IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

Star Predictive Analysis: Machine Learning-Based Stellar Classification Using a Random Forest Pipeline and Flask Deployment

S V Sivaraman[1], Tanish Selvakumaran[2], Vaibhav Shankar[3], Navadeep Bammidi[4]

Student, Department of CSE (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[1,2,3,4]

Abstract: Classifying stars into evolutionary categories normally depends on spectroscopy and expert interpretation, which does not scale to modern survey volumes. This paper presents Star Predictive Analysis, a supervised machine learning system that classifies stars into six types (Brown Dwarf, Red Dwarf, White Dwarf, Main Sequence, Supergiant and Hypergiant) from six tabular observational and physical parameters: temperature, luminosity, radius, absolute magnitude, spectral class and star colour. A scikit-learn Pipeline couples a ColumnTransformer (StandardScaler for numerical and OneHotEncoder for categorical features) with a Random Forest of 100 trees, and is benchmarked against a Decision Tree and Logistic Regression on the 240-record Kaggle Star Dataset using a stratified 80/20 split. In our local experiments the Random Forest reached the highest accuracy of 93.77%, ahead of Logistic Regression (90.00%) and the Decision Tree (89.21%). The trained pipeline is deployed as a Flask web application with REST endpoints for training, real-time prediction with class-probability output, and an analytics dashboard. We also discuss the limits of a small, curated dataset and outline paths towards larger catalogues and explainable models.

Keywords: stellar classification, Random Forest, ensemble learning, scikit-learn pipeline, Hertzsprung–Russell diagram, Flask deployment

I. INTRODUCTION

Stellar classification is one of the oldest organising tasks in astrophysics. The Morgan–Keenan (MK) system assigns a star a spectral type (O, B, A, F, G, K, M) and a luminosity class by comparing absorption-line profiles in its spectrum against standards [8]. The procedure is reliable but slow, partly subjective between observers, and increasingly hard to sustain as sky surveys deliver millions of objects.

Machine learning offers a practical alternative. Deep networks have been applied to raw spectra and photometric images with good results [1]–[3], but these methods need large labelled image or spectral archives and considerable compute. In many settings, such as teaching, quick triage or catalogue cross-checks, an observer already holds a handful of derived physical parameters and simply wants the evolutionary category of the star. For such tabular data, tree-based ensembles are a natural fit: they handle mixed numerical and categorical inputs, need little tuning, and expose feature importances that can be checked against astrophysical expectations.

This paper describes Star Predictive Analysis, an end-to-end system that classifies stars into six evolutionary categories from six input features. The main contributions are:

- a reproducible scikit-learn Pipeline that couples feature-specific preprocessing with a 100-tree Random Forest classifier;
- a benchmark of the Random Forest against a single Decision Tree and Logistic Regression on an identical stratified split;
- a Flask web application with REST endpoints for dataset access, retraining and real-time prediction with per-class probabilities; and
- an automated test suite and input-validation layer that keep the deployed service robust to malformed or out-of-range inputs.

The rest of the paper is organised as follows. Section II reviews related work, Section III describes the dataset and preprocessing, Section IV presents the system design, Section V details the implementation, Section VI gives the experimental setup, Section VII reports and discusses results, and Section VIII concludes.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

II. RELATED WORK

A. Deep Learning for Spectra and Images

Sharma *et al.* trained convolutional neural networks on spectra from the Jacoby Atlas, ELODIE and MILES libraries and reduced the classification error to about 1.23 spectral subclasses, showing that learned features can outperform hand-crafted ones [1]. Kuntzer, Tewes and Courbin combined Principal Component Analysis with artificial neural networks to predict spectral types from single-band images, with typical errors near half a spectral class in simulation [2]. A later CNN study on photometric images produced a catalogue of more than 50 million SDSS stars without spectra, reaching 86.1% accuracy over the O–M classes [3]. These works address spectral type from images or spectra, not the evolutionary category used here.

