Capstone Project
Data Science Nanodegree

Tyler McIntyre
July 31, 2024

Stellar Classification

Definition

Project Overview

In this project I demonstrate how we can train machine learning models to classify Stars, Galaxies and Quasars using data collected by the SDSS (Sloan Digital Sky Survey-V: Pioneering Panoptic Spectroscopy - SDSS-V).

Problem Statement

To efficiently identify interstellar bodies in large datasets, we can classify them by analyzing their spectral characteristics. This enables further analysis to uncover patterns, characteristics, and behaviors that enhance our understanding of the universe.

Scoring

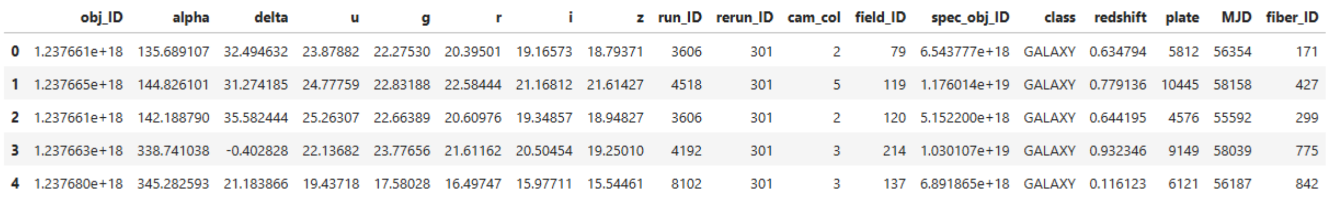
We can evaluate the performance and success of our predictions by selecting the best machine learning models for this problem and analyzing the optimal combination of parameters. Our goal is to identify the most efficient and accurate models, with accuracy being the more crucial factor.

To measure accuracy, we will divide the number of correct predictions by the total number of predictions made. For efficiency, we will measure the time it takes to train the model. We will explore two different types of models, each with potential trade-offs to consider.

