Classification of star types using Principal Component Analysis

GitHub link: https://github.com/EemanJ/star_type_classification

Eeman Jehangir-40251393

***Abstract*—This paper focuses on statistical analysis using Principal Component Analysis (PCA) and Machine learning Classification algorithms to predict the final output using data gathered from multiple variables, i.e. attributes. It is a powerful tool for data analysis and visualization.**

When handling multivariate data, it is important that the data be simplified and correlations removed. This is where PCA is used to create a new set of uncorrelated variables, thus reducing dimensionality while retaining the sample's original variation. The algorithm works by identifying the underlying structure of the data and representing it in a more compact format.

This report follows Principal Component Analysis conducted on a dataset that classifies stars into six different types based on certain attributes. The PCA algorithm will identify the dimensions amongst which the variance is highest and organize the data accordingly. It is important to note that the rows all correspond to observations, and the columns correspond to different attributes. The PCA algorithm will convert the data matrix into a transformed matrix with fewer attributes, known as principal components.

This newly transformed data, as well as the original data, are used as inputs in Classification algorithms, namely Ridge Classifier(ridge), Naive Bayes(NB), Logistic Regression(LR) and Quadratic Discriminant Analysis(QDA). This is all done using the Python PyCaret library.

Lastly, the Extra Trees Classifier(ET) is used to interpret and classify the data. Conducting analysis on the calculated Shapley values to interpret the values.

***Index Terms*—Principal Component Analysis, Ridge Classifier, PyCaret, Naive Bayes, Logistic Regression, Quadratic Discriminant Analysis, Extra Trees Classifier, Shapely**

I. Introduction

Stars are the most widely recognized astronomical objects and represent the most fundamental building blocks of galaxies. The age, distribution, and composition of the stars in a galaxy trace the history, dynamics, and evolution of that galaxy. [1]

This report focuses on the prediction of the type of star, a mass is going to be classified as, from the six most common types of stars. Several attributes are used for the purpose of this classification: the temperature of the star in Kelvin, the luminosity of the star, the radius of the star with respect to the sun, the absolute magnitude of the star, the color of the star and the spectral class assigned based on emissions. The readings taken to be with respect to the sun are based on the following: the average luminosity of the sun is taken to be $3.828x10^{26}$Watts and the average radius of the sun is taken to be $6.9551x10^{8}$m.

Originally, stellar classification or the way astronomers classify stars was done primarily through mass, temperature, and color. Since stars have more characteristics than mass, temperature, and color, scientists have adapted and added them to star classification. For example, luminosity is an extremely important characteristic of stars. Luminosity measures the total amount of energy the star radiates out per second. In other words, it is a measure of the absolute magnitude of the power output of the star. This is vital to understanding the stellar classification of various stars because luminosity is dependent upon both temperature and size. It also tells astronomers about where on its lifecycle the star is. [3]

The attributes mentioned above are used to predict the star types and classify them to a number corresponding to one of six types as follows: red dwarf, brown dwarf, white dwarf, main sequence, super giants and hypergiants. Red dwarfs make up about 75% of all stars in the Milky Way galaxy, while brown dwarfs are thought to be even more common. [2]

II. Principal Component Analysis

Principal Component Analysis (PCA) is based on the mathematical concept of linear algebra and matrix factorization. PCA takes advantage of multicollinearity and combines the highly correlated variables into a set of uncorrelated variables. Therefore, PCA can effectively eliminate multicollinearity between features. [4]

Real-life applications require the monitoring of several different variables at a time. PCA helps to reduce the dimensionality of said data while maintaining the variance of the original data, therefore maintaining the structure of the data. This makes the data easier to compute, the simpler models are far more robust and easier to interpret and explain as well as enable easier and quicker data visualization on two or dimensions, instead of several tens or even hundreds in some cases.

A. Principal Component Analysis Algorithm

The PCA algorithm aims to reduce the size of the data matrix from p x p size to n x p, where n $\ll$ p. The rows in the given data matrix are all observations, whereas the columns describe the data attributes and variables. PCA works in four stages to reduce dimensionality.

1) **Standardization**

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis. More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges, which will lead to biased results. So, transforming the data to comparable scales can prevent this problem. [5]
Compute the centred data matrix of a 'pxp' matrix by subtracting off column means.

$$Y = HX$$

This gives the final centred matrix.

2) **Covariance Matrix**

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix. [5]

$$S = \frac{1}{(n-1)} Y^T Y$$

S gives the covariance matrix.

3) **Eigen Vectors and Eigen Values of S**

Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the principal components of the data. [5]
The eigenvectors are calculated using eigenvalue decomposition(EVD) if the matrix is a square matrix, else square value decomposition(SVD) is used.

$$S = A \Lambda A^T$$

4) **Final transformed matrix**

The final transformed matrix is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction because if we choose to keep only p eigenvectors (components) out of n, the final data set will have only p dimensions. [5]
This matrix has size nxp, where n $\ll$ p.

This is denoted by

$$Z = YA$$

B. Principal Component Analysis on the Stars dataset

In order to conduct PCA on the given dataset, the 'Type' column was removed, as this is the one we will be predicting, hence we do not want to standardize it or conduct any further calculations on this.
This next step is specific to this dataset, as PCA is conducted on continuous data, two further columns were dropped as they contained categorical data, namely:'Color', and 'Spectral_Class'.

