AAI 595 Final Project Report- Machine Learning Approaches for Celestial Object Classification

Chris Muro
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030
cmuro@stevens.edu

Rocco Gannon
Department of Mechanical Engineering
Stevens Institute of Technology
Hoboken, NJ 07030
rgannon@stevens.edu

Marc DiGeronimo
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030
mdigeron@stevens.edu

Abstract

This paper addresses the approaches of classifying celestial objects into one of three categories: stars, galaxies, or quasars using machine learning algorithms. The project evaluates various models. The goal is to determine the most effective method for accurate classification of these celestial objects to aid in astronomy research.

I. Introduction

The machine learning problem is to classify each input into one of the 3 categories presented in the data set. As described by the dataset, the 3 classes are stars, galaxies and quasars. More robust classification methods can help astronomers better identify these celestial objects in order to have a better understanding of the cosmos. As an example, by identifying a specific distribution of stars, scientists were able to notice that the cluster of stars was actually a galaxy. Moreover, based on that distribution that galaxy which was named Andromeda was concluded to be a separate one from the Milky Way which would not have happened without accurate classification mechanisms. Today, many advanced telescopes are being constructed and deployed such as the James Webb Space Telescope and the Vera C. Rubin observatory which are being used to plot different celestial objects in the sky map of the universe. As mentioned, stars, galaxies and quasars are the most fundamental classifications and can teach how each celestial system is changing and can be used as a process for making informed predictions about the history of the universe and other discoveries. The James Webb Space Telescope for example can collect information based on near-infrared astronomy which utilizes metrics in the dataset. The Vera Rubin Observatory is primarily used for mapping changes in celestial positions over a relatively short period of time. While these are primary objectives there is much more information that each can collect. Moreover the source of the dataset, the Sloan Digital Sky Survey[1] has over 40 partners that are used to collect data to make these classifications. Overall, there are many advancements being made in astronomy to collect data and it is a machine learning problem to be able to make use of that data to make accurate classifications which can be used to advance astronomical understanding.

II. Related Work

Some existing techniques for similar problems use different metrics to make predictions on classification. According to the Australia Telescope National Facility[2], stars, galaxies and quasars can be identified by studying different spectrum graphs. The spectrum graphs mentioned are continuum, absorption and emission. Over time scientists have discovered trends that relate to each celestial classification. For stellar objects, absorption lines occur at specific wavelengths that are related to special Balmer series characteristics, with the shape approximating a black body curve which relates to radiation. An example star spectrum is as follows

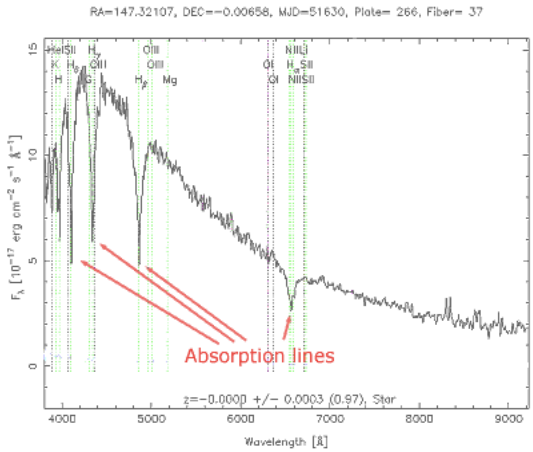


Figure 1: Star Spectrum Showing Absorption Lines

Galaxy spectrum plots on the other hand contain more variance between kinds since it is made up of multiple star spectrums. It resembles an extended object, meaning there are multiple point sources in the spectrum. Example galaxy spectrums are as follows.

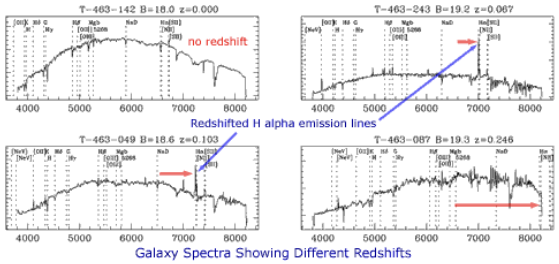


Figure 2: Galaxy Spectra Showing Different Redshifts

Quasar spectrums are also different. The emission lines compared to stars are in different locations and are very strong in specific locations not found in stars. Moreover quasars contain very bright emission features with a relatively low intensity. An example quasar spectrum plot is as follows.

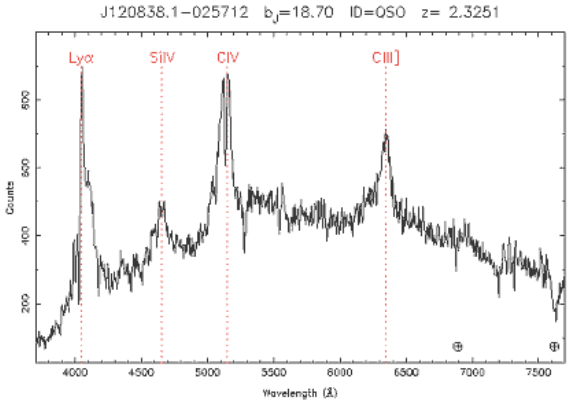


Figure 3: Quasar Spectrum Showing Emission Lines

The data for these plots are collected from different telescopes such as the Anglo-Australian Telescope among others. Another method is to study images of a cosmic field and identify characteristics of each celestial type. For example, the Hubble Extreme Deep Field provides a very high resolution image of space.

Figure 4: Hubble Extreme Deep Field Showing Galaxies and Stars

Through this Scientists have found differentiation characteristics of stars and galaxies. Clearly when the body is large it can be quite simple to identify a galaxy. However, when smaller it can be more difficult to differentiate. In this image there are over 5,500 galaxies and many 1000s of star. The key is to find diffraction spikes. Diffraction spikes are present in stars but not galaxies no matter how big or small in an image. An example is as follows

Figure 5: Diffraction Spikes in a Star

Using characteristics like this programs such as SExtractor[3] can ingest high resolution cosmic images and identify stars and galaxies.

Another method is using machine learning to estimate the point source probabilities of celestial objects based on Pan-STARRS1[4] databases. It is based on data collected from The Panoramic Survey Telescope and Rapid Response System in Hawaii. It is designed to take differences in measurement from previous observations of the same areas of the sky. The model operates on over 1.5 Billion collected celestial objects and uses clustering to separate stars from galaxies by using parameters such as PSFMag and KronMag, psfLikelihood, moments and lensing parameters from the collected measurements.