Analysis

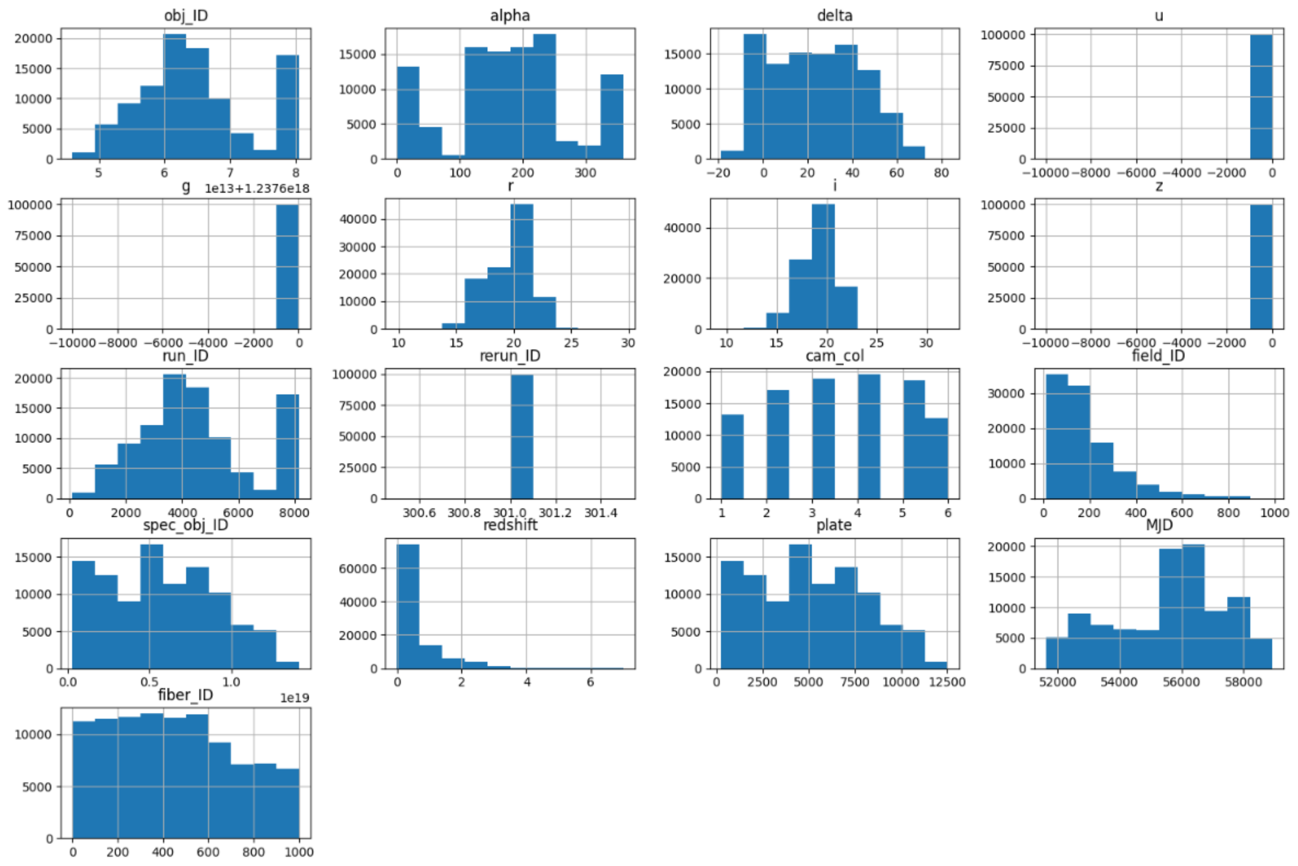
Data Exploration and Visualization

The dataset comprises observations collected by the Sloan Digital Sky Survey (SDSS). Each observed object has a unique identifier (obj_ID). The dataset includes 78,053 unique observations, each classified as a star, quasar, or galaxy.