III. Machine Learning Algorithms for Classification Analysis

A. Ridge Classifier (Ridge)

It is a form of regularization that penalizes model coefficients to prevent overfitting. Overfitting is a common issue in machine learning that occurs when a model is too complex and captures noise in the data instead of the underlying signal. This can lead to poor generalization performance on new data. Ridge classification addresses this problem by adding a penalty term to the cost function that discourages complexity. This results in a model that is better able to generalize to new data. [6]

Ridge classification works by adding a penalty term to the cost function that discourages complexity. The penalty term is typically the sum of the squared coefficients of the features in the model. This forces the coefficients to remain small, which prevents overfitting. The amount of regularization can be controlled by changing the penalty term. A larger penalty results in more regularization and smaller coefficient values. This can be beneficial when there is little training data available. [6]

B. Naive Bayes (NB)

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. [7]

The Naive Bayes classifier works by assuming that the presence of a particular feature in a class is independent of the presence of any other feature. This is the "naive" assumption and is the reason why the algorithm is called "naive." Despite its simplifying assumptions, the Naive Bayes classifier is known to be very accurate in practice and can perform well on a wide range of classification problems.

C. Logistic Regression (LR)

The logistic regression model works by fitting a logistic curve to the data, which is an S-shaped curve that maps the predictor variables onto the outcome variable. The curve represents the probability of the outcome variable being a certain value based on the predictor variables. The logistic function used in logistic regression is defined as:

$$p = \frac{1}{1+exp(-z)}$$

where p is the probability of the outcome variable being 1, exp is the exponential function, and z is a linear combination of the predictor variables. The logistic curve is typically estimated using maximum likelihood estimation, which involves finding the parameter values that maximize the likelihood of the observed data.

Once the model is fitted, it can be used to predict the probability of the outcome variable being a certain value based on new values of the predictor variables.

D. Quadratic Discriminant Analysis (QDA)

Quadratic Discriminant Analysis (QDA) is a classification algorithm that is based on Bayes' theorem. QDA is a variant of Linear Discriminant Analysis (LDA) that assumes that the covariance matrices of the input features are different for each class. In contrast, LDA assumes that the covariance matrices are the same for all classes.

QDA works by modeling the probability density function (PDF) of the input features for each class. This is done by estimating the mean and covariance matrix of the input features for each class. The estimated PDFs are then used to compute the posterior probability of each class for a given set of input features.

The decision boundary between two classes in QDA is a quadratic equation, rather than a linear equation as in LDA. This means that QDA can model more complex decision boundaries than LDA, and may be more accurate when the covariance matrices of the input features are different for each class.

However, QDA requires more parameters to be estimated than LDA, which can make it more prone to overfitting if the training dataset is small. QDA also assumes that the input features are normally distributed, which may not always be the case in practice.

Overall, QDA can be a useful algorithm for classification tasks where the covariance matrices of the input features are different for each class and a quadratic decision boundary is more appropriate than a linear one. However, it is important to carefully evaluate the performance of QDA on a given dataset and consider other classification algorithms as well.

E. Extra Trees Classifier (ETC)

Extra Trees Classifier (ETC) is a type of ensemble learning algorithm that uses a collection of decision trees to make predictions. ETC works by recursively partitioning the feature space into regions that are as pure as possible with respect to the class labels.

For each feature, it generates a large number of random split points and selects the one that results in the largest information gain or decrease in impurity. This approach of randomizing the split points and then combining the predictions of multiple trees is what makes ETC a powerful and robust algorithm. Using many randomized trees reduces the impact of noisy or irrelevant features and can handle high-dimensional datasets with ease.

IV. Dataset Description

The dataset for star type classification for the purposes of this project and report was taken from Kaggle. This dataset has 240 rows of observations regarding attributes of six different types of stars: red dwarf, brown dwarf, white dwarf, main sequence, super giants and hyper giants. Each of these has been assigned a numerical value from 0 to 5 in order. As can be seen from the pie chart below in "Fig. 1", the dataset contains equal numbers of observations for each type of star, this ensures that the classification algorithms we will implement will have no biases towards any specific star type, and will likely be trained and tested equally on all.

The dataset classifies stars on the basis of six attributes:

- Temperature (K): This attribute records the temperature of the star in Kelvin.
- L: This attribute records the luminosity of the star relative to that of the sun.
- R: This attribute notes the radius of the star with respect to the radius of the sun.
- AM (Mv): The absolute magnitude of the star.
- Color: The general observational color of the star, which indicates the star's temperature in its photosphere-the layer where the star emits most of its visible light.
- Spectral_Class: These spectral classes are assigned to aestroid based on their emission spectrum, and are thought to correspond to an asteroid's surface composition.
- Type: This is the attribute the algorithm will be predicting, based on the values for the above.

The readings taken to be with respect to the sun are based on the following: the average luminosity of the sun is taken

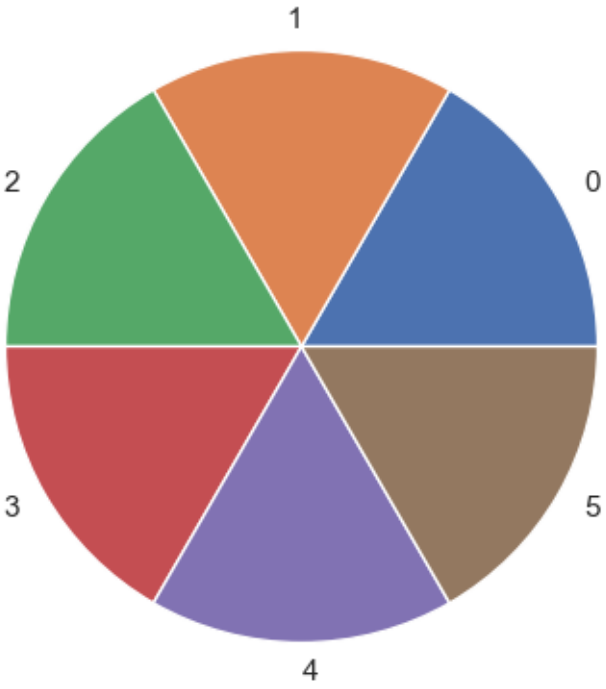

Fig. 1. Pie chart showing the division of star types in the dataset.

to be $3.828x10^{26}$Watts and the average radius of the sun is taken to be $6.9551x10^{8}$m.

