

# Exploring the Role of Redshift in Celestial Object Classification in Combination with Photometric Parameters:

A Comparative Analysis with Linear and Non-Linear Machine Learning Models

Danielle Brennan

ASTR 12600

December 2024

# I. Background

## A. Dataset

For this project, I take advantage of the Sloan Digital Sky Survey (SDSS) to analyze astronomical data using both baseline and more complex machine learning (ML) models. In my proposal, I outlined a plan to use an online dataset to build and train a ML model which classifies an image as a celestial object (a galaxy, star or quasar) from its provided spectral and photographic data. I also included a blurb identifying the inevitability of the evolution of this expectation, which added that I would undertake to explain this development in my paper. In this case, due to time constraints (and my machine's pathetic RAM), collecting and analyzing both photometric and spectral data—and creating a finely-tuned, complex neural network—was not a reasonable undertaking. To maintain relevance to our lecture topics with these limitations in mind, I opted to use an open-source dataset from Kaggle and to use primarily photometric data<sup>1,2</sup>. Using this readily-available dataset<sup>3</sup> circumvented the data collection step, which is CPU-intensive and time consuming but provides exactly the same values from the same source (SDSS) which I initially aimed to utilize.

## B. Problem Definition

My project revolves around a multi-class classification problem where the goal is to categorize celestial objects in the SDSS dataset as stars, galaxies, or quasars based on their UGRIZ photometric (and/or redshift<sup>4</sup>) data from the SDSS dataset. I began looking to build a model with better accuracy than the Dummy or Logistic Regression baseline, which I could then compare with the latter to lend insight into the relationship between the UGRIZ features and the object type. However, while experimenting with additional features, I tried including redshift since we have discussed it in class. I ran my models and saw (as will be later explained) *massive* positive impact on model performance. I then endeavored to write this paper centered on this impact and, in my analysis, address the question of why redshift has such a profound impact on all model performance. Additionally, I undertook a breakdown of the comparative performance between models to fulfill my original curiosity as to the nature of the relationship between the collective features and the object classes. Though model performance increased across all except the complete-baseline, the Dummy model, the models were differentiated in their comparative gain from the extra feature's inclusion. I will lay out the output of my models in terms of their prediction possibility, uncertainty outputs (which are either variance or entropy where respectively applicable to model structure), and confusion matrices. In my analysis, I will dissect the performance of each set of models individually, then end the paper with a synthesis of the two, finally arriving at an understanding of the potential reason driving redshift's immense impact on my models' prediction power.

## C. Relevant Dataset Features

SDSS collects photometric data in the form of UGRIZ magnitude values, which measure a celestial object's magnitude through five filters on SDSS telescope, representing light in the ultraviolet (u), green (g), red (r), near-infrared (i), and infrared (z) wavelengths. In the UGRIZ system, the apparent magnitude

---

<sup>1</sup> ("UGRIZ" magnitude values, which I will expand on later)

<sup>2</sup> (I did experiment with incorporating redshift into the training feature set, which is derived from spectral, not photometric, data.) (SDSS III, accessed 2024) (SDSS IV, accessed 2024)

Links: <https://skyserver.sdss.org/dr12/en/proj/advanced/hubble/redshifts.aspx>  
<https://www.sdss4.org/dr17/algorithms/redshifts/>

<sup>3</sup> (Fedesoriano, 2022), (Abdurro'uf et al., 2022)

Link: <https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17/data>

<sup>4</sup> I ran the model with and without redshift included in the training features. I will discuss this later.

in each band ('u', 'g', 'r', 'i', 'z') indicates the magnitude of an object in those specific wavelengths. The magnitude is related to the flux, which refers to the amount of energy received per unit area per unit time. Flux is proportional to the brightness of an object, but the relationship is logarithmic, meaning that a smaller magnitude corresponds to a higher brightness.<sup>5</sup> Among the available features I also experimented with redshift, which in the SDSS dataset is exactly as it appears: a value representing the redshift of the celestial object at hand. (In this way, I indirectly *do* incorporate spectral data in my analysis since redshift is calculated using it!) Finally, SDSS has class labels which categorize an object as a GALAXY, STAR, or QSO (quasar). These five (six when with redshift) features and single class identifier will constitute the training and testing data for the models, respectively.

## D. Tools

While building both the baseline and more complex classifiers, I relied on a combination of libraries and tools that allowed me to preprocess data, build diverse ML models, and evaluate their performance. I structured and manipulated my dataset using NumPy and pandas. In ML, it is sometimes necessary to preprocess data for specific models in order to cater to their internal function. For example, for my baseline logistic regression model and multi-layer perceptron model, I needed to scale the data to values which the tools are most fit to interpret. Scikit-learn was an essential framework throughout the process—I used `train_test_split` to create training and testing sets, `StandardScaler` to normalize features, and pipelines to streamline preprocessing steps with model training. Recognizing the class imbalance in my data (approx. 60% of the objects are labeled as galaxies, a fact visible in both dummy models' accuracies of that approx. percentage), I implemented SMOTE (synthetic minority oversampling) to balance the data and thus improve model accuracy. To explore a variety of algorithms, I trained a logistic regression, decision tree, K-Nearest-Neighbors (KNN), Multi-Layer Perceptron (MLP), Gaussian Naive Bayes (GaussianNB), and neural network (NN) model. I also combined these models using a VotingClassifier model, to try ensemble learning. Visual tools like confusion matrices and Matplotlib allowed me to interpret the results and identify areas for improvement. To try out a more complexly architected model, I turned to TensorFlow's Keras API to design and fine-tune a (very simple) custom neural network. This toolkit gave me the flexibility to iterate and improve as I deepened my understanding of the problem.

## E. Lecture Connections (3)

### Relevance to Lecture 2: Flux

Right: SDSS site image describing the direct relationship between asinh magnitudes (the UGRIZ features I train on) and flux. →

Magnitudes in the SDSS system are expressed as inverse hyperbolic sine (asinh) magnitudes, a transformation that allows for consistent magnitude reporting even at low or negative flux values. This system smoothly approximates traditional logarithmic magnitudes at high signal-to-noise ratios but remains robust when dealing with faint

#### SDSS Asinh Magnitudes

Magnitudes within the SDSS are expressed as inverse hyperbolic sine (or "asinh") magnitudes, described in detail by [Lupton, Gunn & Szalay \(1999\)](#). They are sometimes referred to informally as *luptitudes*. The transformation from linear flux measurements to asinh magnitudes is designed to be virtually identical to the standard astronomical magnitude at high signal-to-noise ratio, but to behave reasonably at low signal-to-noise ratio and even at negative values of flux, where the logarithm in the [Pogson magnitude](#) fails. This allows us to report a magnitude even in the absence of a formal detection; we quote no upper limits in our photometry.

