**Space Server Dataset Project**

**Introduction:**

**1. Star –** It is a type of astronomical object consisting of a luminous spheroid of plasma which is held together by its own gravity. It is a luminous fixed point in the night sky which has a large remote incandescent body just like Sun. The nearest star to Earth is the Sun.

**2.** **Galaxy** – It is a gravitationally bound system of stars, stellar remnants, interstellar gas, dust, and dark matter. Galaxies are categorized according to their visual morphology as elliptical, spiral, or irregular. Many galaxies are thought to have super massive black holes at their active centers. It is a system of millions or billions of stars, together with gas and dust, held together by gravitational attraction.

3. **Quasars -** A quasar is an extremely luminous active galactic nucleus, in which a super massive black hole with mass ranging from millions to billions of times the mass of the Sun is surrounded by a gaseous accretion disk. It is a massive and extremely remote celestial object, emitting exceptionally large amounts of energy and typically having a star like image in a telescope. It has been suggested that quasars contain massive black holes and may represent a stage in the evolution of some galaxies.

**1. Problem Statement:**

The data consists of 10,000 observations of space taken by the SDSS. Every observation is described by 17 feature columns and 1 class column which identifies it to be either a star, galaxy or quasar. The data released by the SDSS is under public domain. It’s taken from the current data release **RD14**. The dataset offers plenty of information about space to explore. The class identifies an object to be galaxy, star or quasar.

**Dataset Description:**

The following are the descriptions of the features present in the dataset and they are as follows:

1. objid - Object Identifier

2. ra - J2000 Right Ascension (r-band)

3. dec - J2000 Declination (r-band)

4. u - better of DeV/Exp magnitude fit

5. g - better of DeV/Exp magnitude fit

6. r - better of DeV/Exp magnitude fit

7. i - better of DeV/Exp magnitude fit

8. z - better of DeV/Exp magnitude fit

9. run - Run Number

10. rerun - Rerun Number

11. camcol - Camera column

12. field - Field number

13. specobjid - Object Identifier

14. class - object class (galaxy, star or quasar object)

15. redshift - Final Redshift

16. plate - plate number

17. mjd - MJD of observation

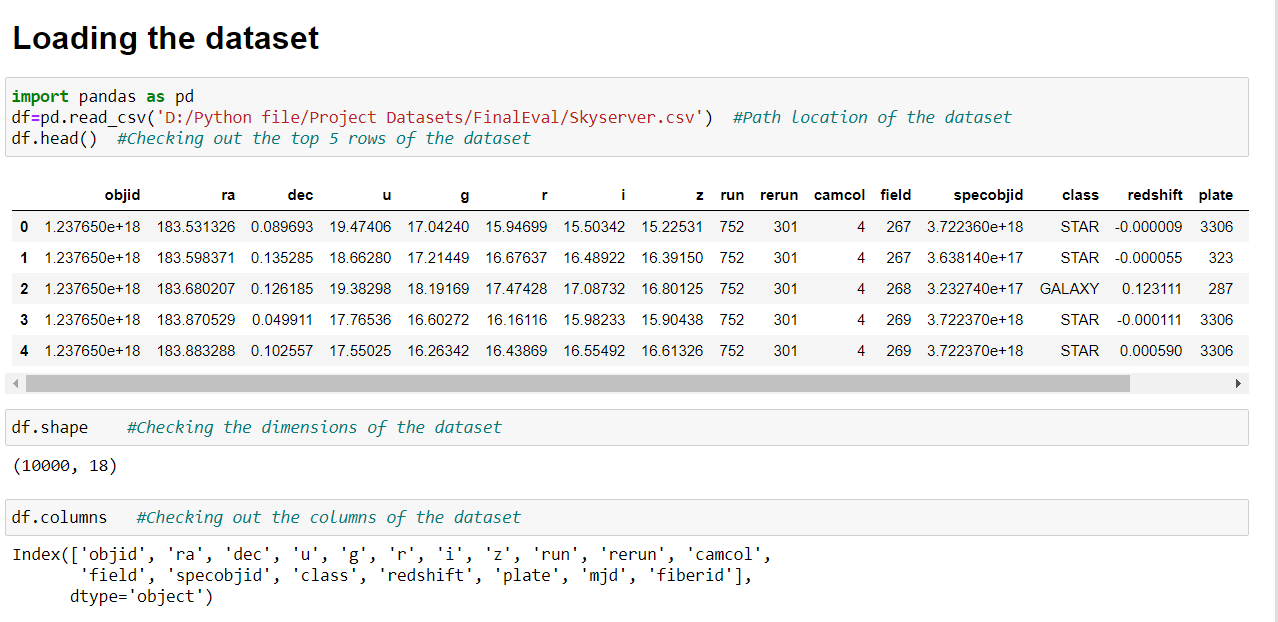
18. fiberid - fiber ID

We need to take class as the target variable and predict whether the data observation is either a star, galaxy or quasar and we are going to approach this problem statement by doing **classification analysis**.

**Importing Warning Library:**



**Load the dataset and check out its features and dimensions:**



**Checking for any null values in the dataset:**



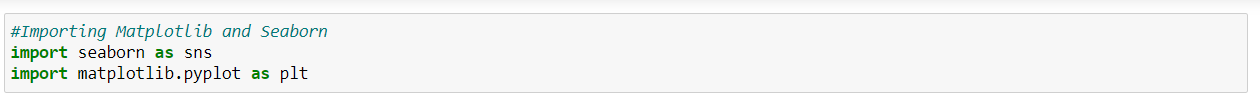
**Observations:**

1. In the Skyserver dataset, we have 10 float64 data type features, 7 int64 data type features and 1 object type feature present.
2. There are no null values present in the dataset.

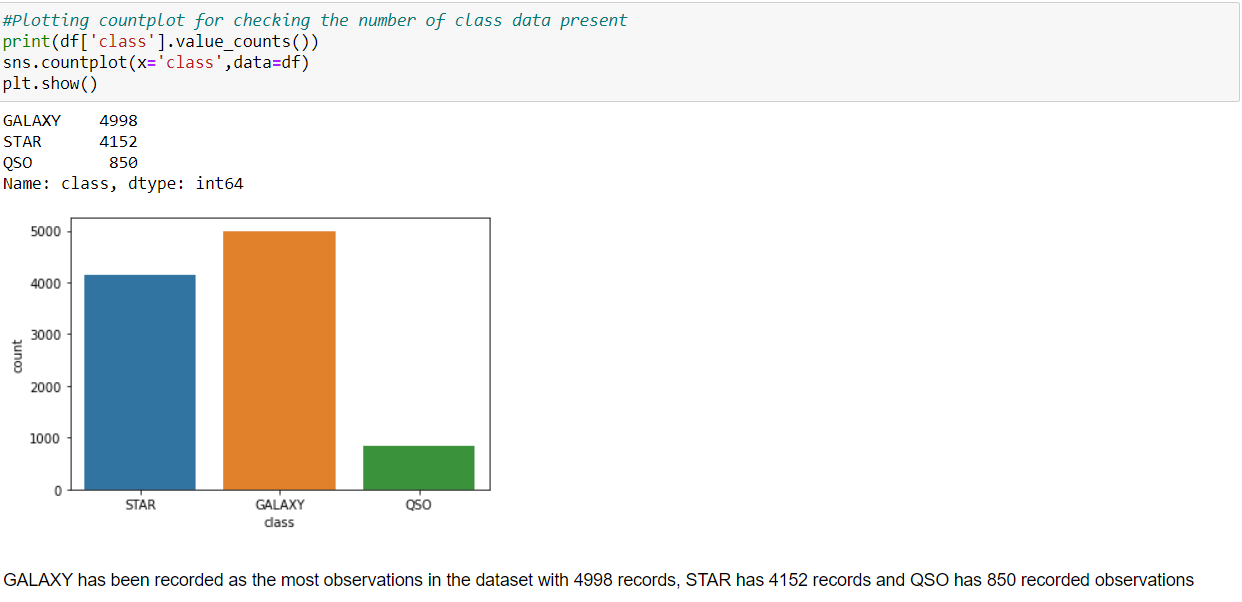
**2. Data Analysis:**

We will be using Exploratory Data Analysis (EDA) to check the relationship between features and also, we can understand the distribution of data in the dataset given. We plot graphs for the features by using data visualization libraries like matplotlib and seaborn.

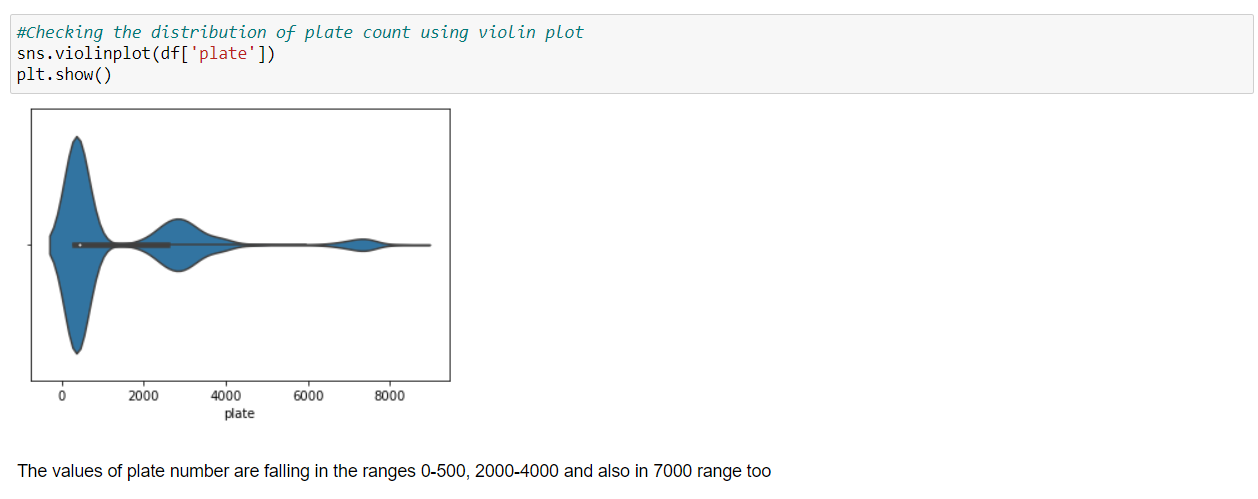
**Importing matplotlib and seaborn library:**



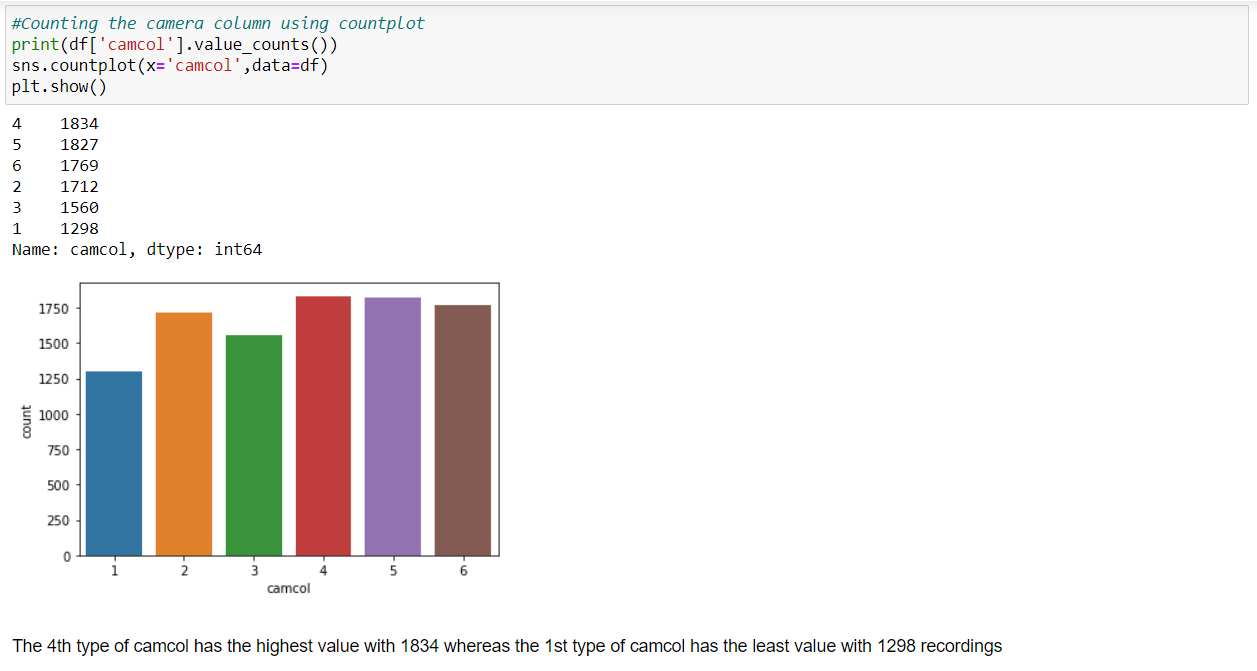
**Checking the number of class data present:**