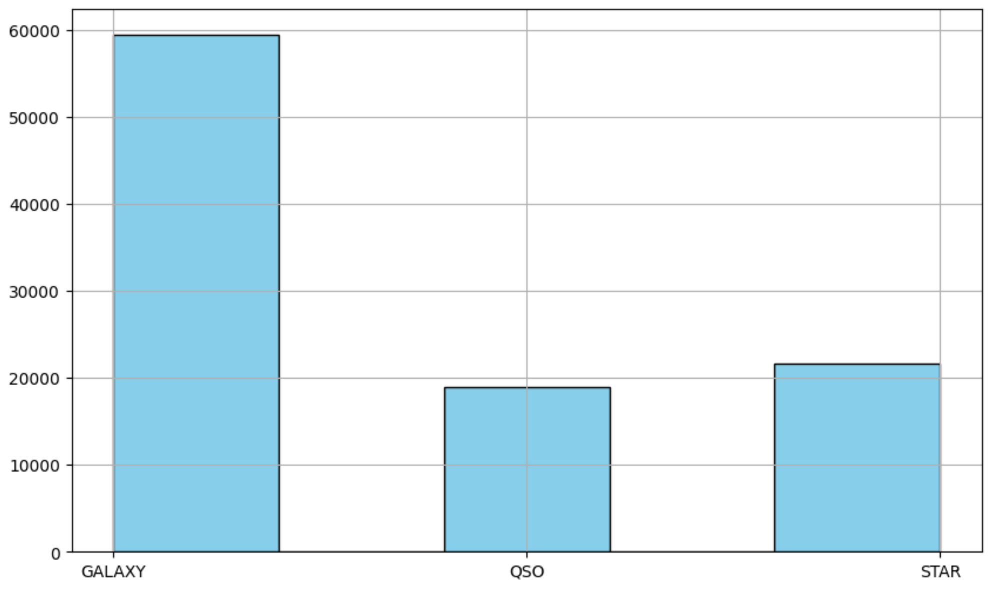


	obj_ID	alpha	delta	u	g	r	i	z	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID	class	redshift	plate	MJD	fiber_ID
0	1.237661e+18	135.689107	32.494632	23.87882	22.27530	20.39501	19.16573	18.79371	3606	301	2	79	6.543777e+18	GALAXY	0.634794	5812	56354	171
1	1.237665e+18	144.826101	31.274185	24.77759	22.83188	22.58444	21.16812	21.61427	4518	301	5	119	1.176014e+19	GALAXY	0.779136	10445	58158	427
2	1.237661e+18	142.188790	35.582444	25.26307	22.66389	20.60976	19.34857	18.94827	3606	301	2	120	5.152200e+18	GALAXY	0.644195	4576	55592	299
3	1.237663e+18	338.741038	-0.402828	22.13682	23.77656	21.61162	20.50454	19.25010	4192	301	3	214	1.030107e+19	GALAXY	0.932346	9149	58039	775
4	1.237680e+18	345.282593	21.183866	19.43718	17.58028	16.49747	15.97711	15.54461	8102	301	3	137	6.891865e+18	GALAXY	0.116123	6121	56187	842

In the initial histogram plot, we can get a broad overview of the data distribution and skew.



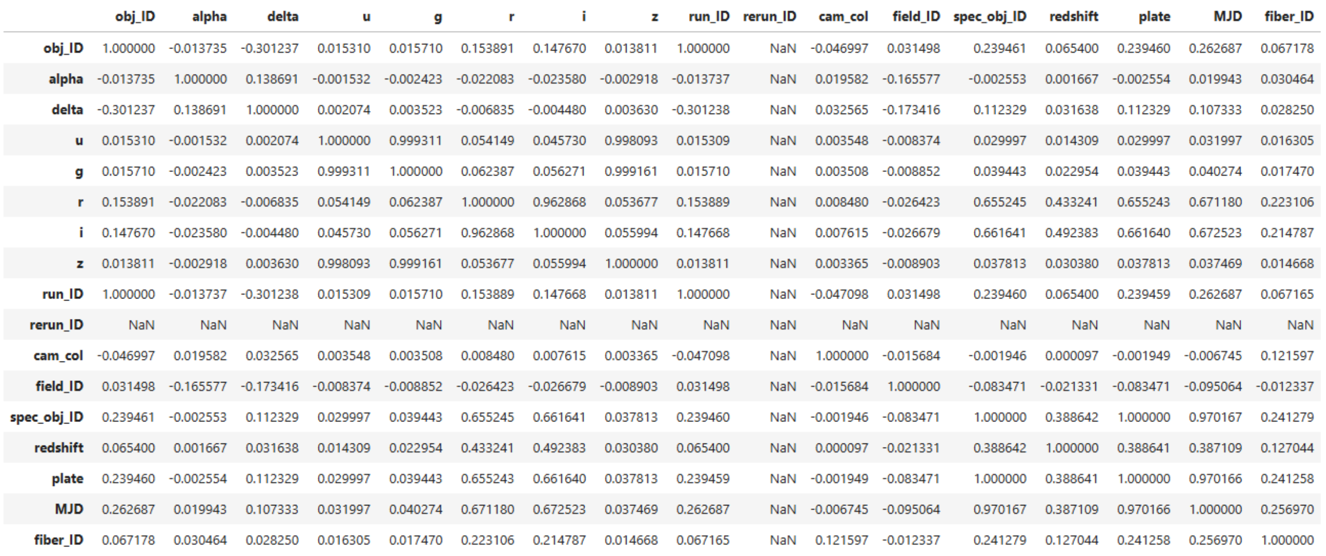
Looking closer at the classifications, we can see that the observations are heavily skewed towards galaxies, with stars being the second most frequent classification. There is a difference of roughly 40,000 observations between galaxies and stars. This imbalance will require data balancing to prevent bias towards classifying galaxies. One explanation for this could be that galaxies emit more light than individual stars, as they are clusters of potentially millions of stars.