The asinh magnitudes are characterized by a softening parameter  $b$ , the typical 1-sigma noise of the sky in a PSF aperture in 1'' seeing. The relation between detected flux  $f$  and asinh magnitude  $m$  is:

$$m = -2.5/\ln(10) * [\operatorname{asinh}((f/f_0)/(2b)) + \ln(b)].$$

Here,  $f_0$  is given by the classical zero point of the magnitude scale, i.e.,  $f_0$  is the flux of an object with *conventional* magnitude of zero. The quantity  $b$  is measured relative to  $f_0$ , and thus is in maggies; it is given in the [table of asinh softening parameters](#) below (Table 21 in the [EDR paper](#)), along with the asinh magnitude associated with a zero flux object. The table also lists the flux corresponding to  $10f_0$ , above which the asinh magnitude and the traditional logarithmic magnitude differ by less than 1% in flux. For details on converting asinh magnitudes to other flux measures, see [converting counts to magnitudes](#).

| Asinh Softening Parameters |                       |  |                  |  |
|----------------------------|-----------------------|--|------------------|--|
| Filter                     | $b$                   | Zero-flux Magnitude [ $m(f/f_0 = 0)$ ] | $m(f/f_0 = 10b)$ |  |
| u                          | $1.4 \times 10^{-10}$ | 24.63                                  | 22.12            |  |
| g                          | $0.9 \times 10^{-10}$ | 25.11                                  | 22.60            |  |
| r                          | $1.2 \times 10^{-10}$ | 24.80                                  | 22.29            |  |
| i                          | $1.8 \times 10^{-10}$ | 24.36                                  | 21.85            |  |
| z                          | $7.4 \times 10^{-10}$ | 22.83                                  | 20.32            |  |

<sup>5</sup> (SDSS IV, accessed 2024)

Link: <https://www.sdss4.org/dr17/algorithms/magnitudes/>

detections. By transforming raw flux data into asinh magnitudes, the SDSS dataset provides a standardized and noise-tolerant measure of an object's brightness. This project's reliance on magnitude values draws on our study of flux as a linear measure of light intensity, as these magnitudes are derived directly from flux through a precise mathematical relationship (described in the image above). Additionally, my use of UGRIZ asinh magnitudes not only reflects the concept of Flux discussed in lecture but also underscores the dataset's practical utility in robust classification tasks (due to its preprocessing-esque control for noise).

## Relevance to Lecture 4: Propagation of Error and Uncertainty

Empirical uncertainty in the SDSS data, stemming from factors like noise in photometric measurements or variability in observational conditions, directly impacts model performance by introducing inconsistencies in the training data. These uncertainties are propagated in a complex way determined by model structure, leading to reduced accuracy as the model may struggle to distinguish between true patterns and noise. Faint objects in the SDSS data with low signal-to-noise ratios can be (and have been<sup>6</sup>) mislabeled or misclassified, which skews ML models' ability to generalize. I addressed this challenge by incorporating techniques like data preprocessing (SMOTE, scaling, etc.) and recording model uncertainty (in the form of variance or entropy) to mitigate the effects of empirical uncertainty and improve the robustness of my predictions. Additionally, incorporating the Gaussian Naive Bayes (GaussianNB) model into my analysis draws on our study of the Gaussian distribution. GaussianNB assumes that the features follow a normal distribution<sup>7</sup>, enabling it to calculate class probabilities efficiently using mean and variance estimates. This aligns with the statistical principles we explored, such as how the Gaussian distribution can model natural phenomena and variability. By leveraging this model, I applied these theoretical insights to classify objects based on their photometric (and redshift) features, linking the probabilistic foundation of Gaussian distributions to practical machine learning applications.

## Relevance to Lecture 17: Light Spectra and Redshift

I experimented with redshift as a sixth additional training feature to enhance model training, which required direct knowledge from our lecture on redshift and light spectra. Redshift, which measures how much the wavelength of light from an object stretches as it moves away from us, is a critical tool in understanding the universe's expansion and an object's distance/velocity relative to Earth. The SDSS calculates redshift using spectral features such as absorption and emission lines, shifting systematically with increasing wavelength. With redshift included, all models became extremely more accurate, with even the baseline logistic regression model pushing past 90% accuracy in its predictions (to be discussed later). A discussion of redshift will be necessary as I attempt to explain this vast improvement so directly related to inclusion of the redshift feature.

---

<sup>6</sup> (SDSS IV, accessed 2024): "The SDSS distinguishes between stars and galaxies based on their shapes: single points of light are stars, and fuzzy patches of light are galaxies. Some stars are bright enough that their light saturates the camera, and thus the light spills over the image, so to the SDSS's camera, they look like fuzzy disks instead of single points of light. Their appearance fools the SDSS's software into classifying them as galaxies."

Link: <https://sdss.org/dr18/help/faq/>

<sup>7</sup> This isn't true for all five UGRIZ features, but the distribution in 'g', 'r', and 'i' features among the data somewhat resembled the Gaussian distribution, so I gave the model a shot. (It performed terribly, actually, underperforming the dummy classifier which always guesses the most frequent class type in the dataset. (Continued from the footnote on the previous page:.) However, an understanding of Lecture 4's Gaussian content nevertheless underpinned my exploration of this model's potential for high accuracy in the classification task.)

