Stellar Classification and Feature Analysis Using the ELODIE Dataset

Esra Şekerci
Middle East Techinal University
Ankara, Turkey
esra.sekerci@metu.edu.tr

***Abstract*— This paper presents a comprehensive analysis of stellar classification using an expanded version of the ELODIE dataset. We implemented various machine learning models, including Decision Tree, Random Forest, Stochastic Gradient Descent (SGD) Classifier, and Artificial Neural Networks (ANN), to classify stars based on their physical and spectral properties. We employed robust evaluation metrics such as the ROC AUC score to handle class imbalance effectively. Our findings highlight the ANN's superior performance, achieving a weighted average ROC AUC score of 0.9741, significantly outperforming the baseline and other models. This study underscores the importance of feature engineering, data preprocessing, and model optimization in enhancing classification accuracy for complex astronomical datasets.**

Keywords—Artificial Neural Network, Classification, PCA, Random Forest, Classification, ROC AUC, Stochastic Gradient

I. INTRODUCTION

Stellar classification is a fundamental task in astronomy, aiding in the understanding of stellar properties and the underlying physics governing stellar evolution. The ELODIE dataset, originally published by Prugniel et al. in 2007, provides a rich source of stellar parameters for this purpose. This study leverages an expanded version of the ELODIE dataset, encompassing 1891 observations and 37 variables, to classify stars based on their spectral characteristics. We employ various machine learning algorithms to achieve this classification, focusing on techniques to address class imbalance and enhance model performance. Our methodology includes rigorous data preprocessing, exploratory data analysis, and model evaluation using robust statistical metrics.

II. METHODOLOGY

A. Dataset

The dataset utilized in this project is an expanded version of the ELODIE library, Version 3.1. Our dataset initially includes 1891 observations and 37 variables, capturing a broad array of stellar parameters and their corresponding quality metrics. The main data types include floating-point numbers, integers, and categorical strings. Each observation is intended to represent a unique star. However, upon further inspection, duplicate entries were identified. These duplicates were subsequently removed to streamline and improve the accuracy of the classification task.

The comprehensive nature of this dataset makes it ideal for detailed analysis and classification of stars based on their physical and spectral properties. Below is a description of the features included in the dataset.

Stellar parameters:

Units	Label	Explanations
---	name	Object identifier
km/s	vr	Heliocentric radial velocity
K	teff	Effective temperature
[cm/s+2]	logg	Decimal log of surface gravity
[Sun]	feh	Iron abundance relative to solar
0.1nm	ca4227	Ca4227 index
0.1nm	g4300	G4300 index
0.1nm	hgamA	H gamma A index
0.1nm	hgamF	H gamma F index
0.1nm	fe4383	Fe4383 index
0.1nm	ca4455	Ca4455 index
0.1nm	fe4531	Fe4531 index
0.1nm	fe4668	Fe4668 index
0.1nm	hbeta	Hbeta index
0.1nm	fe5015	Fe5015 index
mag	mg1	Mg1 index
mag	mg2	Mg2 index
0.1nm	mgb	Mgb index
0.1nm	fe5270	Fe5270 index
0.1nm	fe5335	Fe5335 index
0.1nm	fe5406	Fe5406 index
0.1nm	fe5709	Fe5709 index
0.1nm	fe5782	Fe5782 index
0.1nm	nad	NaD index
mag	tio1	TiO1 index
mag	tio2	TiO2 index
---	sp_type	Morgan-Keenan Spectral type
---	spec	Main spectral type (Target)

Observation quality parameters:

Units	Label	Explanations
---	q_sp_type	[A-E] Quality flag on SpType
---	q_logg	[-1,1] Quality flag on logg
---	q_feh	[-1,4] Quality flag on [Fe/H]
---	q_vr	[-1,0] Quality flag on Vr
[cm/s+2]	loggM	Estimated log of surface gravity
[Sun]	[Fe/H]M	Estimated [Fe/H]

B. *Exploratory Data Analysis and Preprocessing*

Our analysis begins with a detailed examination of the features to understand their distributions and identify potential correlations. Descriptive statistics reveal significant variability across the data, highlighted by some features containing erroneous values such as minimums of -99 in variables like logg and feh, which are likely placeholders for missing data.