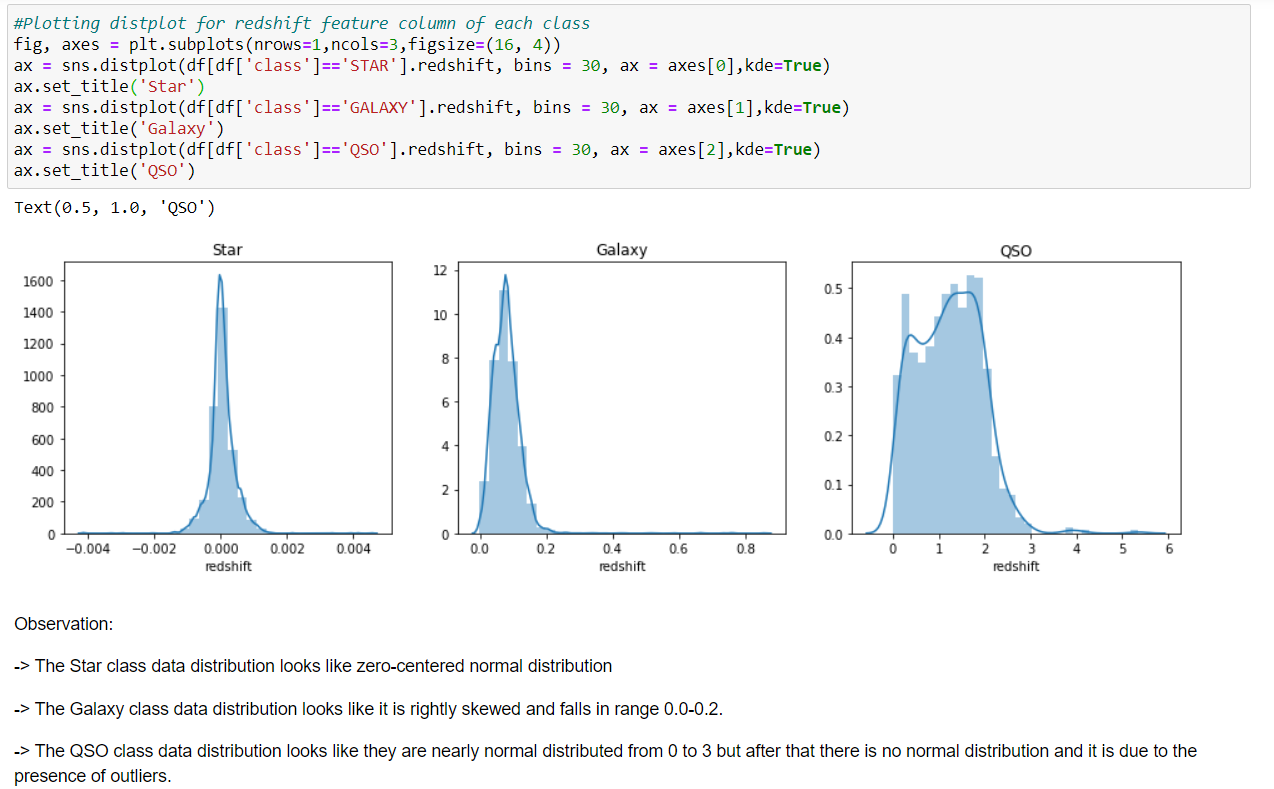
**Distribution of plate using violin plot:**



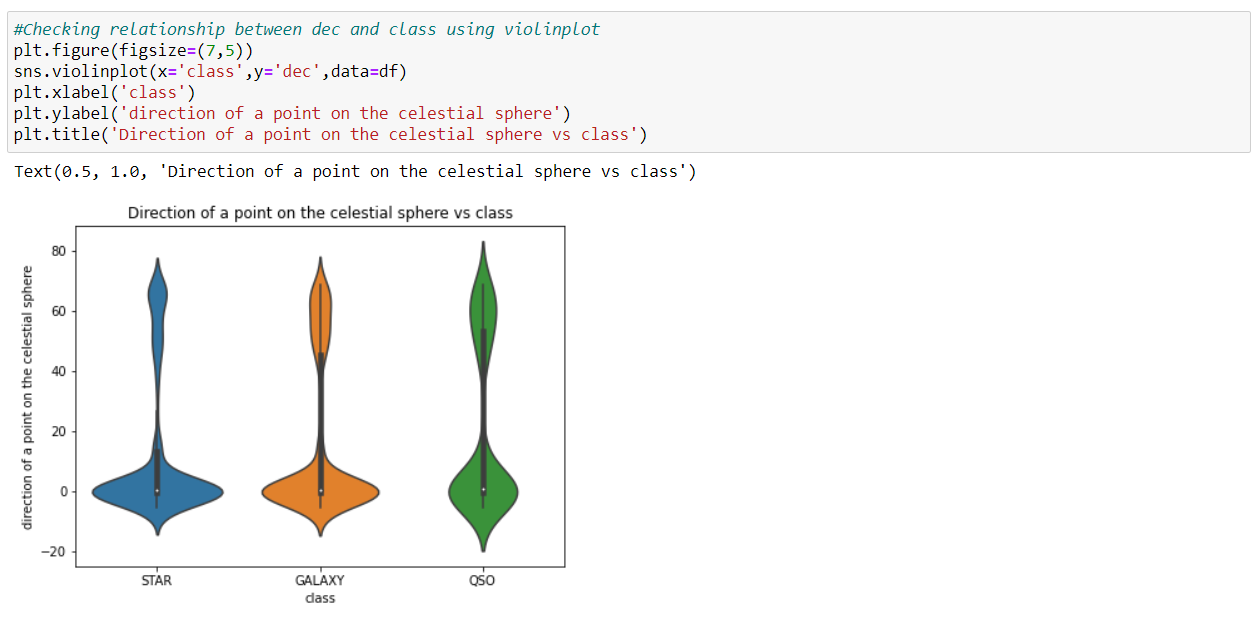
**Checking the camcol feature count:**



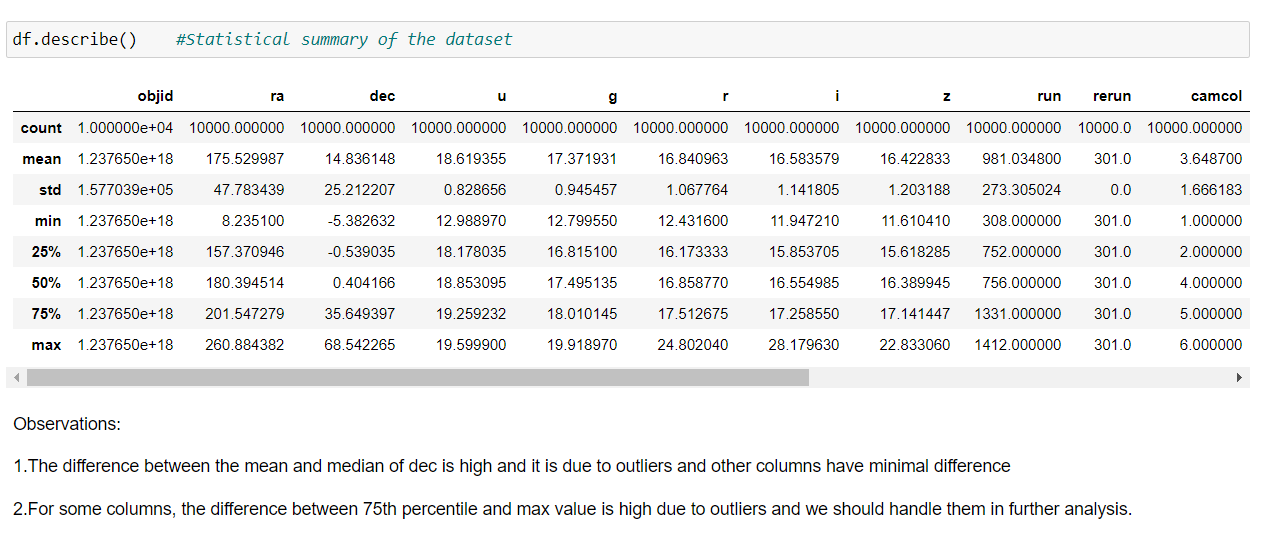
**Distplot for redshift feature column for each class:**



**Checking relationship between dec and class using violin plot:**



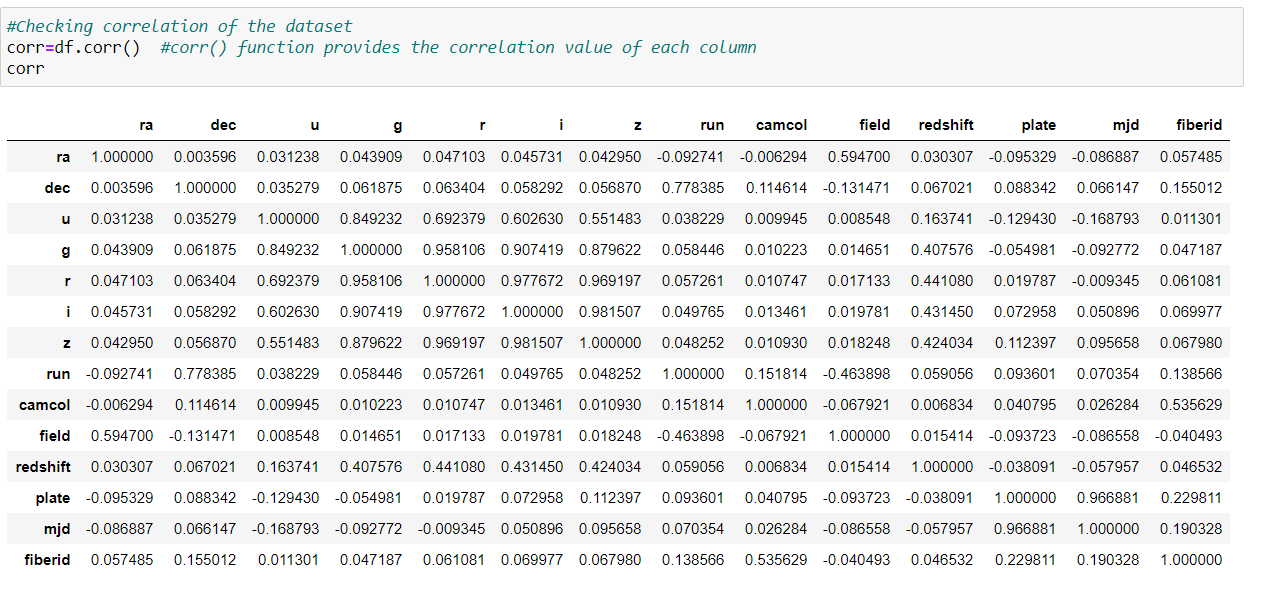
**Checking the Statistical summary of the dataset:**



