PROJECT REPORT

On

Machine Learning based celestial objects classifier

By

Jyothsana M Rao

Under the guidance of

Mr. Rajeev Ranjan

Assistant Vice President, Genpact

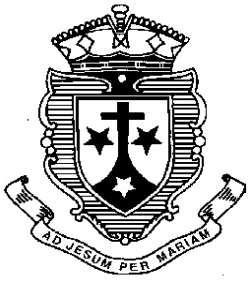
Submitted in partial fulfilment of the requirements of

Bachelor of Vocation (Analytics) course

DEPARTMENT OF STATISTICS AND ANALYTICS

MOUNT CARMEL COLLEGE, AUTONOMOUS,

NO. 58, PALACE ROAD, BENGALURU – 560052



COLLEGE CERTIFICATE

This is to certify that the students

1. _____ **(Register Number :_____)**

of the **BACHELOR OF VOCATION, B.Voc (ANALYTICS)** course, batch (2017 - 2020),

have successfully completed the project titled:

_____

in their sixth semester as prescribed by the Advisory Committee and the Board of Studies of the B.Voc (Analytics) course, Mount Carmel College, Autonomous, Bengaluru, affiliated to **Bangalore University**.

Internal Guide **Project Guide**

Date

Examiners: **College Seal**

1. _____

2. _____

MACHINE LEARNING BASED CELESTIAL OBJECTS CLASSIFIER

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude to all those who helped me complete this project.

My sincere gratitude to Dr.Sr.Arpana , Principal, Mount Carmel College for this opportunity.

A big thank you to Mr. Rajeev Ranjan, Assistant Vice President, Genpact, for patiently mentoring me and guiding me on this project.

I would also like to extend my gratitude to Mr. Nagarajan and Mr. Anirudh Acharya for helping me understand the physics associated with cosmology.

Special thanks to Dr. S.K. Lakshmi, Associate Professor, Department of Statistics and Analytics for her constant encouragement and all the faculty of the Department of Statistics and Analytics without whom this project would not be possible.

In addition, I would like to express my deepest appreciation to Dr. Rajaranjan, Principal Data Scientist, Accenture, for all his valuable guidance and support both in the project and in the course.

Furthermore, I would also like to acknowledge Ms. Shanmuga Priya.J, lab administrator, for always being there in the lab when we needed it.

I express my deepest gratitude to my parents and siblings for helping me out in different ways.

I am grateful to all those who have helped me in some way or the other to complete this project.

ABSTRACT

This project offers a classification method that not only classifies various cosmic objects with excellent accuracy but does so with good precision. Comparisons were made between four classification techniques , two of them being ensemble methods. The classifier is specific to Stars, Galaxies and Quasars.

This project showcases the use of data analysis techniques like Exploratory Data Analysis (EDA) and Information value (IV).

Out of the four classification models developed, Random Forest generated the best results both in terms of the overall model accuracy and class level measures.

TABLE OF CONTENTS

1. SUMMARY OF 10 WORKING WEEKS

WEEK 1

- Data collection
- Project objective was set

WEEK 2

- Work was done on understanding the data and preparing the data dictionary

WEEK 3 & 4

- Exploratory Data Analysis

WEEK 5

- Feature selection techniques – Information Value and weight of evidence

WEEK 6

- Preparation for mid semester project presentation

WEEK 7

- Data pre-processing for model building
- Model building – Multinomial Regression

WEEK 8

- Model building – Decision Tree and Random Forest

WEEK 9 & 10

- Model Building – XGBoost Classifier
- Reporting

2. INTRODUCTION

”Somewhere, something incredible is waiting to be known.”

- Carl Sagan

This project focuses on automated classification of astronomical objects (stars, quasars and galaxies) from data release 16 of the Sloan Digital Sky Survey (SDSS), using various machine learning classification techniques.

The need for identifying and classifying Stars, Quasars and Galaxies is to help aid in the discovery of potential Earths for human life and its dependencies to sustain on. These planets are believed to be revolving around other sun like stars either in our own galaxy or in other galaxies. After years of study and research we have come to an understanding that the Earth might not protect us forever. Like all good things, our planet too will one day meet her end. Scientists all around the world have made it their life mission to discover Earth like planets that will allow life to exist.

This project aims at building a model that not only classifies the astronomical objects accurately but does so with good precision, so as to help the concerned individuals to optimally allocate their time and resources in studies that will ultimately yield productive results.

3. OBJECTIVE

The main objective of this project is to develop a machine learning algorithm to help astronomers accurately classify stars, quasars and galaxies.

4. VARIABLE UNDERSTANDING

4.1 VARIABLE DESCRIPTION

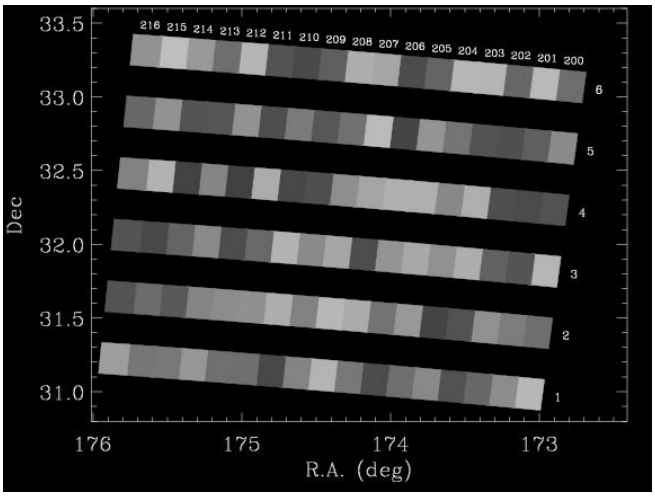
VARIABLE	DESCRIPTION
objID	Object id in the image catalogue used by CAS (Catalogue Archive Server).
Ra & Dec	RA (right ascension) and DEC (declination) are to the sky what longitude and latitude are to the surface of the Earth. RA corresponds to east/west direction (like longitude), while Dec measures north/south directions, like latitude.
u,r,g,i,z	These are SDSS colour filters. u – ultraviolet r – red g – green i – near infrared z – infrared
Run	A Run is a length of a strip observed in a single continuous image observing scan. A strip covers a great circle region from pole to pole; this cannot be observed in one pass. The fraction of a strip observed at one time is a Run.
Rerun	The reprocessing of an imaging run.
Camcol	A Camcol is the output of one camera column of CCDs (each with a different filter) as part of a Run. Therefore, 1 Camcol = 1/6 of a Run.

Fields	A field is a part of a camcol that is processed by the Photo pipeline at one time. Fields are 2048 × 1489 pixels; a field consists of the frames in the 5 filters for the same part of the sky.
SpecObjID	A unique bit-encoded 64-bit ID used for optical spectroscopic objects. It is generated from plateid, mjd, and fibreid.
Plate	These are large aluminium plates into which tiny holes are drilled. Each hole has an optical fibre plugged into it. Each hole corresponds to the sky location where there is an object (a star or a galaxy) which SDSS wants to measure a spectrum for.
MJD	Modified Julian Date, used to indicate the date that a given piece of SDSS data (image or spectrum) was taken.
fibreID	The SDSS spectrograph uses optical fibres to direct the light at the focal plane from individual objects to the slit head. Each object is assigned a corresponding fibreID.
Redshift	Red shift' is a key concept for astronomers. The term can be understood literally - the wavelength of the light is stretched, so the light is seen as 'shifted' towards the red part of the spectrum.
Class	It is the dependent variable and consists of three classes; **Star**: A star is an astronomical object consisting of a luminous spheroid of plasma held together by its own gravity. **Galaxy**: A galaxy is a gravitationally bound system of

	stars, stellar remnants, interstellar gas, dust, and dark matter. **Quasar**: Quasar is an astronomical object of very high luminosity found in the centres of some galaxies and powered by gas spiralling at high velocity into an extremely large black hole.