To initiate this process, we identify the numerical columns within the dataset for correlation analysis. This step is pivotal as it guides subsequent feature engineering and selection processes. Utilizing a correlation matrix, we compute and visualize pairwise correlations among these numerical columns.

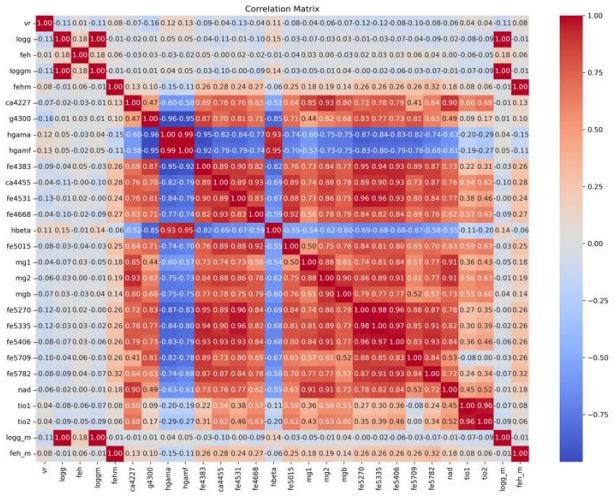

Figure 1-Correlation Heatmap

The heatmap illustrates the strength and direction of correlations between numerical features. Notably, strong correlations such as between g4300 and mgb (0.68), and fe5270 and fe4531 (0.87) are observed, suggesting potential redundancies within the dataset. These findings prompt considerations for feature engineering strategies aimed at reducing multicollinearity.

Further, we map the target 'spec' column to numerical values and calculate Pearson correlation coefficients between 'spec' (target) and all other numerical columns to pinpoint the most correlated features.

Correlations with 'spec':
fe5270: 0.868
fe4531: 0.865
g4300: 0.843
fe5335: 0.840
hgama: -0.826
fe4383: 0.819
hgamf: -0.806
mgb: 0.795
fe5406: 0.794
fe5015: 0.784
mg2: 0.776
fe4668: 0.755
ca4455: 0.744
fe5709: 0.710
fe5782: 0.666
ca4227: 0.665
hbeta: -0.661
nad: 0.6496
mg1: 0.569
tio2: 0.397
tio1: 0.324
vr: -0.171
fehm: 0.0179
feh_m: 0.018
feh: -0.009
loggm: 0.008
logg_m: 0.008
logg: 0.000

Results highlight fe5270 (0.87) and fe4531 (0.86) as the top positively correlated features with 'spec', while hgama (-0.83) exhibits a strong negative correlation. These insights are instrumental in guiding feature selection and engineering efforts to enhance model performance.

Understanding the distribution of our target class 'spec' provides additional context for model training and evaluation.

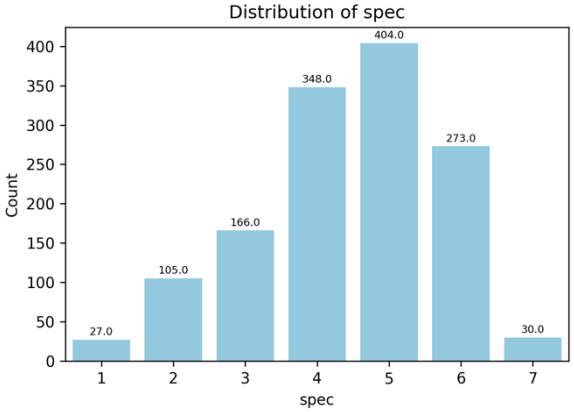

Figure 2-Distribution of Target Class

The distribution shows that category '5' is predominant with 404 occurrences, followed by '4' with 348 occurrences. Categories '6', '3', and '2' also exhibit notable frequencies of 273, 166, and 105 respectively, while categories '1' and '7' appear less frequently, with 27 and 30 occurrences respectively.

Following the correlation analysis, two distinct feature engineering approaches were considered to optimize the dataset for subsequent statistical modeling tasks. The first approach utilized a correlation matrix, and address highly correlated features. A threshold of 0.7 was set to determine highly correlated pairs. For each identified pair, the feature less correlated with the target variable was systematically removed.