For the purpose of this project and to conduct PCA, two of the above attributes are dropped, namely: color and Spectral_Class on the basis of being categorical variables which will not be contributing much to the Principal Component Analysis.

In the box plot figure, "Fig. 2" below, it can be seen that the data is not central and has quite a fair bit of variability among the attributes themselves too; the strip plot dots on it show the spread of each of the points for all four attributes, especially for the first three attributes.

These data visualization tools plotted on one diagram are better used for exploratory data analysis, and useful to compare the distributions of continuous variables.

The box in the plot represents the interquartile range (IQR), which contains the middle 50% of the data. The median is shown as a center line inside the box, and the upper and lower whiskers extend to the maximum and minimum values within 1.5 times the IQR from the upper and lower quartiles, respectively. Any data points beyond the whiskers are considered outliers. Outliers exist in all features, but especially for temperature, luminosity and radius, whereas the absolute magnitude has the least variation.

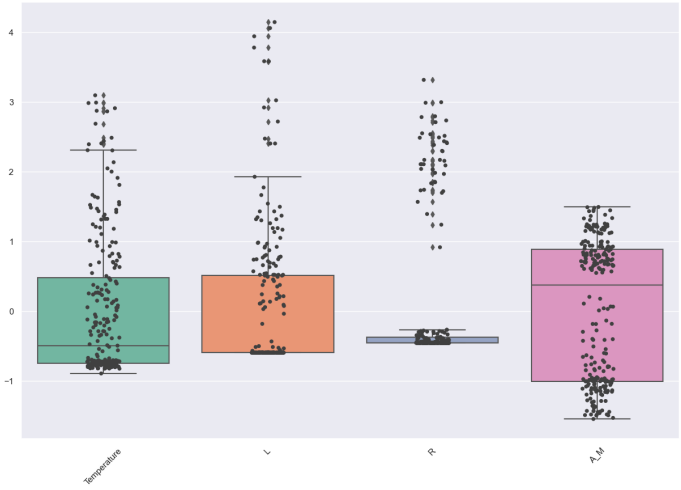

Fig. 2. Box and Strip plot showing the distribution of dataset points.

The strip plot, on the other hand, displays the individual data points for each category or group in a dataset along an axis. This allows for easy comparison of the distribution of a variable across different categories or groups. Strip plots can be further enhanced by adding jitter, which randomly perturbs the position of each point along the axis to prevent overlapping.

In the figure, "Fig. 3" the correlation matrix shows the relationships between the variables, representing the strength and direction of linear relationships between pairs of variables in the dataset. Each cell in the matrix represents the correlation coefficient between the pairs of variables. This plot also features as a color-coded heat map, where the variable pairs with high correlations are visually represented with warmer, orange to red colors, whereas those with lower correlations have cooler, greener shades. A positive value shows a positive correlation and vice versa.

The features with high positive correlation are temperature and luminosity, and luminosity with radius with respect to the sun.

These same relationships can be seen graphically using the pair plot analysis seen below, in "Fig. 4". The pair plot represents the same data differently, explaining the different relationships, and displaying the star types distributions plotted on variable pairs.

V. PCA Results

Principal component analysis is conducted on this star types dataset. PCA is applied using a well-documented Python PCA library, that is running PCA on a previously standardized data matrix.

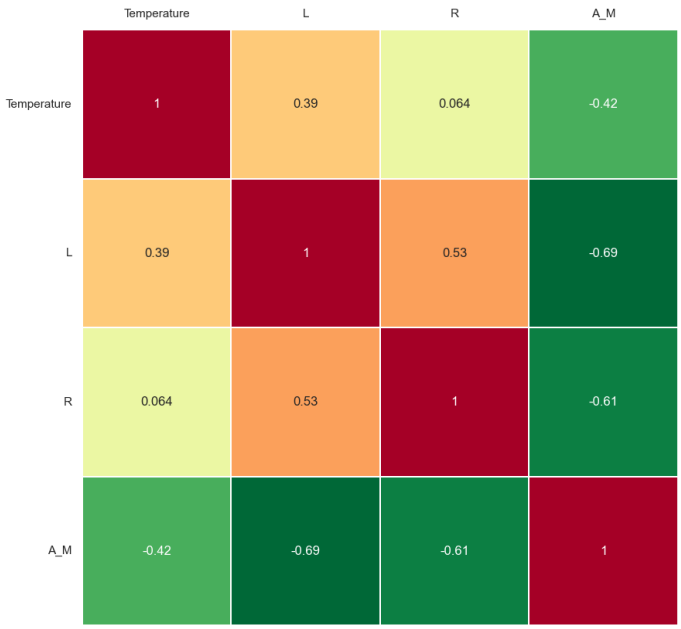

Fig. 3. correlation Matrix of the attributes.