## II. Code

A. [Link to My Colab Notebook](#) (All code here)

B. Model Performance Output and Visualizations

### Baseline Models: Preface

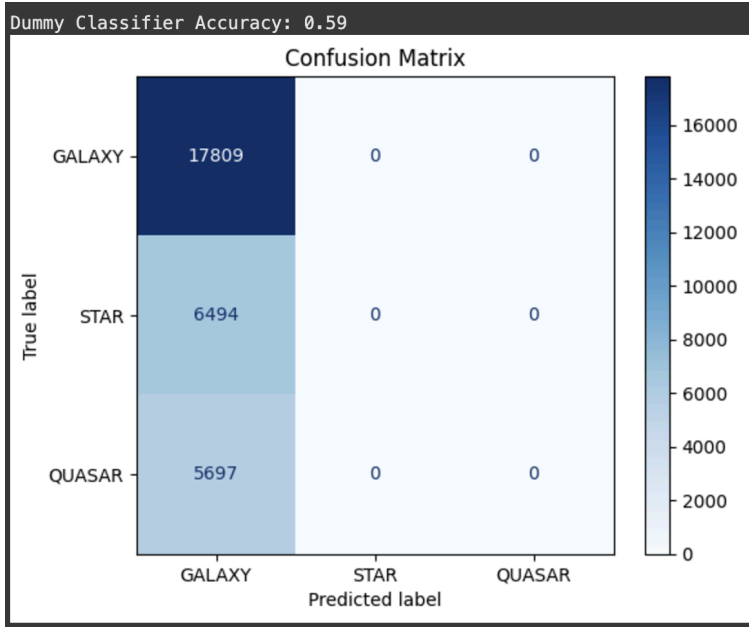
Baseline models serve as, well, a baseline. I use two: the Dummy Classifier, which just predicts the most frequent class among the data, and a Logistic Regression, which predicts the probability of an input belonging to a particular class by fitting a logistic function to the data. The former is as simple as possible and serves as an extreme baseline—a model is at least *somewhat* useful if it outperforms the most rudimentary way of predicting a celestial object's class. The latter is still a simple model but less trivial of a construction, so it provides a secondary, more reasonable baseline which, if a model outperforms it, indicates that that model is useful in a more significant sense; the model's architecture must be making a meaningful contribution to improving classification accuracy.

Now, onto the cool stuff!

## UGRIZ Only

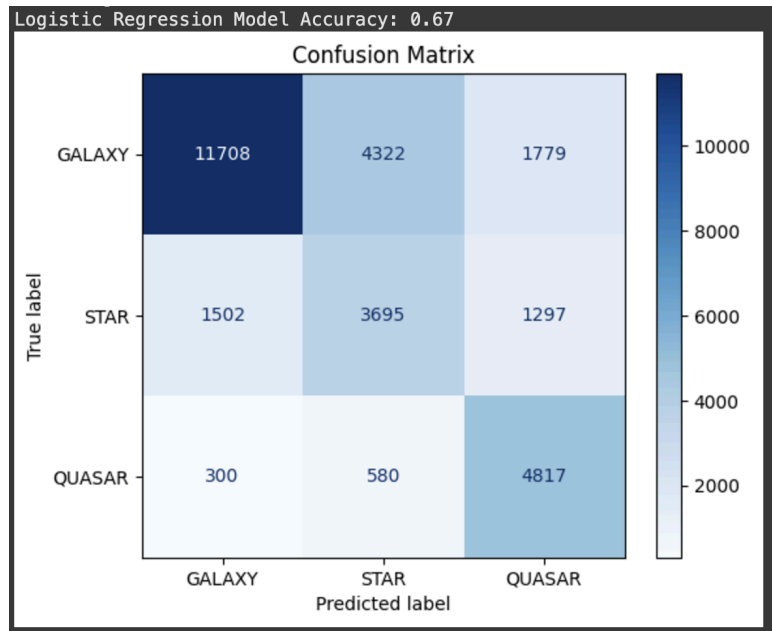
(Baseline)

### Dummy Classifier



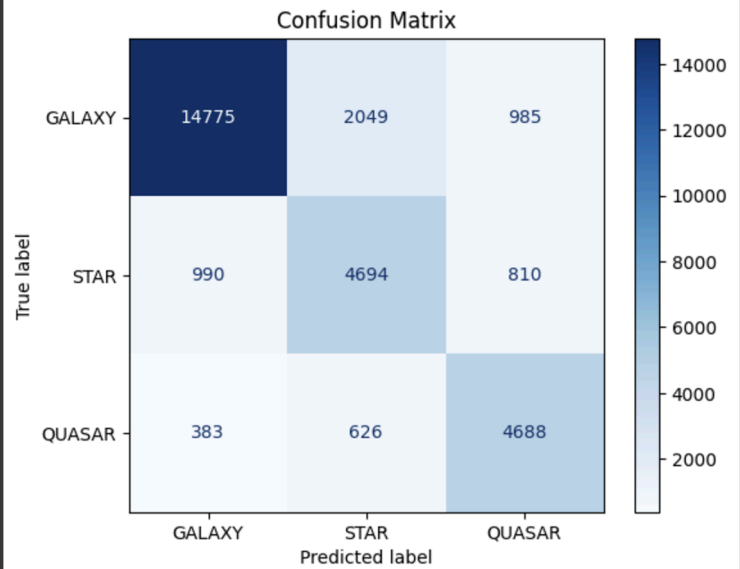
(Baseline)

### Logistic Regression



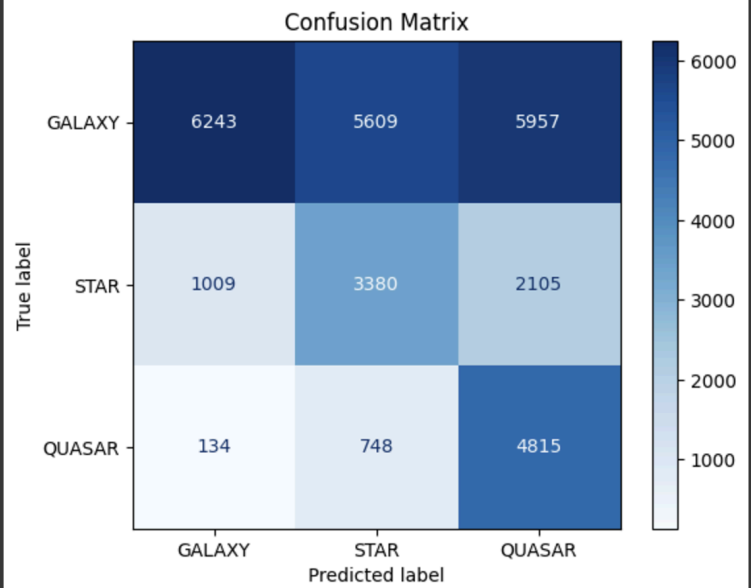
Decision Tree

Decision Tree Model Accuracy: 0.81  
Probabilities for the first few predictions:  
[[0.85138387 0.14259928 0.00601685]  
[0.93918919 0.03909266 0.02171815]  
[0.045053 0.68992933 0.26501767]  
[0.56202532 0.43544304 0.00253165]  
[0.84906034 0.10108803 0.04985163]]  
Uncertainty (variance) for the first few predictions:  
[0.84536703 0.91747104 0.64487633 0.55949367 0.7992087 ]