4.2 GEOMETRY OF A RUN

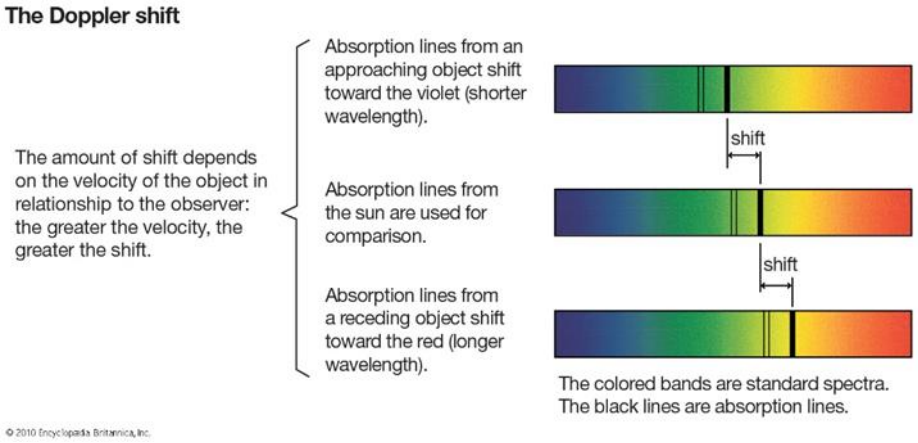
- The Sloan Digital Sky Survey imaging data set is built up from a series of drift scan runs, ranging from an hour to over eight hours in length.
- During each run, images are taken in six long, narrow camcols on the sky.
- Each camcol is divided into individual fields.
- In pixel units, each field is 2048 columns by 1489 rows, with 128 rows overlap with the previous and next field.
- For each field, there is an image in each of the five bands (u, g, r, i, and z) taken within a few minutes of each other.



Visualization of a SDSS run

4.3 UNDERSTANDING REDSHIFT

According to Encyclopaedia Britannica, Redshift is the displacement of the spectrum of an astronomical object toward longer (red) wavelengths. It is generally attributed to the Doppler effect, a change in wavelength that results when a given source of waves (e.g., light or radio waves) and an observer are in rapid motion with respect to each other.



An infographic on redshift

5. ABOUT THE DATA

The dataset used in this project belongs to the Sloan Digital Sky Survey (SDSS) program. It is the 16th data release (DR16) of SDSS.

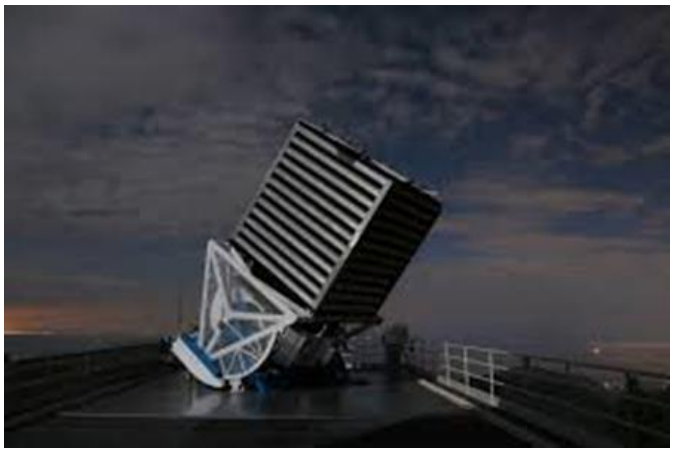
Data Release 16 (DR16) is the fourth data release of the fourth phase of the Sloan Digital Sky Survey (SDSS-IV). DR16 contains SDSS observations through August 2018.

The Sloan Digital Sky Survey has created the most detailed three-dimensional maps of the Universe ever made, with deep multi-colour images of one third of the sky, and spectra for more than three million astronomical objects.

SDSS uses a dedicated 2.5 m wide-angle optical telescope; from 1998 to 2009 it observed in both imaging and spectroscopic modes. The imaging camera was retired in late 2009, since then the telescope has observed entirely in spectroscopic mode.

Images were taken using a photometric system of five filters (named u, g, r, i and z). These images are processed to produce lists of objects observed and various parameters, such as whether they seem point like or extended (as a galaxy might) and how the brightness on the *Charged-Coupled Devices (CCDs)* relates to various kinds of astronomical magnitude.

The data set acquired did not have any missing values.



Optical Telescope used by SDSS

6. EXPLORATORY DATA ANALYSIS

6.1 UNIVARIATE ANALYSIS

	count	mean	std	min	25%	50%	75%	max
ra	100000.0	177.512888	78.039070	0.013061	136.356526	180.411688	224.369107	359.999615
dec	100000.0	25.052056	20.567259	-19.495456	6.770380	23.918611	40.344539	84.490494
u	100000.0	18.637915	0.832284	10.611810	18.212902	18.873250	19.273302	19.599950
g	100000.0	17.407128	0.985921	9.668339	16.852982	17.515860	18.056060	19.996050
r	100000.0	16.881676	1.133337	9.005167	16.196608	16.890640	17.585750	31.990100
i	100000.0	16.625534	1.209532	8.848403	15.865275	16.599885	17.344912	32.141470
z	100000.0	16.467087	1.281788	8.947795	15.619960	16.428385	17.234625	29.383740
run	100000.0	3978.727640	1691.498597	109.000000	2826.000000	3900.000000	5061.000000	8162.000000
camcol	100000.0	3.274010	1.621208	1.000000	2.000000	3.000000	5.000000	6.000000
field	100000.0	187.243080	141.037298	11.000000	85.000000	153.000000	249.000000	982.000000
redshift	100000.0	0.170621	0.437571	-0.004136	0.000001	0.045997	0.095429	7.011245
plate	100000.0	2587.984270	2210.547391	266.000000	1186.000000	2091.000000	2910.000000	11703.000000

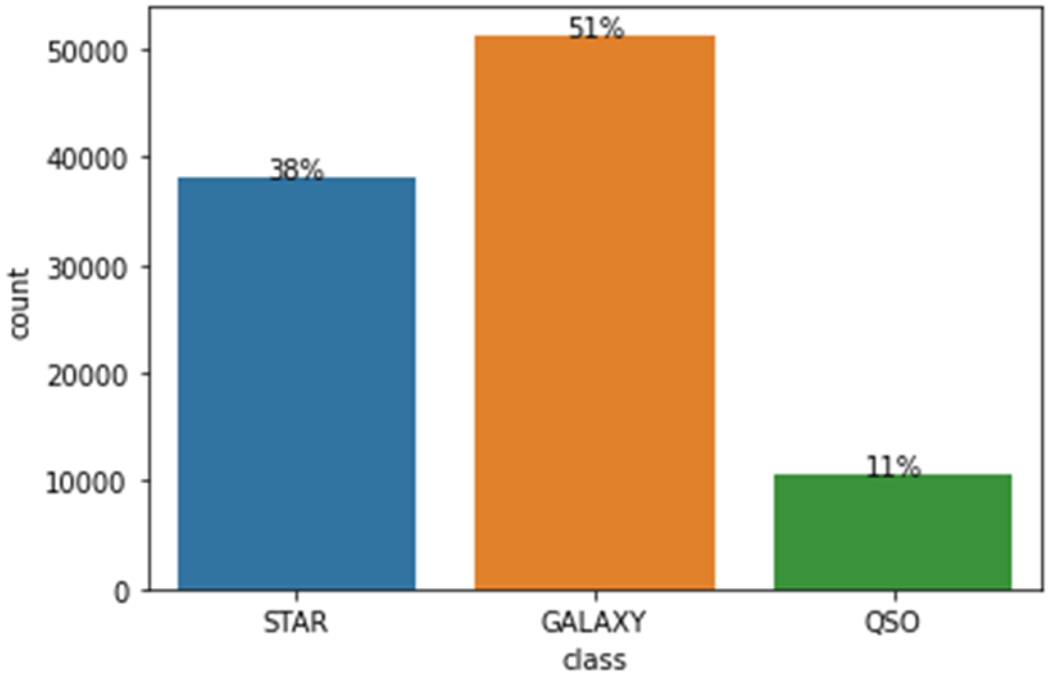
HYPOTHESIS

Most variables show presence of extreme values, indicating the presence of outliers.

Outliers for all variables except the colour filter variables (u,g,r,I,z) were treated.

The colour filter variables are unique to stars, galaxies and Quasars, hence manipulating these variables might result in a loss of information.