III. Description of Dataset

The dataset contains 17 feature columns as well as 1 class column the dataset info is given from the .info() method of the pandas library

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 18 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   obj_ID       100000 non-null  float64
 1   alpha        100000 non-null  float64
 2   delta        100000 non-null  float64
 3   u            100000 non-null  float64
 4   g            100000 non-null  float64
 5   r            100000 non-null  float64
 6   i            100000 non-null  float64
 7   z            100000 non-null  float64
 8   run_ID       100000 non-null  int64  
 9   rerun_ID     100000 non-null  int64  
 10  cam_col      100000 non-null  int64  
 11  field_ID     100000 non-null  int64  
 12  spec_obj_ID  100000 non-null  float64
 13  class        100000 non-null  object 
 14  redshift     100000 non-null  float64
 15  plate        100000 non-null  int64  
 16  MJD          100000 non-null  int64  
 17  fiber_ID     100000 non-null  int64  
dtypes: float64(10), int64(7), object(1)
memory usage: 13.7+ MB
```

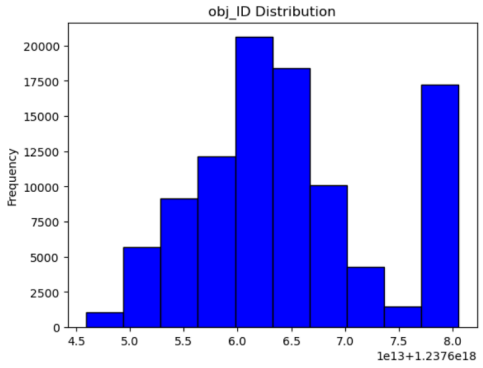
Figure 6: Dataset Information Summary

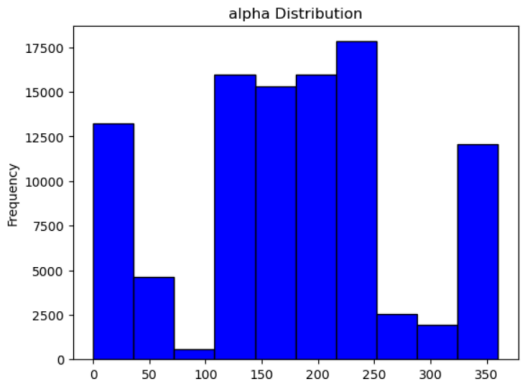
Here obj_ID is the Object Identifier which is the unique value that identifies the object in the image catalog used by the CAS. Alpha is the right ascension angle. Delta is the declination angle. U is the ultraviolet filter in the photometric system. G is the green filter in the photometric system. R is the red filter in the photometric system. I is the near filter in the photometric system. Z is the infrared filter in the photometric system. Run_ID is the run number used to identify the specific scan. Rerun_ID is the rerun number to specify how the image was processed. Cam_col is the camera column to identify the scanline within the run. Field_ID is the field number to identify each field. Spec_obj_ID is the unique ID used for optical spectroscopic objects. Class is the object class which is a galaxy, star or quasar object. Redshift is the redshift value based on the increase in wavelength. Plate is the plate ID. MJD is the modified Julian date when the data was collected. Fiber_ID is the fiber ID that identifies the fiber that points the light at the focal plane in each observation. Of the features, all are numerical with the exception of the class object. 10 are of float64 data type with the remaining 7 being int64 data type. The dataset contains 100000 entries with none of the data missing as each column contains 100000 non null values as well. Here are the first 5 entries in the dataset.

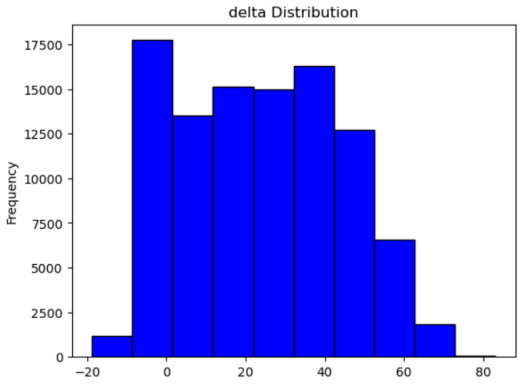
obj_ID	alpha	delta	u	g	r	i	z	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID	class	redshift	plate	MJD	fiber_ID
1.24E+18	135.6891	32.49463	23.87882	22.2753	20.39501	19.16573	18.79371	3606	301	2	79	6.54E+18	GALAXY	0.634794	5812	56354	171
1.24E+18	144.8261	31.27418	24.77759	22.83188	22.58444	21.16812	21.61427	4518	301	5	119	1.18E+19	GALAXY	0.779136	10445	58158	427
1.24E+18	142.1888	35.58244	25.26307	22.66389	20.60976	19.34857	18.94827	3606	301	2	120	5.15E+18	GALAXY	0.644195	4576	55592	299
1.24E+18	338.741	-0.40283	22.13682	23.77656	21.61162	20.50454	19.2501	4192	301	3	214	1.03E+19	GALAXY	0.932346	9149	58039	775
1.24E+18	345.2826	21.18387	19.43718	17.58028	16.49747	15.97711	15.54461	8102	301	3	137	6.89E+18	GALAXY	0.116123	6121	56187	842

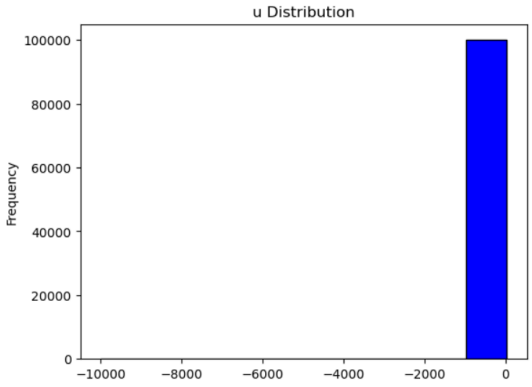
Figure 7: Sample Dataset Entries

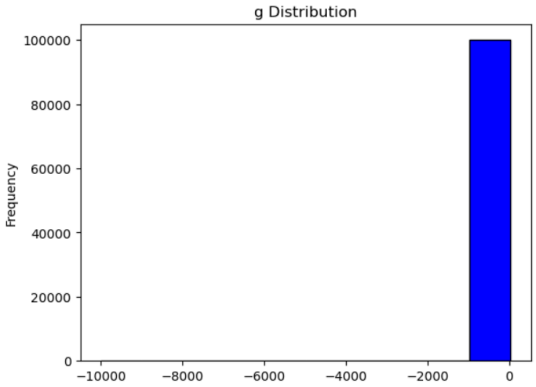
Below are graphs which showcase the distribution of each feature.

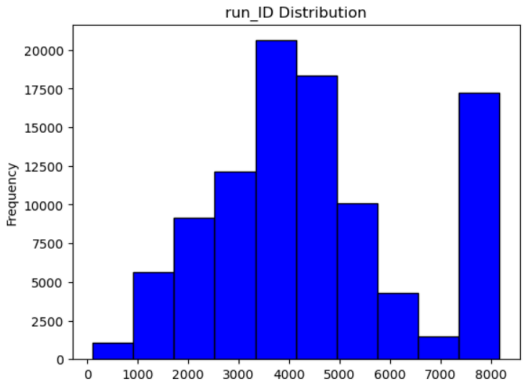


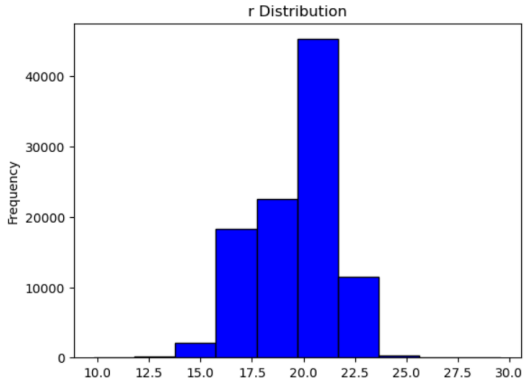


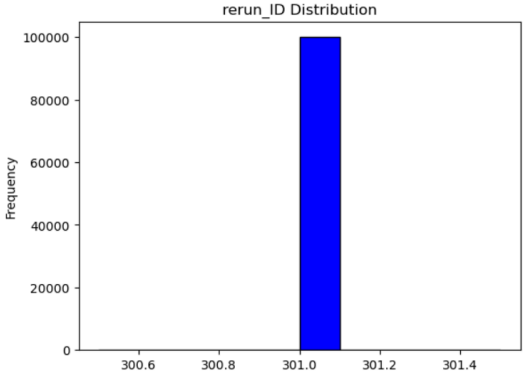


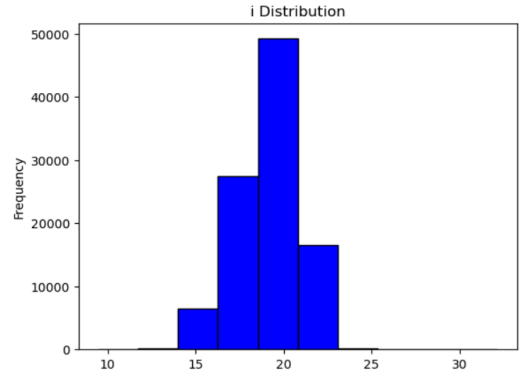


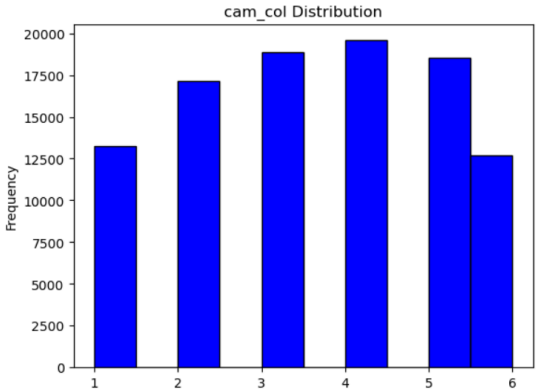

g Distribution
Frequency
100000
80000
60000
40000
20000
0
−10000
−8000
−6000
−4000
−2000
0

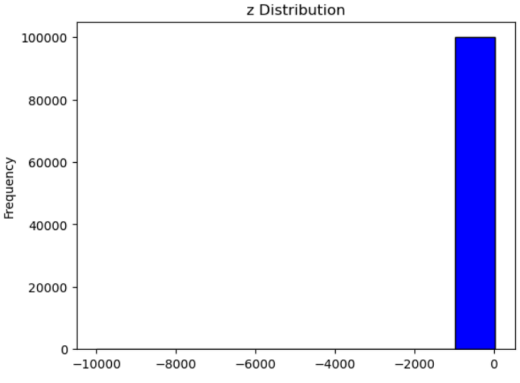

run_ID Distribution
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1000
2000
3000
4000
5000
6000
7000
8000

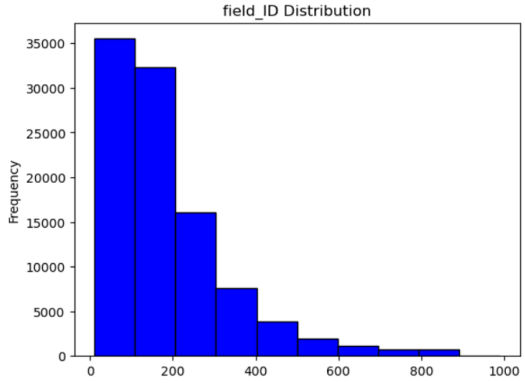

r Distribution
Frequency
40000
30000
20000
10000
0
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0

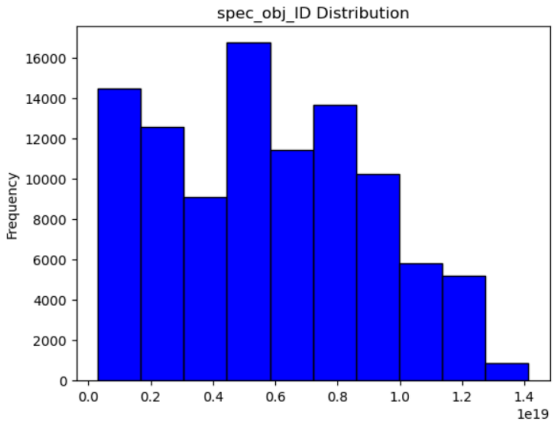

rerun_ID Distribution
Frequency
100000
80000
60000
40000
20000
0
300.6
300.8
301.0
301.2
301.4

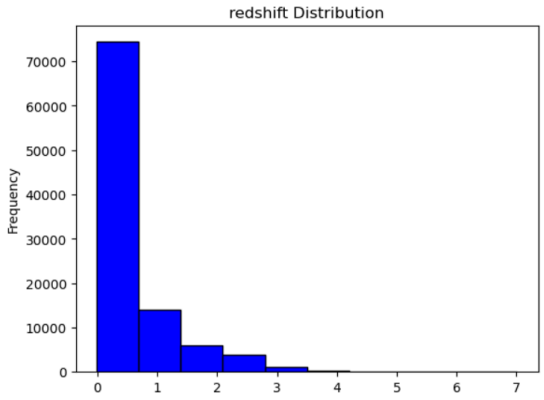

i Distribution
Frequency
50000
40000
30000
20000
10000
0
10
15
20
25
30

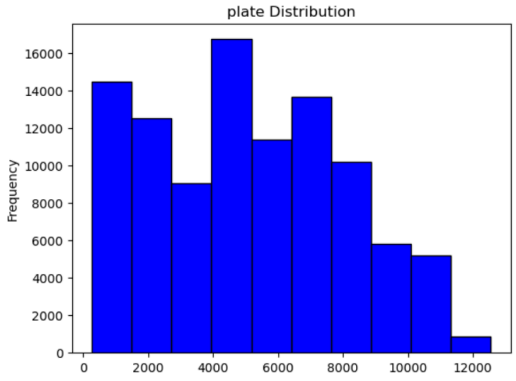

cam_col Distribution
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
1
2
3
4
5
6