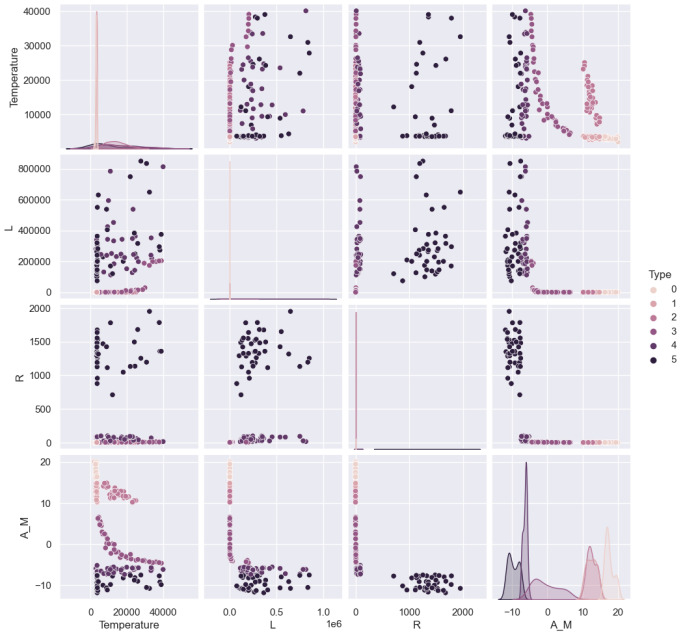

Fig. 4. Pair plot of variable pairs plotted against each other, with star types.

The original pxp dataset is reduced using the eigenvector matrix A. Each column of the eigenvector matrix represents a principal component, with the first column being the first principal component, Z_1, that represents the most variance; accordingly, the second column represents the second principal component, Z_2, which has the second largest variation, and so on and so forth.

The obtained eigenvector matrix, A is given as follows:

$$\begin{bmatrix} 0.35018343 & 0.82161262 & -0.37607837 & 0.24675763 \\ 0.55933789 & 0.00452564 & 0.75509623 & 0.34198002 \\ 0.47477107 & -0.56898115 & -0.53314117 & 0.40818302 \\ -0.58232734 & 0.03453503 & 0.06446119 & 0.80965855 \end{bmatrix}$$

The scree plot in "Fig. 5" and pareto plot in "Fig. 6" demonstrates the scree plot and pareto plot of the PCs. The scree plot and pareto plot display the amount of variance explained by each principal component. The percentage of variance experienced by j-th PC can be evaluated using the following equation:

$$S = \frac{\lambda_j}{\sum_{n=j}^{p}} \lambda_j \cdot 100, j = 1, 2, ..., p$$

Accordingly, the first principal component, PC1, is given by:

$$Z_1 = 0.350X_1 + 0.559X_2 + 0.475X_3 - 0.582X_4$$

the second principal component, PC2, is given by:

$$Z_2 = 0.822X_1 - 0.569X_3$$

as X_2 and X_4 contribute very little to PC2, they are not considered in the equation for PC2, they are 0.004 and 0.03 respectively.

and the third principal component is given by:

$$Z_3 = -0.376X_1 + 0.755X_2 - 0.533X_3$$

As can be seen in "Fig. 5" and "Fig. 6", the first two principal components are explaining a grand total of 83% of the variance, whereas the first three are accounting for almost a grand 94% of the variance. The explained variance matrix of all the principal components is given by: $\begin{bmatrix} 0.60357391 & 0.23500784 & 0.09329645 & 0.0681218 \end{bmatrix}$

"Fig. 7" represents the PC coefficient plot. It visually represents the amount of contribution each feature has on the first two PCs.

From the below "Fig. 8" it can be seen that temperature and R contribute the most to the principal components and they are closely followed by luminosity.

VI. Classification Results

Classification was carried out on the dataset using the Python PyCaret library. Three popular classification algorithms were applied to the dataset, once before conducting PCA, and once after. Both times the trained model was tuned and ran once more.

The model was split into a 70:30 ration, with 70% of the model used for training and the remaining 30% for testing the dataset.

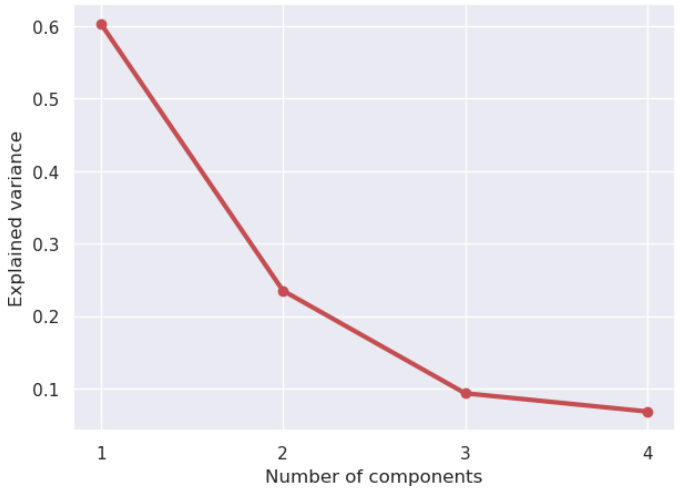

Fig. 5. Scree plot of explained variance.

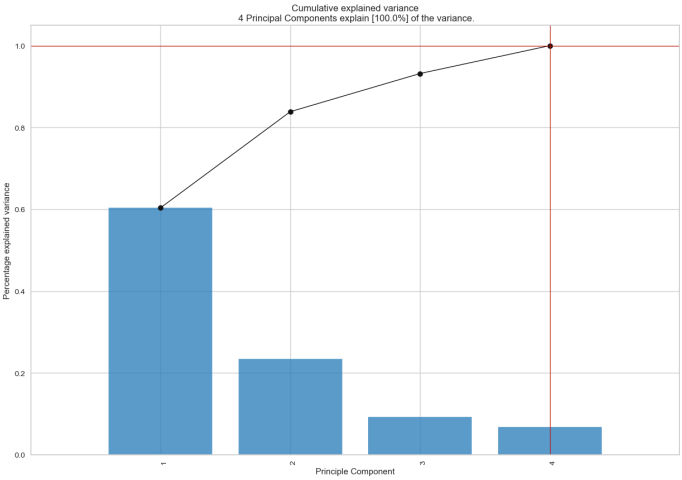

Fig. 6. Pareto chart with cumulative explained variance.