6.1.1 DISTRIBUTION OF CLASS IN THE DATA SET



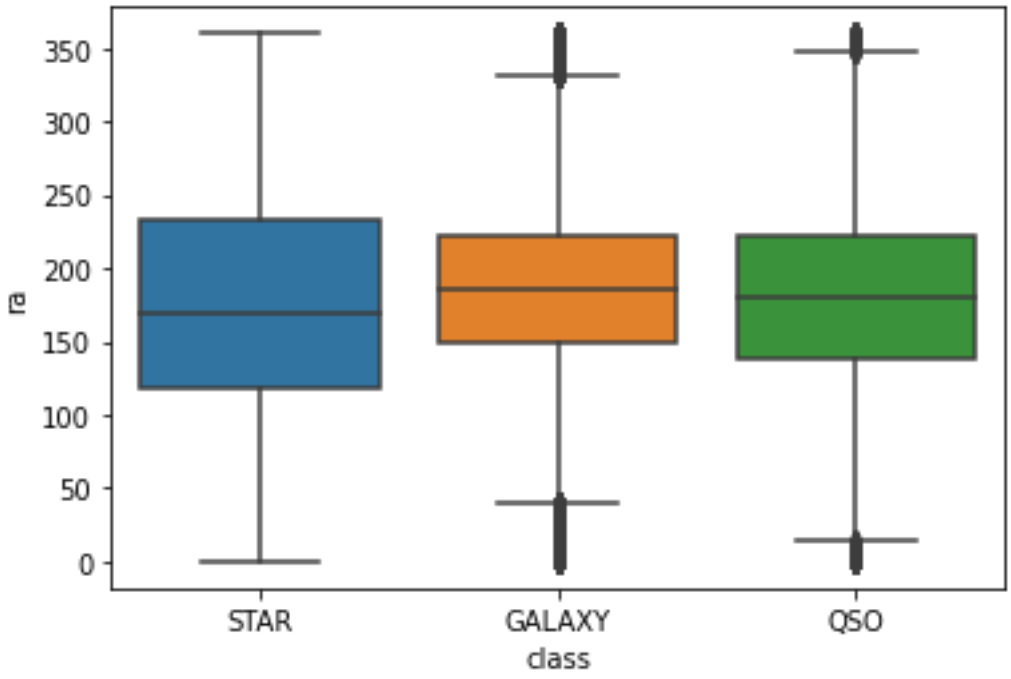
HYPOTHESIS

51% of the records in the data set belong to the class galaxy, 38% belong to the class star and 11% belong to the class quasar.

6.2 BI-VARIATE ANALYSIS

6.2.1 VISUALISING THE DISTRIBUTION OF VARIABLES FOR EACH CLASS USING BOX PLOTS

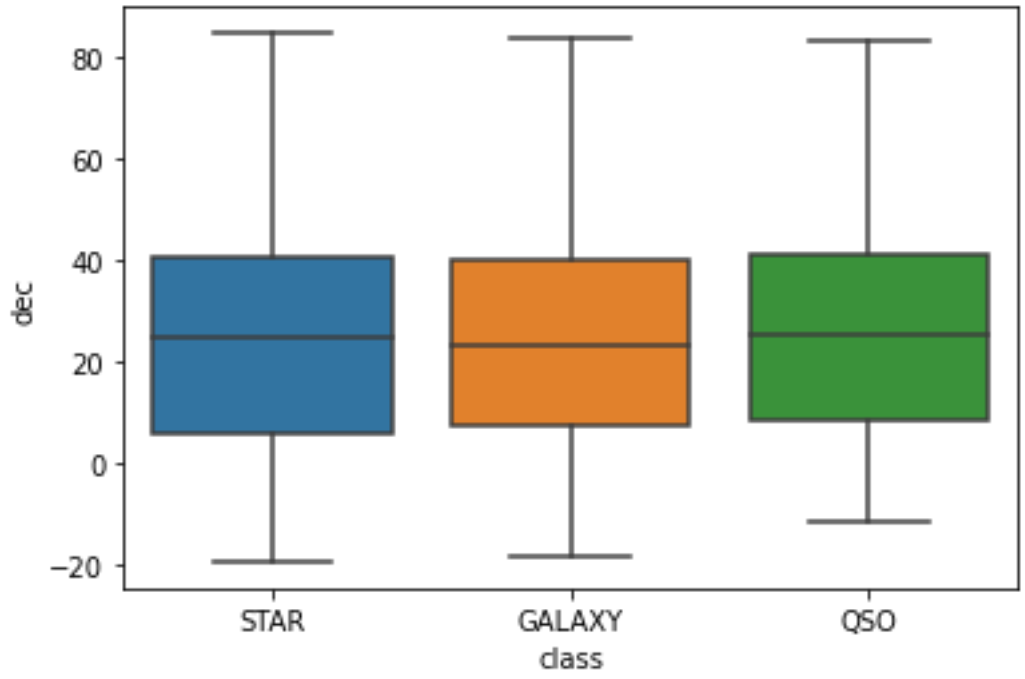
‘ra’ vs ‘class’



HYPOTHESIS

The distribution of ‘ra’ for all the classes is similar, indicating that it’s a poor predictor of the class variable.

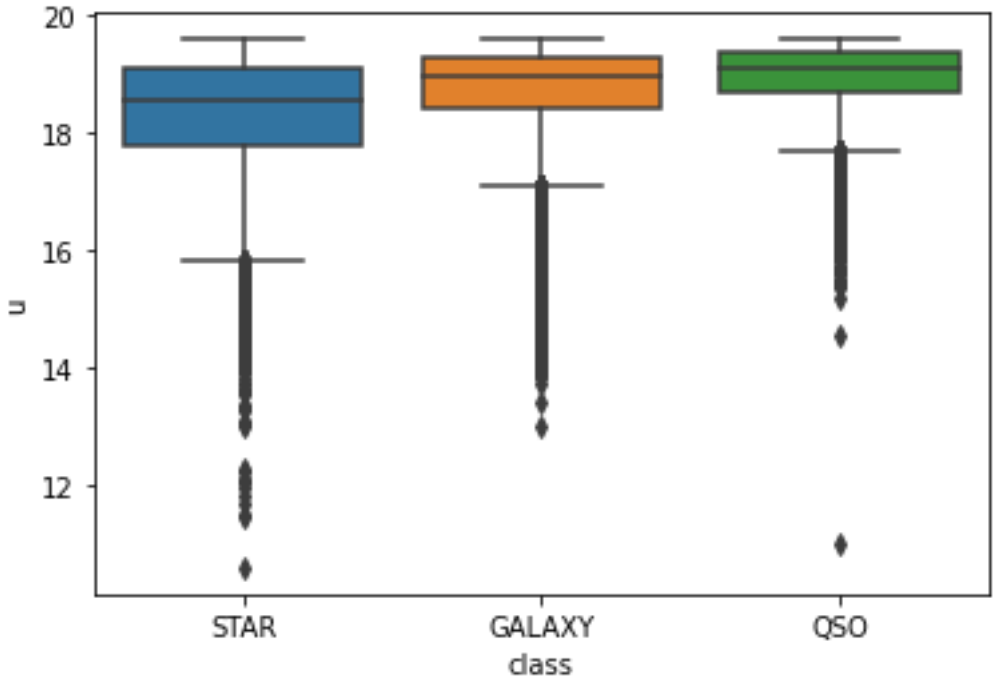
‘dec’ vs ‘class’



HYPOTHESIS

The distribution of ‘dec’ for all the classes is similar, indicating that it’s a poor predictor of the class variable.

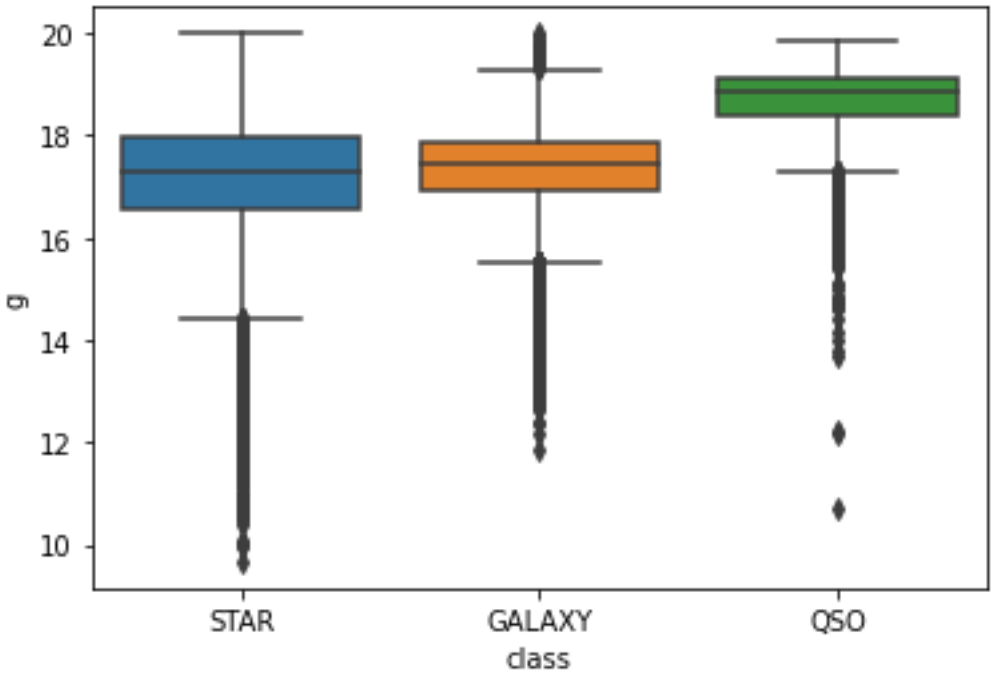
u' vs 'class'



HYPOTHESIS

The distribution of 'u' is slightly different for all the classes, indicating that it's a mild predictor of the class variable.