B. Classical Classifiers

Liu compared Support Vector Machines, Twin SVMs and a Twin Hypersphere Model for spectral classification and reported better handling of noise, although the evaluation concentrated on selected spectral types [4]. Such studies typically target the Harvard spectral sequence rather than categories such as Brown Dwarf, White Dwarf or Hypergiant.

C. Tree Ensembles and Software Tooling

Random Forests, introduced by Breiman, average many decorrelated decision trees grown on bootstrap samples with random feature subsets [5]. They give strong accuracy on tabular data, resist overfitting relative to a single tree, and provide impurity-based feature importances. The scikit-learn library supplies production-quality implementations together with the Pipeline and ColumnTransformer abstractions that keep preprocessing inside the model object [6]. The Star Dataset used in this work was curated and released on Kaggle by Baidya [7].

D. Gap Addressed

Few reports combine (i) evolutionary-category classification from tabular physical parameters, (ii) a transparent comparison of tree and linear baselines, and (iii) a deployed, testable web interface that returns class probabilities. This work targets that combination.

TABLE I REPRESENTATIVE PRIOR WORK AND THIS STUDY

Work	Input	Method	Reported outcome
[1]	Spectra	CNN	Error of about 1.23 spectral subclasses
[2]	Single-band images	PCA + ANN	Error of about half a spectral class
[3]	Photometric images	CNN	86.1% accuracy, O–M classes
[4]	Spectra	SVM / TWSVM / THM	Improved noise handling
This work	Tabular physical parameters	Random Forest	93.77% accuracy, six evolutionary classes

The outcomes in Table I use different data, label sets and metrics and are listed for context only; they are not a like-for-like ranking.

III. DATASET AND PREPROCESSING

A. Dataset Description

We use the Star Dataset to Predict Star Type [7], which contains 240 stellar records. The target is the integer Star type (0–5), mapped to Brown Dwarf, Red Dwarf, White Dwarf, Main Sequence, Supergiant and Hypergiant. The classes are perfectly balanced with 40 records each. The dataset has no missing values and no duplicate records. Table II summarises the classes and Table III lists the six input features.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

TABLE II STAR CLASSES IN THE DATASET

Class	Physical description	Records
Brown Dwarf	Substellar object, too light for sustained hydrogen fusion; cool and very faint	40
Red Dwarf	Low-mass, cool M-type star; faint but long-lived	40
White Dwarf	Hot, dense remnant core of a low-mass star; small radius, low luminosity	40
Main Sequence	Star fusing hydrogen in its core, spanning a wide range of temperature and luminosity	40
Supergiant	Massive, evolved star with very large radius and high luminosity	40
Hypergiant	Extremely luminous, very large and short-lived star at the top of the HR diagram	40

TABLE III INPUT FEATURES AND THEIR TREATMENT

Feature	Type	Preprocessing
Temperature (K)	Numerical	StandardScaler
Luminosity ($L/L_\odot$)	Numerical	StandardScaler
Radius ($R/R_\odot$)	Numerical	StandardScaler
Abs. magnitude (M_v)	Numerical	StandardScaler
Spectral class	Categorical	OneHotEncoder (O, B, A, F, G, K, M)
Star colour	Categorical	OneHotEncoder (9 canonical labels)

The numerical features are physically linked. For a star radiating as a black body, the Stefan–Boltzmann law relates them as

$$L/L_\odot = (R/R_\odot)^2 \, (T/T_\odot)^4 \quad (1)$$

and absolute magnitude falls by 2.5 for every factor of ten in luminosity (ignoring bolometric corrections). Luminosity, radius and magnitude therefore carry overlapping information, which suits tree-based models that can split on whichever feature separates a class most cleanly. Plotted on a Hertzsprung–Russell (HR) diagram, the six categories occupy different regions of temperature and luminosity, which is the physical basis for expecting a good classifier.

On the HR diagram, white dwarfs sit at the hot, faint lower left; brown and red dwarfs at the cool, faint lower right; main-sequence stars along the diagonal band between them; and supergiants and hypergiants in the bright upper region, with hypergiants reaching the largest radii. Because these regions are defined mainly by luminosity, radius and temperature, the numerical features carry most of the discriminating signal, while colour and spectral class mostly refine it. Regions do overlap near their boundaries, which is where classification errors are expected to concentrate.

