Classification of Galaxies, Stars, and Quasars from Sloan Digital Sky Survey Data Release 17 Using Different Machine Learning Techniques

Ahmed Estiak
email:ahmedestiak14@gmail.com

Department of Physics, Shahjalal University of Science and Technology, Sylhet 3114, Bangladesh

Abstract

We worked with a specific chunk of data from Sloan Digital Sky Survey data release 17. Using machine learning models, we aimed to accurately classify stars, galaxies, and quasars (QSO). We applied a total of 15 different machine learning models to our dataset and then applied a voting classifier as an ensemble technique. Finally, we were able to increase our classification accuracy to 98.36%. We also highlighted comparing our research with a previous study on the same dataset, which had an accuracy of 98%.

Keywords: Machine Learning; Stellar Classification; SDSS; Star; Galaxy; Quasar

1 Introduction

Nowadays, it has become much easier to obtain data from various stellar objects. Numerous sky survey projects like the Sloan Digital Sky Survey (SDSS) are successfully in operation [**1**]. Over the past few years, machine learning algorithms have also been improved to process large amounts of data efficiently. The huge amount of data makes manual stellar classification and data labeling inconvenient, tedious, and unfeasible for astronomers. At the rate we are finding new galaxies and stars, it is almost impossible to process such a large amount of data manually. Various machine learning techniques and methods are already being used for star and galaxy classification [**2**, **3**, **4**] to solve this problem. Machine learning finds natural patterns within data sets and makes decisions and predictions accordingly. Finding automated methods for efficient and rapid classification is a very relevant issue for all future astronomical data releases.

In this research, we have used the Python language. Using supervised algorithms like Decision Tree (DT), Random Forests (RF), Kernel Support Vector Machines (SVM), Extra Trees, etc, galaxies, stars, and quasars have been classified. In this paper, data points with photometric observation errors have been excluded. Any suspicious data has also been removed. Also, a comparative picture of the performance of our research with a previous study on classification using machine learning on the same data set has been presented.

2 Method

2.1 Data

For this study, we have used Sloan Digital Sky Survey's (SDSS) observed data of spectral features. SDSS uses a dedicated 2.5-meter wide-angle optical telescope. It acquires images using a photometric system of 5 filters. We have used a chunk of SDSS data release 17, which contains 100,000 stellar objects' spectral data. Seventeen features or eigenvalues of each star sample are mentioned in this dataset. In Figure 1, we can see the first five rows and all the features' names of this dataset.

Among the 100,000 stellar objects, there are 59,445 galaxies, 21,594 stars, and 18,691 quasars. They are classified as 'GALAXY', 'STAR', and 'QSO' respectively.

	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	1.240000e+18	135.689107	32.494632	23.87882	22.27530	20.39501	19.16573	18.79371	3606	301	2	79	6.540000e+18	GALAXY	0.634794	5812	56354	171
1	1.240000e+18	144.826101	31.274185	24.77759	22.83188	22.58444	21.16812	21.61427	4518	301	5	119	1.180000e+19	GALAXY	0.779136	10445	58158	427
2	1.240000e+18	142.188790	35.582444	25.26307	22.66389	20.60976	19.34857	18.94827	3606	301	2	120	5.150000e+18	GALAXY	0.644195	4576	55592	299
3	1.240000e+18	338.741038	-0.402828	22.13682	23.77656	21.61162	20.50454	19.25010	4192	301	3	214	1.030000e+19	GALAXY	0.932346	9149	58039	775
4	1.240000e+18	345.282593	21.183866	19.43718	17.58028	16.49747	15.97711	15.54461	8102	301	3	137	6.890000e+18	GALAXY	0.116123	6121	56187	842

Figure 1: The first five rows, along with all the columns' names of the data set

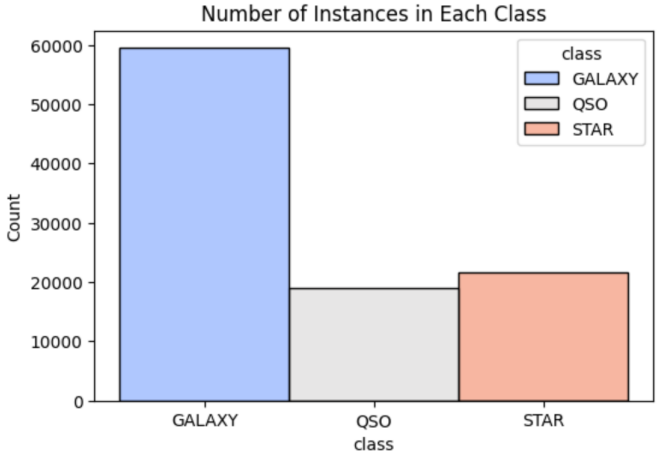


Figure 2: Number of instances in each class

2.2 Preprocessing of the data

We have initially checked that there is no null value in the dataset. Then, we plotted a heatmap (Figure 3) of all the features to check the correlation among those variables. Some features, like u, g, z, i, and r, have very strong correlations with each other (even in some cases the correlation value is 1.0). This is very interesting to note.

At this stage, we have dropped all the irrelevant columns. We have dropped MJD (as it is a time indicator) and cam_col, which has nothing to do with different classes. We have excluded columns with different identifier IDs from the dataset and columns that are redundant and unrelated to the classification problem. Based on our domain knowledge, we know that alpha and delta should not have any significance in our classification problem. So, we have removed those two columns also. When training our models, we have focused on the u, g, r, i, and z bands and redshift features of the SDSS dataset.

In this paper, we have tried to find outlier data points as part of feature engineering and removed the rows containing that data from the dataset. We found that the 'u', 'g', and 'z' values in row 79543 are negative. So, we have excluded this row from our dataset. A latest and similar study [**5**] on the same dataset did not remove this outlier data point. We have plotted another heatmap (Figure 4) of all the important features (u, g, r, i, and z bands, and redshift) to check the correlation among these variables to observe the correlations.

The dataset is very unevenly distributed and unbalanced. The number of stars and quasars is somewhat close, but the number of galaxies is several times greater than them. This will create biases and many deviations when training our models. To solve this problem, we have used the SMOTE (Synthetic Minority Over-sampling Technique) function [**6**] module from the imbalanced-learn library. This function is used for handling imbalanced datasets by generating synthetic examples of the minority class by interpolating between existing minority class instances. Finally, we have rescaled our dataset using the StandardScaler module from the scikit-learn library.

2.3 Dataset splitting

We have split our dataset into a ratio of 0.3 (70% for training data and 30% for testing data). We have fitted our machine learning models to the training set, and then we tested our model on the testing dataset to measure the classification performances.

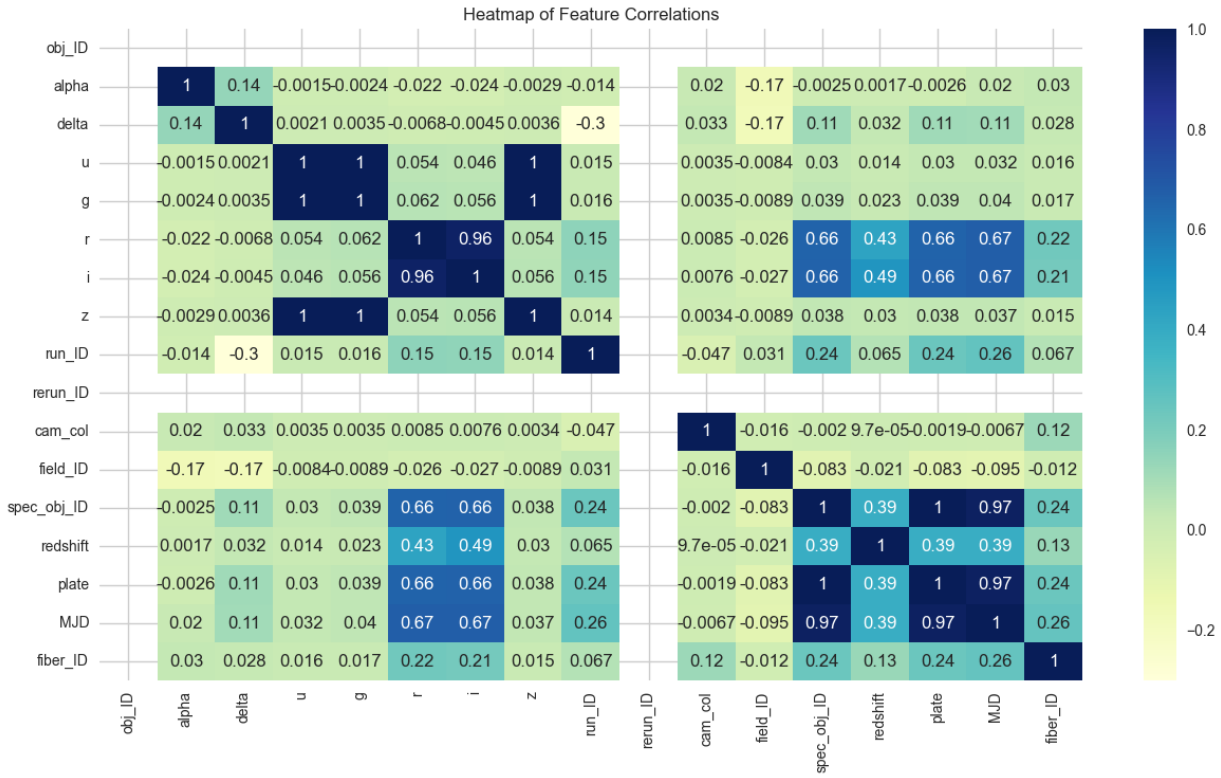


Figure 3: Heatmap of all the features to check the correlation among those variables