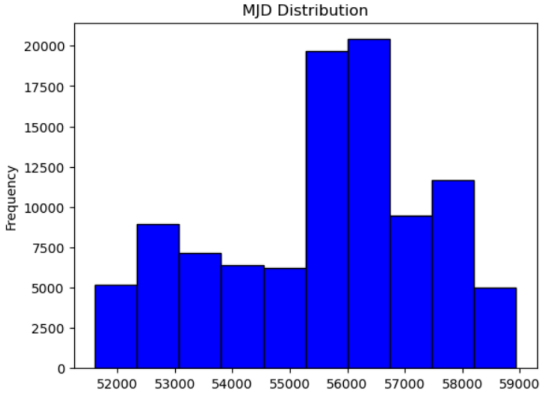

z Distribution
Frequency
100000
80000
60000
40000
20000
0
−10000
−8000
−6000
−4000
−2000
0

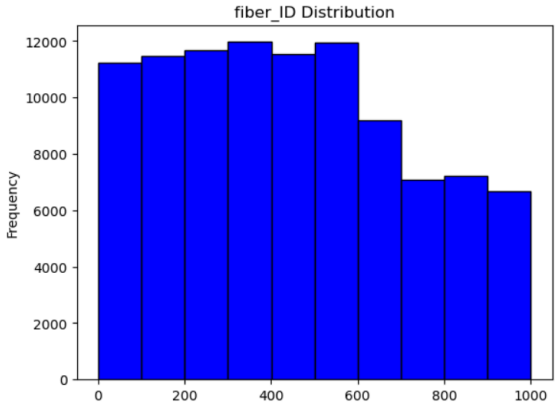

field_ID Distribution
Frequency
35000
30000
25000
20000
15000
10000
5000
0
0
200
400
600
800
1000











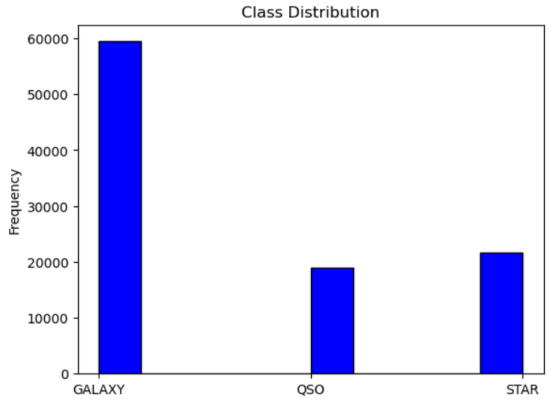


Figure 8: Feature Distributions

Some other measurements of each feature are as follows.

	obj_ID	alpha	delta	u
count	1.000000e+05	100000.000000	100000.000000	100000.000000
mean	1.237665e+18	177.629117	24.135305	21.980468
std	8.438560e+12	96.502241	19.644665	31.769291
min	1.237646e+18	0.005528	-18.785328	-9999.000000
25%	1.237659e+18	127.518222	5.146771	20.352353
50%	1.237663e+18	180.900700	23.645922	22.179135
75%	1.237668e+18	233.895005	39.901550	23.687440
max	1.237681e+18	359.999810	83.000519	32.781390

	g	r	i	z
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	20.531387	19.645762	19.084854	18.668810
std	31.750292	1.854760	1.757895	31.728152
min	-9999.000000	9.822070	9.469903	-9999.000000
25%	18.965230	18.135828	17.732285	17.460677
50%	21.099835	20.125290	19.405145	19.004595
75%	22.123767	21.044785	20.396495	19.921120
max	31.602240	29.571860	32.141470	29.383740

	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID
count	100000.000000	100000.0	100000.000000	100000.000000	1.000000e+05
mean	4481.366060	301.0	3.511610	186.130520	5.783882e+18
std	1964.764593	0.0	1.586912	149.011073	3.324016e+18
min	109.000000	301.0	1.000000	11.000000	2.995191e+17
25%	3187.000000	301.0	2.000000	82.000000	2.844138e+18
50%	4188.000000	301.0	4.000000	146.000000	5.614883e+18
75%	5326.000000	301.0	5.000000	241.000000	8.332144e+18
max	8162.000000	301.0	6.000000	989.000000	1.412694e+19

	redshift	plate	MJD	fiber_ID
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	0.576661	5137.009660	55588.647500	449.312740
std	0.730707	2952.303351	1808.484233	272.498404
min	-0.009971	266.000000	51608.000000	1.000000
25%	0.054517	2526.000000	54234.000000	221.000000
50%	0.424173	4987.000000	55868.500000	433.000000
75%	0.704154	7400.250000	56777.000000	645.000000
max	7.011245	12547.000000	58932.000000	1000.000000

Figure 9: Statistical Measurements of Each Feature

Moreover a correlation heatmap is shown. It does not contain rerun_ID since the correlation values were NaN.

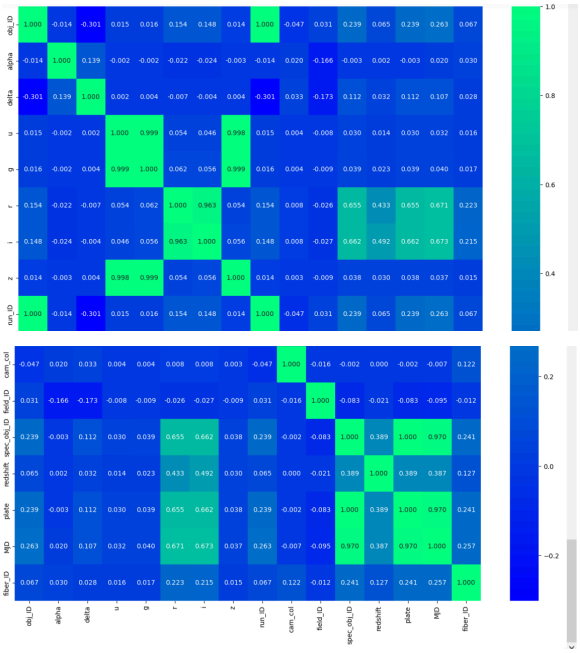


Figure 10: Correlation Heatmap

After observing the data, it needed to be further processed before splitting into train and test sets. First the class values were converted into a numerical system of categorisation with 0 being Galaxy, 1 Star and 2 Quasar. A new dataframe x is initialized as the data frame without the class column and data frame y is the class column. The data is scaled using Scikit-Learn Standardscaler to ensure that the features have equal weight in training. The data is split into a training and testing set using train_test_split() method. Train set is 95% of the data with the test set being the remaining 5%. Typical split would be 80/20 however due to the dataset being rather large at 100000 entries, 5000 entries should be sufficient to gauge how the models perform, 20000 seems unnecessarily large. This allows for more entries to be used for training the model which ideally would result in a higher accuracy going from 80000 entries to 95000 entries.

IV. Machine Learning Algorithms

The first machine learning method utilized was Random Forest Classifier. The basis for such a model is based on Decision Tree Classifiers. Decision trees are well suited for classification problems such as the one presented by the Stellar classification dataset. Since this dataset has 17 feature columns a decision tree seemed to be a viable solution since they are designed to perform inherent feature selection, picking features that are more significant as branch points. Moreover the dataset is rather large so it should be more clear which features are important since noise and outliers are less prone to having an effect on a large dataset. To further improve a decision tree model, a random forest classifier was used since it is an ensemble method based on decision trees. In this way, more trees are used in the prediction which may do a better job at identifying key features resulting in higher accuracy and less prone to overfitting. When implementing the random forest classifier a grid search was used to tune hyperparameters. One of the parameters tuned was n_estimators which specifies the amount of trees used in the ensemble. Therefore it makes it more time consuming to test different parameters. Therefore 3 parameters were tuned, n_estimators, criterion and max_features.