Alpha (right ascension) and delta (declination) represent the object's location in degrees when observed in space. Using Pearson's correlation coefficient, we can see that there is little to no relationship between the alpha and delta values for each category. This suggests that the distribution of objects in the sky is random with respect to these coordinates, and changes in right ascension do not predict changes in declination.

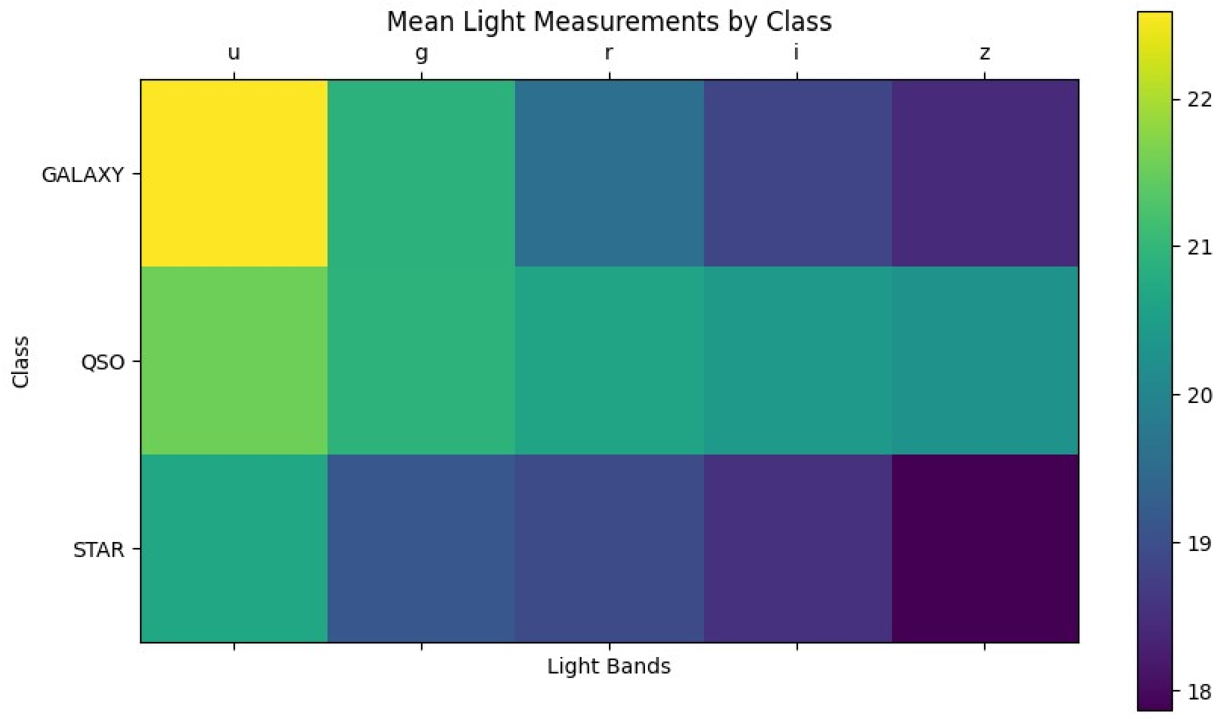
The columns u, g, r, i, and z pertain to the spectrum of light emitted by the observed objects. This spectrum can provide clues about the distance, size, and temperature of these objects.

Using a correlation matrix of our numerical columns, we can identify some strong positive correlations, especially related to the light emitted. Stars, galaxies, and quasars emit a broad spectrum of light, so when one type of light is present, others are likely to be present as well. Galaxies, which contain millions of stars, support a wide spectrum of light being emitted.