B. Cleaning

The raw colour strings are inconsistent (for example “Blue White”, “Blue-white” and “blue white”). The function clean_star_color() lowercases, strips hyphens and maps each value to one of nine canonical labels using pattern matching, while clean_spectral_class() accepts only the seven Harvard letters and falls back to M otherwise. Missing numerical values, if present in new data, are filled with the column median.

C. Feature Transformation

A ColumnTransformer applies different treatments to different columns and drops everything else (remainder=‘drop’), which prevents the target from leaking into the inputs. Each numerical feature x is standardised to

$$z = (x - \mu) / \sigma \quad (2)$$

where μ and σ are estimated on the training partition only. Categorical features are one-hot encoded with handle_unknown=‘ignore’, so an unseen colour label at prediction time produces an all-zero block instead of an error.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

D. Leakage Control

Three design choices guard against optimistic estimates. The target column is excluded from the input feature list and the ColumnTransformer discards all unlisted columns. The scaler and encoder are fitted inside the Pipeline on the training partition only, then applied unchanged to the test partition and to later user input. Finally, the split is stratified so that every class appears in both partitions in equal proportion. Even so, with so few records the held-out estimate remains noisy, a point revisited in Section VII.

IV. PROPOSED SYSTEM

A. Pipeline Overview

Fig. 1 shows the processing chain. The preprocessor and the classifier are wrapped in a single scikit-learn Pipeline object, so that fitting, prediction and serialisation (via joblib) always use identical transformations, and so that scaling statistics are never computed from test data.

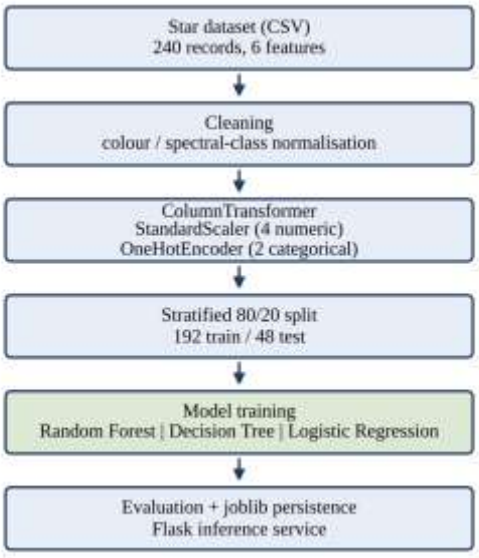


Fig. 1. End-to-end machine learning pipeline.

B. Classifiers

Random Forest (primary model). The forest contains T = 100 trees with max_depth=None, min_samples_split=2 and random_state=42. Each tree splits nodes by reducing Gini impurity,

$$G = 1 - \Sigma_k \, p_k^2 \qquad (3)$$

where p_k is the fraction of class k at the node. The ensemble prediction averages the per-tree class probabilities,

$$P(k|x) = (1/T) \, \Sigma_{t=1}^{T} \, P_t(k|x) \quad (4)$$

and the predicted class is the arg max over k. These averaged probabilities are exposed to the user as the confidence distribution. Impurity decrease aggregated per feature gives the feature importance used for interpretation.

Decision Tree (baseline). A single tree of the same family, trained in an identical pipeline, shows how much the ensemble adds over one high-variance learner.

Logistic Regression (linear benchmark). A multinomial logistic model with max_iter=1000 tests whether the classes are separable by smooth linear boundaries in the standardised feature space.

C. Web Application

The trained pipeline is served by a Flask application organised in three layers (Fig. 2). The client layer provides six pages: a dashboard, dataset explorer, model training view, star prediction workspace, analytics dashboard and an about page. The server layer exposes seven REST endpoints covering dataset access, retraining, prediction, demonstration presets

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

and analytics aggregation, and automatically trains and stores a model on start-up if none exists. The data and model layer holds the dataset, the fitted ColumnTransformer and the Random Forest in a single joblib artifact.

