor	00:00:01.13	0.963368	0.024552	0.156691	0.032306
Support Vector Regression	00:00:00.07	0.243372	0.507112	0.712118	0.579537
Linear Regression	00:00:00.00	0.243360	0.507120	0.712124	0.579509

Table 37 and Table 38 present a performance comparison of regression models trained on clustered and non-clustered datasets. The comparative analysis of regression models reveals that clustering integration has a modest but consistent impact on performance. The Random Forest Regressor show as the top performer in both scenarios, achieving an R^2 score of 0.9761 with clustering and 0.9759 without, and the lowest MSE, RMSE, and MAE values. This highlights

the good performance capability across different data distributions. The Decision Tree Regressor also benefited slightly from clustering, improving its R^2 from 0.9634 to 0.9641 and reducing error metrics slightly. In contrast, Support Vector Regression and Linear Regression showed bad performance between the two setups, both maintaining low R^2 scores (0.243) and high error rates, indicating limited suitability for this task regardless of clustering. Overall, clustering enhances model precision slightly for tree-based regressor, while linear models remain unaffected. These findings support the use of clustering as a preprocessing step when deploying ensemble or tree-based regression algorithms.

Table 39: Execution Time Comparison of Regression Models

Algorithm	**With Clustering**	**Without Clustering**
Random Forest Regressor	00:06:14.882617	00:05:59.636276
Decision Tree Regressor	00:00:01.150720	00:00:01.131381
Support Vector Regression	00:00:00.086013	00:00:00.072010
Linear Regression	00:00:00.012513	00:00:00.009501

Table 40: R^2 Score Comparison of Regression Models

Algorithm	**With Clustering**	**Without Clustering**
Random Forest Regressor	0.976096	0.975925
Decision Tree Regressor	0.964105	0.963368
Support Vector Regression	0.243453	0.243372
Linear Regression	0.243407	0.243360

Table 41: Mean Squared Error (MSE) Comparison of Regression Models

Algorithm	With Clustering	Without Clustering
Random Forest Regressor	0.016021	0.016136
Decision Tree Regressor	0.024058	0.024552
Support Vector Regression	0.507058	0.507112
Linear Regression	0.507089	0.507120

Table 42: Root Mean Squared Error (RMSE) Comparison of Regression Models

Algorithm	With Clustering	Without Clustering
Random Forest Regressor	0.126576	0.127026
Decision Tree Regressor	0.155106	0.156691
Support Vector Regression	0.712080	0.712118
Linear Regression	0.712102	0.712124

Table 43: Mean Absolute Error (MAE) Comparison of Regression Models

Algorithm	With Clustering	Without Clustering
Random Forest Regressor	0.032079	0.032488
Decision Tree Regressor	0.032549	0.032306
Support Vector Regression	0.578549	0.579537
Linear Regression	0.578461	0.579509

6.3 Comparison Analysis

Classification Models: Random Forest Classifier consistently outperforms other models in terms of accuracy, precision, recall, and F1 score, achieving over 98.3% accuracy in both clustered and non-clustered scenarios. KNN and CNN follow closely, with CNN showing slightly improved metrics when clustering is

applied. Notable, Naive Bayes benefits significantly from clustering, with its accuracy rising from 91.47% to 94.03%, suggesting that clustering may enhance performance for some models. Support Vector Machines also show a modest improvement in all metrics post-clustering.

Regression Models: Random Forest Regressor again with the highest R^2 score (0.976), lowest MSE (0.016), and minimal MAE (0.032), regardless of clustering. Decision Tree Regressor performs well but slightly below Random Forest. Linear Regression and Support Vector Regression exhibit poor performance across all regression metrics, with R^2 scores around 0.243 and high error values (MSE ¿ 0.5, RMSE ¿ 0.71, MAE ¿ 0.57), indicating limited suitability for the dataset used. Clustering has small impact on regression performance, except for marginal improvements in execution time and error reduction.

Overall Insight: Random Forest emerges as the best performer and reliable model across both classification and regression tasks. Clustering provides slight improvements in classification metrics for Naive Bayes and SVM, but has minimal influence on regression.

7 Conclusion and Future Work

This study investigated the performance of various machine learning models on SDSS photometric data and evaluated the impact of clustering on both classification and regression tasks. The results demonstrate that clustering can enhance some models performance by improving data structure, reducing noise, and revealing hidden patterns. Artificial Intelligence proves to be a highly effective and scalable solution for stellar classification, capable of handling large datasets with precision and speed. In classification, the Random Forest Classifier trained on non-clustered data achieved the highest accuracy of 98.32%, outperforming all other models. However, lightweight models such as Naive Bayes benefited significantly from clustering, with accuracy increasing from 91.46% to 94.03%. The Convolutional Neural Network (CNN) also showed a modest improvement, achieving 97.43% accuracy on clustered data compared to 97.36% without clustering. In regression tasks, all models with clustering dataset got slightly better results indicating that clustering contributes positively to predictive accuracy across different models.