```
parameter_grid = {
    "n_estimators": [10, 50, 100],
    "criterion": ["gini", "entropy"],
    #"max_depth": [None, 10, 25, 50],
    #"min_samples_split": [2, 5, 10],
    #"min_samples_leaf": [1, 2, 4],
    "max_features": ["sqrt", "log2", None]
}
```

Figure 11: Random Forest Parameter Grid

Resulting in 90 models tested. The results of the accuracy based on each parameter is as follows.

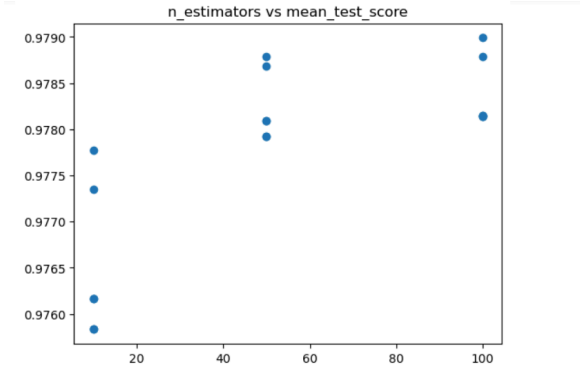


Figure 12: N_estimators vs Mean Test Score

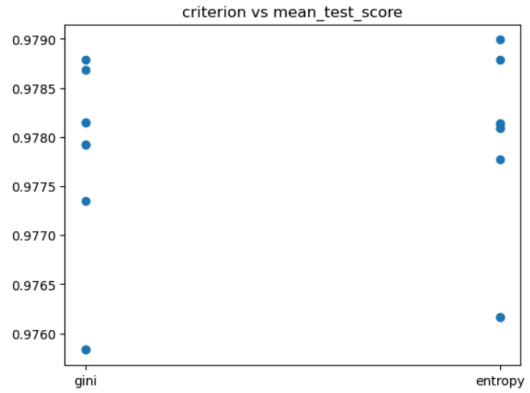


Figure 13: Criterion vs Mean Test Score

From the grid search the best parameters are