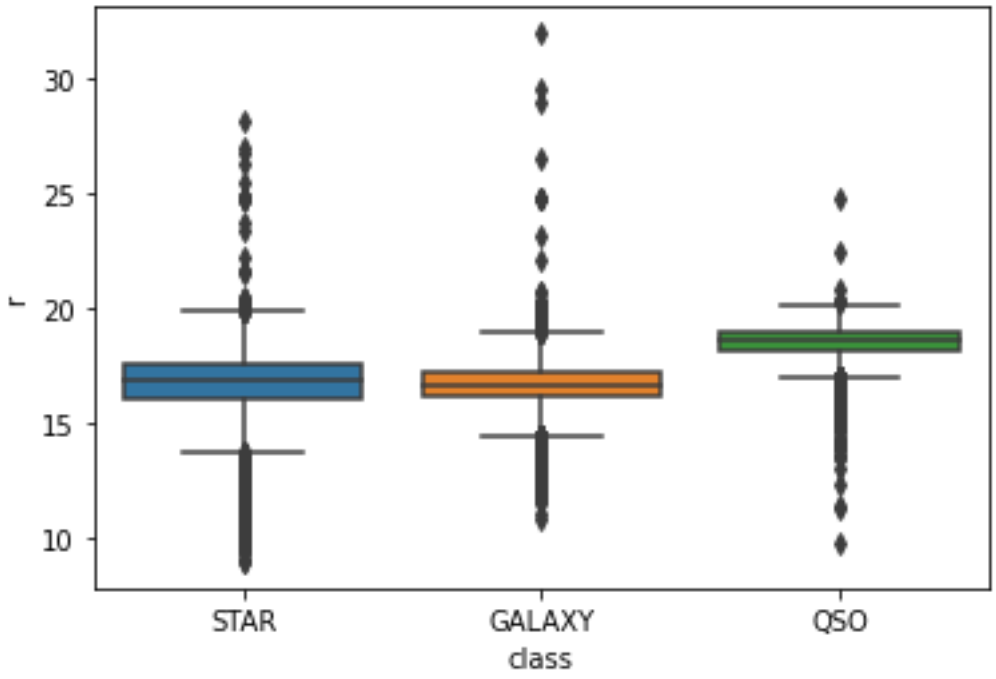
‘g’ vs ‘class’



HYPOTHESIS

The distribution of ‘g’ is slightly different for all the classes, indicating that it’s a mild predictor of the class variable.

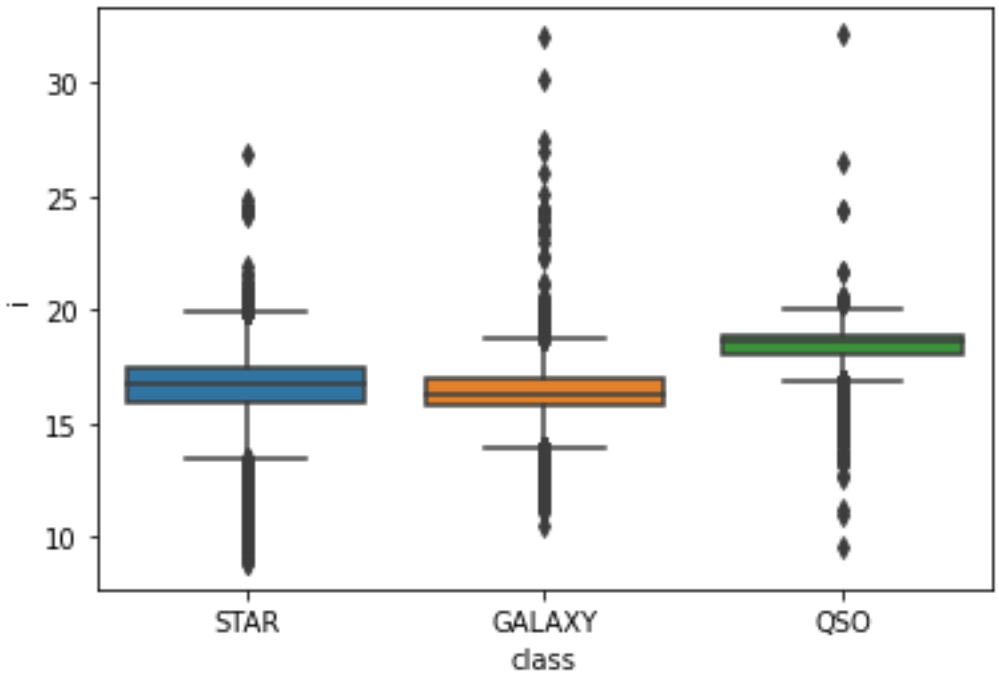
‘r’ vs ‘class’



HYPOTHESIS

The distribution of ‘r’ is slightly different for all the classes, indicating that it’s a mild predictor of the class variable.

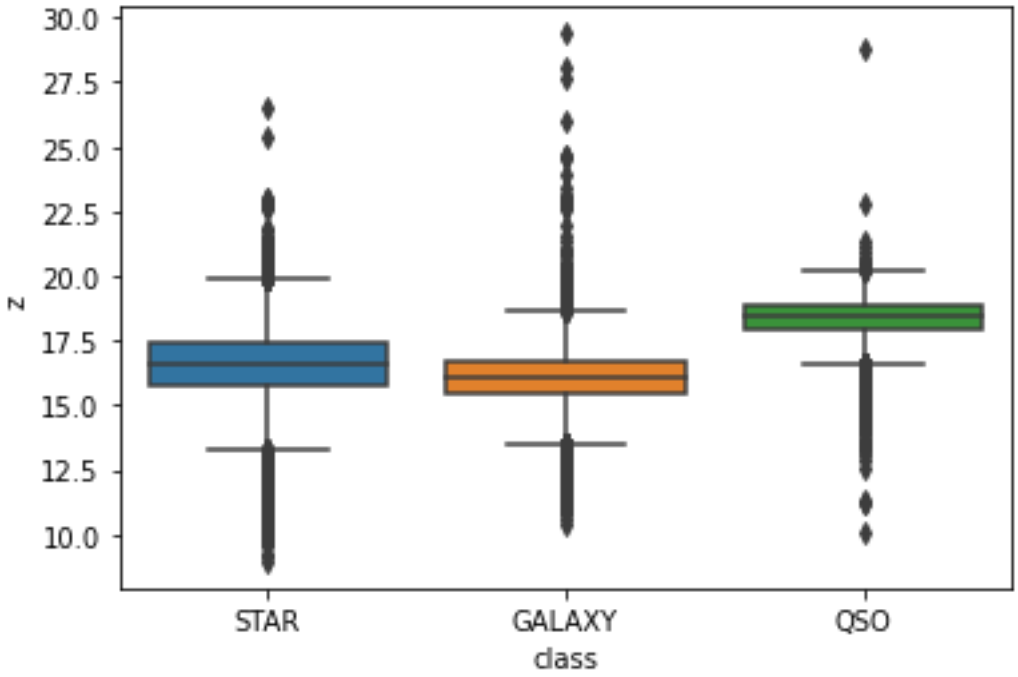
‘i’ vs ‘class’



HYPOTHESIS

The distribution of ‘i’ is slightly different for all the classes, indicating that it’s a mild predictor of the class variable.