Future enhancements could include applying Bayesian optimisation or genetic algorithms for more efficient hyper-parameter tuning. Applying pre-trained models from similar astronomical datasets may accelerate learning and improve accuracy. Additionally, deploying models on cloud platforms with distributed computing could enable real-time classification of massive datasets. Try different clustering algorithms like DBSCAN, K-Means or Spectral Clustering which may uncover different structures in the data.

8 Acknowledgments

I would like to express my appreciation to my supervisor Prof. Devanshu Anand, for his guidance and support throughout the course of this research. His feedback was helpful in shaping the direction and quality of this thesis.

I am also thankful to the faculty and staff at National College of Ireland for providing the resources and environment necessary for academic growth.

Special thanks to the Higher Education Authority (HEA) and the Springboard+ initiative for supporting my participation in this program. Their commitment to providing accessible, high-quality education has enabled me to pursue postgraduate Computer Science in AI and participate in this research.

Thanks to fedesoriano for Dataset, fedesoriano (January 2022). Stellar Classification Dataset - SDSS17. Retrieved [June 2025] from
https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17.

Finally, I am grateful to my family and friends for their patience and encouragement throughout this journey.

References

Rubin Observatory 2025, Data,
https://rubinobservatory.org/explore/how-rubin-works/technology/data

Astrostatistics 2025, Large Synoptic Survey Telescope (LSST), Data,
https://www.cfa.harvard.edu/research/topic/astrostatisticsDeen

Omat, Jood Otey, Amjed Al-Mousa (2022), " Stellar Objects Classification Using Supervised Machine Learning Techniques",
2022 International Arab Conference on Information Technology (ACIT),
https://doi.org/10.1109/ACIT57182.2022.9994215

Zhuliang Qi (2022), " Stellar Classification by Machine Learning " , August 2022SHS Web of Conferences 144(3):03006,
http://dx.doi.org/10.1051/shsconf/202214403006

Tanvi Mehta, Nishi Bhuta, Swati Shinde (2022), "Experimental Analysis of Stellar Classification by using Different Machine Learning Algorithms", 2022 International Conference on Industry 4.0 Technology (I4Tech),
https://doi.org/10.1109/I4Tech55392.2022.995296

Mehmet Bilge Han Taş, Eyyüp Yıldız (2024) "Stellar Classification with Machine Learning Algorithms: Graph-Based Feature Selection Approach", In In-

ternational Studies and Evaluations in Computer Engineering,
Stellar Classification with Machine Learning

SDSS17 Sloan Digital Sky Survey Dataset (2021), SDSS17 Sloan Digital Sky Survey Dataset (2021)

Sabeesh Ethiraj, Bharath Kumar Bolla (2022) "Classification of Quasars, Galaxies, and Stars in the Mapping of the Universe Multi-modal Deep Learning", Presented at Deep Learning Developers Conference, 2021, Bangalore
https://arxiv.org/abs/2205.10745

Ahmed Taha Hassina (2023), Using machine learning to classify and localize stellar objects

Anisha Bhat, Amisha Mallick, Vishvapriya Sangvikar, Shreya Gholap (2023), "Stellar classification and analysis using Vision Transformer", Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune,India, Stellar classification and analysis using Vision Transformer

Michael Brice, Răzvan Andonie (2019), "Classification of Stars using Stellar Spectra collected by the Sloan Digital Sky Survey", Classification of Stars using Stellar Spectra collected by the Sloan Digital Sky Survey

Jing-Hang Shi , Bo Qiu , A-Li Luo , Zhen-Dong He , Xiao Kong , Xia Jiang (2022) , "A photometry pipeline for SDSS images based on convolutional neural networks", A photometry pipeline for SDSS images

Johanna Pasquet 1 , E. Bertin 2 , M. Treyer 3 , S. Arnouts 3 , D. Fouchez (2018) , "Photometric redshifts from SDSS images using a convolutional neural network", Photometric redshifts from SDSS images

M. Treyer, R. Ait-Ouahmed, J. Pasquet, S. Arnouts, E. Bertin, D. Fouchez (2023), "CNN photometric redshifts in the SDSS at $r \leq 20$", CNN photometric redshifts in the SDSS at $r \leq 20$

Krishna Chunduri, Mithun Mahesh (2024), "Deep Learning Approach to Photometric Redshift", Deep Learning Approach to Photometric Redshift Estimation