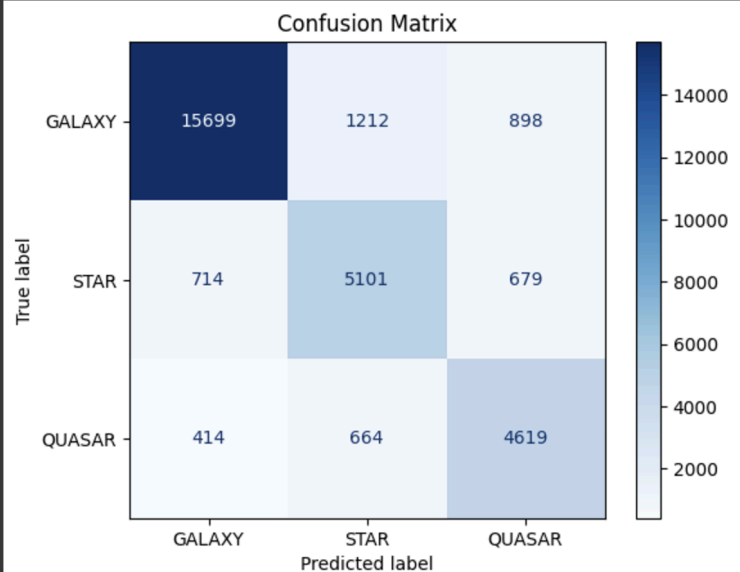
GaussianNB

Gaussian Naive Bayes Model Accuracy: 0.48  
Probabilities for the first few predictions:  
[[3.93889392e-01 6.05890182e-01 2.20426139e-04]  
[1.88506158e-01 8.11493517e-01 3.25440958e-07]  
[1.34058441e-01 7.76615432e-02 7.88280016e-01]  
[3.43334737e-01 6.56632043e-01 3.32204903e-05]  
[8.72806599e-01 8.06225559e-02 4.65708456e-02]]  
Uncertainty (variance) for the first few predictions:  
[0.06297283 0.12024099 0.10401834 0.07190368 0.14570896]



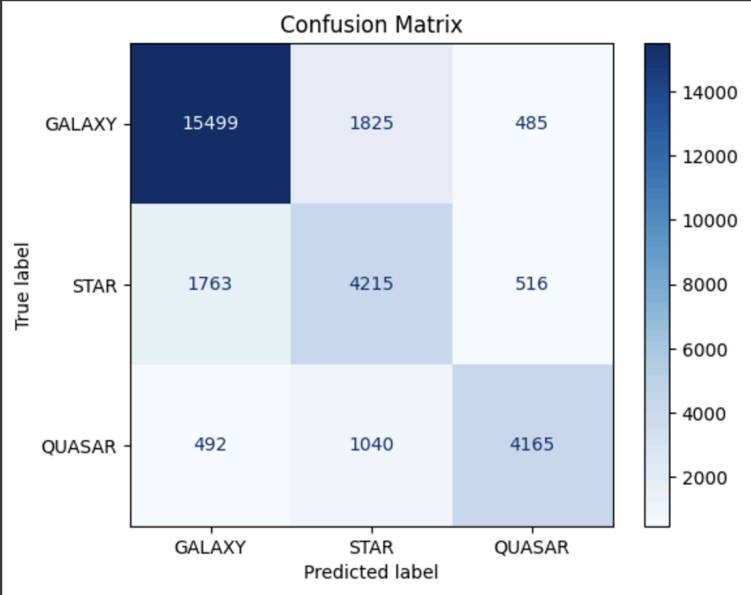
KNN

KNN Model Accuracy: 0.85  
Probabilities for the first few predictions:  
[[1. 0. 0.]  
[1. 0. 0.]  
[0. 0.9 0.1]  
[1. 0. 0.]  
[1. 0. 0.]]  
Uncertainty (entropy not variance here) for the first few predictions  
[-1.00000008e-10 -1.00000008e-10 3.25082973e-01 -1.00000008e-10  
-1.00000008e-10]



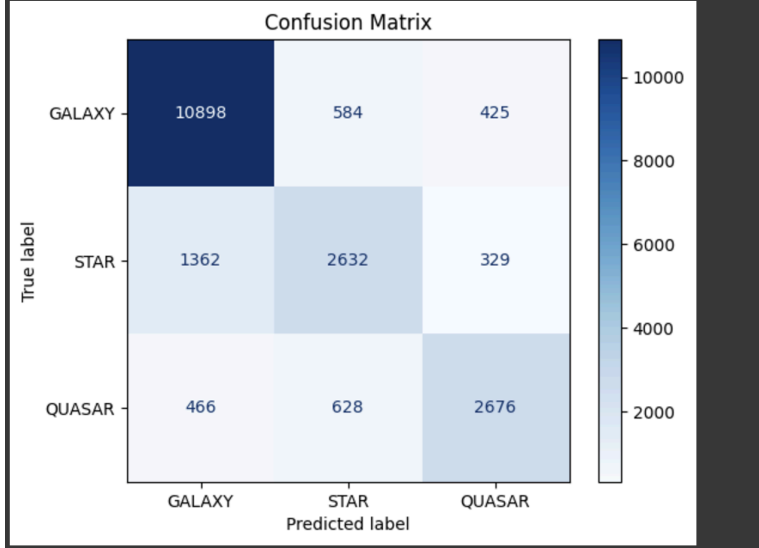
MLP

Probabilities for the first few predictions:  
[[0.7193702 0.20607425 0.07455555]  
[0.72190609 0.20413556 0.07395835]  
[0.72190609 0.20413556 0.07395835]  
[0.72190609 0.20413556 0.07395835]  
[0.72190609 0.20413556 0.07395835]]  
Uncertainty (entropy) for the first few predictions:  
[0.75600562 0.75221223 0.75221223 0.75221223 0.75221223]  
MLPClassifier Model Accuracy: 0.80