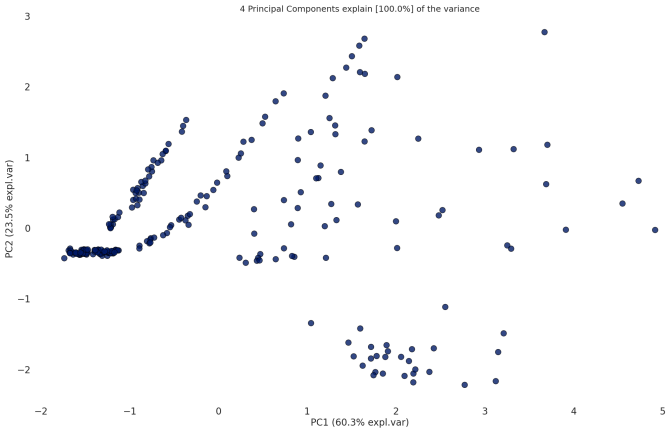

Fig. 7. Principal component analysis.

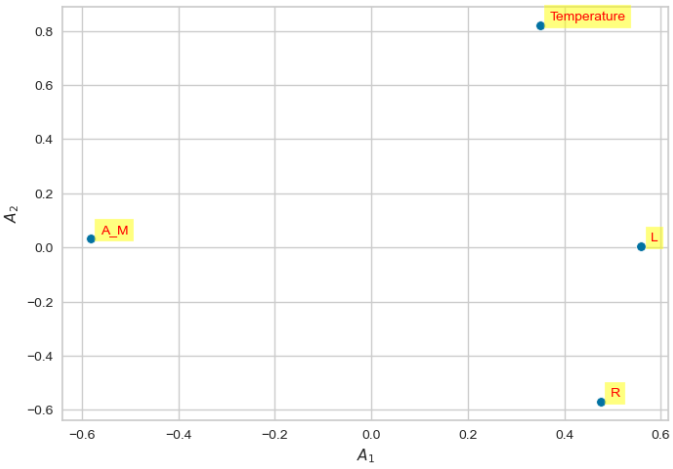

Fig. 8. Coefficient plot.

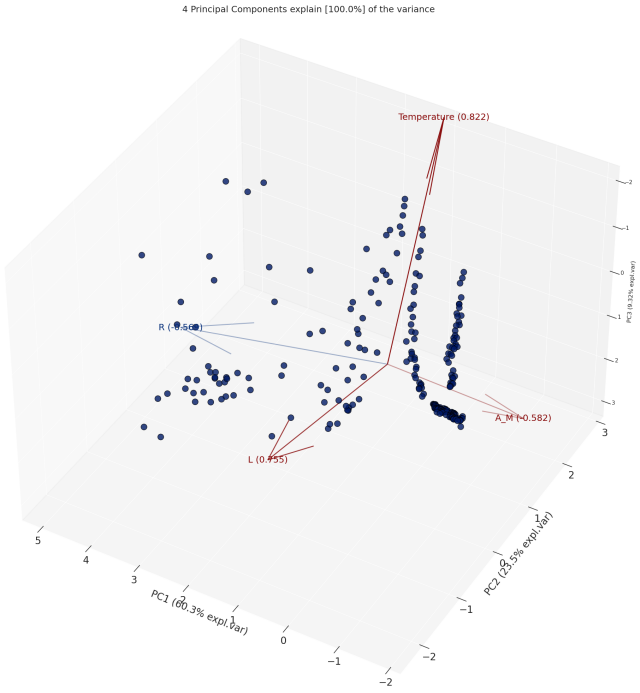

Fig. 9. Biplot of the principal components.

First PyCaret was used to analyze which classification algorithms would form the best models for the given dataset and the accuracy level for them, given in “Fig. 10”.

It can be noted that the best fitting algorithm is the Random Forest Classifier, however, it gives an accuracy of 1.000, which leads to overfitting, i.e to say that the model will be overfitted to the nuances of this specific dataset and may not be able to accurately predict which star type it really is. Given this, we would want to select algorithms that will still be able to accurately predict the type of star, without being overly fitted to this specific dataset, and that will be the best

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.3440
lightgbm	Light Gradient Boosting Machine	0.9938	1.0000	0.9938	0.9953	0.9932	0.9924	0.9929	0.3380
dt	Decision Tree Classifier	0.9933	0.9958	0.9933	0.9950	0.9927	0.9919	0.9924	0.0530
et	Extra Trees Classifier	0.9875	1.0000	0.9875	0.9906	0.9867	0.9848	0.9857	0.0930
gbc	Gradient Boosting Classifier	0.9867	1.0000	0.9867	0.9900	0.9858	0.9838	0.9848	0.1840
lda	Linear Discriminant Analysis	0.9671	0.9994	0.9671	0.9764	0.9661	0.9602	0.9627	0.0520
lr	Logistic Regression	0.9404	0.9924	0.9404	0.9559	0.9384	0.9280	0.9317	1.5010
ridge	Ridge Classifier	0.8946	0.0000	0.8946	0.9062	0.8805	0.8732	0.8855	0.0400
nb	Naive Bayes	0.8408	0.9910	0.8408	0.8892	0.8310	0.8085	0.8203	0.0550
ada	Ada Boost Classifier	0.8275	0.9605	0.8275	0.7385	0.7689	0.7913	0.8218	0.1160
knn	K Neighbors Classifier	0.5562	0.9056	0.5562	0.5655	0.5235	0.4692	0.4856	0.9880
svm	SVM - Linear Kernel	0.2979	0.0000	0.2979	0.1123	0.1588	0.1621	0.2140	0.0620
qda	Quadratic Discriminant Analysis	0.1588	0.0000	0.1588	0.0262	0.0447	0.0000	0.0000	0.0710
dummy	Dummy Classifier	0.1525	0.5000	0.1525	0.0242	0.0416	0.0000	0.0000	0.0480