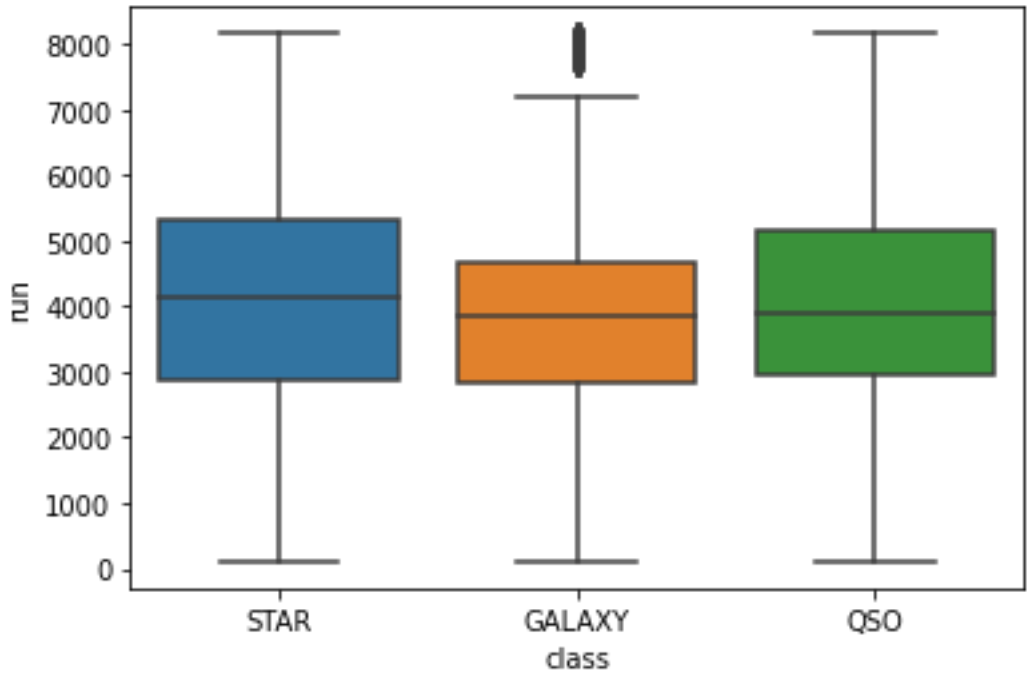
‘z’ vs ‘class’



HYPOTHESIS

The distribution of ‘z’ is slightly different for all the classes, indicating that it’s a mild predictor of the class variable.

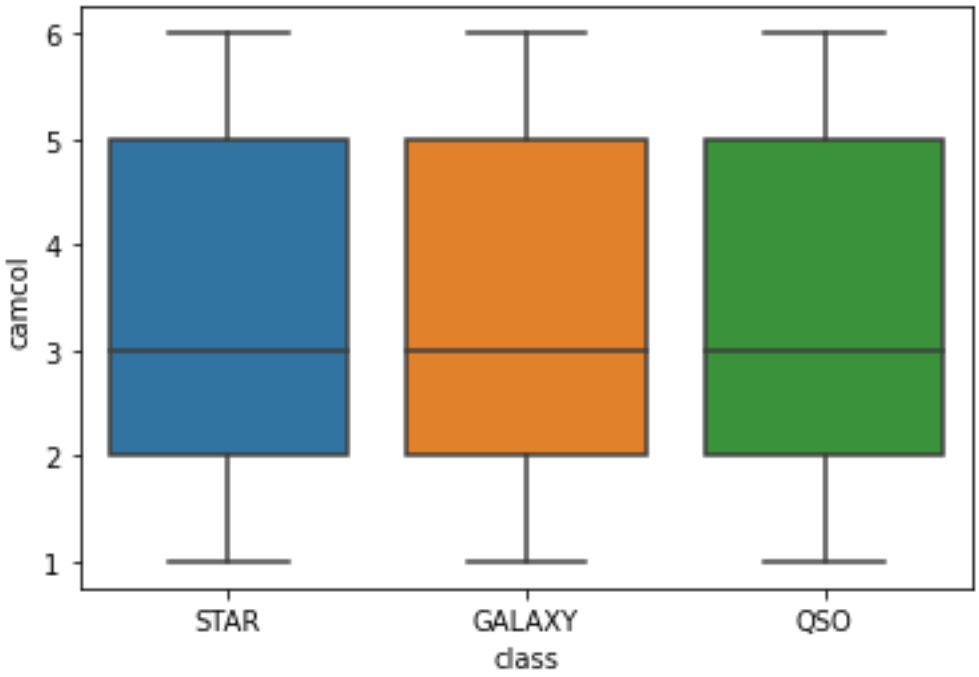
‘run’ vs ‘class’



HYPOTHESIS

The distribution of ‘run’ is similar for all the classes, indicating that it’s a poor predictor of the class variable.

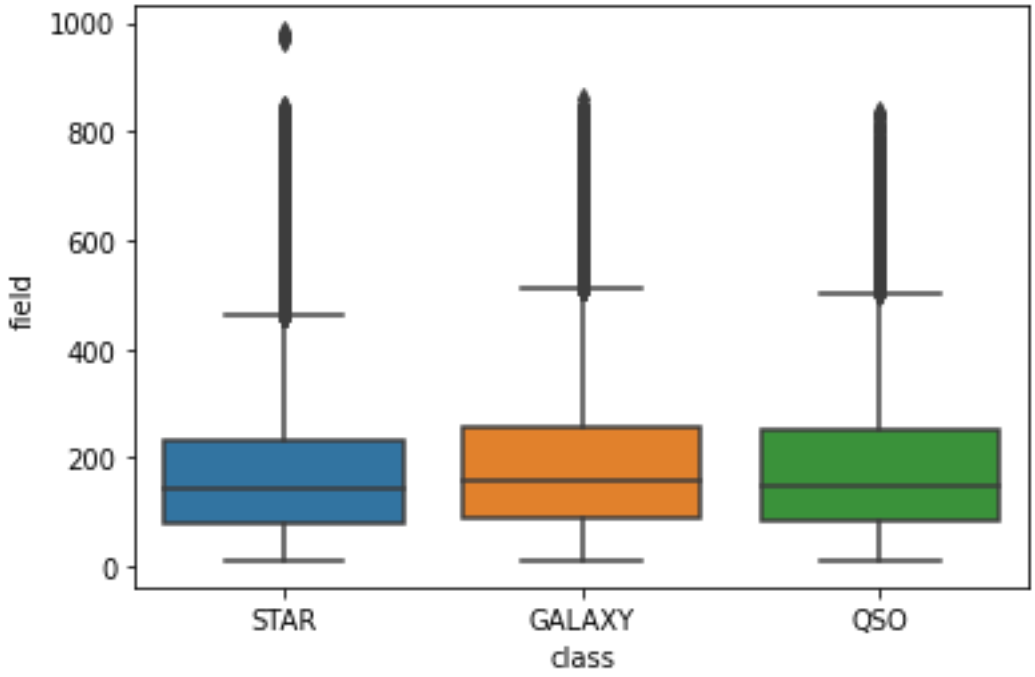
‘camcol’ vs ‘class’



HYPOTHESIS

The distribution of ‘camcol’ is similar for all the classes, indicating that it’s a poor predictor of the class variable.

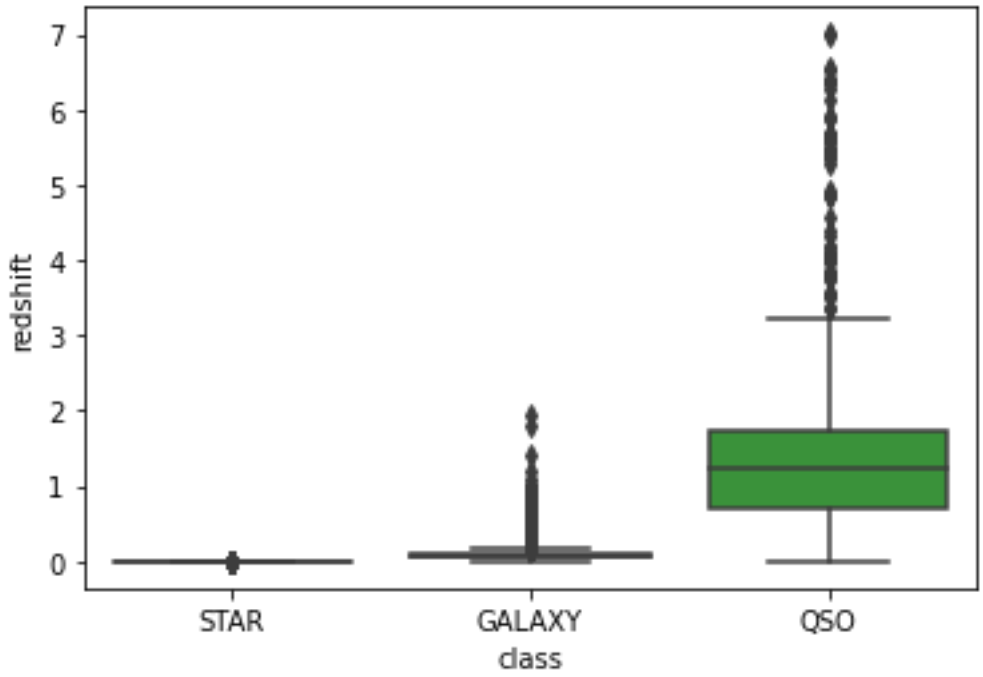
‘field’ vs ‘class’



HYPOTHESIS

The distribution of ‘field’ is similar for all the classes, indicating that it’s a poor predictor of the class variable.