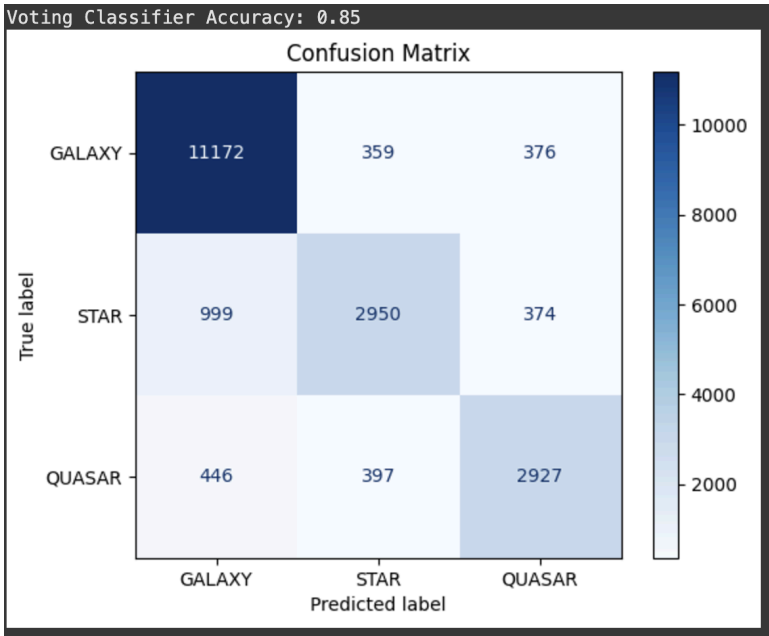


NN

```
625/625 ————— 1s 2ms/step - accuracy: 0.8155 - loss: 0.5353
625/625 ————— 1s 2ms/step
Test Accuracy: 0.8102999925613403
Probabilities for the first few predictions:
[[0.2949849  0.04815694 0.65685815]
 [0.3087138  0.04635661 0.6449296 ]
 [0.8925029  0.0857252  0.02177184]
 [0.23136234 0.6480583  0.12057932]
 [0.8548952  0.09578426 0.04932067]]
Uncertainty (entropy) for the first few predictions:
[0.78226984 0.7880984 0.3954174 0.8748518 0.5071306 ]
```



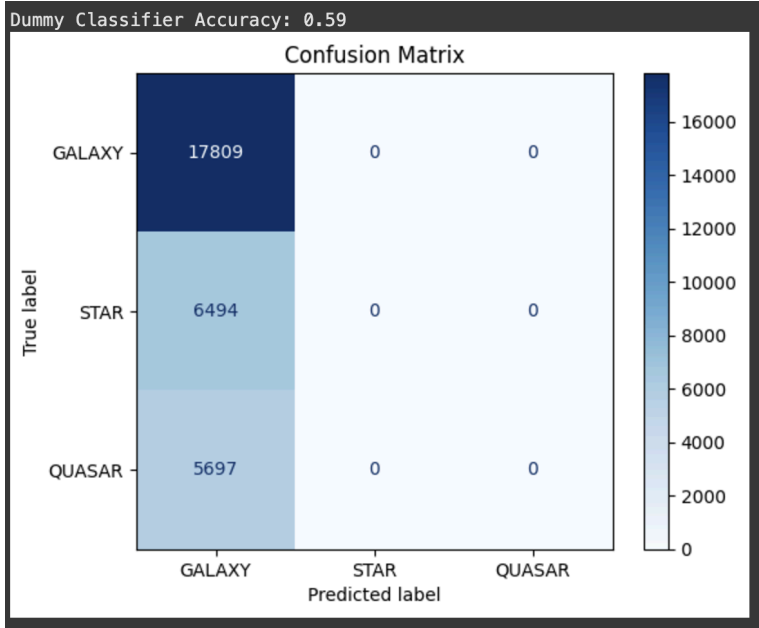
Voting Classifier



UGRIZ-Redshift

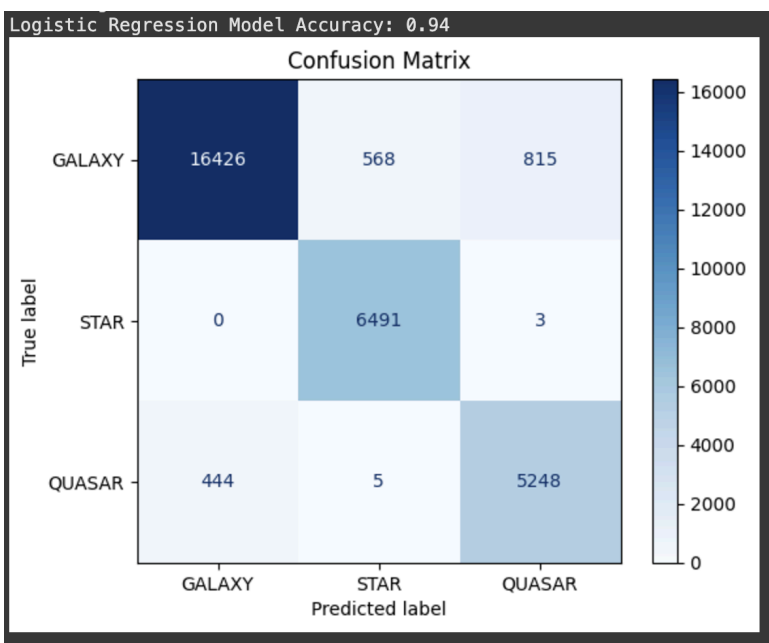
(Baseline)

Dummy Classifier-R



(Baseline)