```
Best parameters found:  {'criterion': 'entropy', 'max_features': None, 'n_estimators': 100}
```

Figure 14: Random Forest Best Parameters

Further optimization was attempted based on the feature importance values. The resulting graph shows the importance values of each feature.

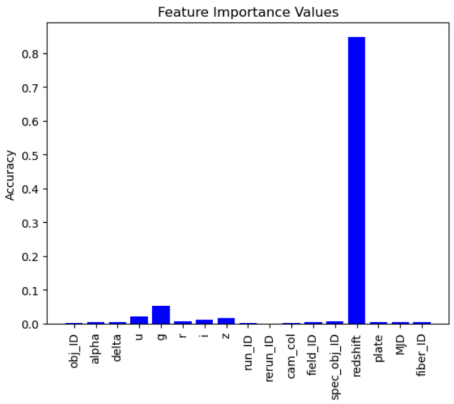


Figure 15: Feature Importance Values

Based on this it appears that there are only a few features out of the 17 total that contribute to the accuracy of the model. Therefore dimensional reduction was performed using Principal Component Analysis (PCA). To test this theory and remove possibly unnecessary features, PCA was used to reduce features from 17 to 5. The overall results of the entire feature space and PCA are discussed in the following section.

Another machine learning model that was implemented was Gaussian Mixture Model (GMM). It is a probabilistic model that assumes the objects in the dataset were generated from a mixture of several gaussian distributions. Unlike random forest classifiers it is an unsupervised learning technique so it is interesting to see how it will handle this dataset. Another reason that it was chosen is that the Stellar Classification dataset is very large and there are large amounts of data with regards to each class. By the central limit theorem if the number of samples is greater than 30 it can be assumed samples follow a gaussian distribution. Therefore it may be plausible to conclude that the classes of this dataset

follow a combination of gaussian distributions. As with random forest, Scikit-Learn's implementation is used. Moreover GridSearchCV was used again to tune the hyperparameters. The following parameters were tested.

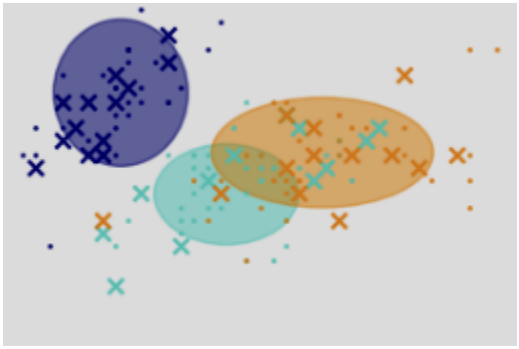
```
parameter_grid = {
    "n_components": [1,2,3],
    "covariance_type": ["tied", "diag", "spherical", "full"],
    #"init_params": ["k-means++"]
}
```

Figure 16: GMM Parameter Grid

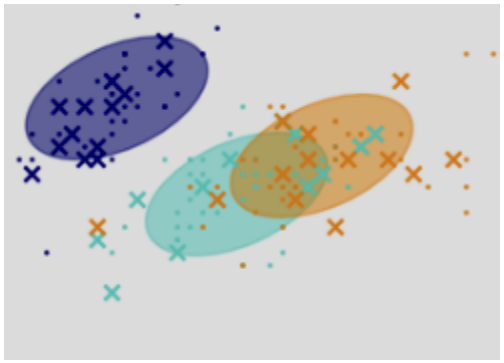
There are less parameters tuned than random forest classifiers. N_components is the number of gaussian distributions that make up the model while the covariance type influences the shape of each cluster. Here is an example of the shapes from the official documentation

Spherical

Diag



Tied



Full

Figure 17: Cluster Shapes

The grid search resulted in 60 models being fit. The results of each feature are as follows

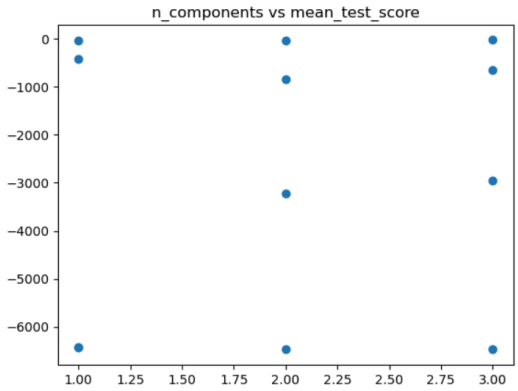


Figure 18: N_components vs Mean Test Score

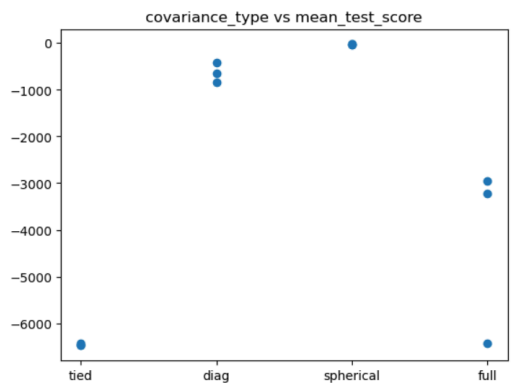


Figure 19: Covariance Type vs Mean Test Score

The best parameters are as follows

```
Best parameters found:  {'covariance_type': 'spherical', 'n_components': 3}
```

Figure 20: GMM Best Parameters

For further optimization PCA was used to reduce the dimensionality of the dataset to see how the optimized model would work with a reduced feature space. For this example the feature space was reduced to the 2 most important features. Moreover it is known that GMM can be sensitive to imbalanced data. As shown earlier there are significantly more instances of galaxy classification in the dataset when compared to stars and quasars. Therefore RandomUnderSampler() was used to reduce the amount of galaxy classifications in the dataset. Undersampling was used rather than oversampling because the dataset is large so it should not have as significant of an impact when compared to smaller datasets.

```
undersampler = RandomUnderSampler(random_state=1)
x_balanced, y_balanced = undersampler.fit_resample(x2, y2)
```

The resulting class amounts are as follows.