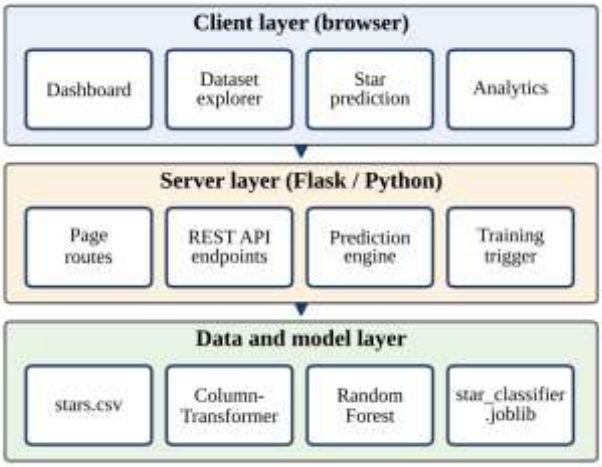


Fig. 2. Three-layer web application architecture.

Before inference, validate_and_sanitize_inputs() range-checks every numerical field (for example, temperature must lie between 500 K and 150,000 K) and cleans the categorical fields. Invalid input raises a ValueError that the API returns as HTTP 422; unexpected failures return HTTP 500. A seven-test automated suite checks that every demonstration preset is classified into its expected class with confidence above 70%, that probabilities sum to 100% within tolerance, and that out-of-range or non-numeric inputs fail cleanly.

D. Inference Procedure

Algorithm 1 summarises how a single prediction request is processed. The same cleaning and transformation steps used in training are applied to user input, so a request cannot reach the classifier in a form it has not seen.

Algorithm 1. Single-star inference

Input: JSON payload with six stellar parameters
Output: class name, confidence, probability vector
1. Load the saved artifact; if none exists, train and save one
2. Range-check temperature, luminosity, radius and magnitude
3. Clean star colour and spectral class strings
4. If any check failed, raise ValueError (HTTP 422)
5. Build a one-row DataFrame with the training column names
6. $p \leftarrow$ pipeline.predict_proba(row)
7. class $\leftarrow$ arg max of p; confidence $\leftarrow 100 \times \max(p)$
8. Attach description and HR-diagram positioning note
9. Return JSON with status, class, confidence and p

V. IMPLEMENTATION

A. Software Stack and Code Organisation

The system is implemented in Python. Data handling uses pandas and NumPy, modelling uses scikit-learn, persistence uses joblib, and the web layer uses Flask with Jinja2 templates; the browser interface renders charts with Chart.js. Code is split by responsibility. The preprocessing module holds the feature lists, canonical label sets, cleaning functions and the ColumnTransformer builder. The training module performs the split, fits the three pipelines, computes metrics and writes the artifact. The prediction module validates input, runs inference and generates an explanation. The application module defines the routes. Tests live in a separate file.

B. Model Artifact and Start-up Behaviour

Training stores a single joblib artifact containing the fitted pipeline together with its metrics, confusion matrix, class labels, aggregated feature importances, the comparison results and a training timestamp. The web pages draw on these

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

stored values, so the metrics shown in the interface come from the deployed model itself. If the artifact is missing at start-up, the application trains and saves one automatically, which makes a fresh deployment self-contained. A retraining endpoint lets a user refresh the model from the interface and returns the new metrics in its response.

C. Error Handling and Demonstration Presets

Errors are separated by cause. Missing or malformed JSON returns HTTP 400, invalid parameter values return HTTP 422 with a message naming each offending field, and unexpected exceptions are logged and returned as HTTP 500. The interface also provides one preset observation per class. Loading a preset fills the form with representative values so that a user can demonstrate all six outcomes without typing parameters, and the same presets serve as fixtures for the automated tests.