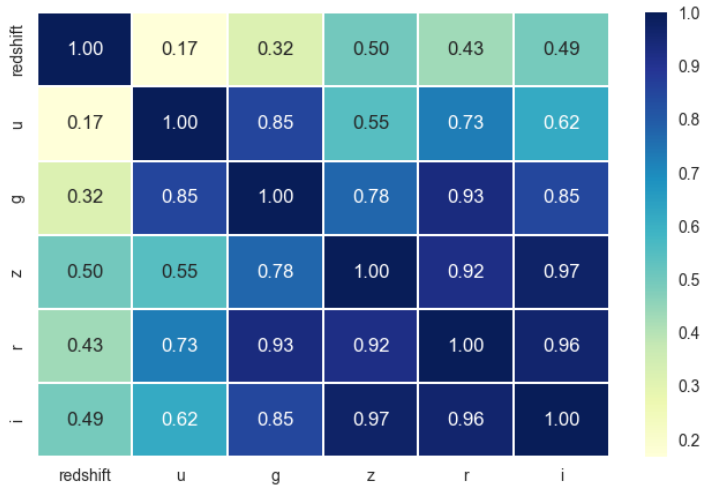


Figure 4: Heatmap of all the important features (u, g, r, i, and z bands, and redshift) to check the correlation among those variables

2.4 Models

Our goal was to find the best-performing model to classify the dataset. We have tried different machine learning models and checked their accuracy initially. The models are:

1. Logistic Regression
2. SVC (Support Vector Classifier)
3. Random Forest Classifier
4. Extra Trees Classifier

5. Gaussian Naive Bayes
6. Decision Tree Classifier
7. Ada Boost Classifier
8. K Neighbors Classifier
9. Bagging Classifier
10. Gradient Boosting Classifier
11. MLP Classifier
12. Extreme Gradient Boosting (XGB) Classifier
13. Stochastic Gradient Descent (SGD) classifier
14. Quadratic Discriminant Analysis
15. Light Gradient Boosting Machine (LGBM) Classifier

We have applied each of these models primarily with default parameters and found out the accuracy of each model. Based on the performance of the models, we have selected the 6 highest accuracy models for further investigation (like hyperparameter tuning). Finally we used an ensemble technique called Voting Classifier which helps us increase the accuracy by different combinations of the best models we have previously found.

3 Metrics to quantify the classification performance

We will now introduce some simple metrics to assess the classification performance.

3.1 The confusion matrix

The confusion matrix is a crucial machine learning tool for assessing the effectiveness of classification algorithms. It offers a clear and straightforward assessment of how a model's predictions are compared to the actual labels from the ground truth. The diagonal components of the matrix are made up of things that have been correctly identified, whereas the off-diagonal elements are made up of incorrectly classified objects. A thorough breakdown of the classification mistakes can be obtained from the distribution of the items in the confusion matrix, which can show systematic biases.

3.2 Accuracy Score

The accuracy score is a commonly used metric in machine learning to evaluate the performance of a classification model. It measures the proportion of correctly classified instances (or data points) out of the total number of instances in the dataset. In other words, it calculates how accurate the model's predictions are compared to the true labels.

The accuracy score is expressed as a ratio or percentage and is defined as:

$$Accuracy = \frac{Numberofcorrectpredictions}{Totalnumberofpredictions} \tag{1}$$

Accuracy is suitable for evaluating classification models when the classes are roughly balanced, meaning that each class has roughly equal numbers of instances. It provides a general measure of overall model performance.

Accuracy can be misleading when dealing with imbalanced datasets, where one class significantly outnumbers the others. In such cases, a model that predicts the majority class for all instances may achieve a high accuracy score even though it doesn't perform well in terms of correctly identifying minority classes.

In our case, we have used the SMOTE function to produce a balanced dataset. So, an accuracy score would be a good choice to compare among different machine learning models' performance. Other metrics such as precision, recall, and F1-score are more important in case of imbalanced dataset classification problems. Although we will also mention these scores in the performance of each model, we will only consider the accuracy score in our paper when making decisions.

4 Performances of Different Models

4.1 All the models with default parameters

Here, we will present all our applied models' performances with default parameters. The table below shows all the 15 machine learning models we have applied.

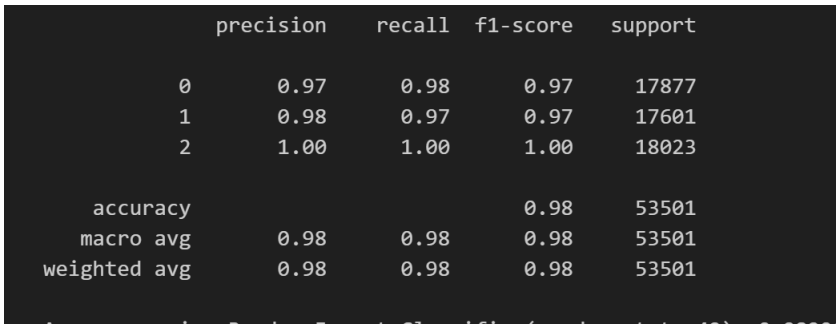
Model Name	Accuracy Score with default parameters
Random Forest Classifier	98.21%
Extra Trees Classifier	98.27%
Gaussian Naive Bayes	92.04%
Decision Tree Classifier	97.09%
Logistic Regression	94.94%
Support Vector Classifier (SVC)	96.44%
Ada Boost Classifier	45.92%
K Neighbors Classifier	96.99%
Bagging Classifier	98.03%
Gradient Boosting Classifier	97.03%
MLP Classifier	97.3%
Extreme Gradient Boosting (XGB) Classifier	97.83%
Stochastic Gradient Descent (SGD) classifier	91.03%
Quadratic Discriminant Analysis	94.99%
Light Gradient Boosting Machine (LGBM) Classifier	97.59%

Table 1.1: The accuracy scores of all 15 machine learning models

The above table shows that the best performing model is the Extra Trees Classifier with an accuracy 98.15%. Random Forest Classifier, Bagging Classifier, XGB Classifier, LBGM Classifier, and MLP Classifier are the 2nd, 3rd, 4th, 5th, and 6th best performing models respectively according to accuracy score. We will further analyze these six best performing models using hyperparameter tuning to increase their accuracy scores.

4.2 Hyperparameter tuning of Random Forest Classifier

At first, we checked the accuracy score of the Random Forest Classifier with all the default parameters and random state equal to 42. It gave an accuracy score of 98.20%. Figures 5, 6, and 7 show the classification report, confusion matrix, and ROC curves in this case, respectively.



	precision	recall	f1-score	support
0	0.97	0.98	0.97	17877
1	0.98	0.97	0.97	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 5: Classification report of Random Forest Classifier with random state 42

Then, we did hyperparameter tuning for the Random Forest Classifier in the case of our feature-engineered dataset using a randomized search CV. The best parameters, according to our search, were 'n_estimators': 50, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': 50, 'bootstrap': False. The accuracy score of the Random Forest Classifier using these parameters is 98.29%, which is better than the random forest classification accuracy score with the default parameters and random state 42.

Figures 8, 9, and 10 show the classification report, confusion matrix, and ROC curves of the Random Forest Classifier after hyperparameter tuning respectively.