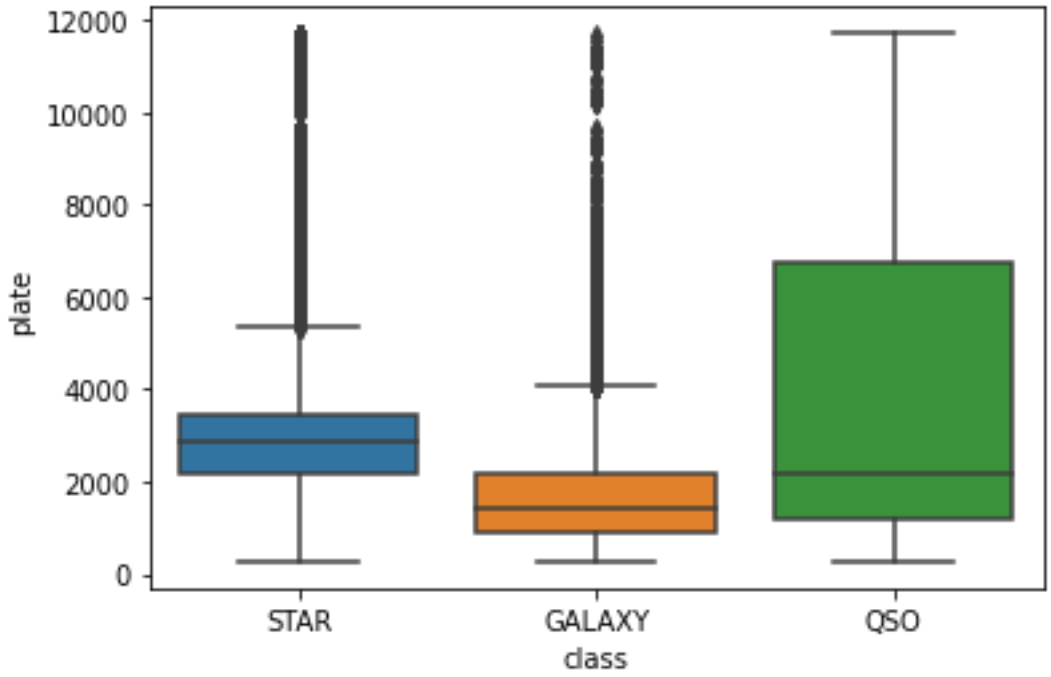
‘redshift’ vs ‘class’



HYPOTHESIS

The distribution of ‘redshift’ is different across all the classes, indicating that it’s a good predictor of the class variable.

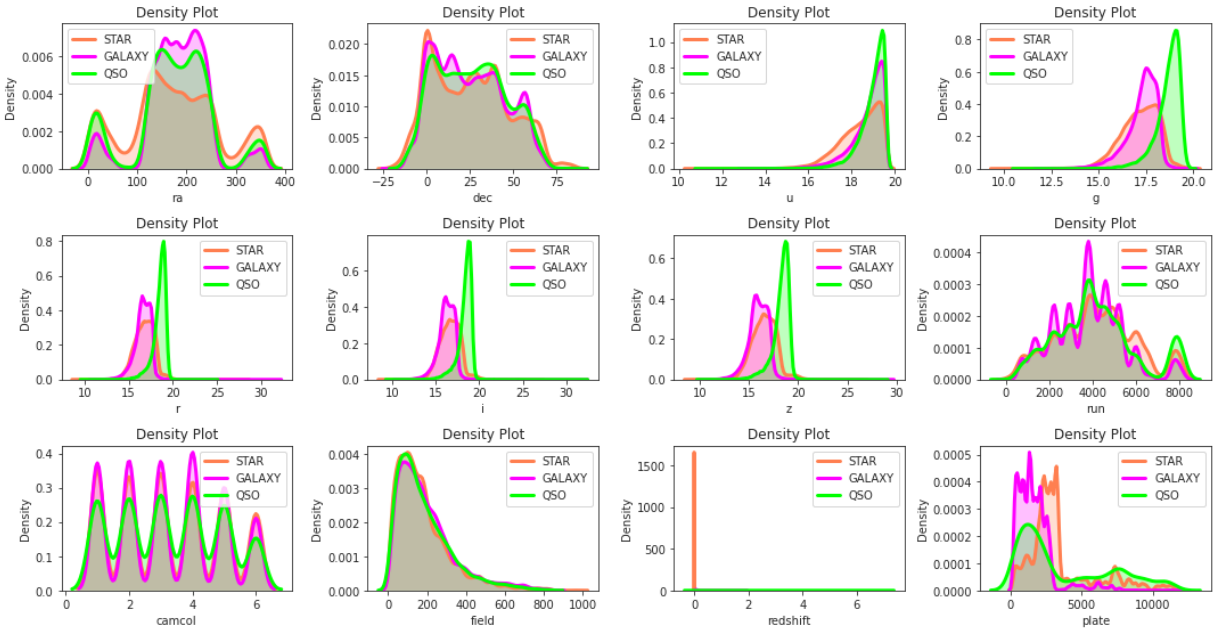
‘plate’ vs ‘class’



HYPOTHESIS

The distribution of ‘plate’ is slightly different across all the classes, indicating that it’s a good predictor of the class variable.