Additionally, a second approach leveraged the Variance Inflation Factor (VIF) which quantifies the severity of multicollinearity by assessing how much the variance of a regression coefficient is inflated due to collinearity with other predictors. Features were iteratively evaluated and removed if their VIF exceeded a specified threshold at 10.

After careful evaluation and comparison of both approaches, the VIF-based method was chosen to reduce the dataset. This approach not only effectively addressed multicollinearity concerns but also provided a systematic framework for optimizing dataset quality. The resulting dataset consists of 1353 observations and 17 key features.

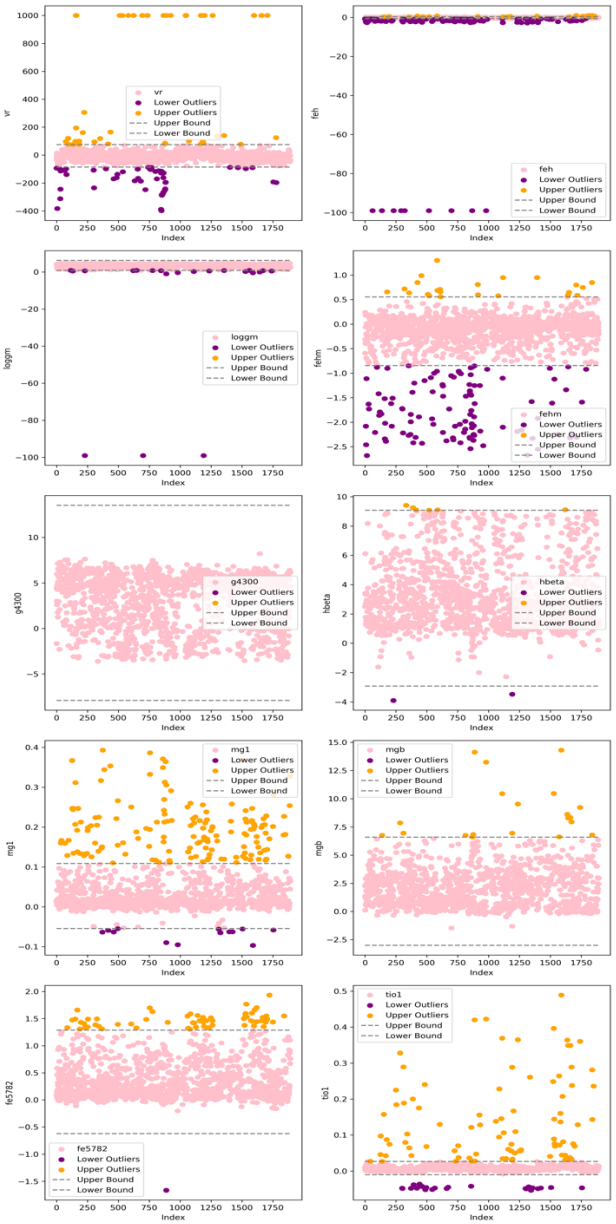

Figure 3- Outlier Analysis for Numerical Features

Furthermore, outliers were identified using the Interquartile Range (IQR) method, establishing upper and lower bounds based on quartiles. Despite their identification, we opted not to remove outliers from the dataset. This approach acknowledges outliers' potential to offer valuable insights into extreme or unusual stellar phenomena, ensuring our dataset remains comprehensive for subsequent statistical modeling tasks.

C. Missingness

In the context of statistical analysis, understanding the missingness mechanism is pivotal as it directly impacts the reliability and interpretability of the data. Initially, we identify missing values in specific features (q_logg, q_feh, q_vr), denoted by -1. This step allows us to grasp the scope and potential impact of missing data on our analyses.

To systematically handle missing values, we create a new DataFrame, df_imputed. Here, we replace placeholder values (-99.00 for loggm and feh, 999.99 for vr) with NaN using NumPy's np.nan. This approach effectively marks missing data, facilitating transparent handling and analysis in subsequent statistical procedures.

loggm	0.222
feh	0.665
vr	1.626

Table 1-Missingness Percentages

In the imputation stage, we utilized the SimpleImputer from scikit-learn with a strategy set to 'median'. We verify the impact of our data handling by revisiting descriptive statistics for loggm, feh, and vr. These statistics offer a comprehensive view of the variables' distributions and characteristics post-imputation, ensuring that our data preprocessing efforts maintain data integrity and enhance the reliability of subsequent statistical analyses.