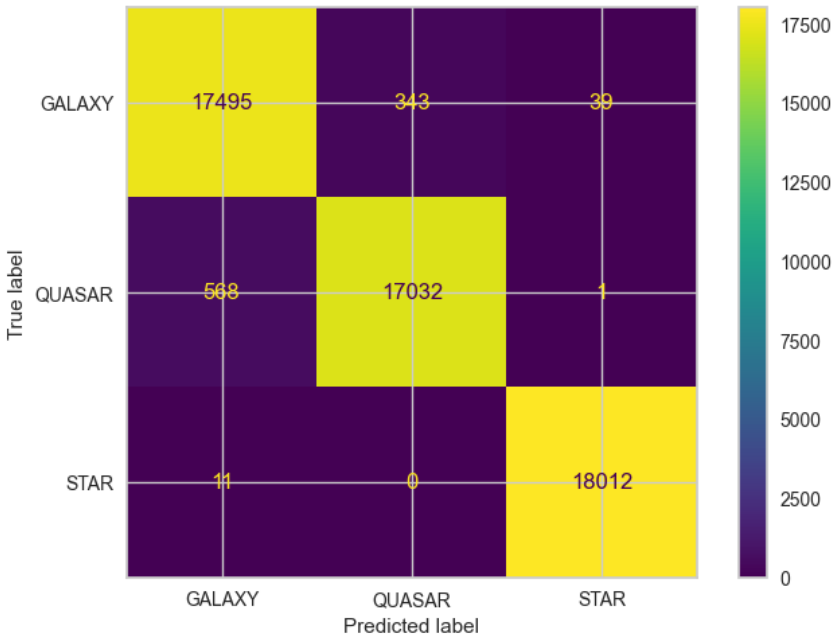


Figure 6: Confusion Matrix of Random Forest Classifier with random state 42

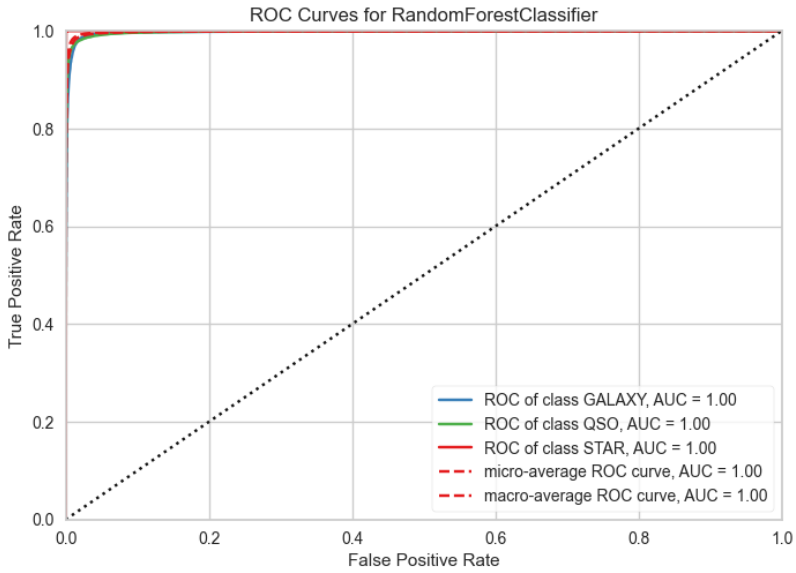


Figure 7: ROC Curves of Random Forest Classifier with random state 42

	precision	recall	f1-score	support
0	0.97	0.98	0.97	17877
1	0.98	0.97	0.98	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 8: Classification report of Random Forest Classifier after hyper parameter tuning

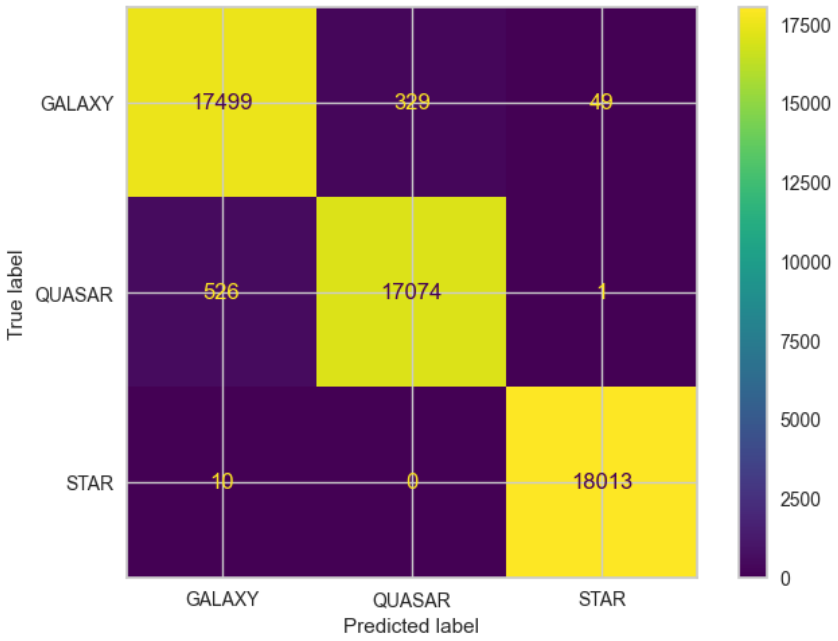


Figure 9: Confusion Matrix of Random Forest Classifier after hyper parameter tuning

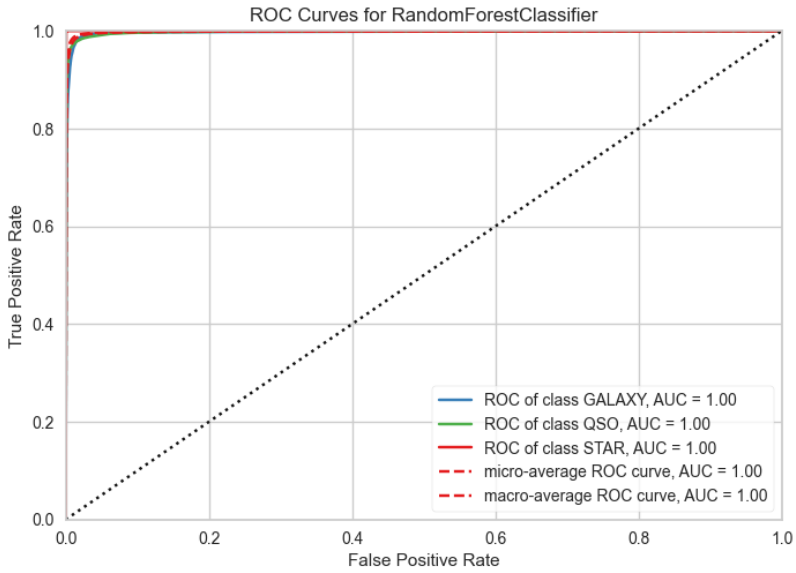


Figure 10: ROC Curves of Random Forest Classifier after hyper parameter tuning

4.3 Hyperparameter tuning of Extra Trees Classifier

At first, we checked the accuracy score of the Extra Trees Classifier with all the default parameters and random state equal to 42. It gave an accuracy score of 98.25%. Figures 11, 12, and 13 show the classification report, confusion matrix, and ROC curves in this case, respectively.

Then, we did hyperparameter tuning for the Extra Trees Classifier in the case of our feature-engineered dataset using a randomized search CV. The best parameters, according to our search, were 'warm_start': False, 'n_estimators': 50, 'max_features': 'sqrt', 'max_depth': 32, 'bootstrap': False. The accuracy score of the Extra Trees Classifier using these parameters is 98.19%. The Extra Trees Classification accuracy score with the default parameters and random state 42 was better than this score.

Figures 14, 15, and 16 are the classification report, confusion matrix, and ROC curves of the Random Forest Classifier after hyperparameter tuning, respectively.

	precision	recall	f1-score	support
0	0.97	0.97	0.97	17877
1	0.98	0.97	0.98	17601
2	0.99	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 11: Classification report of Extra Trees Classifier with random state 42