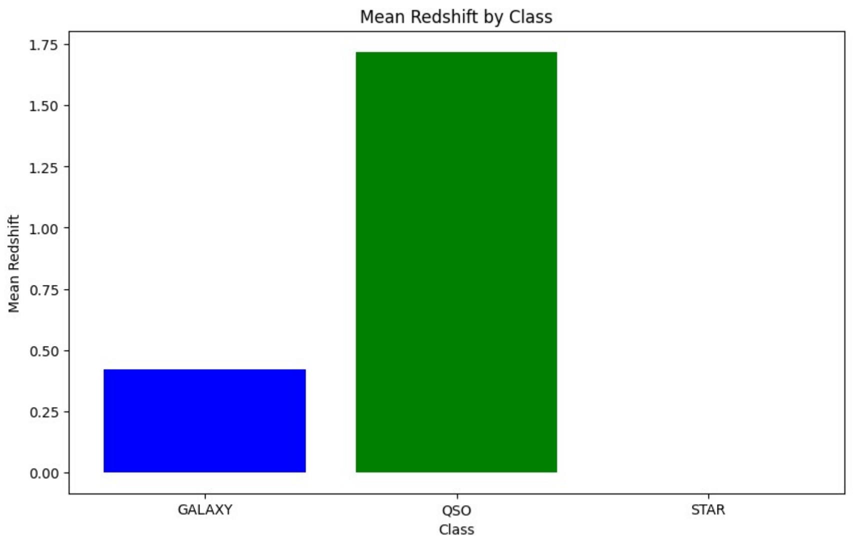


	obj_ID	alpha	delta	u	g	r	i	z	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID	redshift	plate	MJD	fiber_ID
obj_ID	1.000000	-0.013735	-0.301237	0.015310	0.015710	0.153891	0.147670	0.013811	1.000000	NaN	-0.046997	0.031498	0.239461	0.065400	0.239460	0.262687	0.067178
alpha	-0.013735	1.000000	0.138691	-0.001532	-0.002423	-0.022083	-0.023580	-0.002918	-0.013737	NaN	0.019582	-0.165577	-0.002553	0.001667	-0.002554	0.019943	0.030464
delta	-0.301237	0.138691	1.000000	0.002074	0.003523	-0.006835	-0.004480	0.003630	-0.301238	NaN	0.032565	-0.173416	0.112329	0.031638	0.112329	0.107333	0.028250
u	0.015310	-0.001532	0.002074	1.000000	0.999311	0.054149	0.045730	0.998093	0.015309	NaN	0.003548	-0.008374	0.029997	0.014309	0.029997	0.031997	0.016305
g	0.015710	-0.002423	0.003523	0.999311	1.000000	0.062387	0.056271	0.999161	0.015710	NaN	0.003508	-0.008852	0.039443	0.022954	0.039443	0.040274	0.017470
r	0.153891	-0.022083	-0.006835	0.054149	0.062387	1.000000	0.962868	0.053677	0.153889	NaN	0.008480	-0.026423	0.655245	0.433241	0.655243	0.671180	0.223106
i	0.147670	-0.023580	-0.004480	0.045730	0.056271	0.962868	1.000000	0.055994	0.147668	NaN	0.007615	-0.026679	0.661641	0.492383	0.661640	0.672523	0.214787
z	0.013811	-0.002918	0.003630	0.998093	0.999161	0.053677	0.055994	1.000000	0.013811	NaN	0.003365	-0.008903	0.037813	0.030380	0.037813	0.037469	0.014668
run_ID	1.000000	-0.013737	-0.301238	0.015309	0.015710	0.153889	0.147668	0.013811	1.000000	NaN	-0.047098	0.031498	0.239460	0.065400	0.239459	0.262687	0.067165
rerun_ID	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cam_col	-0.046997	0.019582	0.032565	0.003548	0.003508	0.008480	0.007615	0.003365	-0.047098	NaN	1.000000	-0.015684	-0.001946	0.000097	-0.001949	-0.006745	0.121597
field_ID	0.031498	-0.165577	-0.173416	-0.008374	-0.008852	-0.026423	-0.026679	-0.008903	0.031498	NaN	-0.015684	1.000000	-0.083471	-0.021331	-0.083471	-0.095064	-0.012337
spec_obj_ID	0.239461	-0.002553	0.112329	0.029997	0.039443	0.655245	0.661641	0.037813	0.239460	NaN	-0.001946	-0.083471	1.000000	0.388642	1.000000	0.970167	0.241279
redshift	0.065400	0.001667	0.031638	0.014309	0.022954	0.433241	0.492383	0.030380	0.065400	NaN	0.000097	-0.021331	0.388642	1.000000	0.388641	0.387109	0.127044
plate	0.239460	-0.002554	0.112329	0.029997	0.039443	0.655243	0.661640	0.037813	0.239459	NaN	-0.001949	-0.083471	1.000000	0.388641	1.000000	0.970166	0.241258
MJD	0.262687	0.019943	0.107333	0.031997	0.040274	0.671180	0.672523	0.037469	0.262687	NaN	-0.006745	-0.095064	0.970167	0.387109	0.970166	1.000000	0.256970
fiber_ID	0.067178	0.030464	0.028250	0.016305	0.017470	0.223106	0.214787	0.014668	0.067165	NaN	0.121597	-0.012337	0.241279	0.127044	0.241258	0.256970	1.000000