D. Modelling

In the modelling phase, our approach focuses on preparing the dataset for predictive analytics, aiming to construct robust models capable of accurately predicting the spec variable based on stellar observations.

Ensuring robust evaluation, we stratify the data split to maintain proportional representation of spec categories in both training and testing sets, with a 20% test size and fixed random state for consistent evaluation metrics across experiments.

We employ ColumnTransformer to streamline feature preparation, ensuring efficient preprocessing of diverse feature types. Numeric features undergo standardization with StandardScaler to harmonize scales and facilitate model convergence. Ordinal features are encoded using OrdinalEncoder to preserve their inherent order, while categorical features are transformed into one-hot vectors.

The baseline method we employed randomly selects predictions based on the distribution of classes observed in the training data. This approach requires no model training and serves as a straightforward benchmark for evaluating more complex predictive models.

The chosen approach yields a weighted average ROC AUC score of about 0.506, indicating the minimum performance models should surpass to demonstrate meaningful predictive ability beyond random chance. This baseline sets a starting point for evaluating more complex models effectively.

Furthermore, we have implemented a custom scoring function to compute the ROC AUC score for evaluating multiclass classification models. This function is designed to accept true class labels and predicted probabilities as inputs. It incorporates parameters such as average="weighted" to calculate the weighted average ROC AUC score across classes and multi_class='ovr' to treat the classification task as one-vs-rest.

1. *Decision Tree*

The decision tree model underwent tuning via a randomized search over parameters including criteria, maximum depth, and minimum samples required for splitting and leaf nodes. The best model selected based on ROC AUC performance across 20-fold cross-validation on training set exhibits varied precision, recall, and F1-scores across spectral classes.

Classes with limited support, such as 1 and 7, show lower metrics, while classes 2, 3, 4, 5, and 6 demonstrate more balanced predictive performance. The model achieves a weighted average F1-score of 0.76 on the test set.

2. *Random Forest*

The random forest model was optimized with parameters including the number of estimators, maximum depth, minimum samples for splitting and leaf nodes, features to consider per split, and bootstrap sampling. After tuning and 10-fold cross-validation on training set, the best model maximizes the ROC AUC score.

On the test set, the model performs strongly across all spectral classes, achieving high precision, recall, and F1-scores. Notably, precision improvements are observed for classes 1, 2, 3, and 7, with a weighted average F1-score of 0.81.

3. *Stochastic Gradient Descent*

The SGD classifier was optimized using randomized search over parameters such as loss function, penalty, learning rate schedule, and regularization strength. The best model, identified through ROC AUC optimization across 20-fold cross-validation on training set, utilizes parameters alpha=0.001, loss='log_loss', and penalty='elasticnet'.

On the test set, the SGD classifier demonstrates robust performance across all spectral classes, with balanced precision, recall, and F1-scores. It achieves improved precision for classes 1, 2, 3, and 7, yielding a weighted average F1-score of 0.81.

4. *Artificial Neural Networks*

Using a grid search approach with various combinations of hidden layers and neurons per layer, we optimized the MLPClassifier for predicting star spectral classes. The best-performing model was found with 3 hidden layers each containing 512 neurons, utilizing parameters such as 'relu' activation function, 'sgd' solver, alpha=0.5, 'adaptive' learning rate, max_iter=900, batch_size=128, and random_state=42. This configuration yielded a best cross-validation score of 0.977 based on ROC AUC.

On the test set, the model demonstrates strong performance with precision, recall, and F1-scores ranging from 0.79 to 0.93 across different spectral classes. Notably, classes 1, 2, 3, and 7 show improved precision, with a weighted average F1-score of 0.83, highlighting its effectiveness in predicting star spectral classes.