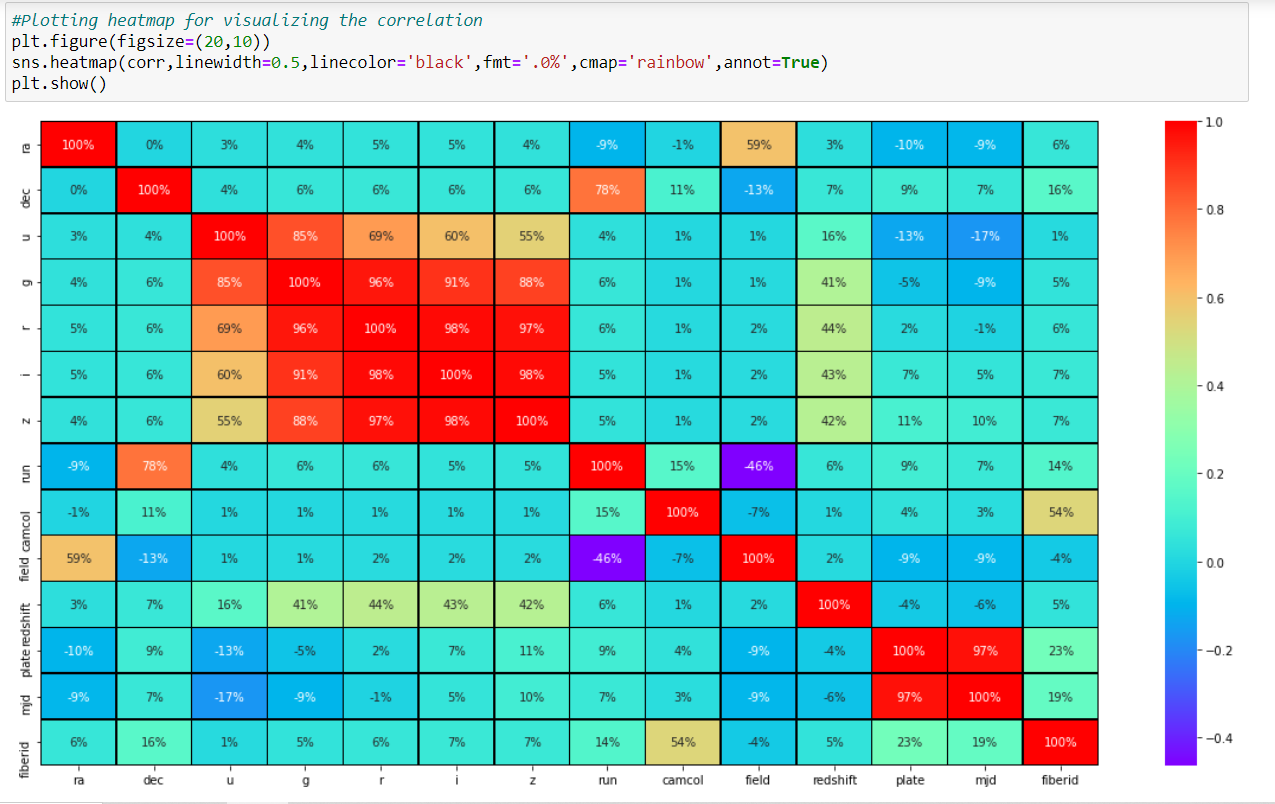
**Correlation:**

The statistical relationship between two variables is referred to as their correlation. The correlation factor represents the relation between columns in a given dataset. A correlation can be positive, meaning both variables are moving in the same direction or it can be negative, meaning that when one variable's value increasing, the other variable’s value is decreasing.

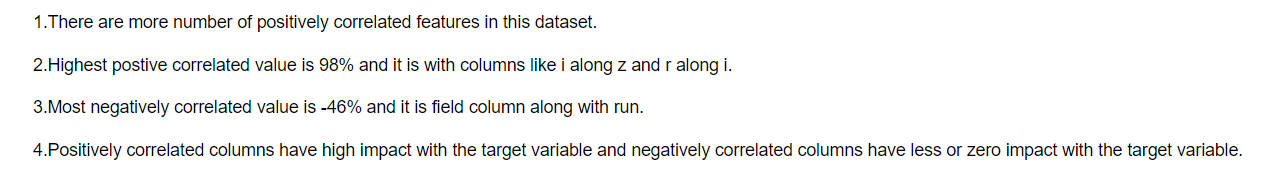
**Checking the correlation using corr:**



**Plotting heatmap for visualizing the correlation:**



**Observations:**



We can see that rerun, objid and specobjid columns are not contributing well in the dataset as it is same across the columns and it is not correlated to any of the other columns. Considering its nil contribution towards the model building, we can drop these columns for further analysis.

**3. EDA Concluding Remarks:**

-> GALAXY has been recorded as the most observations in the dataset with 4998 records, STAR has 4152 records and QSO has 850 recorded observations.

-> The 4th type of camcol has the highest value with 1834 whereas the 1st type of camcol has the least value with 1298 recordings.

-> There are outliers in the dataset as it is skewed on both right and left tail.

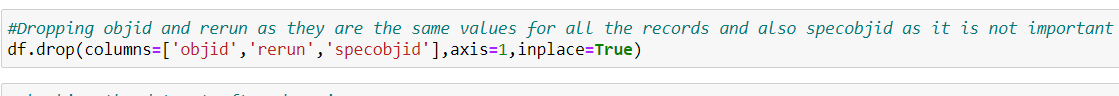
-> The difference between the mean and median of dec is high and it is due to outliers and other columns have minimal difference.

->For some columns, the difference between 75th percentile and max value is high due to outliers and we should handle them in further analysis.

->There are more positively correlated features in this dataset, with a highest value of 98%.

**4. Pre-processing Pipeline:**

**Dropping the columns:**

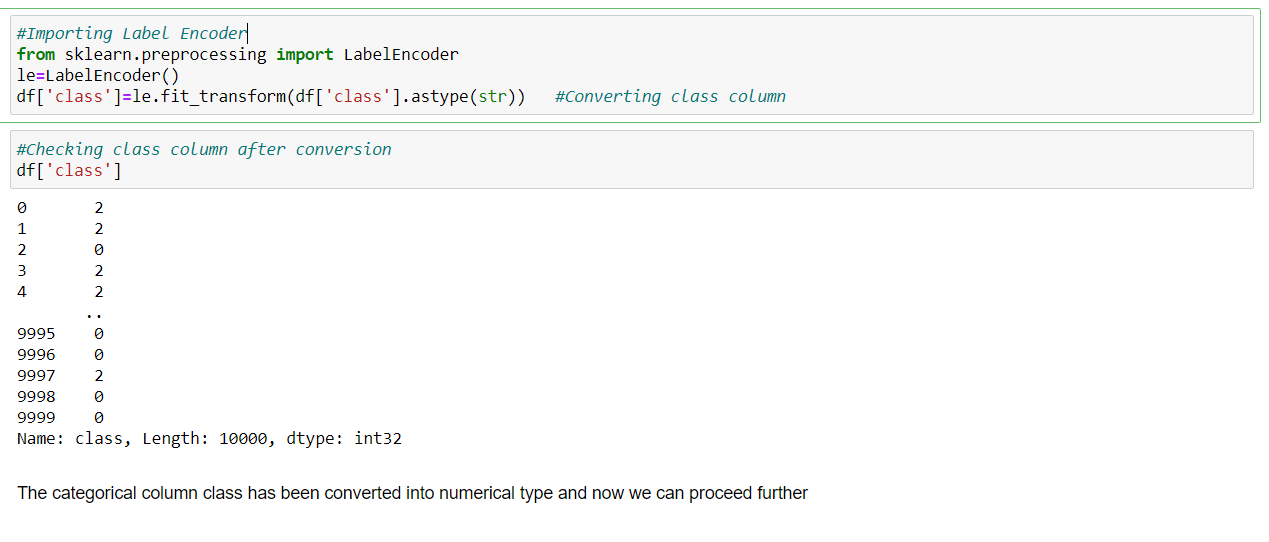


**Label Encoding:**

As class column has categorical data, we need to convert into numerical for proceeding further analysis and it can be done by using Label Encoder.

**Label Encoder** - refers to converts the labels into numeric form i.e., the columns which are in alphabetical or categorical values are assigned with numbers so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated during the process.

Import label encoder for converting categorical data (object) into numeric (int 32) data.

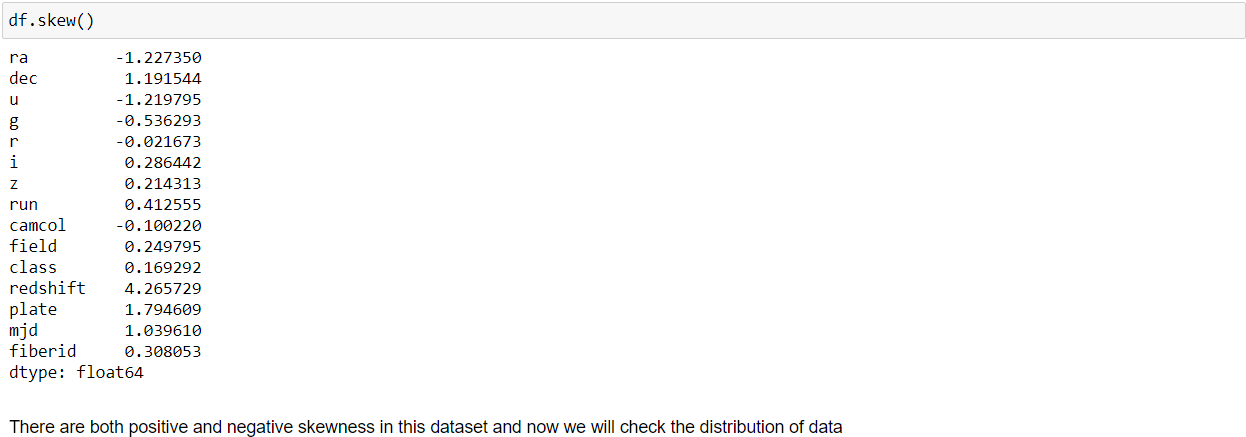


**Skewness:**

Skewness is a measure of the symmetry of a distribution. The highest point of a distribution is its mode. The mode marks the response value on the x-axis that occurs with the highest probability. A distribution is skewed if the tail on one side of the mode is fatter or longer than on the other: it is asymmetrical.

In an asymmetrical distribution a **negative skew** indicates that the tail on the left side is longer than on the right side (left-skewed), conversely a **positive skew** indicates the tail on the right side is longer than on the left (right-skewed). Asymmetric distributions occur when extreme values lead to a distortion of the normal distribution.