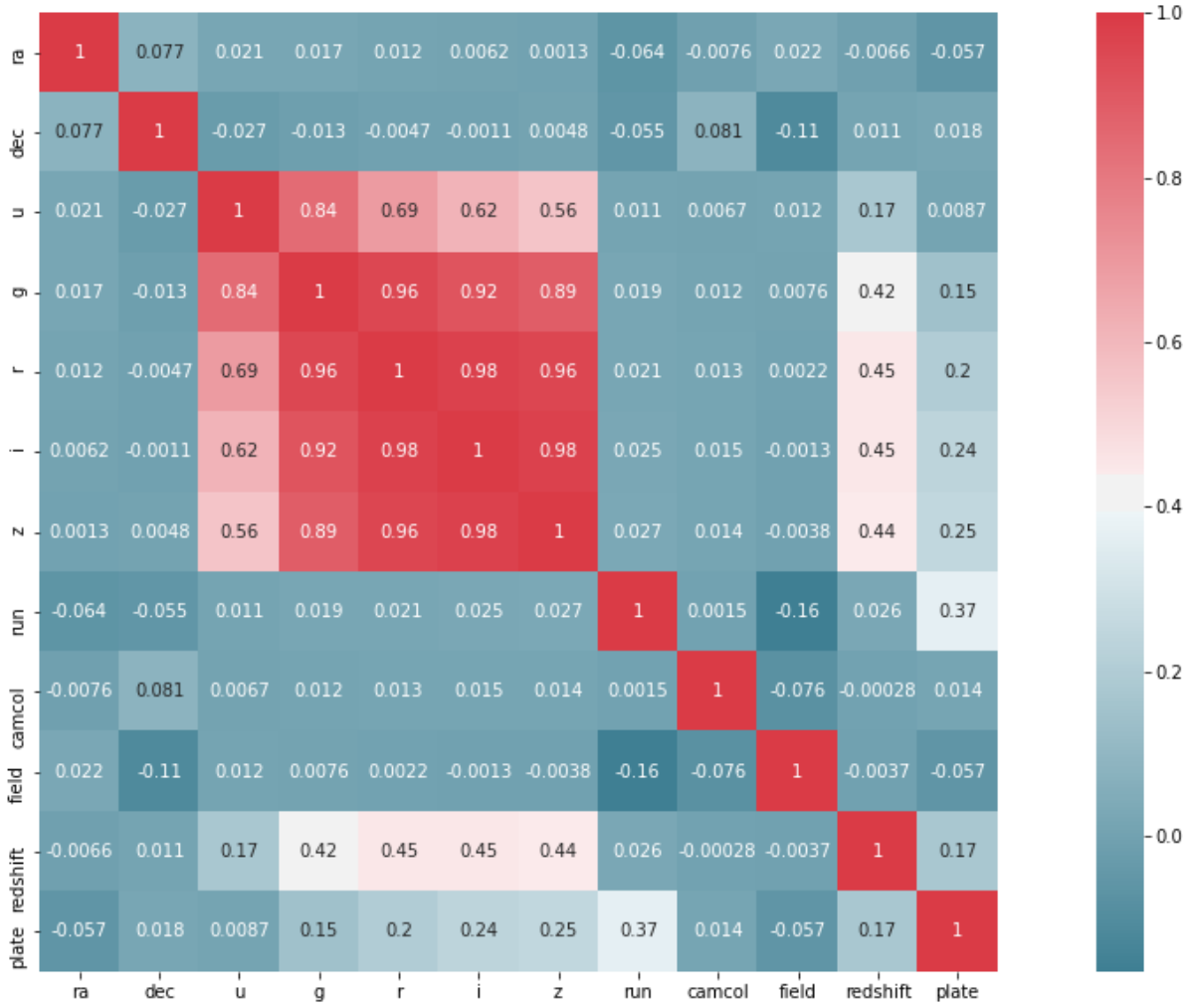
6.2.2 VISUALISTATION OF THE DISTRIBUTION OF DATA USING DENSITY PLOT



HYPOTHESIS

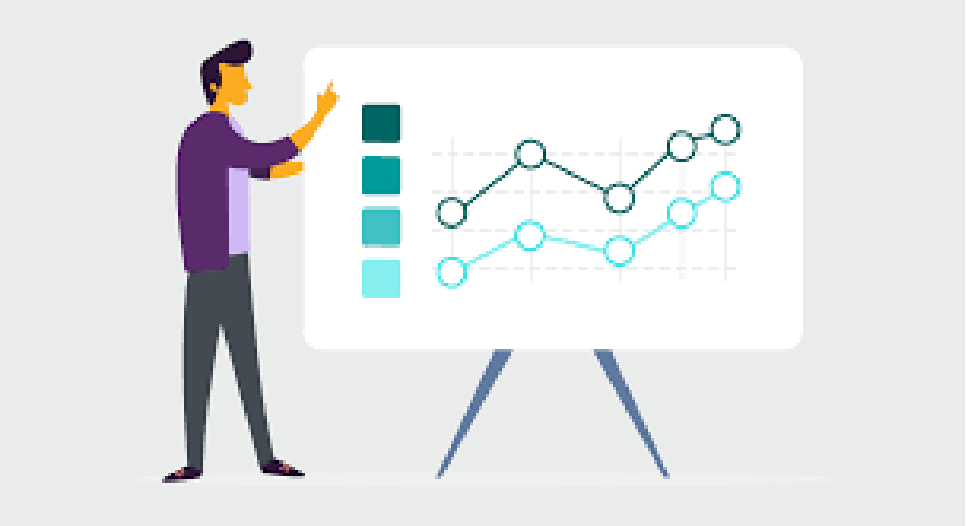
The density distribution of variables ‘g’, ‘r’, ‘i’, ‘z’, ‘redshift’ and ‘plate’ is different for the three classes. This indicates that these variables will be able to differentiate the three classes (stars, quasars and galaxy) well. The above mentioned variables can be hypothesised to be good predictors of the variable ‘class’.

6.2.3 HEAT MAP

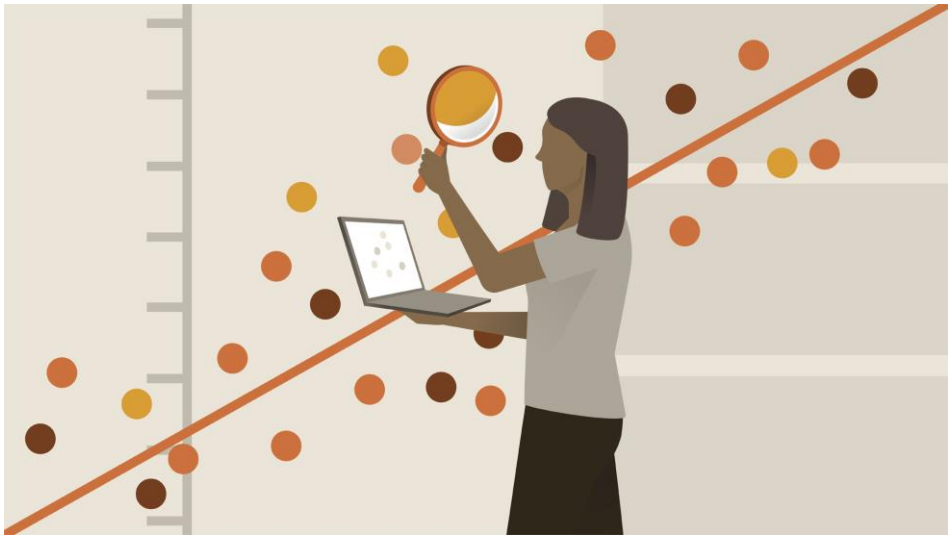


HYPOTHESIS

The heat map indicates presence of multi collinearity in the data set. It can be hypothesised that all variables that represent the colour filters (u, g, r, i, z) are correlated to each other.



7. MODEL BUILDING



7.1 MODEL BUILDING APPROACHES

A total of four models were built as a part of this project. Out of the four, two of them use ensemble methods. Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.

The models and methods used for classification of Galaxies, Stars and Quasars are

- Multinomial Logistic Regression
- Decision Tree Classifier
- Random Forest
- XGBoost Classifier

The models were built on the train data which is a random sample consisting of 70% of the records from the population data.

The models were used to predict classes for the test data which is a random sample consisting of 30% of the records form the population data.

Confusion matrices were generated for both train and test data.

FEATURE SELECTION

The purpose of this project is to build a model that generates good measures of classification, i.e. Accuracy, Precision, sensitivity, specificity and f1-score.

The feature selection process was carried out in two steps, One of the primary steps was to get rid of variables such as –

- objID
- specobjid
- fiberid
- rerun

- mjd (Modified Julian date)

The first three variables were excluded because they are ID (Identity) features. 'Rerun' was excluded because it had the same value (301) for all the records in the data set and 'mjd' was excluded because its value consists of dates, which is not an appropriate feature to use during the model building stages.

INFORMATION VALUE

Information value provides a measure of how well a variable 'X' is able to distinguish between a binary response (e.g. good vs bad) in some target variable 'Y'. Since the project deals with multiclass classification, two of the minority classes (stars – 38% , quasars – 11% of the total data) were combined in order to get the information values of the features.

Information values is derived by calculating the weight of evidence (WOE).

These two concepts - weight of evidence (WOE) and information value (IV) evolved from the logistic regression technique. These two terms have been in existence in credit scoring world for more than 4-5 decades. They have been used as a benchmark to screen variables in the credit risk modelling projects such as probability of default. They help to explore data and screen variables.

Weight of evidence tells the predictive power of an independent variable in relation to the dependent variable. It is calculated by taking the natural logarithm (log to base e) of division of % of non-events and % of events.

$$WOE = \ln\left(\frac{\%\text{ of non-events}}{\%\text{ of events}}\right)$$

Weight of Evidence (WOE) helps to transform a continuous independent variable into a set of groups or bins based on similarity of dependent variable distribution i.e. number of events and non-events.

Information value is one of the most useful technique to select important variables in a predictive model. It helps to rank variables on the basis of their importance.

The IV is calculated using the following formula :

$$IV = \sum (\%\ \text{of non-events} - \%\ \text{of events}) * WOE$$

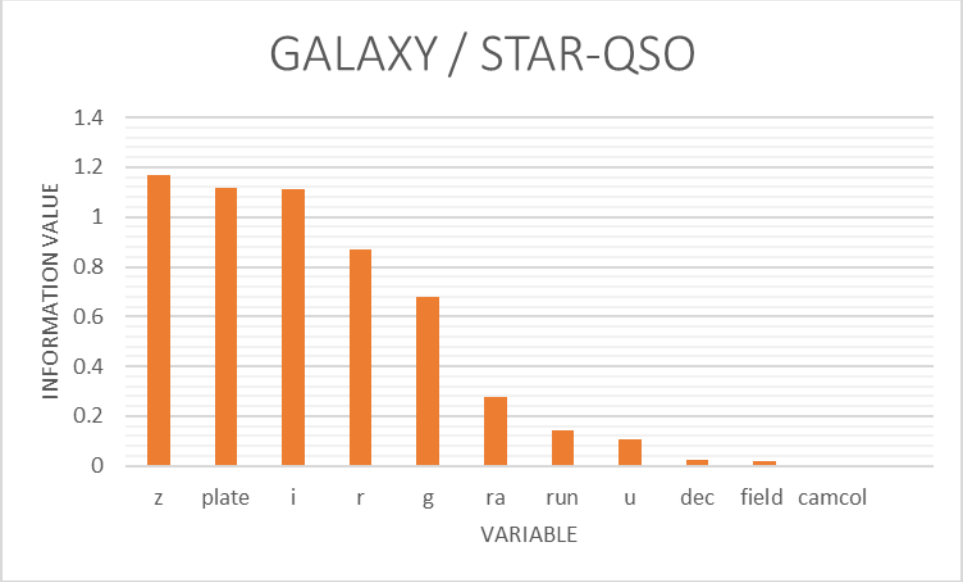
RULES RELATED TO INFORMATION VALUE (IV)

Information Value	Variable Predictiveness
Less than 0.02	Not useful for prediction
0.02 to 0.1	Weak predictive Power
0.1 to 0.3	Medium predictive Power
0.3 to 0.5	Strong predictive Power
>0.5	Suspicious Predictive Power

INFORMATION VALUE (IV) GENERATED FOR THE SDSS DATA

GALAXY / STAR-QSO	
Variable	**IV**
z	1.169
plate	1.119
i	1.113
r	0.87
g	0.677
ra	0.278
run	0.142
u	0.105
dec	0.027
field	0.017
camcol	0.006

VISUALIZATION OF IV GENERATED FOR THE SDSS DATA



HYPOTHESIS

All the variables which have information values less than 0.3 were not included during model building.