Using a heat map, we can understand the relationship between light emitted and the different classes of objects. Galaxies are the strongest emitters of ultraviolet light. Quasars show a stronger correlation with infrared light compared to galaxies and stars. Additionally, quasars consistently emit relatively similar frequencies of light across all measured spectrum's.



Additionally, we can observe that a high Redshift value correlates strongly with quasars and has a nearly perfect negative linear relationship with stars.



The columns **run_ID**, **rerun_ID**, **cam_col**, and **field_ID** pertain to how the observation was captured. If observations and measurements were taken in a consistent manner, these identifiers can help boost our confidence in the classification. Consistent results from rerunning observations in the same way lend credibility to our findings.

Algorithms and techniques

Models

For this project, I'll be using two different models and comparing their results. The first model will be **Support Vector Machines** (SVM). SVM's are effective for small datasets like ours because they are robust to over-fitting and can manage high-dimensional data efficiently. While SVMs can be computationally intensive, this is manageable given the current dataset size.

The second model I'll use is **K-Nearest Neighbors** (KNN). KNN is effective for classification tasks and does not require a separate training phase, as it uses the training data directly for making predictions. This can be beneficial for handling various data distributions and potentially improve performance, particularly when the data has complex patterns or relationships.

Class imbalance handling
We'll use SMOTE (Synthetic Minority Over-sampling Technique) to help rebalance the classification samples. The dataset consists of approximately 60,000 galaxy samples, 20,000 star samples, and 18,000 quasar samples.

While rebalancing, it's important to maintain the natural prevalence of galaxies to avoid creating a synthetic bias toward less frequent classifications. Therefore, we'll maintain a 2:3 ratio of minority classes to the majority class.

Accuracy & Benchmarks
To ensure the accuracy of these models, I'll use cross-validation and establish benchmarks for comparison. Cross-validation is valuable for multi-class classification problems as it provides a robust evaluation of model performance by splitting the dataset into multiple folds, training the model on each fold, and testing it on the remaining data. By averaging the results across all folds, we can assess the model's overall performance and its ability to generalize to unseen data.

Methodology

Data Pre-processing
The dataset contains 100,000 observations, with 21,947 identifying the same object. These objects may emit varying levels and spectra of light at different times and can show different characteristics at different life cycle phases, as indicated by the Modified Julian Date (MJD). To avoid introducing synthetic bias towards certain classes, we will retain objects with multiple classifications. This approach is acceptable for our context. Approximately 15,000 of the 100,000 samples have multiple classifications, which is a significant portion and may affect data balance or introduce inconsistencies.