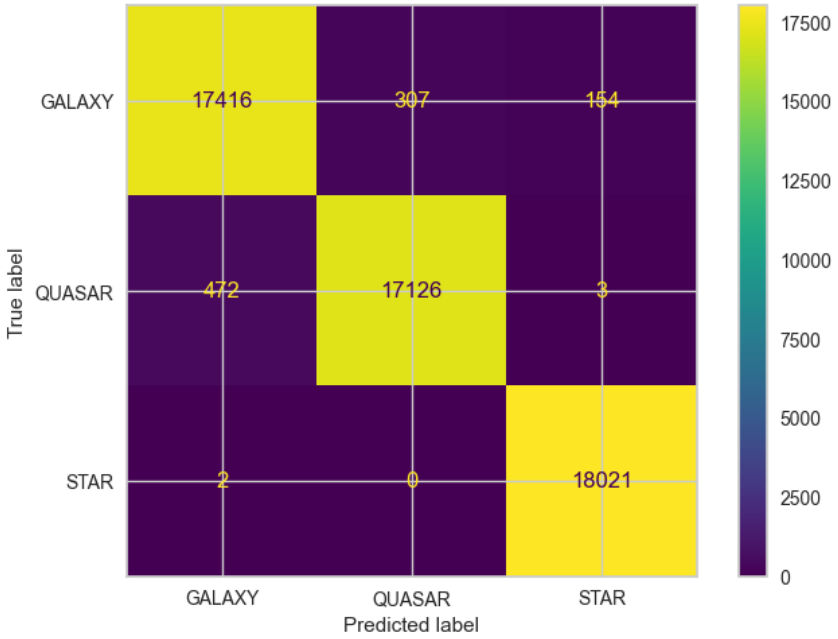


Figure 12: Confusion Matrix of Extra Trees Classifier with random state 42

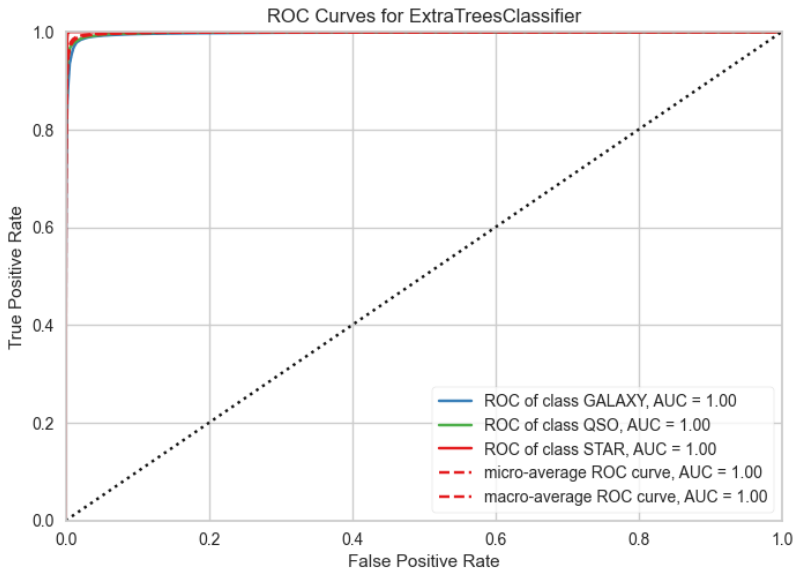


Figure 13: ROC Curves of Extra Trees Classifier with random state 42

	precision	recall	f1-score	support
0	0.97	0.97	0.97	17877
1	0.98	0.97	0.98	17601
2	0.99	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 14: Classification report of Extra trees Classifier after hyper parameter tuning

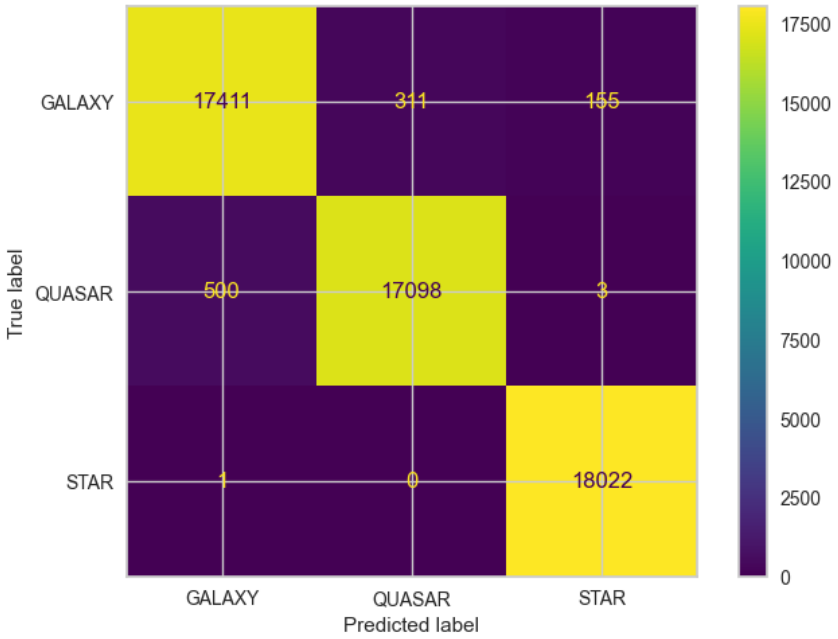


Figure 15: Confusion Matrix of Extra trees Classifier after hyper parameter tuning

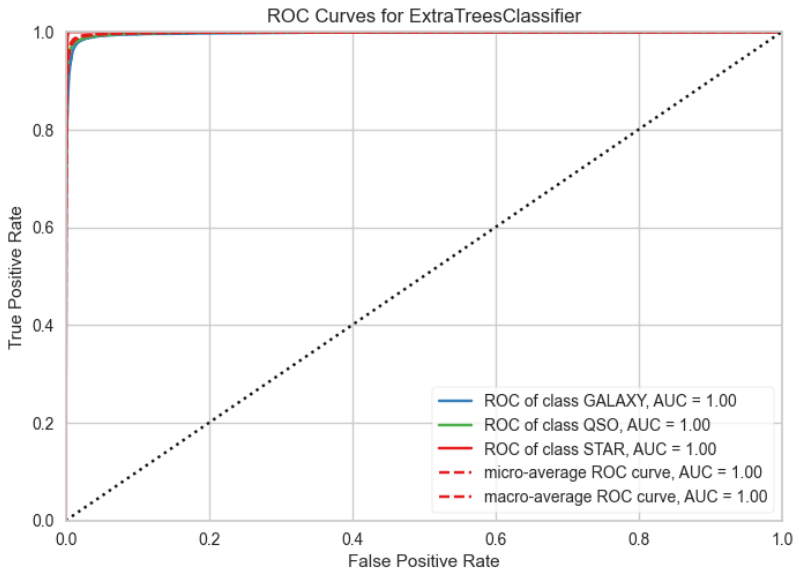


Figure 16: ROC Curves of Extra trees Classifier after hyper parameter tuning

4.4 Hyperparameter tuning of Bagging Classifier

At first, we checked the accuracy score of the Bagging Classifier with all the default parameters and random state equal to 42. It gave an accuracy score of 98%. Figures 17, 18, and 19 are the classification report, confusion matrix, and ROC curves in this case, respectively. Figures 14, 15, and 16 are the classification report, confusion matrix, and ROC curves of the Random Forest Classifier after hyperparameter tuning, respectively.

	precision	recall	f1-score	support
0	0.96	0.98	0.97	17877
1	0.98	0.96	0.97	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 17: Classification report of Bagging Classifier with random state 42

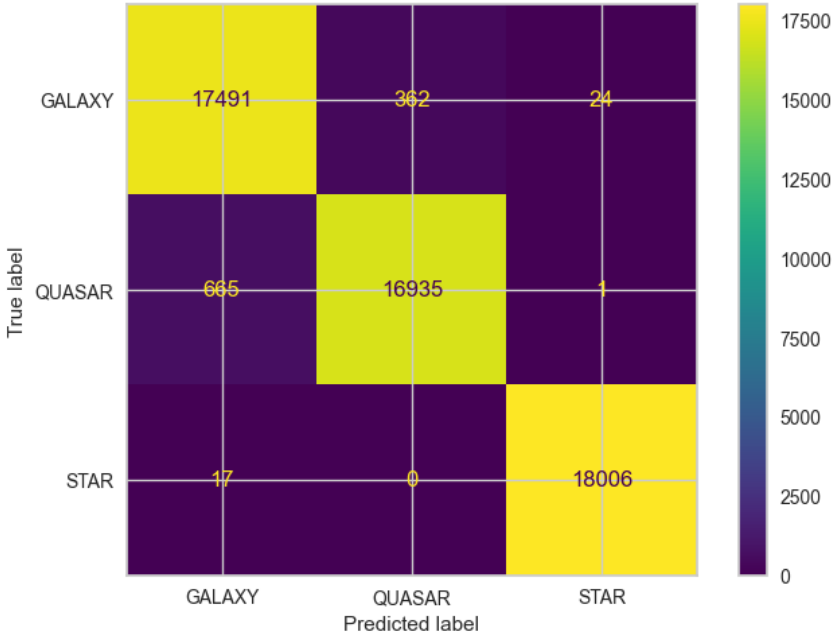


Figure 18: Confusion Matrix of Bagging Classifier with random state 42

Then, we did hyperparameter tuning for the Bagging classifier in the case of our feature-engineered dataset using a randomized search CV. The best parameters, according to our search, were 'warm_start': True, 'oob_score': False, 'n_estimators': 50, 'max_samples': 0.5, 'max_features': 0.92, 'bootstrap_features': False, 'bootstrap': True. The accuracy score of the Bagging Classifier using these parameters is 97.97%. The Bagging Classification accuracy score with the default parameters and random state 42 was better than this score.

Figures 20, 21, and 22 show the classification report, confusion matrix, and ROC curves of the Bagging Classifier after hyperparameter tuning, respectively.

4.5 Hyperparameter tuning of MLP Classifier

At first, we checked the accuracy score of the MLP Classifier with all the default parameters and a random state equal to 42. It gave an accuracy score of 97.29%. Figures 23,24,25 are the classification report, confusion matrix, and ROC curves in this case, respectively.