Fig. 10. Best model before PCA.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9408	0.9904	0.9408	0.9567	0.9383	0.9286	0.9327	0.2530
rf	Random Forest Classifier	0.9404	0.9978	0.9404	0.9603	0.9377	0.9282	0.9334	0.1610
et	Extra Trees Classifier	0.9338	0.9970	0.9338	0.9553	0.9329	0.9203	0.9254	0.1120
gbc	Gradient Boosting Classifier	0.9279	0.9952	0.9279	0.9458	0.9257	0.9131	0.9176	0.1880
dt	Decision Tree Classifier	0.9271	0.9558	0.9271	0.9420	0.9248	0.9118	0.9157	0.0530
knn	K Neighbors Classifier	0.8750	0.9826	0.8750	0.8909	0.8650	0.8493	0.8573	0.0630
qda	Quadratic Discriminant Analysis	0.8154	0.9556	0.8154	0.8202	0.7961	0.7777	0.7932	0.0630
nb	Naive Bayes	0.7683	0.9511	0.7683	0.7363	0.7402	0.7216	0.7348	0.0530
lr	Logistic Regression	0.7288	0.9592	0.7288	0.7403	0.7186	0.6737	0.6826	0.0660
ada	Ada Boost Classifier	0.6558	0.9203	0.6558	0.5033	0.5529	0.5835	0.6282	0.0790
svm	SVM - Linear Kernel	0.5567	0.0000	0.5567	0.4502	0.4702	0.4664	0.4991	0.0520
ridge	Ridge Classifier	0.5100	0.0000	0.5100	0.4341	0.4270	0.4114	0.4433	0.0520
lda	Linear Discriminant Analysis	0.5042	0.9264	0.5042	0.4472	0.4457	0.4041	0.4258	0.0490
dummy	Dummy Classifier	0.1525	0.5000	0.1525	0.0242	0.0416	0.0000	0.0000	0.0450

Fig. 11. Best model after PCA.

model for our classification needs.

Accordingly, the three algorithms selected had accuracies of just below 90%, Ridge Classifier, Naive Bayes, and K-Neighbors Classifier, before conducting pca on the dataset.

In "Fig. 11" it can be noted that there is a smaller chance of overfitting, as the data is standarized and the best model suggested is the Light Gradient Boosting Machine algorithm, at 94% accuracy.

However, even 94% is a fairly high number where overfitting can be an issue, so we picked the K-Neighbours Classifier, Naive Bayes and Quadratic Discriminant Analysis for the PCA dataset.

A. *Classification before PCA*

The decision boundaries of the classification models are seen in "Fig. 12", "Fig. 13", and "Fig. 14".
The decision boundaries of the classification models are the boundaries that separate the different star types in the input feature space. These boundaries are based on the values of the

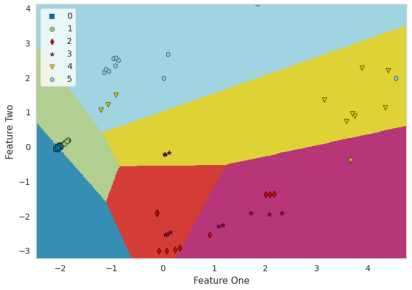

Fig. 12. Decision Boundary for Logistic Regression before PCA.

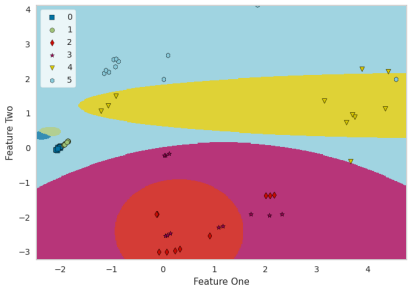

Fig. 13. Decision Boundary for Naive Bayes before PCA.

input features and the model parameters and are used to make predictions for new data points.

The decision boundaries provide valuable insights into the relationships between the input features and the output classes. By visualizing the decision boundaries, it is possible to understand how the model is making predictions, which features are most important in determining the class of a data point, and which regions of the feature space are most informative.

The decision boundaries of the classification models can also be used to evaluate the performance of the model. It can be seen that the decision boundaries for Naive Bayes are quite irregular, whereas Logistic Regression is performing better, and Ridge Classifier is performing the best of all the models on the dataset before PCA is conducted. These diagrams are useful tools to giving a deeper understanding of the performance of all three models on the dataset, prior to PCA.

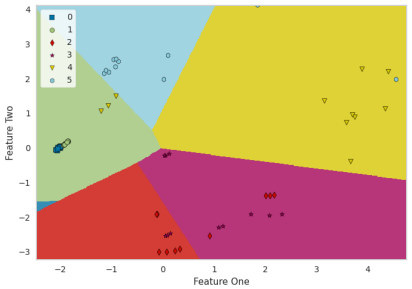

Fig. 14. Decision Boundary for Ridge Classifier before PCA.