```
0    18961
1    18961
2    18961
Name: class, dtype: int64
```

Figure 21: Balanced Class Amounts

Since the dataset is now smaller when splitting the new balanced data into train and test sets, the ratio was changed to be 90% train and 10% test in an effort to better capture the variances of the data.

```
train_test_split(x_balanced, y_balanced, test_size = 0.1, random_state = 1)
```

The balanced data was used to fit a model that was constructed using the same best parameters found in the previous grid hyperparameter search. The overall results of all of these different models are in the following section.

The last machine learning model used is XGBoost Classifier. It was developed by Tianqi Chen and Carlos Guestrin in their paper “XGBoost: A Scalable Tree Boosting System”[5] in 2016. It is similar to random forest models in that it is an ensemble method based on decision trees. Rather than ensembling the results of many independent decision trees, XGBoost builds trees sequentially, with each minimizing the errors of the previous one. It should be a good fit for this well structured, complex dataset for the same reasons as Random Forest. Moreover, it is plausible to assume that XGBoost could even perform better than random forest. This is because of the sequential nature. Random forest ensembling generates many different trees and combines the results to achieve the predictions, however the nature of this means that the trees themselves are all inherently optimized. In fact, the importance is having many different trees to help mitigate the poor performing trees. XGBoost keeps building on the previous tree iteration, ideally improving with each additional tree built. Moreover, XGBoost is also one of the most popular choices for classification

problems where accuracy is the top priority. Grid search was used to optimize the different hyperparameters with the following 3 parameters being tuned.

```
parameter_grid = {
    "booster": ["gbtree", "gblinear", "dart"],
    "max_depth": [0, 2, 4],
    "tree_method": ["auto", "exact", "approx"]
}
```

Figure 22: XGBoost Parameter Grid

The boosted parameter determines what boosting algorithm is used. GBTree is the standard boosting algorithm. GBLinear utilizes a linear classifier algorithm for the trees. Dart is a boosting algorithm that utilizes drop out to limit overfitting. It was proposed in a paper titled "DART: Dropouts meet Multiple Additive Regression Trees."[6] by Korlakai Vinayak Rashmi, Ran Gilad-Bachrach. Max_depth is the maximum depth of the tree with 0 meaning no limit in this context. Tree_method is the algorithm for updating the trees. Auto is a histogram optimized greedy algorithm. Exact is an exact greedy algorithm. Approx uses an approximate greedy algorithm using quantile sketch and gradient histogram from the documentation. It resulted in 135 models being fit. The following are the results for each parameter.

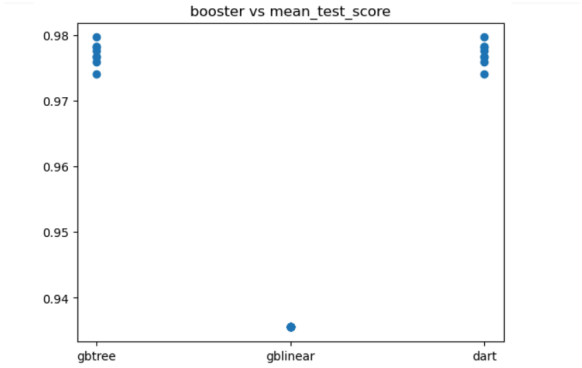


Figure 23: Booster vs mean_test_score

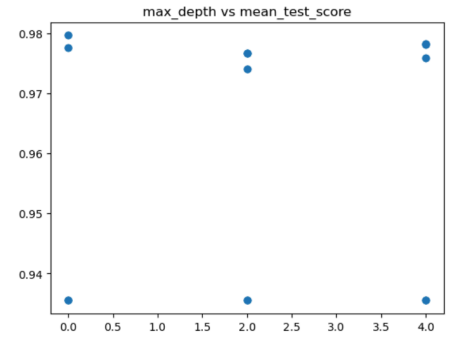


Figure 24: Max_depth vs mean_test_score

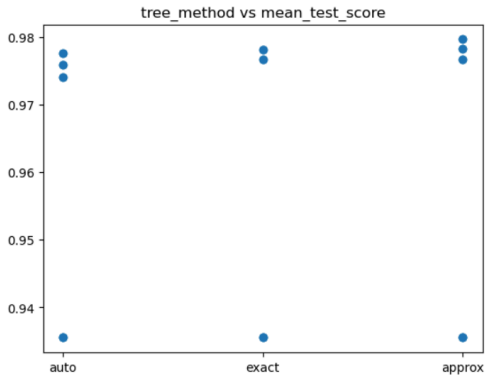


Figure 25: Tree_method vs mean_test_score

From the graph the worst performing parameter is gblinear. It does not perform well with multi classification tasks. The best model parameters are