**Checking skewness in the dataset:**

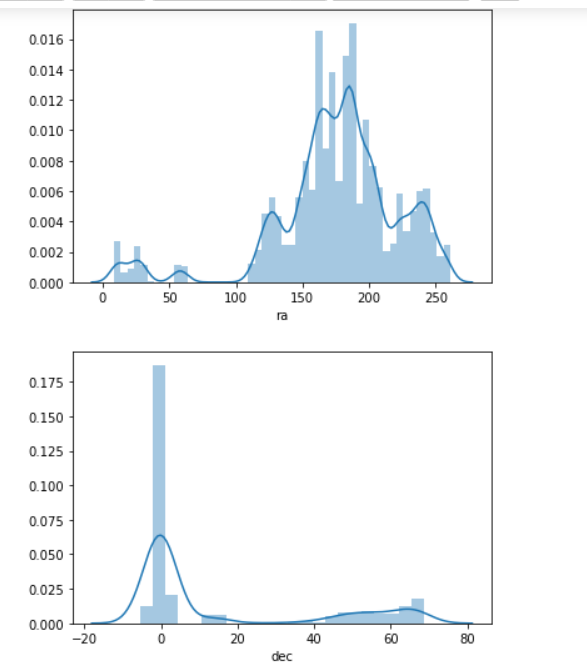


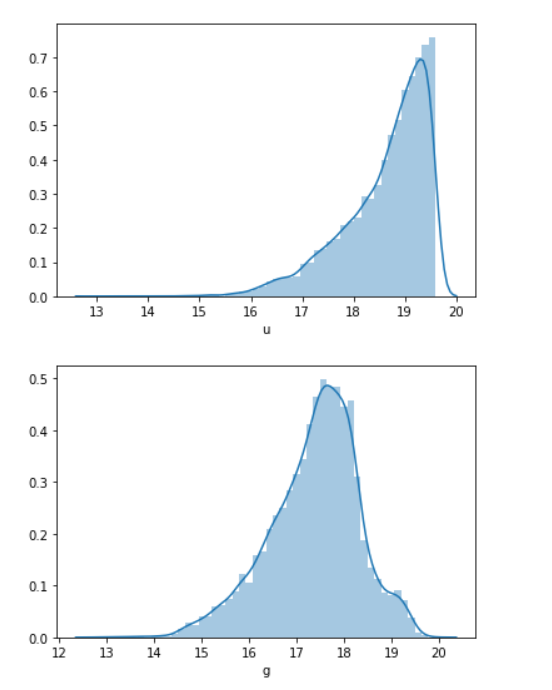
The range of skewed data is between -0.55<df<0.55 and if the value is obtained above the range, then it is said to be skewed. We can check the distribution of the features data using distplot.

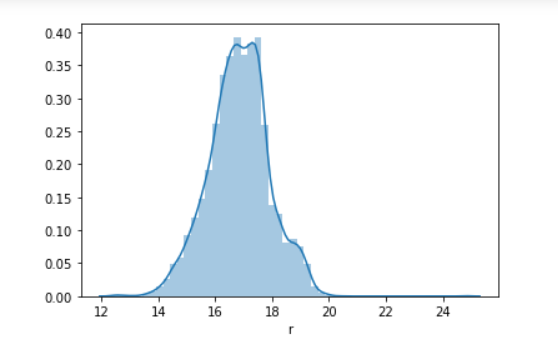
**Plotting distplot:**

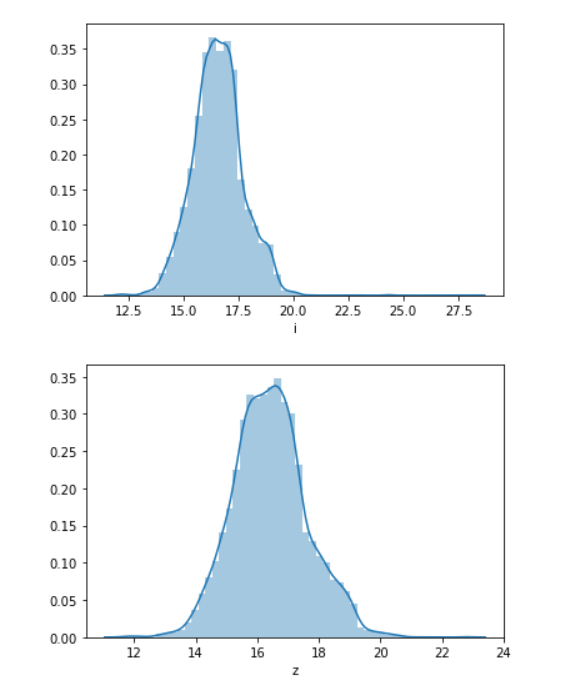


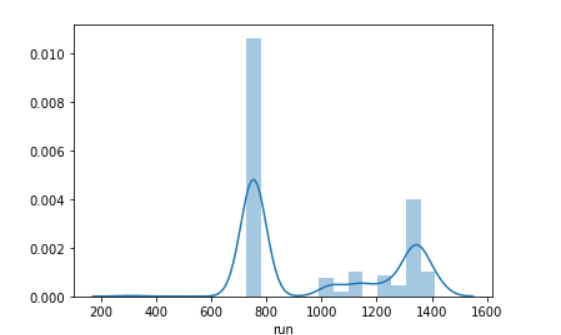
If we run a loop with all the columns for plotting the distplot, we can see that all the columns would be plotted simultaneously. Below is the graphical distribution of the columns data according to the skewness of each column:

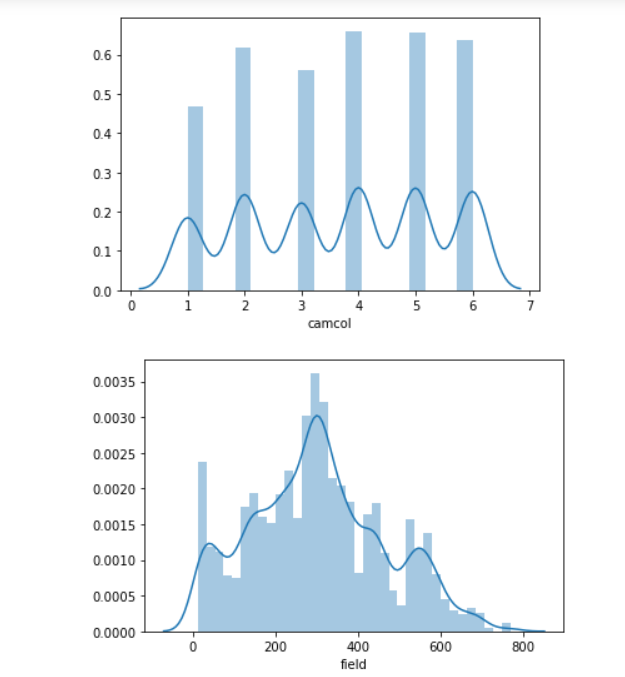


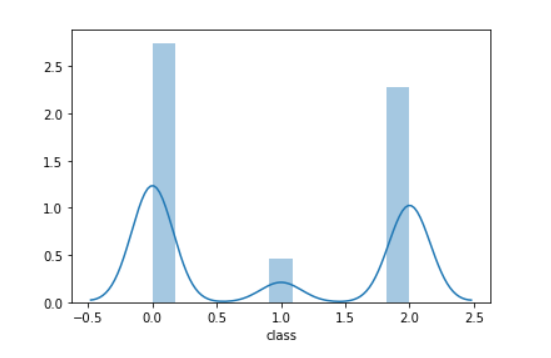


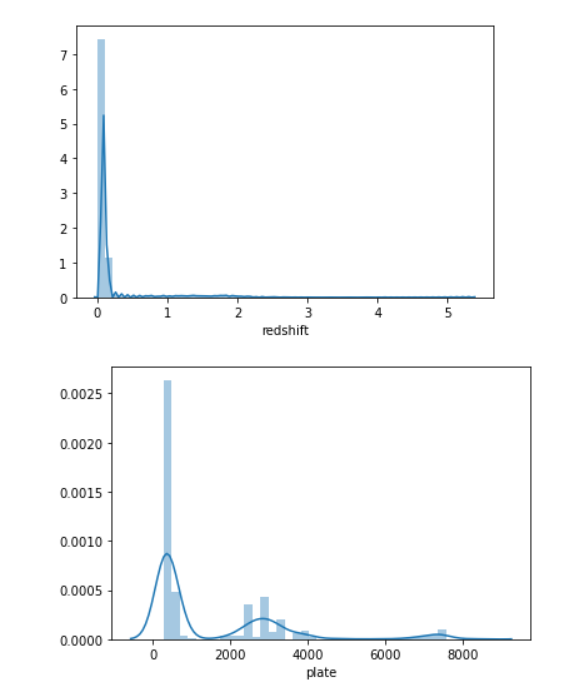


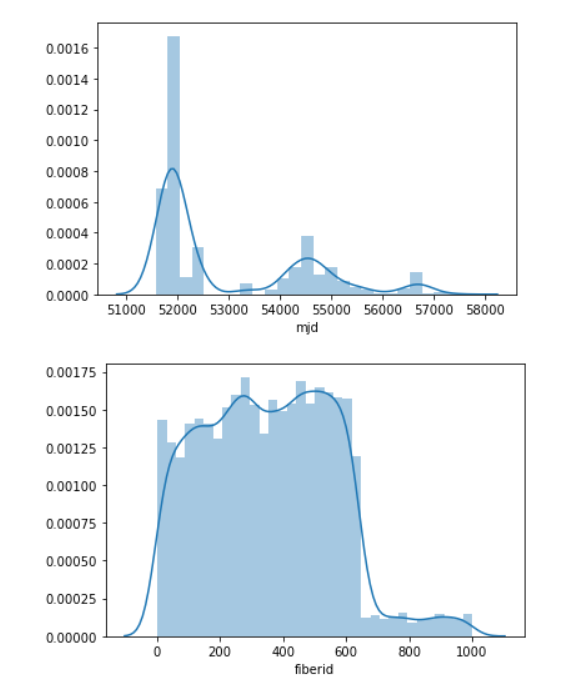










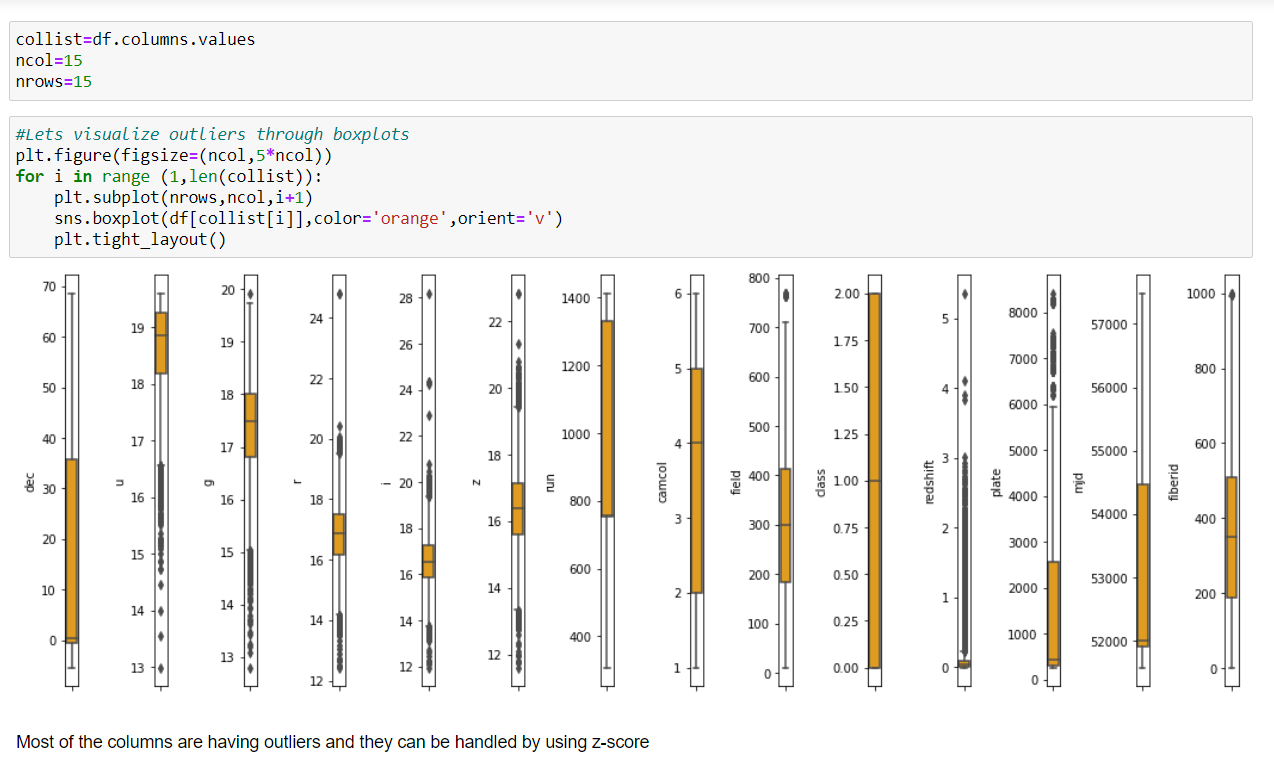


Most of the features are not normally distributed and it is due to the presence of outliers.