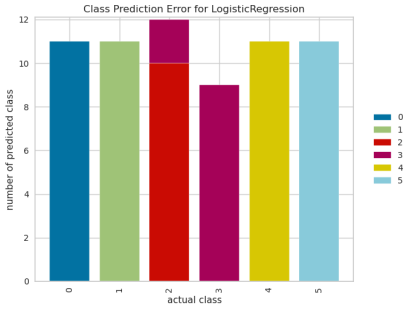

Fig. 15. Prediction error for Logistic Regression before PCA.

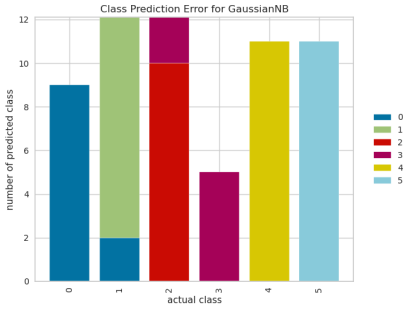

Fig. 16. Prediction error for Naive Bayes before PCA.

The prediciton errors of the classification models, before PCA can be seen in "Fig. 15", "Fig. 16", and "Fig. 17". Class prediction error is another very useful metric for evaluating the overall performance of a classification model. It can be seen here in "Fig. 16" that Naive Bayes has the most errors when predicting between star type 1 and star type 2, closely followed by the Logistic regression algorithm and finally the Ridge Classifier here performs best. This is the ideal solution for the dataset prior to PCA, as it seems there is little overfitting on the exact nuances of the data, hence it is giving us more accurate predictions.

B. Classification after PCA

The decision boundaries of the classification models on the dataset after conducting PCA are seen in "Fig. 18", "Fig. 19", and "Fig. 20".

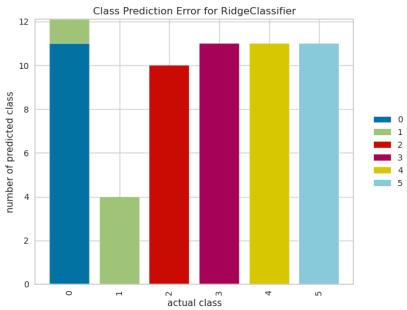

Fig. 17. Prediction error for Ridge Classifier before PCA.

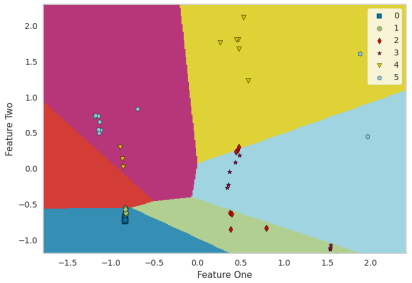

Fig. 18. Decision Boundary for Logistic Regression after PCA.

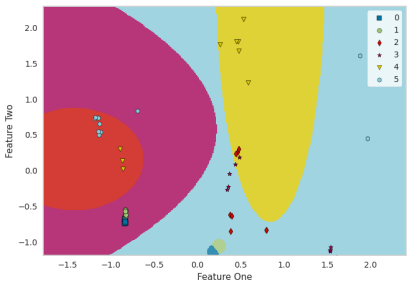

Fig. 19. Decision Boundary for Naive Bayes after PCA.

From the above-mentioned diagrams, it can be seen that both Naive Bayes and Quadratic Discriminant Analysis have very specific decision boundaries, which are very likely to be overfitting the data. On the other hand the Logistic Regression model seems to have far better boundaries, that are not too generic, hence seems to be the best fit for the PCA dataset.

The prediction errors of the classification models, after PCA can be seen in "Fig. 21", "Fig. 22", and "Fig. 23".

From these diagrams, it can be noted that none of the models are predicting the cell types too accurately, however, Logistic Regression still seems to have a better prediction error compared to the other two, which have been overfitting the dataset.

VII. Explainable AI with Shapley Values

Explainable AI (XAI) is an emerging field of AI research that focuses on developing methods and techniques to make

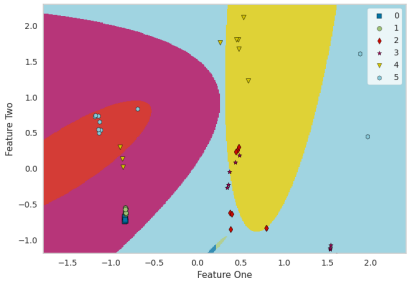

Fig. 20. Decision Boundary for Quadratic Discriminant Analysis Classifier before PCA.

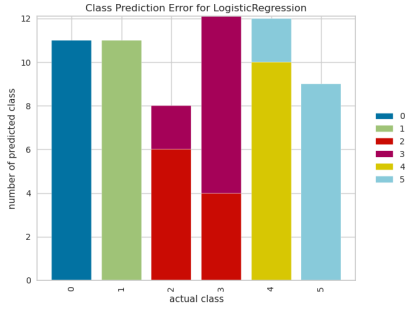

Fig. 21. Prediction error for Logistic Regression after PCA.

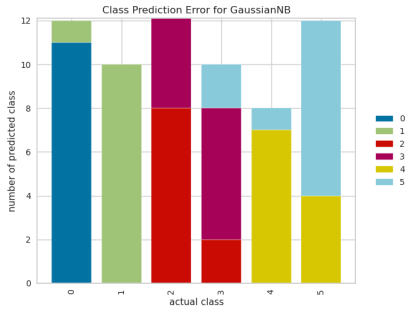

Fig. 22. Prediction error for Naive Bayes after PCA.