D. Testing

The project includes seven unit and integration tests. Two are representative of the checks performed. The preset test asserts both the expected class and a confidence above 70%, and verifies that the six probabilities sum to 100% within a tolerance of one point. The validation tests assert that out-of-range and non-numeric temperatures raise ValueError. Running the suite after every change guards against regressions in the preprocessing contract between training and inference.

VI. EXPERIMENTAL SETUP

The 240 records were divided with a stratified 80/20 split and random_state=42, giving 192 training and 48 test records with equal class representation (8 test stars per class). All three pipelines were fitted on the same training partition and evaluated on the same held-out records. The principal metric is classification accuracy,

$$\text{Accuracy} = (1/N)\,\Sigma_{i=1}^{N}\,1[\,\hat{y}_i = y_i\,] \quad (5)$$

where N is the number of evaluated samples, $\hat{y}_i$ the predicted and y_i the true class. Experiments were run locally on a laptop using the released Python code, scikit-learn and pandas, and the complete source, data and tests are available in the project repository.

TABLE IV MODEL CONFIGURATION

Model	Key settings
Random Forest	100 trees, max_depth=None, min_samples_split=2, random_state=42
Decision Tree	Single tree, scikit-learn default settings
Logistic Regression	max_iter=1000, standardised inputs

VII. RESULTS AND DISCUSSION

A. Model Comparison

Table V and Fig. 3 summarise the accuracy of the three models. The Random Forest achieved the highest accuracy at 93.77%. Logistic Regression followed at 90.00%, and the single Decision Tree obtained 89.21%. The Random Forest therefore leads the Decision Tree by 4.56 percentage points and Logistic Regression by 3.77 points.

TABLE V ACCURACY COMPARISON OF THE EVALUATED MODELS

Algorithm	Type	Accuracy (%)
Random Forest	Ensemble (bagging)	93.77
Decision Tree	Single tree	89.21
Logistic Regression	Linear model	90.00

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511106

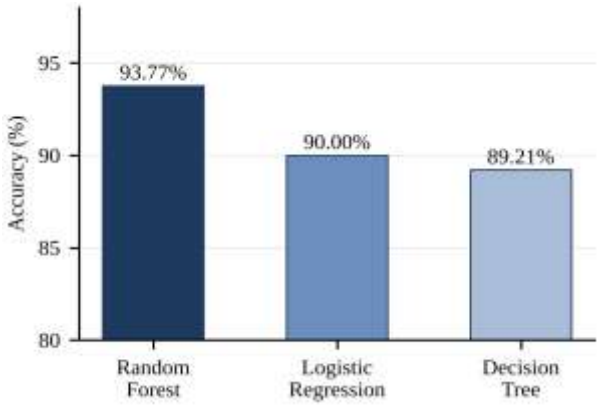


Fig. 3. Accuracy of the three classifiers.

B. Discussion

The ranking agrees with the usual behaviour of bagged ensembles. A single fully grown tree has high variance, and with only 192 training records its splits are sensitive to the particular sample. Averaging 100 decorrelated trees smooths those fluctuations, which is the most likely reason for the Random Forest's advantage over the Decision Tree. Logistic Regression performs slightly better than the single tree, which suggests that much of the class structure is captured by smooth boundaries once the numerical features are standardised, but it still trails the forest, whose axis-aligned splits can follow the irregular, non-linear shapes of the class regions on the HR diagram.

The accuracies reported here are lower than the near-perfect values often quoted for this dataset. This is plausible: the dataset is small, and the test set contains only 48 stars, so each misclassified star moves accuracy by roughly two percentage points and results are sensitive to the chosen split. We therefore treat the figures as indicative of relative model quality rather than as precise estimates of generalisation performance on real survey data.

For the Random Forest, the impurity-based importance computed by the pipeline is dominated by the physically informative parameters (absolute magnitude and radius), consistent with how the classes spread across the HR diagram. This agreement between statistical importance and astrophysical expectation is a useful sanity check on the model.