All columns, except for our target column (class), are numerical, which will simplify processing and pipeline creation.

The current number of features is relatively small, and all should be considered when building our pipeline. Each feature provides significance in either identifying the classification or reinforcing our understanding of the class.

As stated earlier I'm using **SMOTE** to help balance the issue of bias towards the galaxy classification. To maintain a slight bias towards galaxies and avoid a synthetic bias I'm keeping a 2/3rds ratio of minority classes (quasars, and stars) to the majority class.

Implementation

For the initial model, I'm implementing Support Vector Classification (SVC) from Scikit-learn. I chose SVC for several reasons. First, it's straightforward to use, which suits my smaller dataset with a slight class imbalance. SVC's allowance for some misclassifications is beneficial in handling class imbalance. Moreover, SVC is effective for multi-class classification problems like this one, where the goal is to identify stars, quasars, and galaxies. Finally, the ability of SVC to handle misclassifications is advantageous in this dataset, as the same object may present with varying feature values.

In order to explore as many possible combinations for my models I'll be implementing GridSearchCV which allows me to pass a dictionary of params to evaluate my models with.

A few considerations I've made in regards to parameters passed on the SVC model…

Kernel: I'm using the "**Linear**" and "**Radial Basis Function**" (RBF) kernels. The linear kernel is straightforward, simple, and computationally efficient, which is beneficial for a smaller dataset where efficiency and avoiding over-complication are priorities. The RBF kernel, while more complex, can capture non-linear relationships in the data, potentially improving classification accuracy if such relationships exist.

Class Weight: I'm using the "balanced" setting. This automatically adjusts the weights of each class to account for class imbalances, ensuring that all classes are treated equally during training. This approach is appropriate here as it maintains fairness among classes, which are all equally important.

Gamma: I'm using the "auto" value which defaults to 1 / n_features.

Cache Size: Lastly, I've increased the default cache size from 200 mb to 500 to help improve the models performance during training.

For the initial model, I'm implementing K-Nearest Neighbors (KNN) from Scikit-learn. I chose KNN for several reasons. First, it's a simple, intuitive algorithm that works well with the smaller dataset at hand. KNN classifies new data points based on the majority class among the nearest neighbors, which makes it straightforward to interpret and implement. Additionally, KNN is effective for multi-class classification problems like distinguishing stars, quasars, and galaxies, especially when the data has a well-defined local structure.

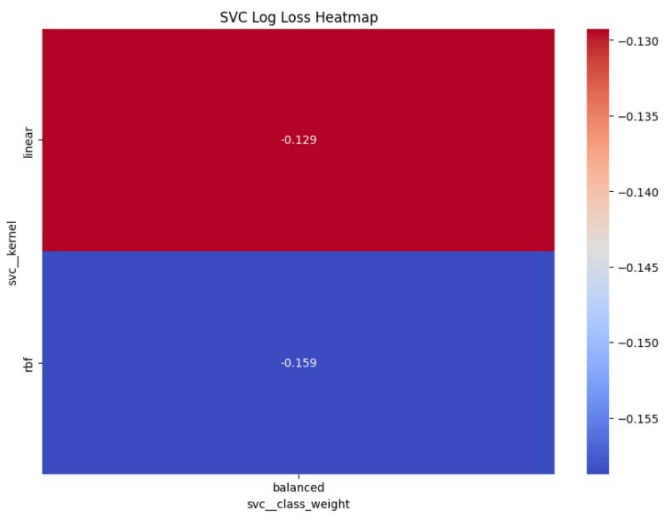
To optimize the performance of the KNN model, I'll be using GridSearchCV. This allows me to explore various combinations of parameters to find the best setup for the model. Here are a few considerations regarding the parameters I've set for the KNN model:

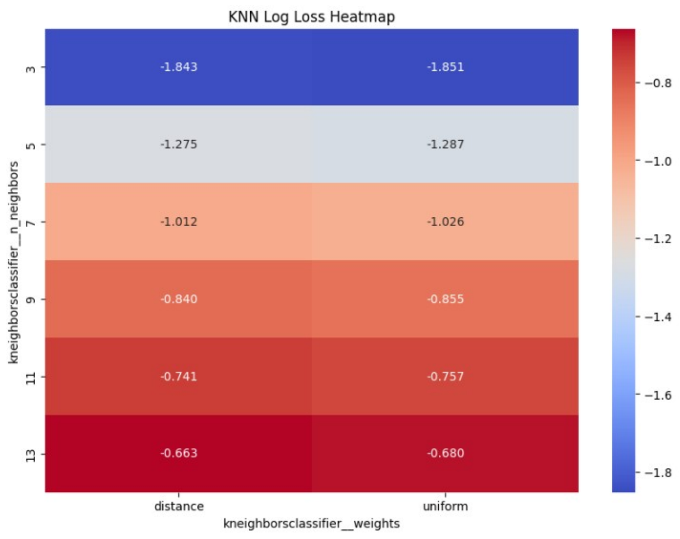
- **n_neighbors:** This parameter, which can be set to 3, 5, 7, or 9, defines the number of nearest neighbors used in classification. Testing different values helps in finding the optimal balance between sensitivity to local data and generalization.
- **weights:** By testing "uniform" and "distance" settings, I'm examining whether treating all neighbors equally or giving more weight to closer neighbors improves the model's performance.
- **algorithm:** I'm trying "auto," "ball_tree," "kd_tree," and "brute" algorithms to determine the most efficient method for finding neighbors, depending on the dataset's dimensionality and size.

Refinement

The cross-entropy loss results for the SVC and K-NN models reveal a significant performance gap, with SVC achieving a much lower loss of 0.129 compared to K-NN's 0.626. This suggests that SVC is likely to have a higher accuracy, approximately 87.1%, while K-NN's accuracy is estimated to be around 37.4%. Despite both models achieving high scores in training, with SVC's results aligning with the reported high accuracy, K-NN's performance is notably lower than the expected 85% accuracy. This disparity emphasizes the need to validate these models with a larger dataset to determine if overfitting is occurring and if the current dataset is leading to such high discrepancies in performance.

The heatmaps below depict the log loss generated by the chosen hyperparameters.





Several strategies could be explored to mitigate overfitting and enhance performance:

1. **Introduce New Kernels:** Experimenting with different kernels could help better distribute the data across larger hyperplanes, potentially improving model generalization.

2. **Use gamma='scale':** Instead of using gamma='auto', switching to gamma='scale' could reduce the risk of overfitting. Scale adjusts based on the variance of the features, offering better adaptability to unseen data.
3. **Increase Cache Size:** Increasing the cache size could improve training performance, especially for larger datasets or more complex models.

Results

Model Evaluation and Validation

SVC outperformed K-NN significantly in this assessment, which might be attributed to the class imbalances present in the data. K-NN tends to struggle with class imbalances more than SVC does. To address this, rebalancing the data could potentially improve performance, but SVC's effectiveness in this scenario is noteworthy. SVC excels at finding a hyperplane that maximizes the margin between classes, which contributes to its superior performance. Additionally, SVC generally requires fewer parameters to tune, simplifying the process of achieving high performance.

Conclusion

Reflection & Improvement

At the outset of this capstone project, I anticipated that KNN might perform as well as, if not better than, the SVC model. However, after comparing the two, I've concluded that my dataset's size and structure do not favor KNN. With a smaller number of observations and features, the complexity of the KNN algorithm leads to poorer performance compared to the simpler SVC algorithm. The hyperparameters that optimized SVC performance—such as linear, balanced, and scale—further support this conclusion.

To make a more definitive comparison between these models, additional real-world data is necessary. Synthetic data augmentation might not yield accurate results, particularly when introducing new outliers for classification.