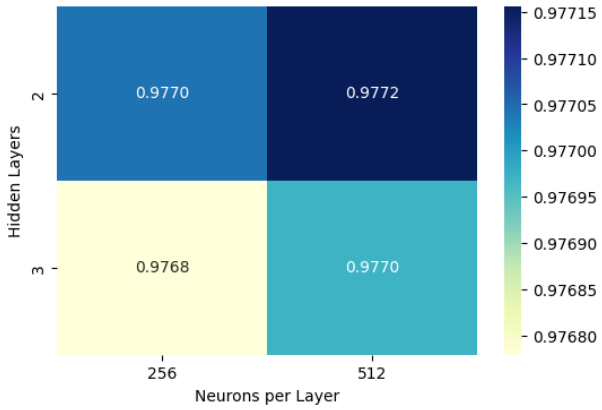

Figure 4- ROC AUC Scores by Neural Network Configuration

This configuration shows the model's strong ability to classify different spectral classes of stars effectively. It visually demonstrates how adjusting the network's architecture impacts its performance, displaying the optimal setup for accurate spectral classification.

According to the docs, MLPClassifier optimizes the log-loss function using LBFGS or stochastic gradient descent, which is essentially the same as cross-entropy. The default loss function for classification is categorical cross-entropy. However, for ordinal data like star categories, ordinal-specific loss functions such as ordinal hinge loss or proportional odds models might be more suitable. These functions consider the ordered nature of classes, ensuring predictions align better with the ordinal structure of the data.

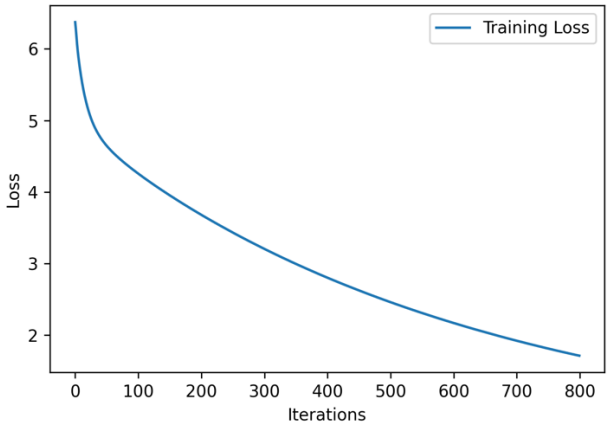

Figure 5-Training Loss Curve

As the above plot illustrates, the training loss starts at around 4.172 and gradually decreases over iterations. Initially, the loss decreases steeply, indicating rapid learning. As training progresses, the rate of decrease in loss slows

down, showing the model is converging. The loss curve stabilizes around 4.095 after numerous iterations, suggesting minimal further improvement beyond this point.

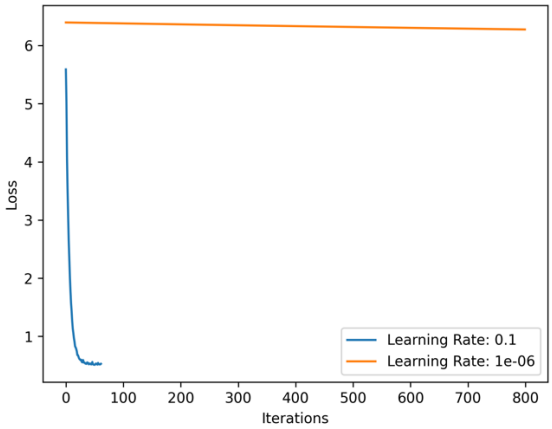

Figure 6-Training Loss Curve for Different Learning Rates

The first model with a learning rate of 0.1 shows effective learning, while the second model with a very small learning rate of 0.000001 struggles to learn from the data effectively, leading to slow and inadequate improvement in loss. Adjusting the learning rate is crucial in optimizing neural network training to achieve better model performance.

5. *K-Means Clustering*

To analyze the distribution of stars based on their observed parameters, we employed clustering algorithms to identify natural groupings within the dataset. Using metrics such as the sum of squared errors (SSE), silhouette score, and Calinski-Harabasz score, we determined that K=5 provides optimal clustering. This choice was validated by high silhouette scores and Calinski-Harabasz scores, indicating well-defined and distinct clusters within our data.