**Outliers:**

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

We can check outliers present in the dataset by plotting boxplot.



We can see that there are outliers present in the data and we can handle them by using z-score method.

**Z-score:**

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. If a Z-score is 0, it indicates that the data point's score is identical to the mean score. A Z-score of 1.0 would indicate a value that is one standard deviation from the mean. Z-scores may be positive or negative, with a positive value indicating the score is above the mean and a negative score indicating it is below the mean.

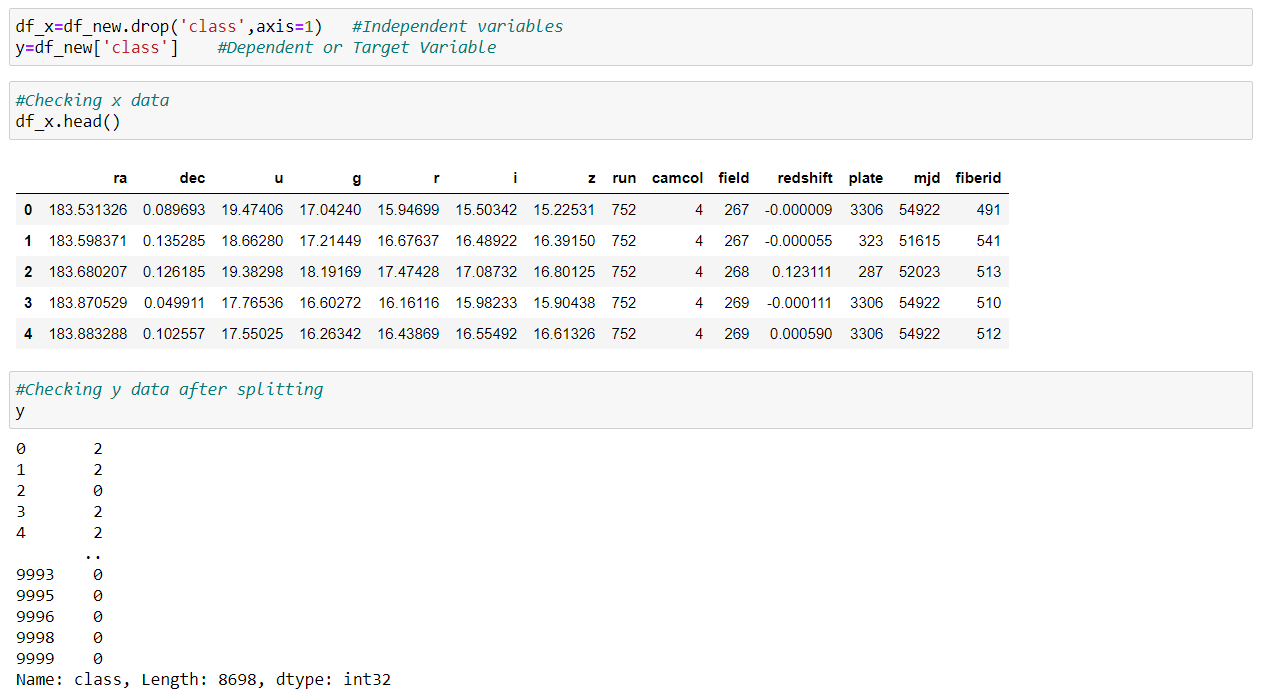
First, we will convert all values to absolute values and then we will set a threshold value for identifying the outliers and with that value, we will be removing the outliers from the dataset and check the number of rows affected. Below is the procedure to remove outliers:



We can see that the original dataset had 10000 rows of data and after removal of outliers, we can see that nearly 1300+ rows of data had outliers and they had been removed by using z-score method.

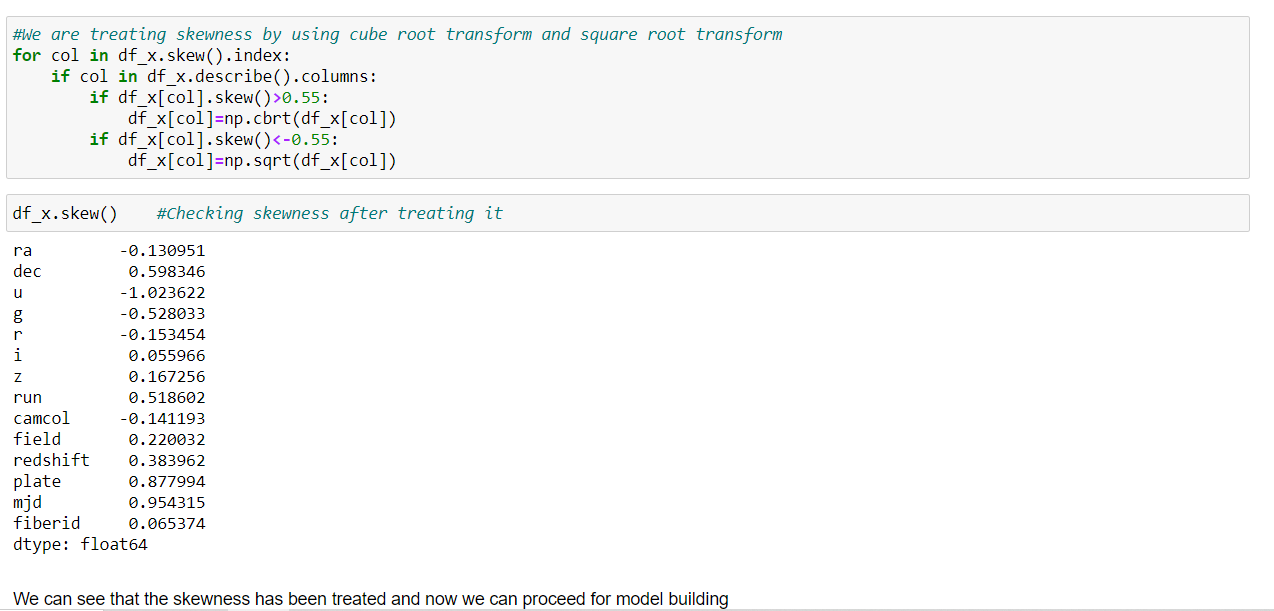
**Model Training:**

We need to split independent and dependent variables from the dataset in order to perform further analysis



**Treating skewness:**

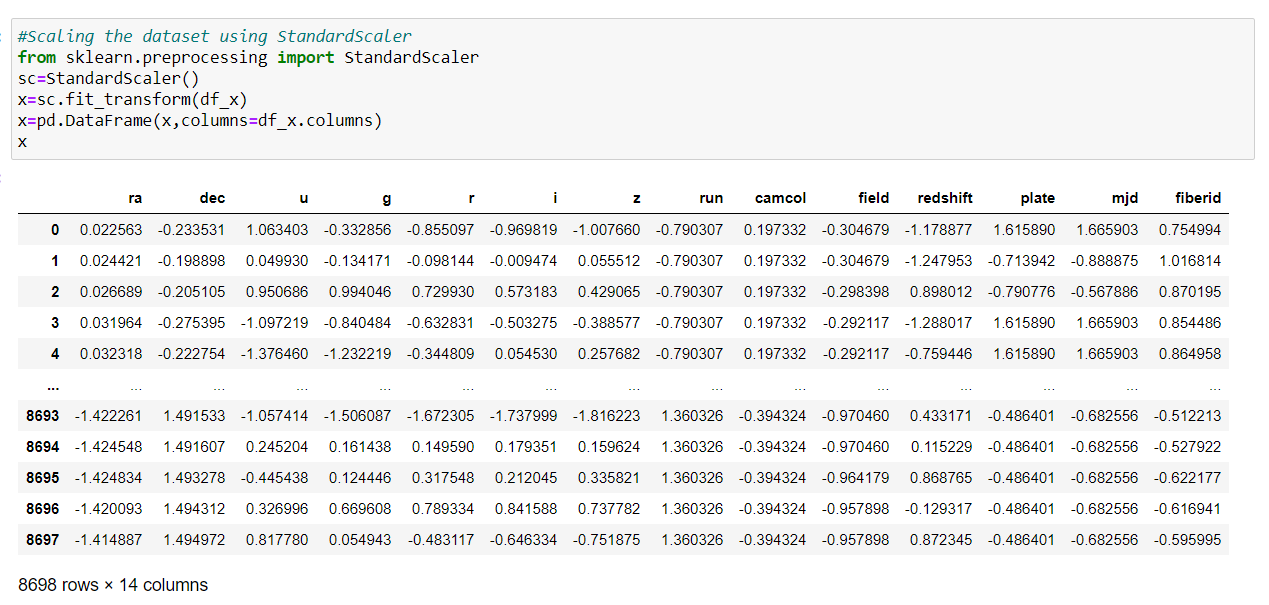
We can treat the skewness by using methods like log-transform, square-root transform, cube-root transform, etc. Below is the way to treat skewness:



**Scaling the data:**

Sometimes model can be biased to higher values in dataset, so it is better to scale the dataset so that we can bring all the columns in common range. We can use Standard Scaler here.

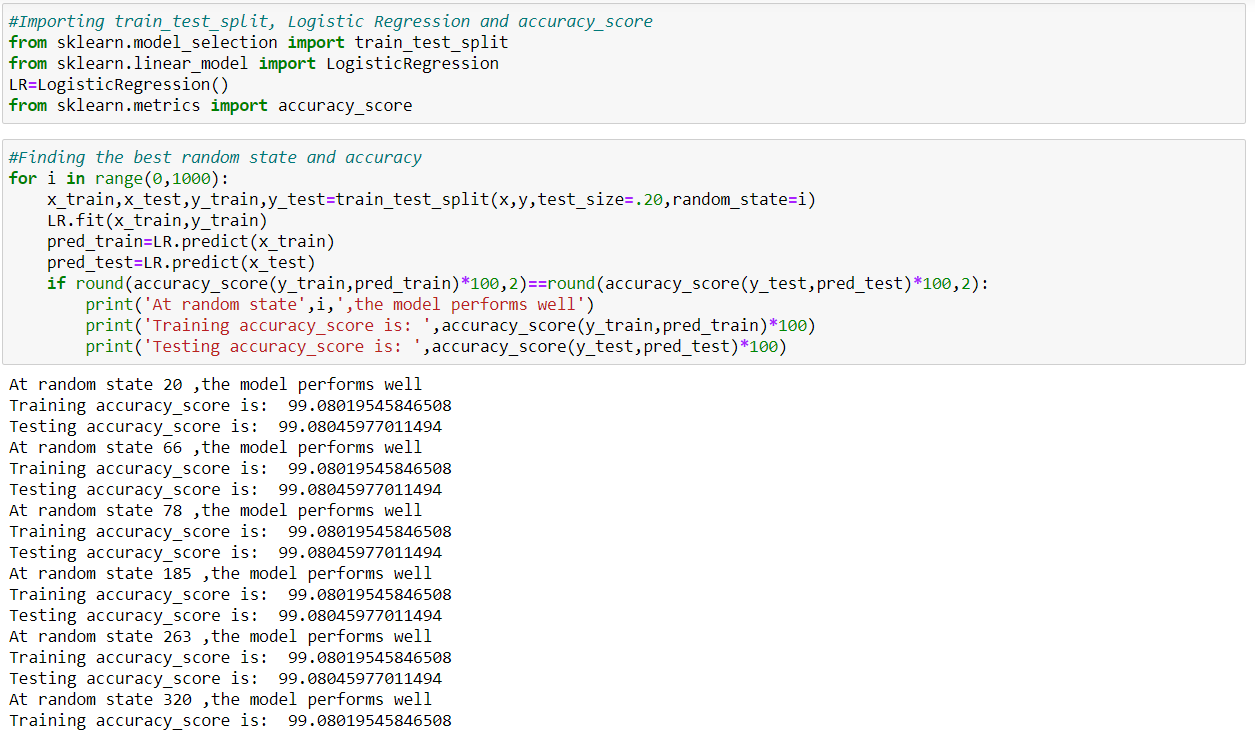
**Standard Scaler:** It performs the task of Standardization. Usually, a dataset contains variables that are different in scale. It will **normalize the feature,** i.e., each column of X, **individually,** so that each column/feature/variable will have μ = 0 and σ = 1.