C. Prediction Example

The deployed interface was exercised with a star of temperature 4800 K, luminosity 2, radius 24, absolute magnitude 1, spectral class F and colour Blue-White. The system returned Main Sequence with 76% confidence, assigning the remaining probability mainly to Supergiant (13%), White Dwarf (7%) and Hypergiant (4%). The spread illustrates a benefit of the ensemble: the probability vector indicates how decisive a prediction is, and a flatter distribution flags borderline stars for expert review.

D. Limitations

The study has four main limitations. First, the dataset has only 240 records, with 40 per class, so it does not reflect the class imbalance and noise of real catalogues. Second, results come from a single stratified split on one machine; repeated splits or cross-validation would give variance estimates. Third, only accuracy is reported in this version; per-class precision, recall, F1-score and a confusion matrix are needed to show which categories are confused. Finally, the model uses derived physical parameters, so errors in those inputs propagate directly to the prediction.

E. Threats to Validity

Internal validity is affected by the single split and by the small test set, as noted above. Construct validity depends on the dataset labels, which were assigned by the dataset curator and are taken as ground truth here; any label noise would cap achievable accuracy for every model. External validity is limited because all classes are equally represented and the feature distributions come from a curated catalogue rather than a survey, so the numbers should not be read as expected performance on raw observational data. We kept all models near default hyperparameters to avoid tuning on the test set; a nested cross-validation search would be a fairer way to tune them in future work.

F. Practical Implications

Despite these caveats, the results support the premise that a lightweight tabular model can give useful first-pass evolutionary classification. The pipeline is small enough to train and run on a laptop, and the probability output lets a user judge how far to trust each answer. For teaching, the preset observations and astrophysical positioning notes connect

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511106

a numerical prediction to its place on the HR diagram, which helps students relate the model to stellar physics rather than treating it as a black box.

VIII. CONCLUSION AND FUTURE WORK

This paper presented a Random Forest-based stellar classification system built as a reproducible scikit-learn pipeline and deployed through a Flask web application. On the Star Dataset the Random Forest achieved 93.77% accuracy, outperforming Logistic Regression (90.00%) and a single Decision Tree (89.21%), while the application supplies real-time predictions with class probabilities, input validation and an automated test suite.

Future work includes (i) evaluating on larger catalogues such as SDSS and Gaia with more spectral subtypes and luminosity classes; (ii) repeated stratified cross-validation with full per-class metrics and confusion matrices; (iii) testing gradient-boosted trees and neural networks on tabular data; (iv) adding SHAP and LIME explanations; (v) classification directly from spectra using CNNs; and (vi) containerised cloud deployment.

ACKNOWLEDGMENT

The authors thank Dr. S. Anitha Jebamani, Assistant Professor, Department of CSE (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, for guidance during the course 21CSC305P Machine Learning, and Dr. P. Chitra, Professor and Head of the Department, for her encouragement.

REFERENCES

[1] K. Sharma, A. Kembhavi, A. Kembhavi, T. Sivarani, S. Abraham, and K. Vaghmare, “Application of convolutional neural networks for stellar spectral classification,” *Mon. Not. R. Astron. Soc.*, vol. 491, no. 2, pp. 2280–2300, 2019.

[2] T. Kuntzer, M. Tewes, and F. Courbin, “Stellar classification from single-band imaging using machine learning,” *Astron. Astrophys.*, vol. 591, A54, 2016.

[3] “Stellar classification with convolutional neural networks and photometric images: A new catalogue of 50 million SDSS stars without spectra,” *Mon. Not. R. Astron. Soc.*, 2023.

[4] Z. Liu, “Stellar spectra classification with twin hypersphere model,” *New Astron.*, vol. 89, 101644, 2021.

[5] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[6] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[7] D. Baidya, “Star dataset to predict star type,” Kaggle Datasets, 2020. [Online]. Available: https://www.kaggle.com/datasets/deepu1109/star-dataset

[8] W. W. Morgan, P. C. Keenan, and E. Kellman, *An Atlas of Stellar Spectra, with an Outline of Spectral Classification*. Chicago, IL, USA: Univ. Chicago Press, 1943.

 This work is licensed under a Creative Commons Attribution 4.0 International License