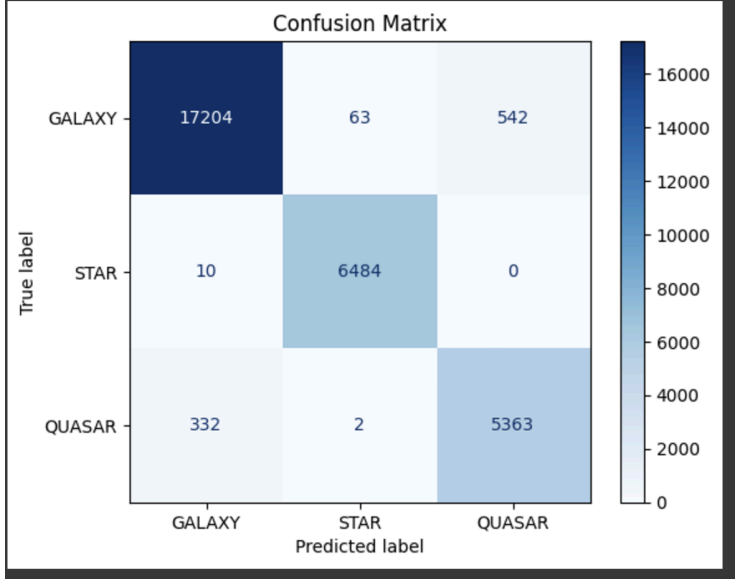
Logistic Regression-R





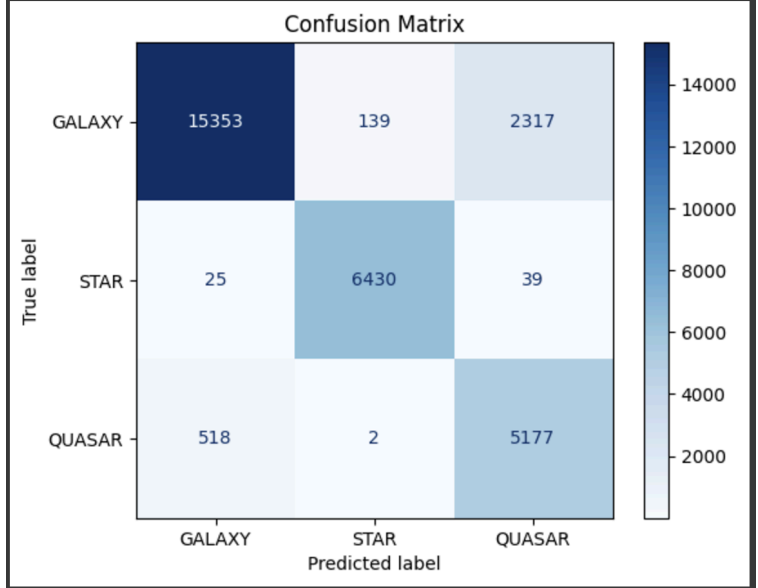
Decision Tree-R

Decision Tree Model Accuracy: 0.97  
Probabilities for the first few predictions:  
[[9.88692580e-01 0.00000000e+00 1.13074205e-02]  
[9.92989165e-01 0.00000000e+00 7.01083493e-03]  
[2.08159867e-04 9.99791840e-01 0.00000000e+00]  
[9.88692580e-01 0.00000000e+00 1.13074205e-02]  
[9.79220333e-01 0.00000000e+00 2.07796668e-02]]  
Uncertainty (variance) for the first few predictions:  
[0.98869258 0.99298917 0.99979184 0.98869258 0.97922033]



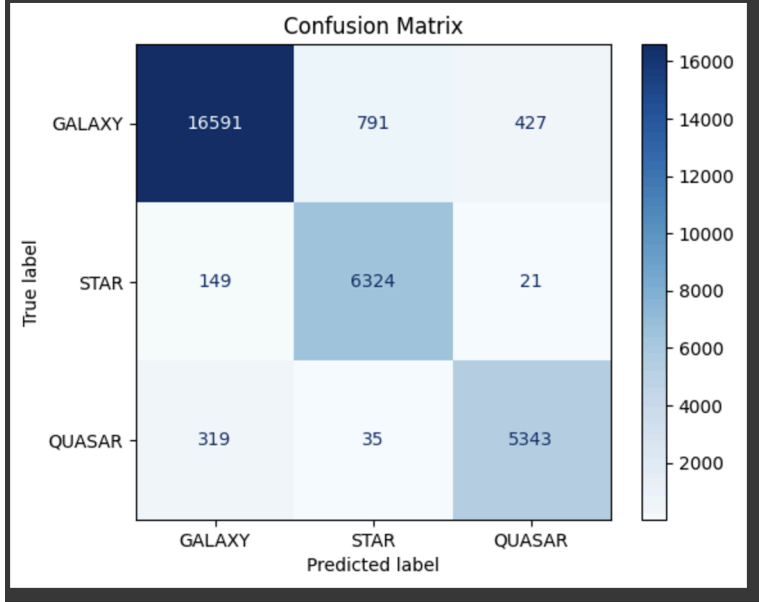
GaussianNB-R

Gaussian Naive Bayes Model Accuracy: 0.90  
Probabilities for the first few predictions:  
[[9.99948512e-01 0.00000000e+00 5.14881382e-05]  
[9.99999813e-01 0.00000000e+00 1.86528613e-07]  
[2.25428639e-03 9.95459654e-01 2.28606004e-03]  
[9.99991037e-01 0.00000000e+00 8.96308309e-06]  
[9.94017022e-01 0.00000000e+00 5.98297831e-03]]  
Uncertainty (variance) for the first few predictions:  
[0.2221879 0.2222221 0.21920563 0.22221625 0.21825743]




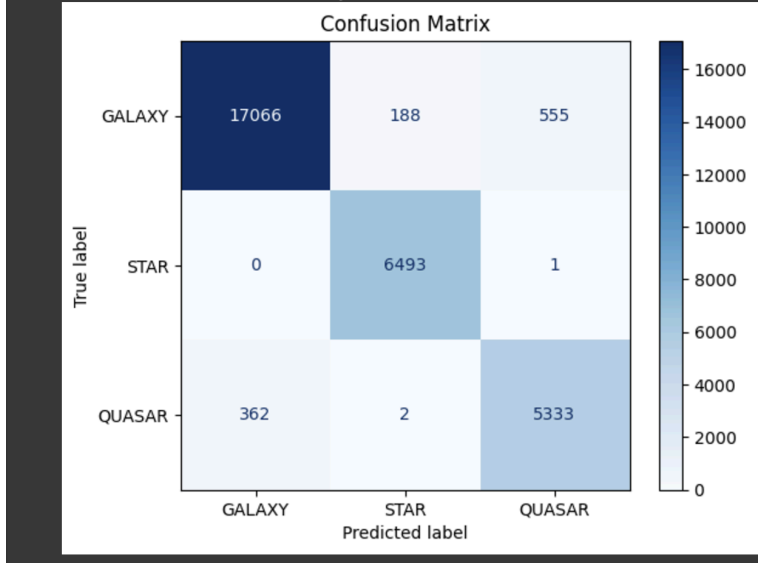
KNN-R

KNN Model Accuracy: 0.94  
Probabilities for the first few predictions:  
[[9.99948512e-01 0.00000000e+00 5.14881382e-05]  
[9.99999813e-01 0.00000000e+00 1.86528613e-07]  
[2.25428639e-03 9.95459654e-01 2.28606004e-03]  
[9.99991037e-01 0.00000000e+00 8.96308309e-06]  
[9.94017022e-01 0.00000000e+00 5.98297831e-03]]  
Uncertainty (entropy not varaince here) for the first few predictions:  
[5.59888681e-04 3.07653001e-06 3.21710831e-02 1.13135347e-04  
3.65909338e-02]