We will scale the data first and then pass the data into a new Data Frame for further analysis.

**5. Building the Machine Learning models:**

After scaling the data, we can proceed with model building process. First, we will import the metrices and model required for finding the best random state according to the training and testing accuracy scores.



We will run a for loop with range 0-1000 and then create a train\_test\_split with the random states and then we will fit the model, predict the scores according to the if condition mentioned. After running the loop, we can see that at random\_state=20, the best scores are obtained so that we can use in our train\_test\_split.

#Creating train\_test\_split using best random\_state

x\_train, x\_test, y\_train, y\_test =train\_test\_split(x,y,random\_state=20,test\_size=.20,stratify=y)

**Finding out the best model:**

We will be using classification models like Logistic Regression, SVC, GaussianNB, DecisionTreeClassifier and KNeighborsClassifier algorithms for finding out the best model among those. Below are the definitions of the models mentioned:

**1. Logistic regression** - is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

**2.** **SVC** - “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. It is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving a SVM model sets of labeled training data for each category, they're able to categorize new text.

**3.** **Gaussian Naive Bayes** - algorithm is a special type of NB algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e., normal distribution. A Gaussian classifier is a generative approach in the sense that it attempts to model class posterior as well as input class-conditional distribution. Therefore, we can generate new samples in input space with a Gaussian classifier.

**4.** **Decision Tree Classifier** - Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

**5. KNeighbors** **Classifier** - It is a method based on k-nearest neighbors. In the KNeighbors model target is predicted by local interpolation of the targets which associated to the nearest neighbors in the training set. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification).

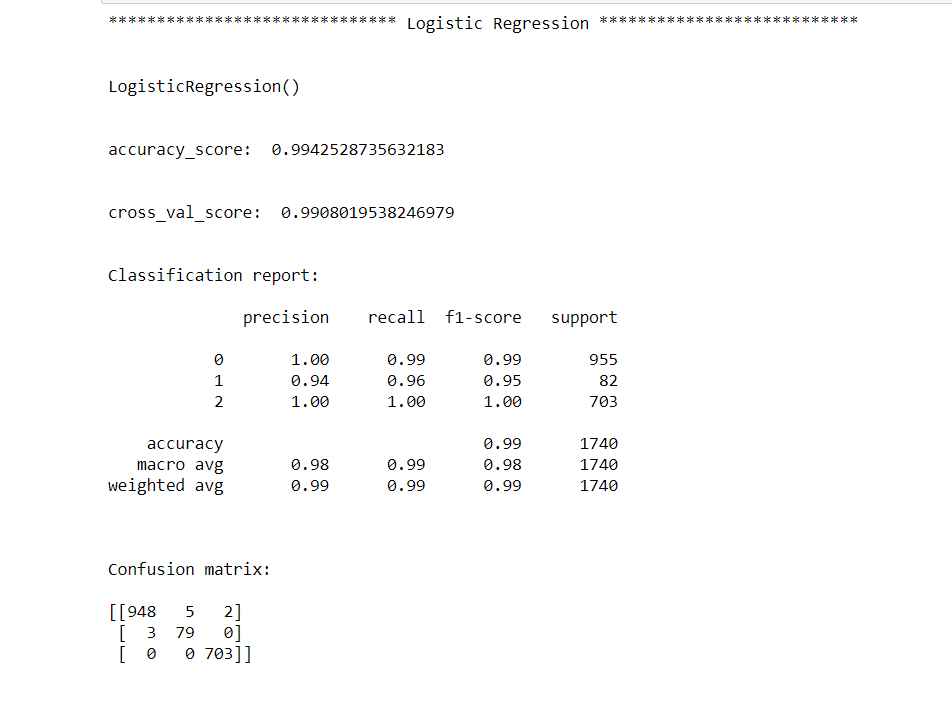
Importing all necessary libraries and metrices required

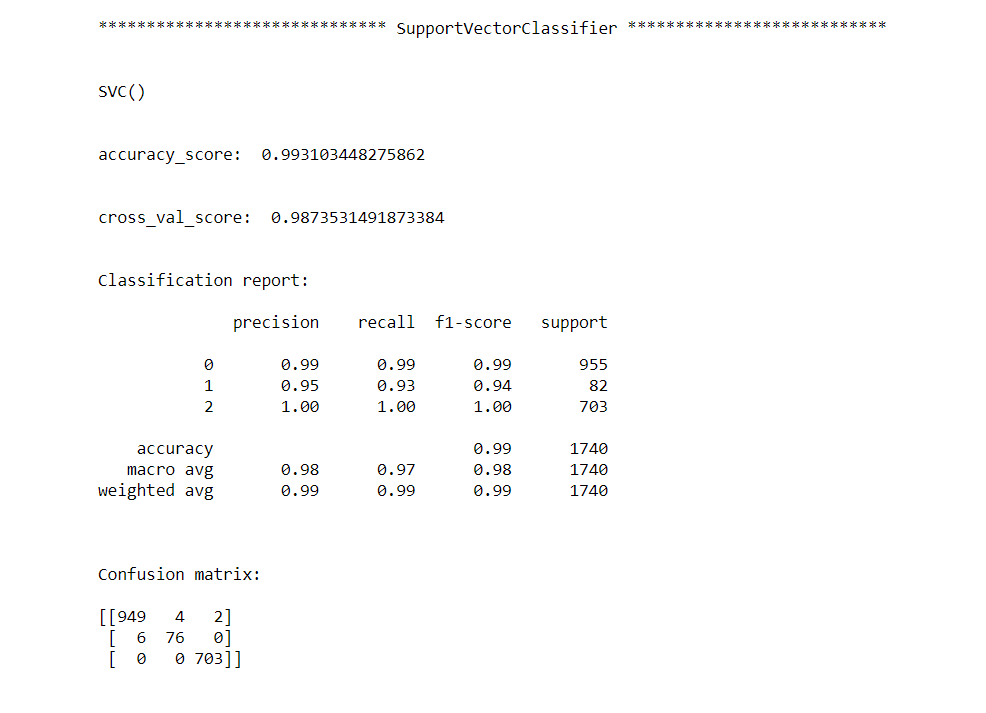


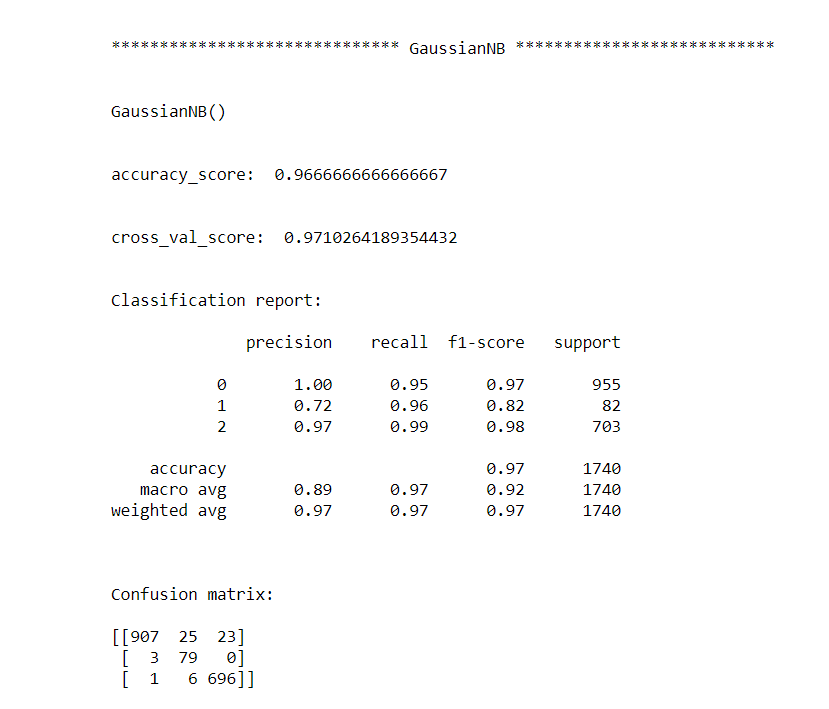
We will be using a for loop for checking the performance metrics of all algorithms together

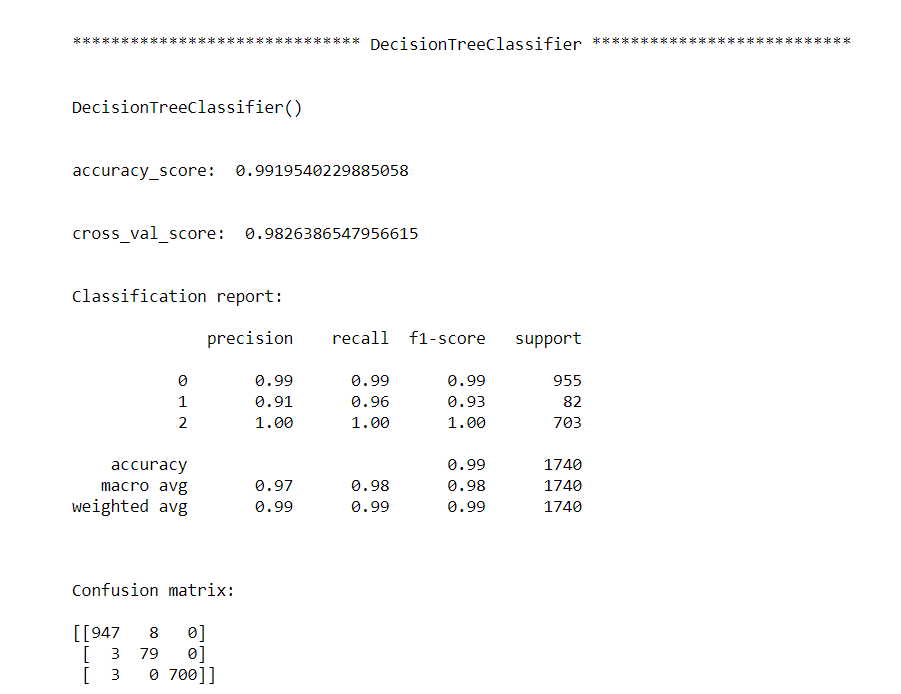


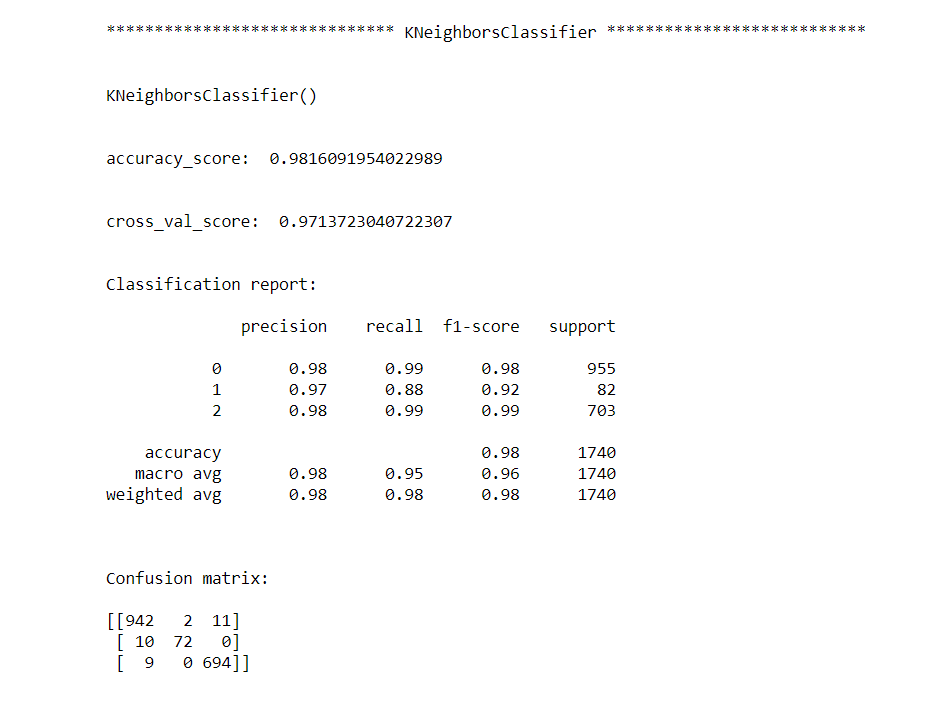
After running this loop, the following outputs will be obtained:











**Definitions of the metrices:**

**1. Cross Validation:**

Cross-validation helps to find out the over fitting and under fitting of the model.In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

**2. Confusion Matrix:**

 A **confusion matrix**, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a **matching matrix**). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabeling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

**3. Classification Report:**

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive

**Precision:** Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below:

Precision = TP/ (TP + FP)

**Recall:** Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below:

Recall = TP/(TP+FN)

**F1 score:** The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is:

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**Support:** Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn’t change between models but instead diagnoses the evaluation process.