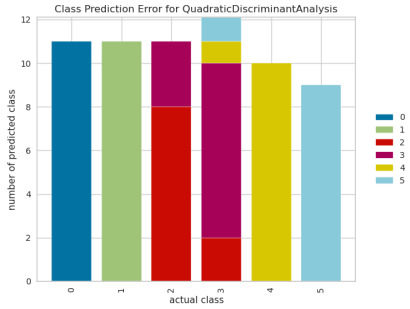

Fig. 23. Prediction error for Quadratic Discriminant Analysis Classifier after PCA.

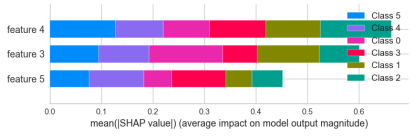

Fig. 24. Summary plot.

Fig. 25. Force plot.

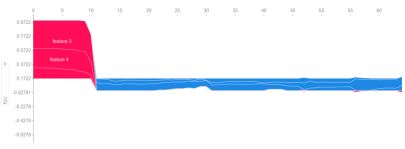

Fig. 26. Combined Force Plot.

machine learning models more transparent and interpretable. The goal of XAI is to help humans understand the decision-making process of AI models and to identify potential biases or errors.

The interpretability of a model is one of the most important things to consider for Machine Learning. Among other important metrics, is explainability of a model and feature importance is one of them, In order to get an overview of the most important features on each of the principal components, we use the python library 'shap'.

Shapley values are a method to quantify each individual feature's contribution to the final prediction of a machine learning model. They can also be used for feature selection, as features with low Shapely values can be removed from the model without significantly affecting the performance.

Shapely values, named after Lloyd Shapley, is a method to quantify the contribution of individual features to the prediction of a machine learning model. It is a method of assigning a numerical value to each feature of a model, indicating how much that feature contributed to the final prediction.

The idea behind Shapely values is to calculate the average change in the model's output when a feature is included, compared to when it is excluded, for all possible combinations of features. The Shapely value of a feature is then the weighted sum of these changes, with weights determined by the number of feature combinations that include the feature.

Using Shapley, we can visualize the contribution of each feature to the model's output, both globally and locally. Shap can also be used to diagnose and mitigate biases in machine learning models. By examining the contributions of each feature to the model's prediction, we can identify features that are driving biased decisions and adjust the model accordingly.

As the Python, shap library is relatively new and still under development, it only support tree-based models, for this purpose the most effective tree model was chosen, the Extra Trees Classifier model for multivariate classification of star types.

As with the previous models, first an extra trees model is created with ideal hyperparameters, and then the tuned model is passed to the 'shap' library for producing the interpretation plots seen in "Fig. 24", "Fig. 25", and "Fig. 26".

The summary plot, "Fig. 24"feature importance with feature effects on the magnitude of each type of star classification. Overall,feature 4 seemed to have the highest impact in terms of magnitude, followed by feature 3 and feature 5.

"Fig. 25" represents the force plot for a single observation only, and the value in bold is seen as the models score for that observation. For this particular observation, feature 3 seemed to be of most importance.

"Fig. 26" displays the combined force plot of all PCs. This plot is a combination of all individual force plots with 90 degree rotation and are stacked horizontally. In this plot, y-axis is the x-axis of the individual force plot. There are 64 data points in the transformed test set, hence x-axis has 64 observations. This combined force plot shows the influence of each PC on the current prediction. Values in the blue color are considered to have a positive influence on the prediction whereas values in the red color have a negative influence on the prediction.

VIII. Conclusion

PCA and four popular classification algorithms are applied to a star type classification dataset. The star type dataset holds information on attributes of stars to identify them as Red dwarfs, Brown dwarfs, White dwarfs, Main sequence, Super giants or Hyper giants. At first, PCA is applied to the original dataset. The first three principal components accounted for 94% of the variance, and the two principal components apprehend 83of the data. Hence, the feature set is reduced accordingly. Extensive experiments are conducted on the first two principal components and different plots are generated to validate the obtained results from different perspectives. To move forward, three classification algorithms, Ridge Classifier, Naive Bayes and LR are applied to the original dataset, and Logistic Regression, Quadratic Discriminant Analysis and Naive Bayes are applied to the transformed dataset with first three components. Each algorithm is tuned with the ideal hyperparameter settings and performance evaluation is conducted by comparing decision boundaries and prediction error.

It is observed that after the hyperparameter tuning performance metrics score of each algorithm improved significantly. There were issues with overfitting of data to the models, but Ridge Classifier performed the best, and according to prediction error it was noted that the models performed better on the original data. Finally, in order to increase the interpretability of the model, three interpretation plots are produced using explainable AI shapley values. To summarize, all four algorithms can successfully determine the star types, with some degree of error.

References

[1] NASA, How do stars form and evolve, Science.Nasa, 'https://science.nasa.gov/astrophysics/focus-areas/how-do-stars-form-and-evolve', May 2023.

[2] Michael A. Seeds and Dana E. Backman, Foundations of Astronomy, 14th ed., vol. 2. Brooks Cole, 2018.

[3] Rachel Kreuger and Maria Airth, "Star classification types and luminosity class," in https://study.com/learn/lesson/star-classification-types-luminosity-class.html, 2022.

[4] Rukshan Pramoditha, "How do you apply PCA to logistic regression to remove multicollinearity?," Towards Data Science, 2021.

[5] Zakaria Jaadi, "A step-by-step explanation of principal component analysis (PCA)," Builtin, 2023.

[6] Ajitesh Kumar, "Ridge classification concepts and python examples," VitalFlux, 2022.

[7] Hastie, Trevor The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations. Tibshirani, Robert., Friedman, J. H. (Jerome H.). New York: Springer, 2001.