MLP-R

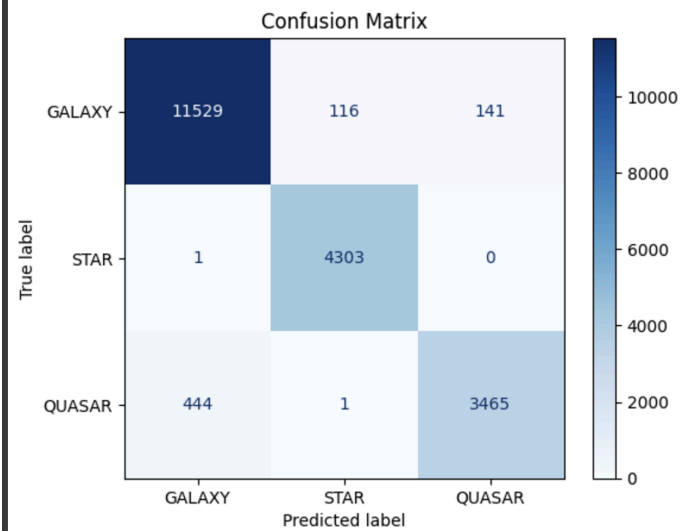
 /usr/local/lib/python3.10/dist-packages/sklearn/base.py:486: UserWarning: warnings.warn(  
Probabilities for the first few predictions:  
[[9.26791120e-01 1.80094884e-36 7.32088795e-02]  
[9.26791120e-01 1.80094884e-36 7.32088795e-02]  
[9.26791120e-01 1.80094884e-36 7.32088795e-02]  
[9.26791120e-01 1.80094884e-36 7.32088795e-02]  
[9.82695214e-01 5.60793993e-28 1.73047859e-02]]  
Uncertainty (entropy) for the first few predictions:  
[0.26186133 0.26186133 0.26186133 0.26186133 0.08735576]  
MLPClassifier Model Accuracy: 0.96





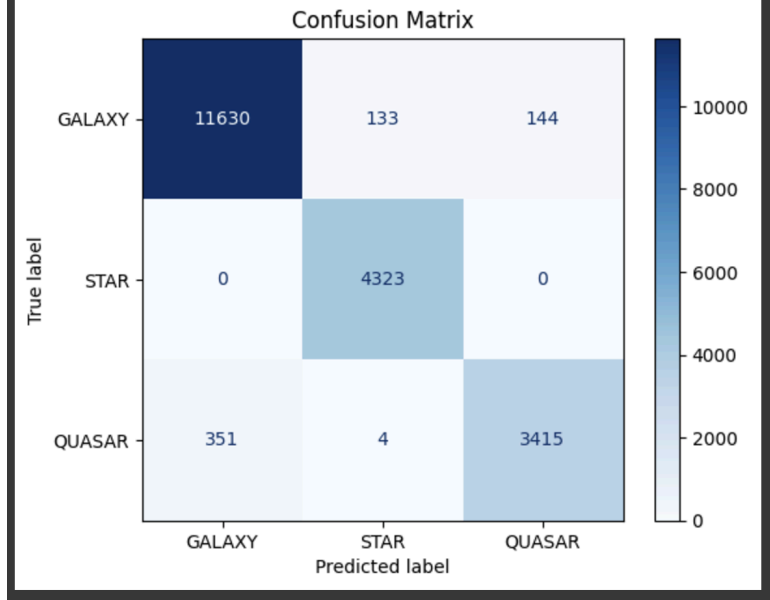
## NN-R

```
625/625 ————— 1s 1ms/step - accuracy: 0.9657 - loss: 0.7495
625/625 ————— 1s 1ms/step
Test Accuracy: 0.964850084877014
Probabilities for the first few predictions:
[[5.4677755e-01 0.0000000e+00 4.5322233e-01]
 [9.8189092e-01 0.0000000e+00 1.8109029e-02]
 [5.7521433e-01 0.0000000e+00 4.2478573e-01]
 [9.9395788e-01 0.0000000e+00 6.0421014e-03]
 [1.8682478e-02 9.8128963e-01 2.7866858e-05]]
Uncertainty (entropy) for the first few predictions:
[0.6887645 0.09058566 0.68178976 0.03689295 0.09318593]
```



## Voting Classifier-R

Voting Classifier Accuracy: 0.97



## III. Takeaways

### IV. UBRIZ Only

When evaluating the performance of no-redshift models against their no-redshift baselines (the Dummy and Logistic Regression models), some notable patterns emerge. The Dummy baseline—unsurprisingly—achieves a low accuracy (59%), as expected since it predicts the most frequent class without learning any patterns in the data. The Logistic Regression (LR), though it is limited to linear decision boundaries (not the best fit here but it's better than a linear regression), provides a better benchmark with an accuracy of 0.67—an indication that the dataset may actually contain linearly separable features. (I initially expected the relationship between UGRIZ and class to be complex and totally non-linear.) From there, the Decision Tree (DT) model shows a substantial improvement, reaching 0.81 accuracy (14% higher than LR!!), leveraging its ability to capture non-linear interactions. However, its variances (ranging up to 0.92) points to some overfitting that I would ideally go back and adjust for. On the other hand, the GaussianNB model struggles—delivering just 0.48 accuracy, which is somehow even worse than even the Dummy model. Its reliance on feature independence assumptions (which I'm realizing is definitely a poor fit here) likely contributes to its overconfident but inaccurate probability estimates. The K-Nearest Neighbors (KNN) model, though, emerges as the strongest performer, with an accuracy of 0.85—with a very significant 26% improvement over the LR. While its confidence levels (4 out of the 5 sampled having suspiciously impressive entropy near  $-1e-10$ ) skew toward what one might call over-certainty, the KNN's ability to exploit localized data patterns is evident. The neural

network-based models—the Multi-Layer Perceptron (MLP, 0.80 accuracy) and the NN (0.8155 accuracy)—either match or slightly outperform the DT, with more calibrated confidence levels (entropies consistently moderate) suggesting better handling of decision boundaries. Overall, the KNN stands out as the top performer, though its confidence really evokes my suspicion. Neural network models and the DT classifier, on the other hand, provide competitive alternatives with greater stability.