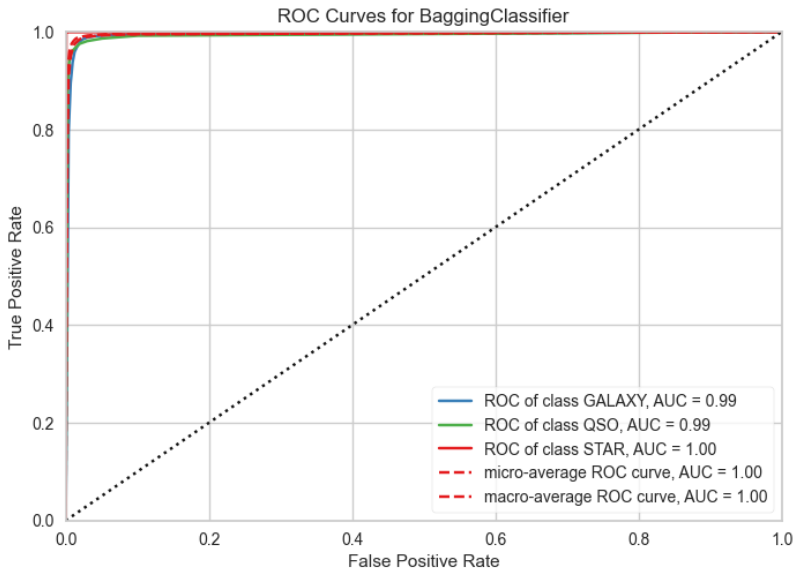


Figure 19: ROC Curves of Bagging Classifier with random state 42

	precision	recall	f1-score	support
0	0.96	0.98	0.97	17877
1	0.98	0.96	0.97	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 20: Classification report of Bagging Classifier after hyper parameter tuning

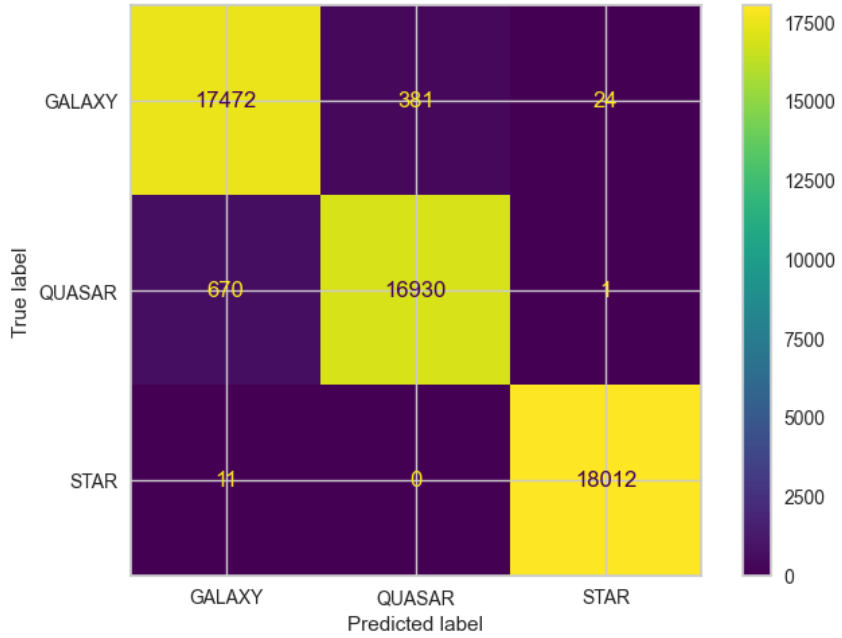


Figure 21: Confusion Matrix of Bagging Classifier after hyper parameter tuning

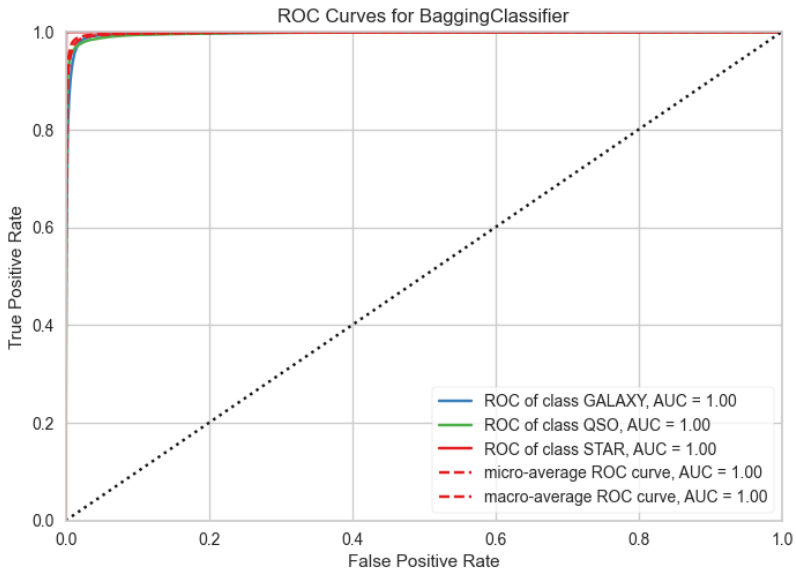


Figure 22: ROC Curves of Bagging Classifier after hyper parameter tuning

	precision	recall	f1-score	support
0	0.96	0.96	0.96	17877
1	0.97	0.96	0.96	17601
2	0.99	1.00	0.99	18023
accuracy			0.97	53501
macro avg	0.97	0.97	0.97	53501
weighted avg	0.97	0.97	0.97	53501

Figure 23: Classification report of MLP Classifier with random state 42

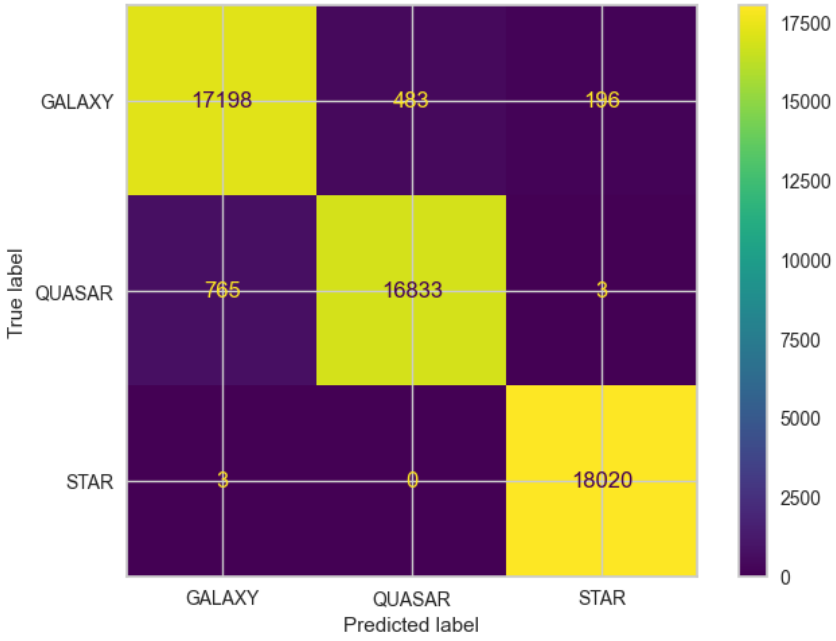


Figure 24: Confusion Matrix of MLP Classifier with random state 42

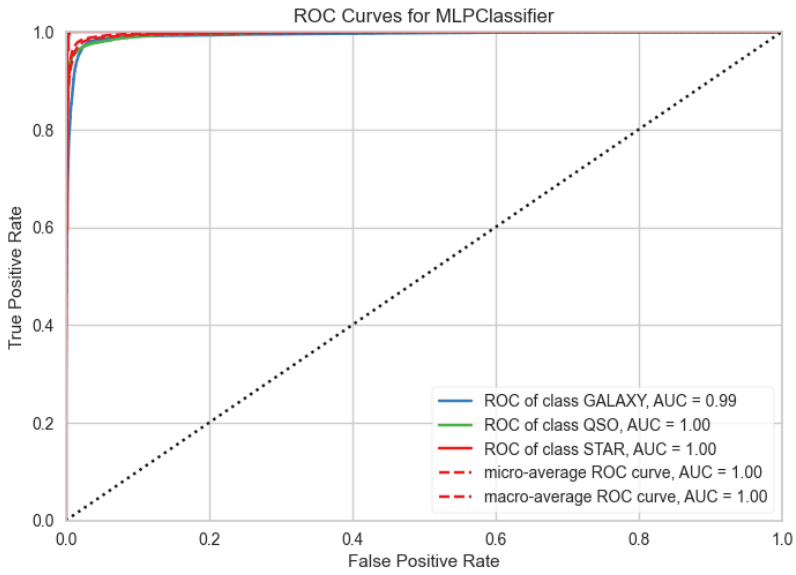


Figure 25: ROC Curves of MLP Classifier with random state 42

Then, we did hyperparameter tuning for the MLP Classifier in the case of our feature-engineered dataset using a randomized search CV. The best parameters according to our search were 'solver': 'adam', 'learning_rate': 'constant', 'hidden_layer_sizes': (10, 30, 10), 'alpha': 0.0001, 'activation': 'relu'. The accuracy score of the MLP Classifier using these parameters is 97.16%. The MLP Classification accuracy score with the default parameters and random state 42 was better than this score.