```
Best parameters found:  {'booster': 'gbtree', 'max_depth': 0, 'tree_method': 'approx'}
```

Figure 26: XGBoost classifier best parameters

Further optimizations were also used in an attempt to improve scores. As with the other models PCA was used to reduce the feature set to see how the model performed. Moreover, the best parameters were also tested on a balanced dataset. The overall results are described in the following section.

V. Results and Comparisons

The accuracy score of the best parameters tested for Random Forest Classifier was .9822 which was significantly higher than expected and reached the goal of an excess of 90% or greater accuracy score. A breakdown of the results are as follows.

	precision	recall	f1-score	support
0	0.98	0.99	0.99	2963
1	1.00	1.00	1.00	1105
2	0.96	0.94	0.95	932
accuracy			0.98	5000
macro avg	0.98	0.98	0.98	5000
weighted avg	0.98	0.98	0.98	5000

Figure 27: Classification Report for Random Forest Classifier

This model performed at its best when dealing with stars but still performed well with galaxies and quasars. The confusion matrix is as follows.

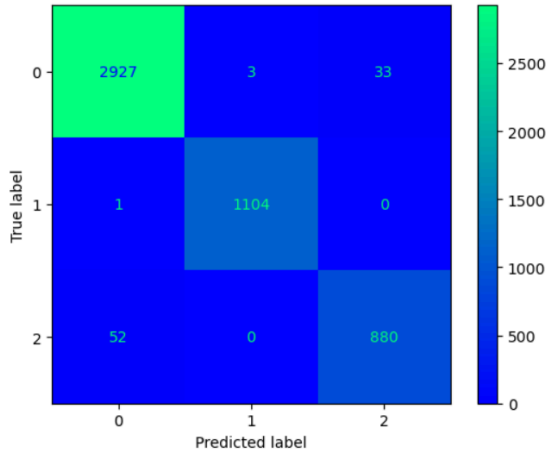


Figure 28: Confusion Matrix for Random Forest Classifier

Even though there are more galaxy data points than stars and quasars the model still handles the information well, this may be able to be explained by random forest classifiers handling overfitting well since it is an ensemble method. An optimized decision tree would likely perform worse in this scenario due to that. Next is the assumption that by performing PCA the accuracy would increase due to removal of potentially useless or misleading features. The results are as follows.

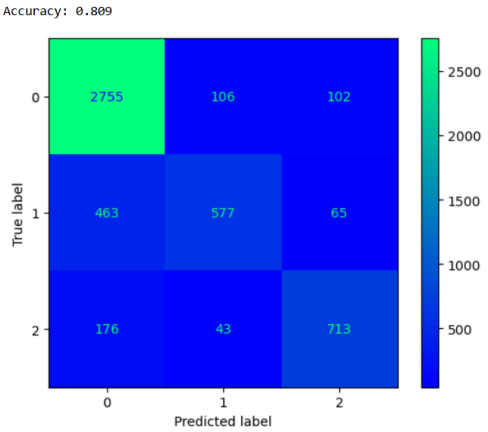


Figure 29: Confusion Matrix for Random Forest Classifier with PCA

	precision	recall	f1-score	support
0	0.81	0.93	0.87	2963
1	0.79	0.52	0.63	1105
2	0.81	0.77	0.79	932
accuracy			0.81	5000
macro avg	0.81	0.74	0.76	5000
weighted avg	0.81	0.81	0.80	5000

Figure 30: Classification Report for Random Forest Classifier with PCA

Surprisingly the model performed significantly worse with a score of .809. This does not meet the goal of greater than 90% accuracy. The model's strongest classification came from galaxies instead of stars at this point with stars performing significantly worse. This could mean that even though the features may have less importance they are still used effectively with larger amounts of decision trees in the random forest ensemble. The features still have importance or at least there are still

features with importance that have been left out. It is also plausible that the top feature importances are related to galaxy classification since those are the most data points. Since by reducing the feature space star classification goes from the strongest to weakest with the same hyperparameters. Therefore those excluded features likely are used to classify stars from galaxies and quasars. Overall Random Forest Classifiers are effective for this dataset when utilizing the entire feature space. The accuracy score of the best parameters for Gaussian Mixture was .2998 which is significantly less than the goal of 90%. Clearly it is a choice of model. The results are as follows.

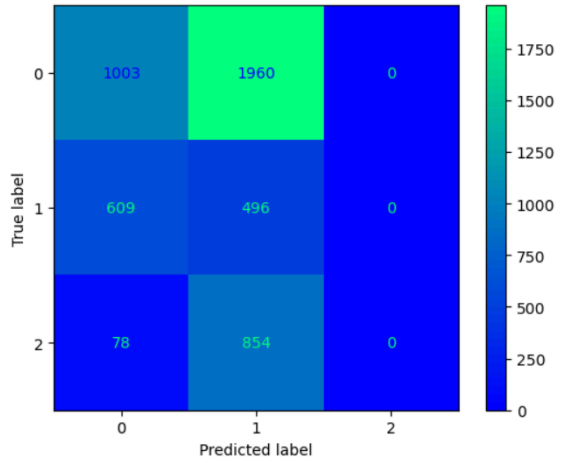


Figure 31: Confusion Matrix for GMM

	precision	recall	f1-score	support
0	0.59	0.34	0.43	2963
1	0.15	0.45	0.22	1105
2	0.00	0.00	0.00	932
accuracy			0.30	5000
macro avg	0.25	0.26	0.22	5000
weighted avg	0.38	0.30	0.31	5000

Figure 32: Classification Report for GMM

The model had an especially difficult time deciphering a star from a galaxy. Moreover the model did not even make a single prediction for a quasar. This could mean that the data is too intertwined with each other so it cannot be easily clustered with this technique. GMM does struggle with high dimensional data therefore the PCA results should in theory offer a higher accuracy score. The PCA results are as follows.

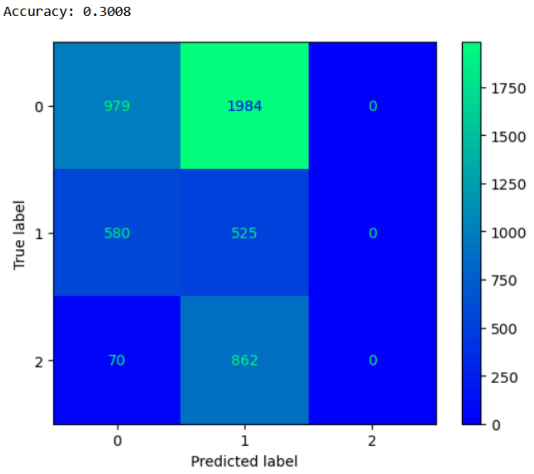


Figure 33: Confusion Matrix for GMM with PCA

	precision	recall	f1-score	support
0	0.60	0.33	0.43	2963
1	0.16	0.48	0.23	1105
2	0.00	0.00	0.00	932
accuracy			0.30	5000
macro avg	0.25	0.27	0.22	5000
weighted avg	0.39	0.30	0.30	5000

```
0    18961
1    18961
2    18961
Name: class, dtype: int64
```

Figure 34: Classification Report for GMM with PCA

The accuracy is slightly improved but in an actual environment the change is very insignificant and does not fully confirm that in a deployment scenario it will perform better. Therefore reducing the feature space did not help. SImilar predictions are still made, difficulties with stars vs galaxies and quasar predictions. In another attempted optimization the dataset was balanced since GMM can be sensitive to imbalance data. The results are as follows.

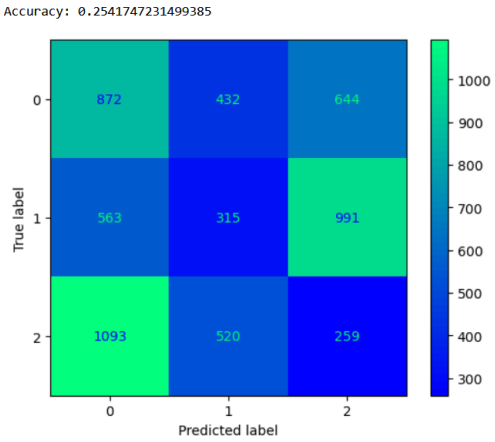


Figure 35: Confusion Matrix for GMM with Balanced Data

	precision	recall	f1-score	support
0	0.34	0.45	0.39	1948
1	0.25	0.17	0.20	1869
2	0.14	0.14	0.14	1872
accuracy			0.25	5689
macro avg	0.24	0.25	0.24	5689
weighted avg	0.24	0.25	0.24	5689

Figure 36: Classification Report for GMM with Balanced Data

The predictions follow a different distribution with stars being the least predicted and quasars being predicted highly as well. Problem areas with predicting quasars as galaxies and predicting stars as quasars. Though the accuracy score is actually worse than the previous attempts. Through the Gaussian Mixture and optimization attempts it appears that either the classes are not distinct enough for the model to identify or the classes do not follow a gaussian distribution.

The accuracy score of the best parameters tested for XGBClassifier was 0.983 which is better than the best random forest model. The confusion matrix and classification report are as follows.

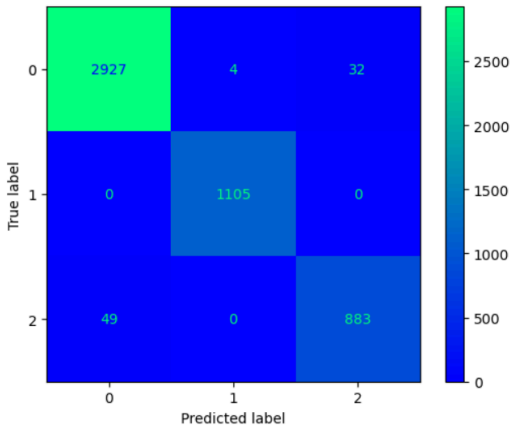


Figure 37: Confusion Matrix for XGBoost Classifier

	precision	recall	f1-score	support
0	0.98	0.99	0.99	2963
1	1.00	1.00	1.00	1105
2	0.97	0.95	0.96	932
accuracy			0.98	5000
macro avg	0.98	0.98	0.98	5000
weighted avg	0.98	0.98	0.98	5000

Figure 38: Classification Report for XGBoost Classifier

Similar to Random Forest the model performs the best with predicting stars. It appears that the way the data is structured and the features used are very good for predicting stars. The next model was by reducing the feature space to from 17 to 5 using PCA. The results are as follows.