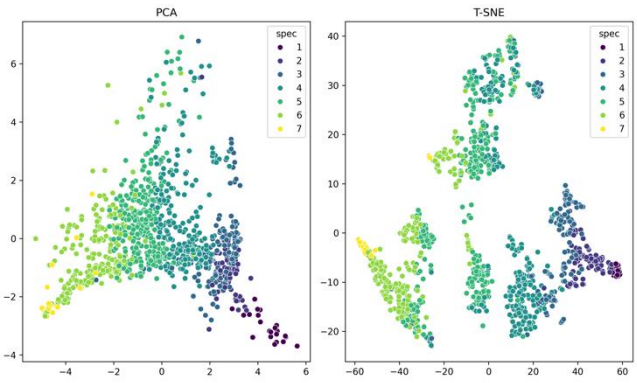

Figure 7-Comparison of PCA and T-SNE Embeddings for Star Distribution

To validate our clustering results and visualize the distribution of stars in a reduced space, we utilized Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (T-SNE). PCA highlighted variance and separability in a two-dimensional space, while T-SNE preserved local structures, providing complementary perspectives on star distribution. However, despite these efforts, we encountered challenges in achieving robust clustering results, which were reflected in moderate clustering metrics. This suggests the need for further exploration and refinement of clustering methodologies to better capture the inherent patterns and complexities within the star dataset.

Figure 8-PCA Plots with K-Means Clustering

The PCA plots with clustering results for k=optimal and k=7 clusters show some separation among data points, indicating potential clustering patterns. However, overlap between clusters suggests challenges in fully separating distinct groups based on the current features and clustering approach.

The DBSCAN algorithm identified an optimal epsilon value of 1.90, resulting in 5 clusters. It classified 59 points as outliers or in the "unknown" cluster (-1), indicating these points deviate from typical clusters' characteristics. Further analysis could explore these outlier points to understand their unique attributes compared to the clustered data, potentially revealing abnormal characteristics in their features.

E. Performance Comparison

We chose the ROC AUC comparison method for evaluating our models due to its robustness in handling class imbalance. The ROC AUC score effectively measures the discrimination ability of a model across all classification thresholds, providing a comprehensive evaluation that is independent of any single threshold.

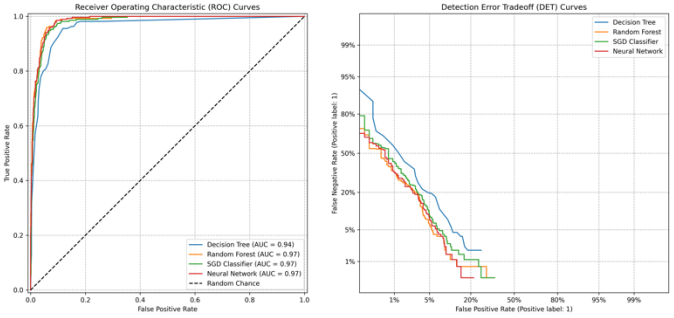

Figure 9-ROC and DET Curves for Model Comparison

Statistically, the models show varying performance metrics compared to the baseline. The Decision Tree achieved an ROC AUC score of 0.9449, while the Random Forest and SGD Classifier scored 0.9719 and 0.9653, respectively. The Neural Network (NN) had the highest score at 0.9741, indicating superior discriminative ability. All models significantly outperformed the baseline ROC AUC of 0.5059, with the NN showing the highest predictive power.

III. CONCLUSION

In summary, this study shows how machine learning models can classify stars using an expanded ELODIE dataset. Through rigorous preprocessing and feature engineering, including addressing multicollinearity and missing values, we significantly enhanced dataset quality. The Artificial Neural Network (ANN) excelled with a high ROC AUC score of 0.9741, outperforming other models and the baseline. Future work will focus on refining clustering methodologies and integrating additional stellar parameters to further improve classification accuracy, while addressing current shortcomings in parameter tuning and validation within our clustering analysis.

IV. REFERENCES

[1] Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(85), 2825-2830.

[3] Prugniel, P., Soubiran, C., Koleva, M., & Le Borgne, D. (2007). New release of the ELODIE library: Version 3.1. https://arxiv.org/abs/0704.0270