Figures 26, 27, and 28 are the classification report, confusion matrix, and ROC curves of the MLP Classifier after hyperparameter tuning, respectively.

	precision	recall	f1-score	support
0	0.95	0.96	0.96	17877
1	0.97	0.95	0.96	17601
2	0.99	1.00	0.99	18023
accuracy			0.97	53501
macro avg	0.97	0.97	0.97	53501
weighted avg	0.97	0.97	0.97	53501

Figure 26: Classification report of MLP Classifier after hyper parameter tuning

4.6 Hyperparameter tuning of XGB Classifier

At first, we checked the accuracy score of the XGB Classifier with all the default parameters and a random state equal to 42. It gave an accuracy score of 97.83%. Figures 29, 30, and 31 are the classification report, confusion matrix, and ROC curves in this case, respectively.

Then, we have done hyperparameter tuning for the XGB Classifier in the case of our feature-engineered dataset using a randomized search CV. The best parameters according to our search were 'subsample': 0.6, 'n_estimators': 100, 'min_child_weight': 0.5, 'max_depth': 5, 'learning_rate': 0.03, 'gamma': 0.0, 'colsample_bytree': 0.5. The accuracy score of the XGB Classifier using these parameters is 97.65%. The XGB Classification accuracy score with the default parameters and random state 42 was better than this score.

Figures 32, 33, and 34 are the classification report, confusion matrix, and ROC curves of the XGB Classifier after hyperparameter tuning, respectively.

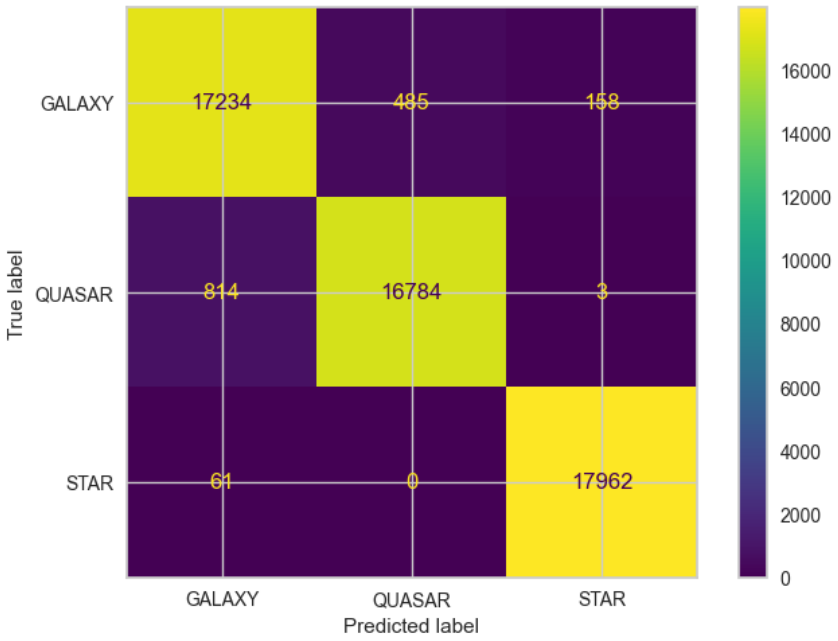


Figure 27: Confusion Matrix of MLP Classifier after hyper parameter tuning

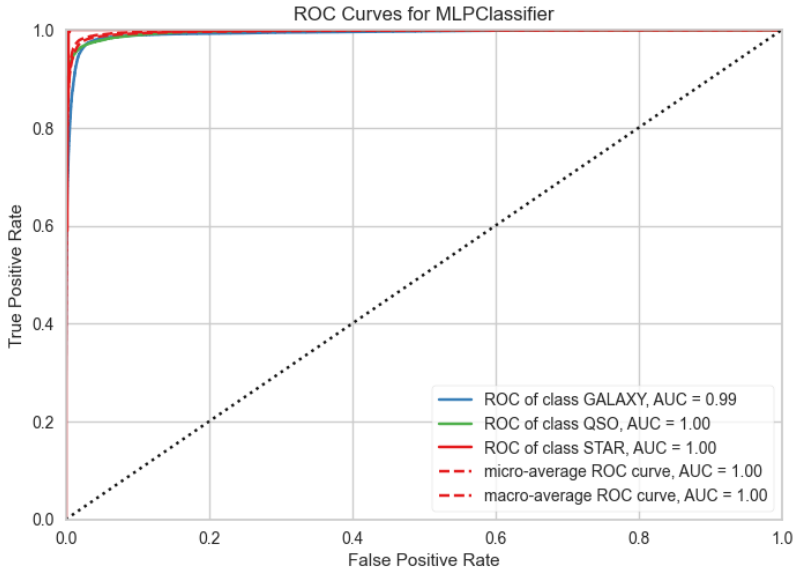


Figure 28: ROC Curves of MLP Classifier after hyper parameter tuning

	precision	recall	f1-score	support
0	0.96	0.97	0.97	17877
1	0.98	0.96	0.97	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 29: Classification report of XGB Classifier with random state 42

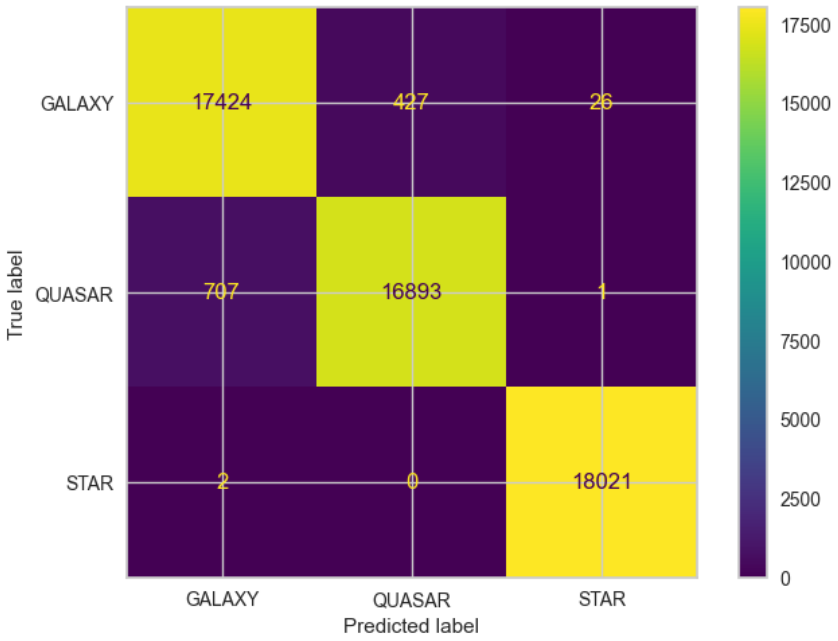


Figure 30: Confusion Matrix of XGB Classifier with random state 42

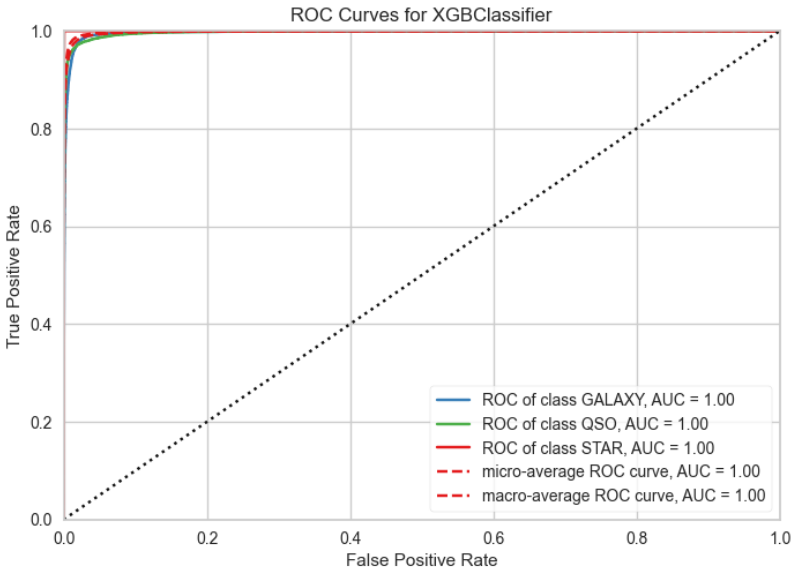


Figure 31: ROC Curves of XGB Classifier with random state 42

	precision	recall	f1-score	support
0	0.96	0.97	0.97	17877
1	0.97	0.96	0.96	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 32: Classification report of XGB Classifier after hyper parameter tuning