From the above algorithms, Logistic Regression, SVC and DecisionTreeClassifier algorithms are working well as they are giving good accuracy score and cross validation score. Now we will try to find out the best parameters and improve our scores by using Hyperparameter Tuning technique.

**Hyperparameter Tuning:**

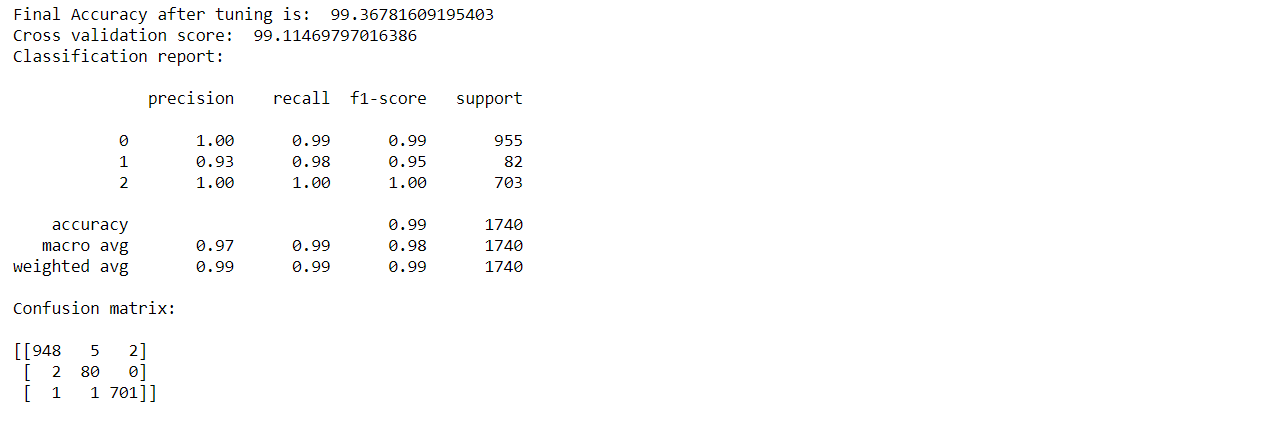
There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.**These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

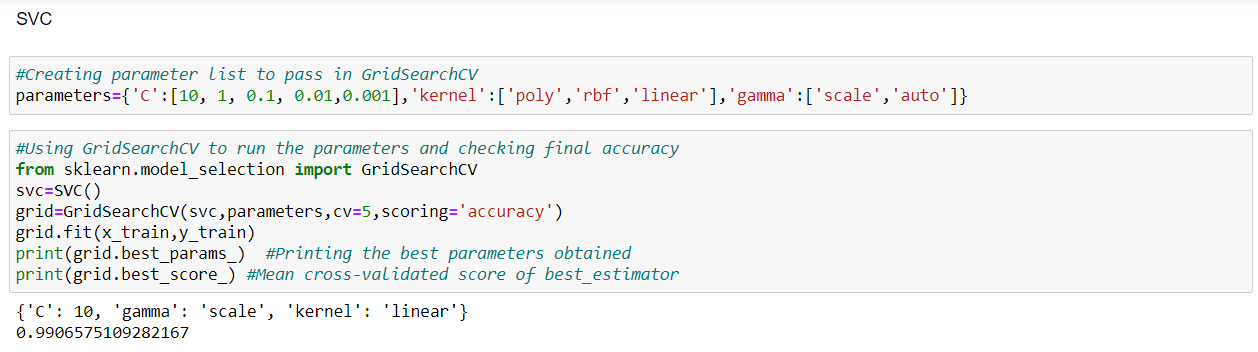
We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as**Hyperparameter Tuning. We can do tuning by using GridSearchCV.**

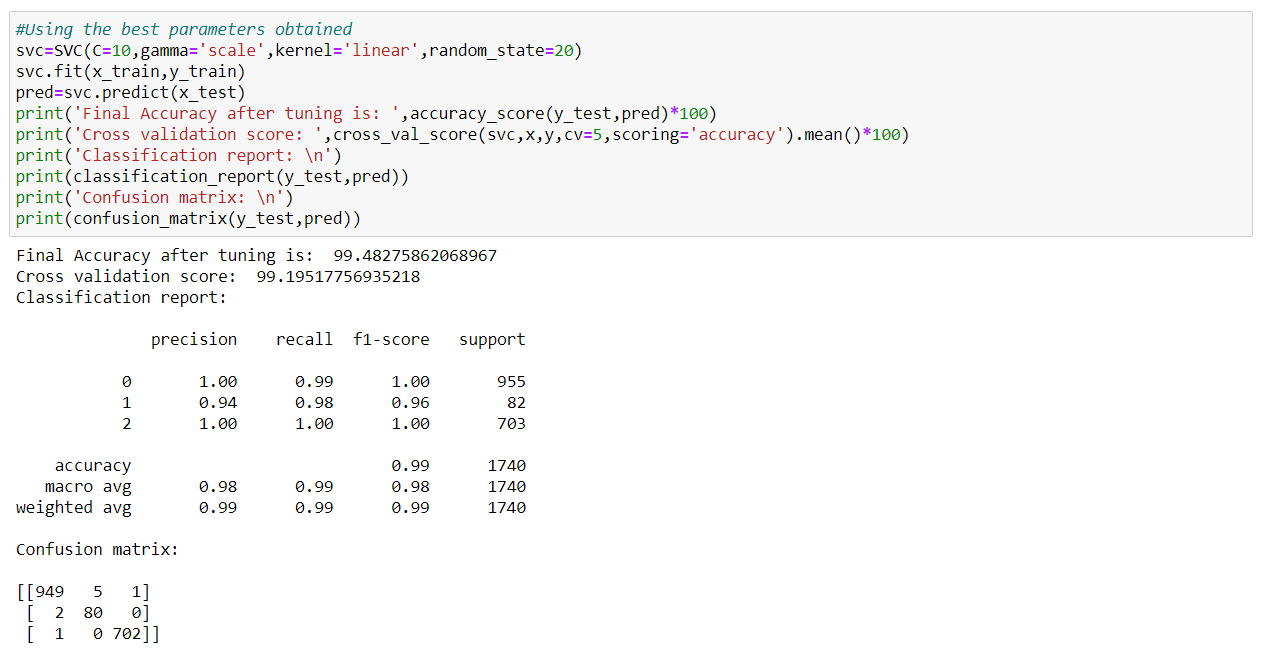
GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Following are the hyperparameter tuning applied to the best algorithms we obtained, i.e., Logistic Regression, SVC and DecisionTreeClassifier:

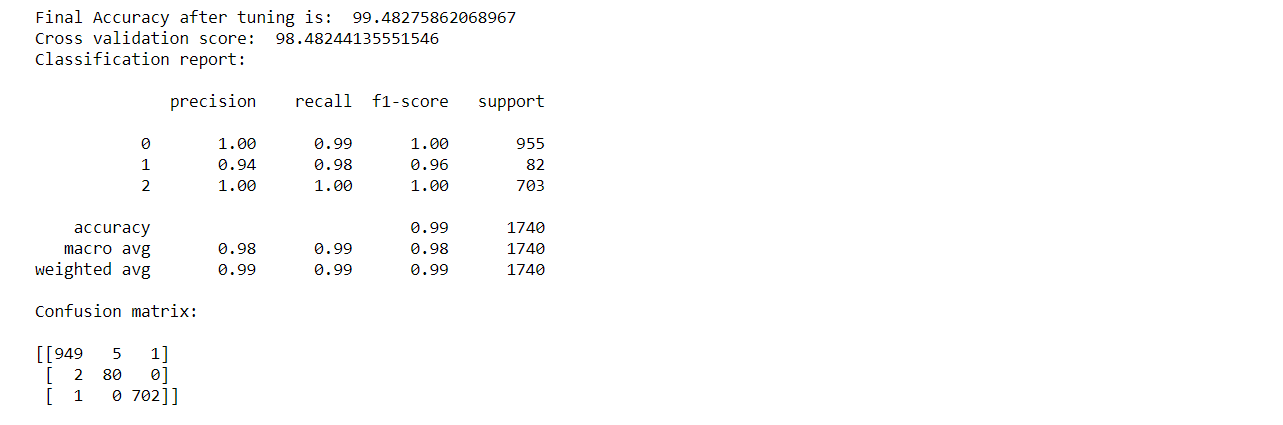












After using Hyperparameter Tuning on the best 3 algorithms we found, we can see that SVC algorithm is the best working algorithm as the scores have been increased, i.e., accuracy score has been increased from 99.31 to 99.48 and cross\_val\_score has been increased from 98.73 to 99.19 and also the metrics values have been improved. We will also try Ensemble Techniques to boost up our scores.

**Ensemble Techniques:**

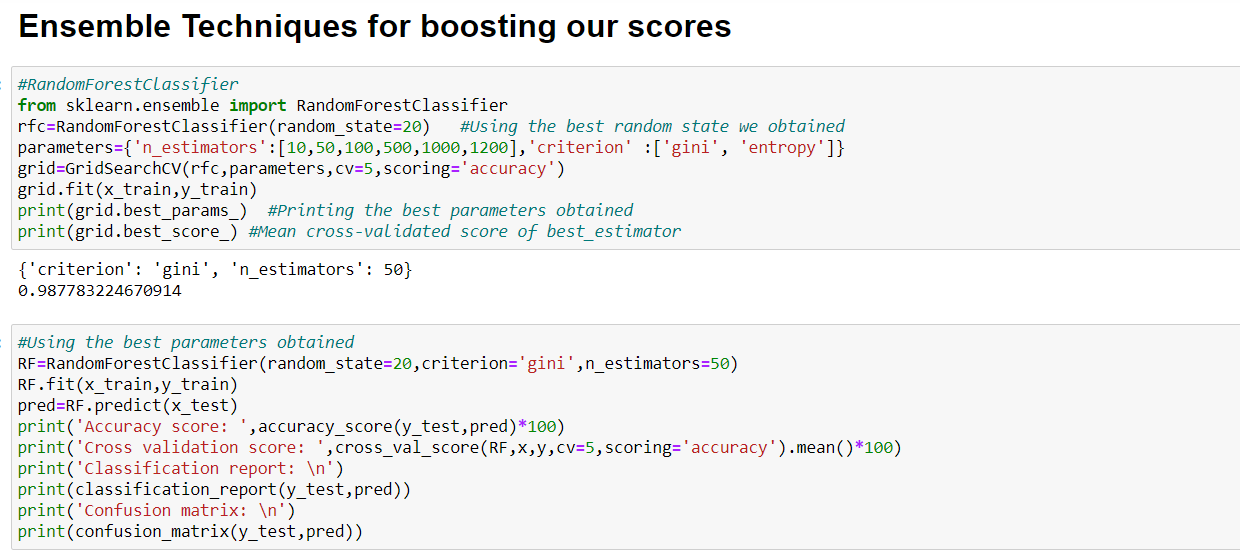
**Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model. Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at each split, and make a final predictor based on the aggregated results of the sampled Decision Trees.