```
Accuracy: 0.811
```

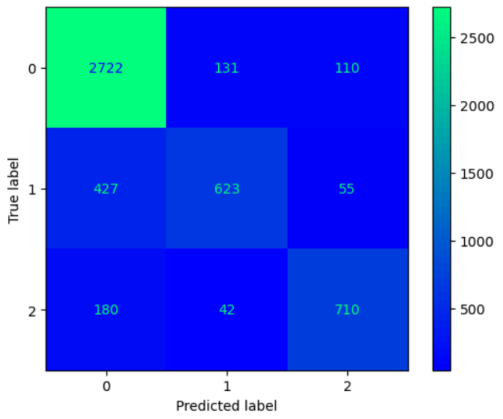


Figure 39: Confusion Matrix for XGBoost Classifier with PCA

	precision	recall	f1-score	support
0	0.82	0.92	0.87	2963
1	0.78	0.56	0.66	1105
2	0.81	0.76	0.79	932
accuracy			0.81	5000
macro avg	0.80	0.75	0.77	5000
weighted avg	0.81	0.81	0.80	5000

Figure 40: Classification Report for XGBoost Classifier with PCA

Interestingly, the accuracy dropped to .811, better than PCA with random forest. Overall though PCA has always lowered the accuracy. The reason could be that most of the features are significant to some extent. If there were significantly more features such as 50+ with many poor features PCA may be of use. However, for this dataset PCA has not shown significant usefulness. The last optimization attempt was to use the balanced dataset. The balanced dataset was constructed first for GMM. The results are as follows.

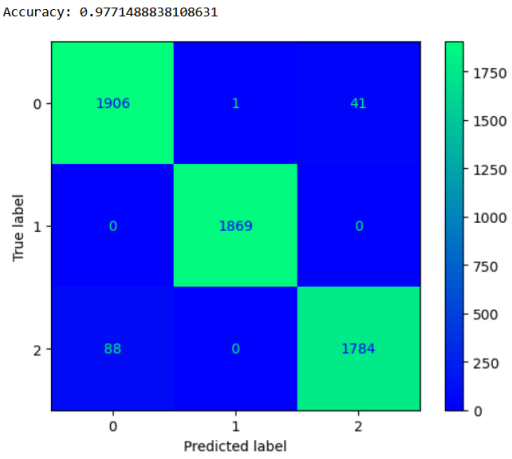


Figure 41: Confusion Matrix for XGBoost Classifier with Balanced Data

	precision	recall	f1-score	support
0	0.96	0.98	0.97	1948
1	1.00	1.00	1.00	1869
2	0.98	0.95	0.97	1872
accuracy			0.98	5689
macro avg	0.98	0.98	0.98	5689
weighted avg	0.98	0.98	0.98	5689

Figure 42: Classification Report for XGBoost Classifier with Balanced Data

The balanced dataset slightly underperforms the imbalance dataset. Galaxy data is very useful as it does help the accuracy scores. Typically imbalanced datasets are not ideal for decision trees however in these models it helps. Overall XGBClassifier is the best model tested and achieves the goal of over 90% accuracy score.

VI. Conclusions and Future Work

The purpose of this project was to utilize multiple machine learning techniques to classify objects as either a galaxy, star or quasar based on the Stellar Classification Dataset - SDSS17. The dataset contains 17 numerical features as well as a categorical classification value. It is 100,000 entries in

length. 3 different machine learning methods were used to see what would best classify the dataset, with those being Random Forest, Gaussian Mixture and XGBoost. Before model invocation, the dataset was prepped. This included visualizing the distributions of the features, checking for null values, calculating correlations, normalizing the data, converting categorical features and splitting into the train and test sets. Since the dataset is very large, the split was 95% train and 5% test in order to have more data points available for tuning the model while still being able to gauge variances in the data with a significant test set. The goal was to obtain models with a performance greater than or equal to 90% accuracy score on the test set. Random Forest proved to be a very effective technique with a 98.22% accuracy score after tuning using grid search. Further attempted optimization was made using PCA to reduce the feature space to the most important features. This did not perform as anticipated with a 80% accuracy score. The next approach was to use an unsupervised clustering algorithm Gaussian Mixture. The thought process of using this model was to see how an unsupervised clustering algorithm will perform on the dataset. Moreover, there was an assumption that the classes would follow a gaussian distribution that could be predicted well using gaussian mixture. The final results after tuning were very poor with a 29.98% accuracy score . It does in fact struggle with complex data so again PCA was used. Accuracy increased to 30.11% but that would be margin of error for a deployed dataset however it did not bring scores down like with Random Forest. It is also sensitive to imbalanced data. Thus the dataset was reduced to balance the class outcomes to even amounts, resulting also in a 90/10 train test split configuration due to the reduction in data. This did not aid in performance, dropping down to approximately 25%. None of these scores are acceptable for deployment however it is beneficial to test various types of models to see how they perform on each dataset so the efforts were not in vain. The final model was XGBoost since it is similar to random forest, meaning a high score would be expected for the dataset, however it can perform better due to the sequential tree building nature of the algorithm. The tuned model was 98.3% accurate, slightly better than random forest. With PCA the score dropped to 81.1% XGBoost was also built using the balanced dataset, resulting in a lower 97.7% score. Overall random forest and XGBoost exceeded the goal of 90% accuracy. For future work, it is suggested to focus on the tuning of XGBoost classifiers. The reasoning is not due to the slightly higher score but rather how it consists of more parameters to tune than random forest. In the scope of this project there was not enough compute power available to tune each XGBoost hyperparameter extensively. With the more fine grained tuning the score should increase. The data preparing process could also be expanded upon in the future. From the results, gaussian mixture performed poorly when classifying among the 3 classes. However a strength of GMM can be anomaly detection. Therefore future work can use it to classify outliers in each of the 3 categories separately and remove those from the dataset which will then be ingested by the XGBoost model.

VII. References

[1] Kaggle “Stellar Classification Dataset - SDSS17.” Available: https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17

[2] Australia Telescope National Facility. "Spectra of Stars, Galaxies, and Quasars." Available: https://www.atnf.csiro.au/outreach/education/senior/astrophysics/spectra_astro_types.html.

[3] Bertin, E. & Arnouts, S. "SExtractor: Software for Source Extraction." Available: http://www.astromatic.net/software/sextractor.

[4] The Pan-STARRS Project. "How to Separate Stars and Galaxies." Available: https://outerspace.stsci.edu/display/PANSTARRS/How+to+separate+stars+and+galaxies.

[5] Chen, Tianqi & Guestrin, Carlos. “XGBoost: A Scalable Tree Boosting System” Available: https://arxiv.org/abs/1603.02754

[6] Rashmi, Korlakai Vinayak & Gilad-Bachrach, Ran. “DART: Dropouts meet Multiple Additive Regression Trees” Available: https://proceedings.mlr.press/v38/korlakaivinayak15.pdf