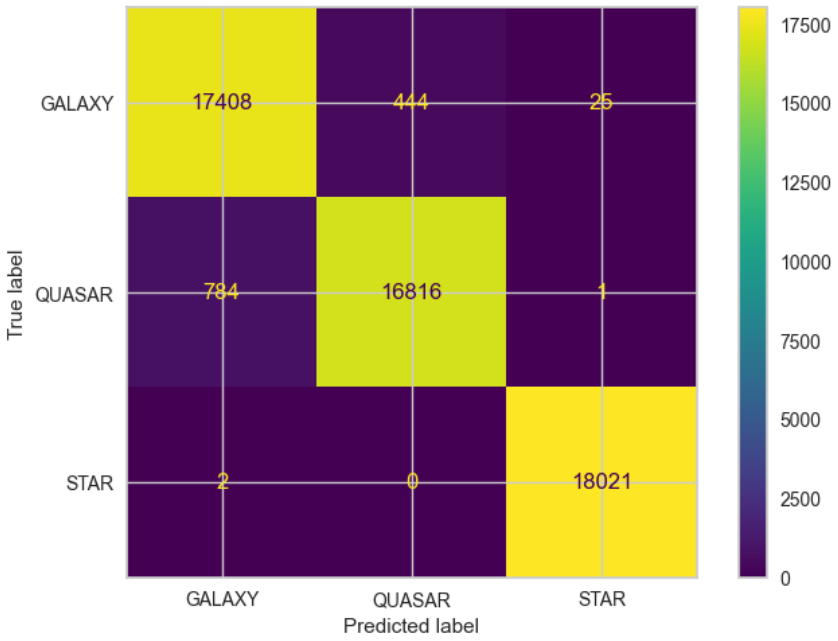


Figure 33: Confusion Matrix of XGB Classifier after hyper parameter tuning

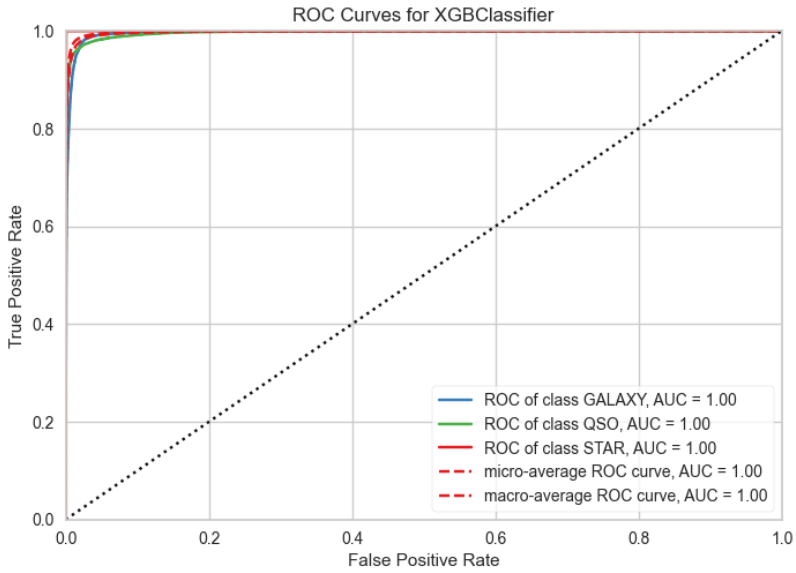


Figure 34: ROC Curves of XGB Classifier after hyper parameter tuning

4.7 Hyperparameter tuning of LGBM Classifier

At first, we checked the accuracy score of the LGBM Classifier with all the default parameters and a random state equal to 42. It gave an accuracy score of 97.57%. Figures 35, 36, and 37 are the classification report, confusion matrix, and ROC curves in this case, respectively.

Then, we did hyperparameter tuning for the LGBM Classifier in the case of our feature-engineered dataset using a randomized search CV. The best parameters according to our search were 'subsample': 0.2, 'num_leaves': 31, 'n_estimators': 200, 'max_depth': 10, 'learning_rate': 0.1, 'colsample_bytree': 0.9. The accuracy score of the LGBM Classifier using these parameters is 97.73%. After the hyperparameter tuning, the LGBM Classification accuracy score is better than the accuracy score with the default parameters and random state 42.

Figures 38, 39, and 40 are the classification report, confusion matrix, and ROC curves of the LGBM Classifier after hyperparameter tuning, respectively.

	precision	recall	f1-score	support
0	0.96	0.97	0.96	17877
1	0.98	0.96	0.97	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 35: Classification report of LGBM Classifier with random state 42

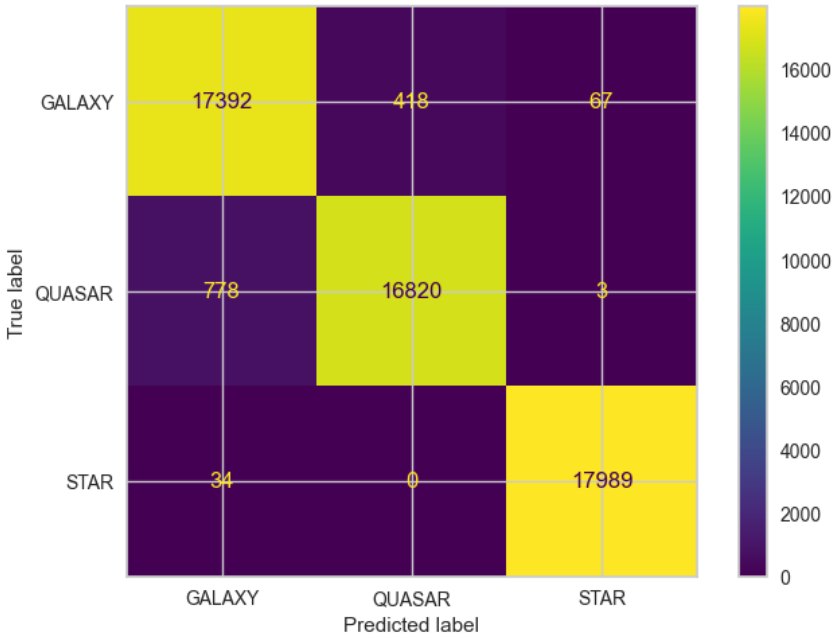


Figure 36: Confusion Matrix of LGBM Classifier with random state 42

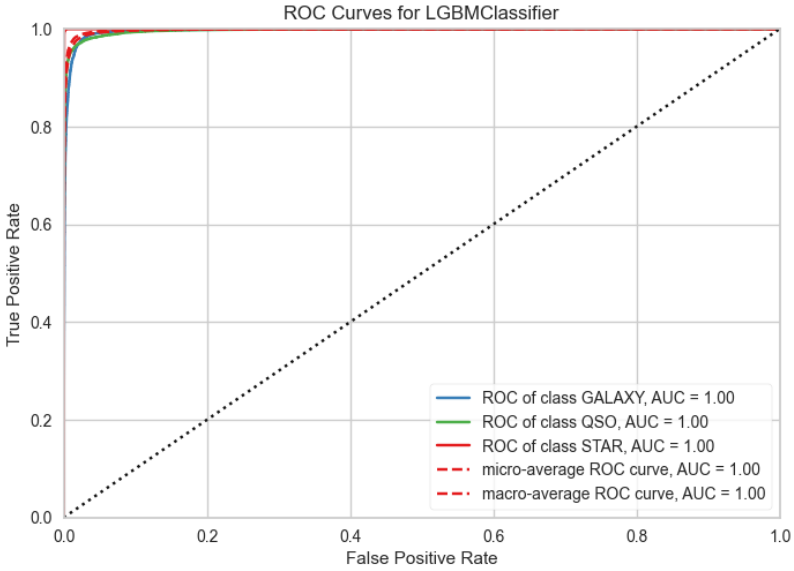


Figure 37: ROC Curves of LGBM Classifier with random state 42

	precision	recall	f1-score	support
0	0.96	0.97	0.97	17877
1	0.98	0.96	0.97	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 38: Classification report of LGBM Classifier after hyper parameter tuning