## V. UBRIZ + Redshift

The inclusion of the redshift feature reshapes model performance—revealing a notable influence on their predictive power. The Dummy model, still grounded in majority-class prediction, achieves an unchanged accuracy of 0.59, confirming its inability to leverage additional information (and truly fulfilling its duties as our ultimate baseline for comparison between the two sets of models). The Logistic Regression, though, benefits from an apparent linear separability introduced by redshift, improving to an eye-boggling 0.94 accuracy—27% gain over its no-redshift counterpart, signaling the feature’s massive importance even for relatively simple models. Among the other models, Decision Trees and the Voting Classifier stand out with both having accuracies of 0.97 (achieving 3% higher than the not-so-baseline-now Logistic Regression). The redshift feature, then, slightly enhances the model’s ability to capture non-linear patterns, but due to the massive increase in performance of the LR this may be negligible in the face of its impact on linear pattern identification. Gaussian Naive Bayes remains an outlier...despite a massive bump to 0.9 accuracy, it still underperforms the LR! This indicates that its simplifying assumptions outweigh even the benefits of redshift. For K-Nearest Neighbors, the impact of redshift is, as expected, significant. With accuracy jumping to 0.94—a 9% improvement from the no-redshift version—the model capitalizes on localized similarity relationships, which appear to be enriched by the feature. Neural networks also benefit from the added value of redshift, with the Multi-Layer Perceptron reaching 0.96 accuracy and the NN slightly surpassing it at 0.9657 (both surpassing the LR by >2%). Both models exhibit confidence and variance/entropy profiles that suggest redshift not only improves predictions, but also contributes to more nuanced decision-making processes. Altogether, the inclusion of redshift amplifies performance across nearly all models, with particularly pronounced effects for Logistic Regression and KNN. While Decision Trees and neural networks still exhibit added value with or without the redshift feature included, the GaussianNB’s still-underperforming accuracy and overconfidence reinforce its lack thereof.

## VI. Synthesis and Conclusion

Including the redshift feature transforms the predictive landscape across all models, indicating that it plays a critical role in connecting features to the class labels of celestial objects. The most striking improvement occurs with Logistic Regression (LR), where accuracy jumps from 0.67 to 0.94—a staggering 27% increase. This suggests that redshift introduces a powerful linear relationship between the features and the class label, allowing LR, which specializes in linear decision boundaries, to exploit this structure effectively. This linear separability might stem from the role of redshift as a proxy for distance or movement, which could correlate directly with physical properties of celestial objects used in classification (see paragraph below). Beyond LR, other models also capitalize on the inclusion of redshift, but to varying degrees. The Decision Tree and Voting Classifier achieve an impressive 0.97 accuracy, slightly surpassing LR, though their non-linear capabilities suggest diminishing returns from redshift's

influence once other relationships are already captured. Neural network-based models (the MLP and NN) also see performance boosts, reaching 0.96 and 0.9657 accuracy, respectively, reflecting their ability to synthesize both linear and complex, non-linear patterns. The KNN benefits significantly too, improving to 0.94 accuracy, where redshift enriches its localized similarity calculations. Interestingly, The GaussianNB model, despite an increase to 0.90 accuracy, remains a poor performer relative to other models, underperforming the LR yet again, this time by 4%. This suggests that its rigid assumptions about feature independence are incompatible with the intricate relationships redshift introduces. Overall, redshift's inclusion not only boosts accuracy across models but also reveals key insights about the relationship between the five features (UGRIZ) and the class label. While neural networks, decision trees, and KNN show improvements driven by enriched non-linear patterns, the enormous gain in LR performance highlights redshift's pivotal role in reinforcing linear patterns within the dataset. This suggests that the combination of redshift with the other features creates a layered relationship, where both simple and complex models can exploit complementary information for improved classification. (Though minimal benefit, the presence of added value by this combination of linear and complex relationship consideration *is* exemplified by the 2 to 3 percent gains over the 0.94 LR baseline accuracy by all models which displayed significant added value in the non-redshift set.)

As for the reason for its huge improvement on prediction power, redshift may have this impact because it is related directly to distance. Stars, being closest to Earth, exhibit negligible redshifts, while quasars, the most distant and luminous objects, have the highest redshifts. Galaxies would be somewhere in between. I would conclude that this introduces a type of clear order to the data—a linear type of pattern—a truth reflected in how even the baseline Logistic Regression (which does not capture complex relationships very well but does capture linear ones) achieves over-90% accuracy. By incorporating redshift, the models easily identify this fundamental astrophysical relationship, transforming a complex classification problem into one with clearer boundaries. Then, the added perspective offered by the more complexly-class-related UGRIZ features contributes additional information to fine tune model decisions, boosting prediction power.

## VII. Bibliography

1. Abdurro'uf, A., et al. 2022. "The Seventeenth Data Release of the Sloan Digital Sky Surveys: Complete Release of MaNGA, MaStar and APOGEE-2 Data." *The Astrophysical Journal Supplement Series* 259: 35. <https://iopscience.iop.org/article/10.3847/1538-4365/ac4414>.
2. Chollet, François, et al. 2015. "Keras." Accessed December 8, 2024. <https://keras.io>.
3. Fedesoriano. 2022. "Stellar Classification Dataset - SDSS17." *Kaggle*. Accessed December 8, 2024. <https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17>.
4. Fukugita, M., et al. 1996. "The Sloan Digital Sky Survey Photometric System." *The Astronomic Journal* 111: 1748. <https://doi.org/10.1086/117915>.
5. GeeksforGeeks. "ML: Voting Classifier Using Sklearn." GeeksforGeeks, November 25, 2019. <https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/>.
6. GeeksforGeeks. "Naive Bayes Classifiers." GeeksforGeeks, July 10, 2024. <https://www.geeksforgeeks.org/naive-bayes-classifiers/>.
7. Hunter, J. D.. "Matplotlib: A 2D graphics environment." *Computing In Science & Engineering* 9 , no. 3 (2007): 90--95.
8. Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python." *The Journal of machine Learning research* 12 (2011): 2825-2830.
9. SDSS III. "Redshifts." *SDSS DR12*. Accessed December 8, 2024. <https://skyserver.sdss.org/dr12/en/proj/advanced/hubble/redshifts.aspx>.
10. SDSS IV. "Frequently Asked Questions" *SDSS DR18*. Accessed December 8, 2024. <https://sdss.org/dr18/help/faq/>.
11. SDSS IV. "Measures of Flux and Magnitude" *SDSS DR17*. Accessed December 8, 2024. <https://www.sdss4.org/dr17/algorithms/magnitudes/>.
12. SDSS IV. "Redshifts, Classifications and Velocity Dispersions." *SDSS DR17*. Accessed December 8, 2024. <https://www.sdss4.org/dr17/algorithms/redshifts/>.