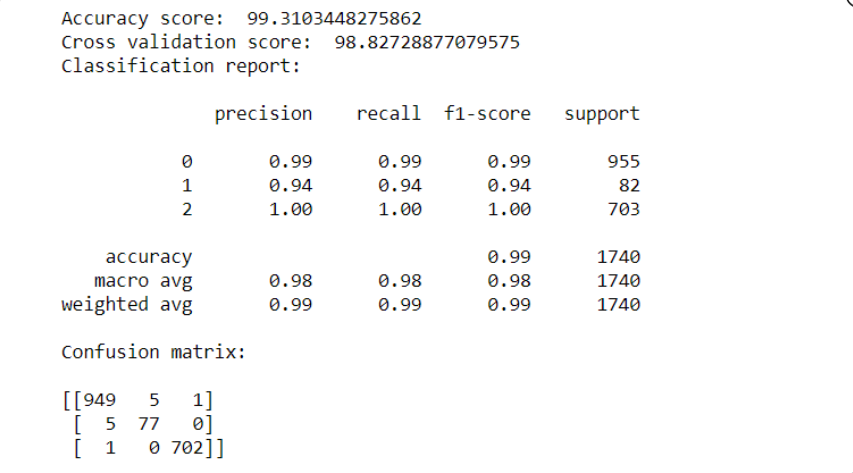
We used Random Forest Classifier, AdaBoost Classifier and Gradient Boosting Classifier algorithms to boost up our scores and finding the best model.

**1. Random Forest:**

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

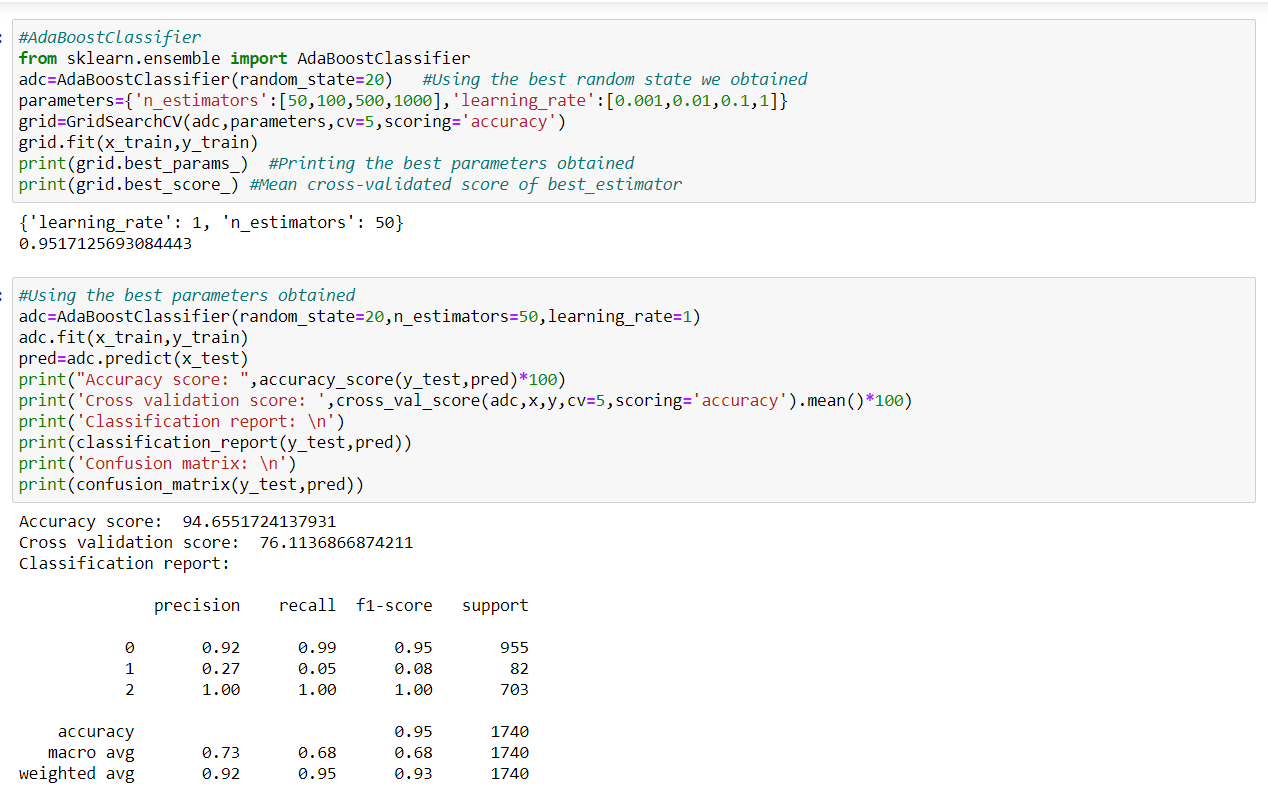
Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max\_features.





**2. AdaBoost:**

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.





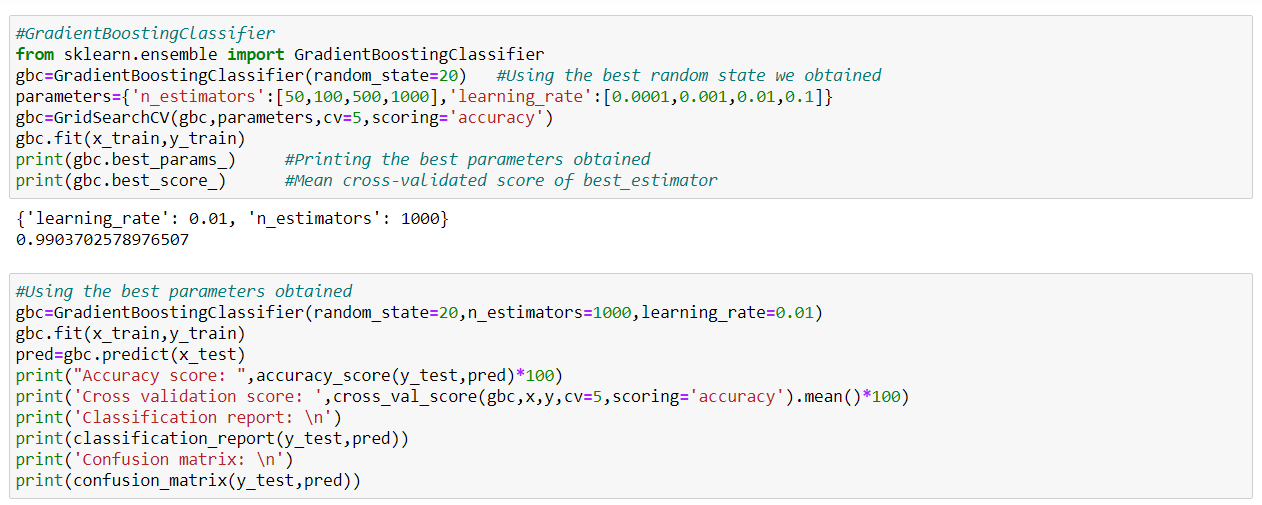
**3. Gradient Boosting algorithm:**

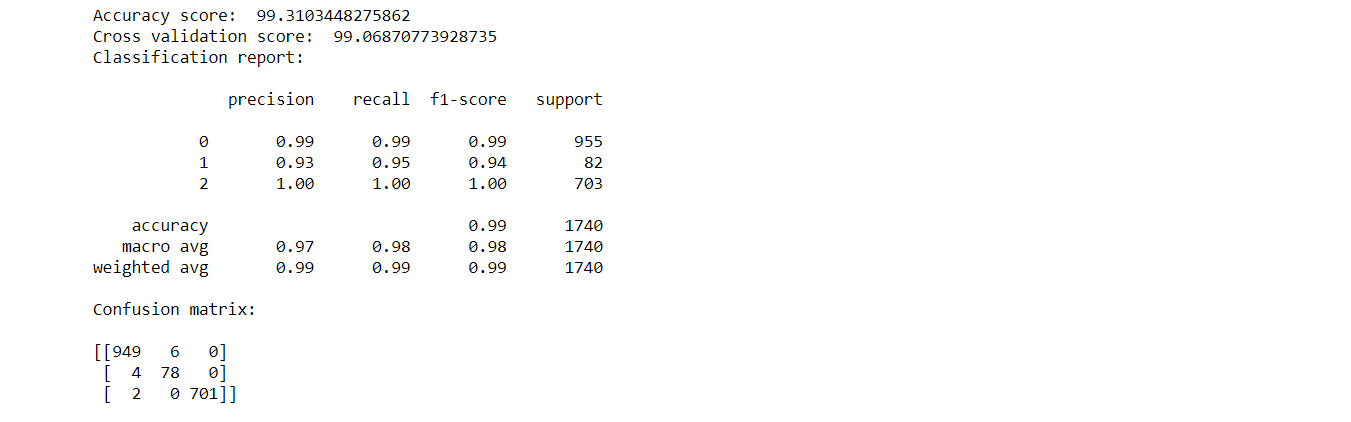
Gradient boosting is a [machine learning](https://data-flair.training/blogs/machine-learning-tutorial/)technique for regression and classification problems. That produces a prediction model in the form of an ensemble of weak prediction models.

The accuracy of a predictive model can be boosted in two ways:

a. Either by embracing feature engineering or

b. By applying boosting algorithms straight away.





After applying the Tuning and Ensemble Techniques, we can say that SVC is the best algorithm we got as the accuracy score is 99.48 and cross validation score is 99.19. We will now finalize the model by saving the predictions we got, saving the best model and also creating a new Data Frame for saving the predictions:



**6. Concluding Remarks:**

-> After exploring the SDSS data, we came to know that how SDSS image is working and how to navigate and plot the sky image using camera data.

-> First, we loaded the dataset and did the EDA process and other pre-processing techniques like skewness check and removal, handling the outliers present, filling the missing data, visualizing the distribution of data, etc.

-> Then we did the model training, building the model and finding out the best model on the basis of different metrices scores we got like Classification Report, Confusion matrix, cross validation score, etc.

-> We got SupportVectorClassifier Algorithm, Logistic Regression and Decision Tree Classifier as the best algorithms among all as the accuracy score was 99% and cross validation score was between 98-99%. Then for finding out the best parameter and improving the scores, we performed Hyperparameter Tuning in all these 3 algorithms.

-> After Tuning, we got that SVC algorithm is the best working algorithm as the scores have been increased, i.e., accuracy score has been increased from 99.31 to 99.48 and cross\_val\_score has been increased from 98.73 to 99.19 and also the metrics values have been improved.

-> We also tried Ensemble Techniques to boost up the scores and we used algorithms like Random Forest Classifier, AdaBoost Classifier and Gradient Boosting Classifier.

-> Finally, we concluded that SVC was the best algorithm among all other algorithms as its scores were good and we finalized the model. We saved the predicted values in a new data frame and also, we saved the model too.

-> Overall, we can say that this dataset is an excellent dataset for predicting the category in the space as the average accuracy of the dataset was nearly 97% and the error rate was minimal too, which is very good and also the best model can be used again and again during further deployment process.

**GitHub Link:** <https://github.com/Jerish7/Projects-M20/blob/main/SDSS_Project.ipynb>