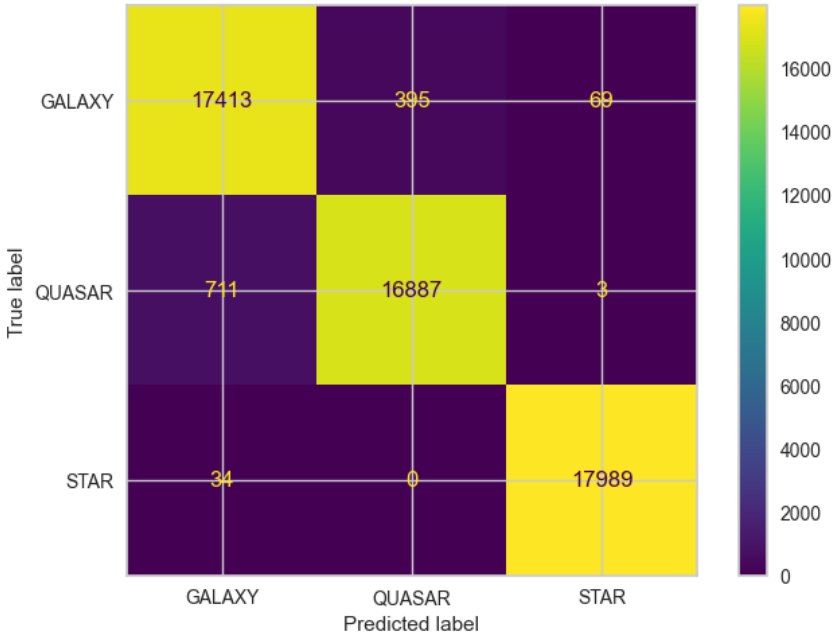


Figure 39: Confusion Matrix of LGBM Classifier after hyper parameter tuning

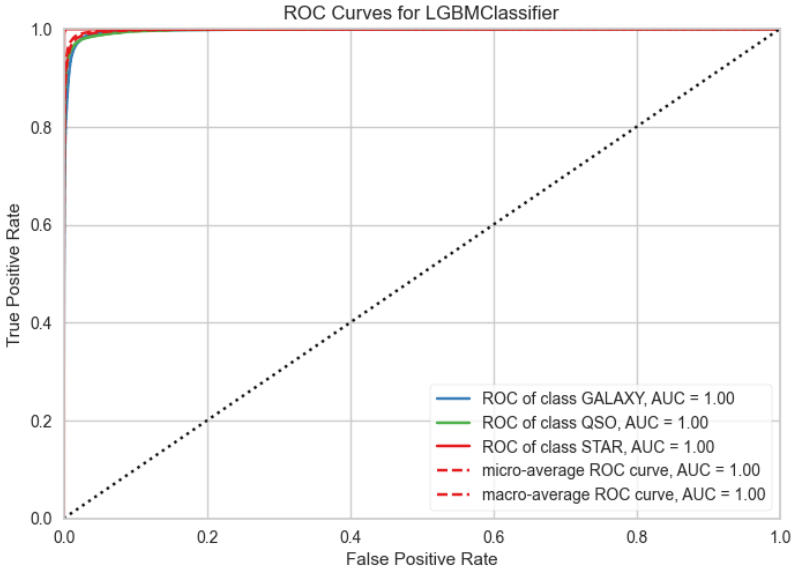


Figure 40: ROC Curves of LGBM Classifier after hyper parameter tuning

4.8 Voting Classifier to increase accuracy

A Voting Classifier is a machine learning ensemble technique in which multiple base models (classifiers or regressors) are trained to make predictions, and their individual predictions are combined through a "voting" mechanism to produce a final prediction. We have used our six best types of base models in our voting classifier as an ensemble. In each model type, either we have chosen the parameters as default with random state 42 or the parameters we got after hyperparameter tuning. This selection was solely based on the greater accuracy score.

We have tried different possible and effective combinations of the models. We have also tried soft and hard voting individually in all possible cases. We have found that the best performing model consists of 2 machine learning techniques. They are Random Forest Classifier and Extra Trees Classifier. The voting type was hard voting. We have used the parameters we got after hyperparameter tuning in the case of the Random Forest Classifier. In the case of te Extra Trees Classifier, we have used default parameters with random state 42. These choices were based on our previous accuracy score analysis. The accuracy score we have got from this combination is 98.36%, which is higher than any other machine learning technique and voting classifier combination used in our research.

Figures 41 and 42 are the classification report and confusion matrix of the voting Classifier in our best ensemble combination with hard voting type.

	precision	recall	f1-score	support
0	0.97	0.98	0.98	17877
1	0.99	0.97	0.98	17601
2	1.00	1.00	1.00	18023
accuracy			0.98	53501
macro avg	0.98	0.98	0.98	53501
weighted avg	0.98	0.98	0.98	53501

Figure 41: Classification report of Voting Classifier

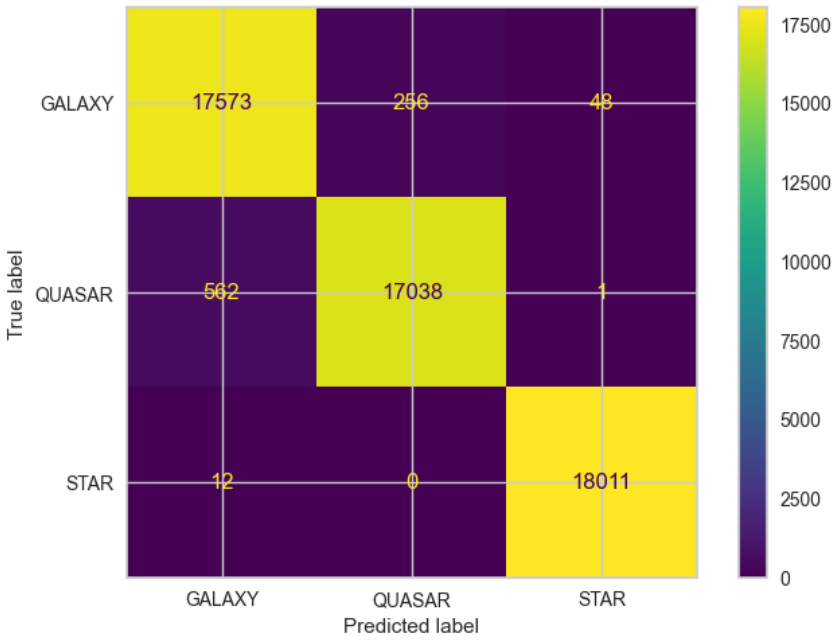


Figure 42: Confusion Matrix of Voting Classifier

5 Result

Among all the machine learning techniques we tested, the voting classifier (with the ensemble combination of Random Forest and Extra Trees Classifier and voting type "hard") performed best and gave the best accuracy

score. The accuracy score was 98.36%. The best solo performing model was the Extra Trees Classifier with all the default values and random state equal to 42. In this case, the accuracy score was 98.25%. After our hyperparameter tuning, the Random Forest Classifier became the best solo performing model with an accuracy score of 98.29%.

6 Comparison with other research

We found one recent research paper [**5**] by Zhuliang Qi on this similar topic and the same dataset. We will now analyze how our taken steps are different from the steps in this paper and how the accuracy of our best machine learning model is better than the best accuracy of the mentioned paper.

(i) One row of the dataset had a negative or error value, which was removed in our paper but not in the other mentioned paper.

(ii) We also removed columns containing alpha and delta data because our domain knowledge says they should have no effect on classification. But the other paper trained a machine learning model with these two columns.

(iii) We applied 15 different machine learning classification techniques or models to our dataset, compared to 3 in the other paper.

(iv) Among the three models in the mentioned paper, Support Vector Machines and Decision Trees did not even make it to the top 6 performing models (based on accuracy) in our research.

(v) Before using the ensemble model of the voting classifier, our best performing base model was the Extra Trees classifier. It gave better accuracy (98.18%) than the random forests classifier (98.15%). But in the mentioned paper, the Extra Trees Classifier was not even considered, and the Random Forests Classifier was the best performing model in that paper with an accuracy of 98%.

(vi) Finally, using the Voting Classifier ensemble, we achieved an accuracy score of 98.36%, which is 0.36% more than the mentioned paper.

7 Conclusion

In this research, we have tried to automatically detect stars, galaxies, or quasars by implementing different machine learning models on a chunk of Sloan Digital Sky Survey's data (DR17) consisting of 100,000 data points. We applied 15 different machine learning models or techniques for classification on our feature-engineered dataset. Among these models, the Extra Trees classifier performs the best. This is before using a voting classifier as an ensemble technique. Using the voting classifier, we increased the classification accuracy even more than the Extra Trees Classifier.

References

[1] Abdurro'uf, N., Accetta, K., Aerts, C., Silva Aguirre, V., Ahumada, R., Ajgaonkar, N., ... & Kneib, J. P. (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(2).

[2] Xiao-Qing, W., & Jin-Meng, Y. (2021). Classification of star/galaxy/QSO and star spectral types from LAMOST data release 5 with machine learning approaches. Chinese Journal of Physics, 69, 303-311.

[3] Rouan, D., Tasca, L., Soucail, G., & Le Fèvre, O. (2008). A robust morphological classification of high-redshift galaxies using support vector machines on seeing limited images-i. method description. Astronomy & Astrophysics, 478(3), 971-980.

[4] Tasca, L., Rouan, D., Pelat, D., Kneib, J. P., Le Fèvre, O., Capak, P., ... & Willott, C. (2009). A robust morphological classification of high-redshift galaxies using support vector machines on seeing limited images-II. Quantifying morphological k-correction in the COSMOS field at 1< z< 2: Ks band vs. I band. Astronomy & Astrophysics, 497(3), 743-753.

[5] Qi, Z. (2022). Stellar Classification by Machine Learning. In SHS Web of Conferences (Vol. 144, p. 03006). EDP Sciences.

[6] Sreejith, S., Nehemiah, H. K., & Kannan, A. (2020). Clinical data classification using an enhanced SMOTE and chaotic evolutionary feature selection. Computers in Biology and Medicine, 126, 103991.