The variable 'redshift' was excluded from model building as well since it has a suspiciously high information value (0.8).

7.2 UNDERSTANDING THE CONFUSION MATRIX

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

The above image is a visualization of a confusion matrix for a multi class classification model. It is used to describe the performance of the model.

In this visualization, we have two sections which have been outlined. We have the predicted classifications section which contains three subsections for each of the classes we want to classify into and the actual classifications section which has three subsections for each of the classes.

The following metrics can be visualized using the confusion matrix :

- True Positives, False Positives, True Negatives and False Negatives
- Accuracy
- Precision
- True Positive Rate also known as Sensitivity or Recall
- False Positive Rate also known as Specificity

Lets start by defining the variables in the visualization.

GG - This variable represents the number of predictions where Galaxies were correctly classified as Galaxies. This is also the True Positive for Galaxy Class.

QQ - This variable represents the number of predictions where Quasars were correctly classified as Quasars. This is also the True Positive for Quasars Class.

SS - This variable represents the number of predictions where Stars were correctly classified as Stars. This is also the True Positive for Star Class.

QG - Represents the number of predictions where Galaxies were incorrectly classified as Quasars.

SG - Represents the number of predictions where Galaxies were incorrectly classified as Stars.

GQ - Represents the number of predictions where Quasars were incorrectly classified as Galaxies.

SQ - Represents the number of predictions where Quasars were incorrectly classified as Stars.

GS - Represents the number of predictions where Stars were incorrectly classified as Galaxies.

QS - Represents the number of predictions where Stars were incorrectly classified as Quasars.

TRUE POSITIVES

The true positives is the number of predictions where the data labelled to a particular class was predicted as the said class.

From the definition of the matrix variables, we have already identified the true positives as;

GG – For the Galaxy Class

QQ – For the Quasar Class

SS – For the Star Class

TRUE NEGATIVES

A true negative is an outcome where the model correctly predicts the negative class.

Here we calculate the True Negatives for each class in the confusion matrix unlike the general or absolute True Negatives in the 2-class confusion matrix.

The True Negatives for a particular class is calculated by taking the sum of the values in every row and column except the row and column of the class we're trying to find the True Negatives for.

For Example, Calculation of True negatives for Galaxy Class.

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

True Negative (GALAXY) = QQ+QS+SQ+SS

Similarly for classes Quasar and Star

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

True Negative (QUASAR) = GG+GS+SG+SS

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

True Negative (STAR) = GG+GQ+QG+QQ

FALSE POSITIVES

False Positive is an outcome where the model incorrectly predicts the positive class.

Here we calculate the False Positives for each class in the confusion matrix unlike the general or absolute False Positives in the 2-class confusion matrix.

The False Positives for a particular class can be calculated by taking the sum of all the values in the row corresponding to that class except the True Positives value.

For Example, the False Positives for Galaxy Class is computed as follows.

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

False Positive (GALAXY) = GQ+GS

Similarly for Quasar and Star

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

False Positive (QUASAR) = QG+ QS

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

False Positive (STAR) = SG+SQ

FALSE NEGATIVES

False Negative is the outcome where that model incorrectly predicts a negative class.

Here we calculate the False Negatives for each class in the confusion matrix unlike the general or absolute False Positives in the 2-class confusion matrix.

The False Negatives for a particular class can be calculated by taking the sum of all the values in the column corresponding to that class except the True Positives value.

For Example, the False Negatives for Galaxy Class is computed as follows.

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

False Negative (GALAXY) = QG+SG

Similarly for Quasar and Star

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

False Negative (QUASAR) = GQ+SQ

		ACTUAL		
		GALAXY	QUASAR	STAR
PREDICTED	GALAXY	GG	GQ	GS
	QUASAR	QG	QQ	QS
	STAR	SG	SQ	SS

False Negative (STAR) = GS+QS

ACCURACY

Accuracy is calculated as the ratio of the number of correct classifications to the total number of classifications. From our confusion matrix, the correct classifications are the True Positives for each class and the total number of classifications is the sum of every value in the confusion matrix, including the True Positives.

Therefore Accuracy is,

ACCURACY = (GG+QQ+SS)/(GG+GQ+GS+QG+QQ+QS+SG+SQ+SS)

PRECISION

Precision is a multi-class confusion matrix is the measure of the accuracy relative to the prediction of a specific class. It is calculated as the ratio of the True

Positives of the class in question to the sum of its True Positives and False Positives.

Precision $_G$ (GALAXY) = $TP_G/(TP_G+FP_G)$

Precision $_Q$ (QUASAR) = $TP_Q/(TP_Q+FP_Q)$

Precision $_S$ (STAR) = $TP_S/(TP_S+FP_S)$

SENSITIVITY

Sensitivity is calculated as the ratio of the True Positives of a specific class to the sum of its True Positives and False Negatives.

Sensitivity $_G$ (GALAXY) = $TP_G/(TP_G+FN_G)$

Sensitivity $_Q$ (QUASAR) = $TP_Q/(TP_Q+FN_Q)$

Sensitivity $_S$ (STAR) = $TP_S/(TP_S+FN_S)$

SPECIFICITY

Specificity is calculated as the ratio of the True Negatives of a specific class to the sum of its True Negatives and False Positives.

Specificity $_G$ (GALAXY) = $TN_G/(TN_G+FP_G)$

Specificity $_Q$ (QUASAR) = $TN_Q/(TN_Q+FP_Q)$

Specificity $_S$ (STAR) = $TN_S/(TN_S+FP_S)$

7.3 MULTINOMIAL LOGISTIC REGRESSION

A detailed history of the logistic regression is given in Cramer (2002). The logistic function was developed as a model of population growth and named "logistic" by Pierre François Verhulst in the 1830s and 1840s, under the guidance of Adolphe Quetelet. In his earliest paper (1838), Verhulst did not specify how he fit the curves to the data.

The logistic function was also independently developed in chemistry as a model of *autocatalysis*. The logistic function was independently rediscovered as a model of population growth in 1920 by Raymond Pearl and Lowell Reed, published as Pearl & Reed (1920), which led to its use in modern statistics. They were initially unaware of Verhulst's work and presumably learned about it from L. Gustave du Pasquier, but they gave him little credit and did not adopt his terminology.

The multinomial logit model was introduced independently in Cox (1966) and Thiel (1969), which greatly increased the scope of application and the popularity of the logit model. In 1973 Daniel McFadden linked the multinomial logit to the *theory of discrete choice*, specifically *Luce's choice axiom*, showing that the multinomial logit followed from the assumption of *independence from irrelevant alternatives* and interpreting odds of alternatives as relative preferences; this gave a theoretical foundation for the logistic regression.

INTRODUCTION TO MULTINOMIAL REGRESSION

Multinomial logistic regression (often just called 'multinomial regression') is used to predict a nominal dependent variable given one or more independent variables. It is sometimes considered an extension of binomial logistic regression to allow for a dependent variable with more than two categories. As with other types of regression, multinomial logistic regression can have nominal and/or continuous

independent variables and can have interactions between independent variables to predict the dependent variable.

For example, you could use multinomial logistic regression to understand which type of drink consumers prefer based on location in the UK and age (i.e., the dependent variable would be "type of drink", with four categories – Coffee, Soft Drink, Tea and Water – and your independent variables would be the nominal variable, "location in UK", assessed using three categories – London, South UK and North UK – and the continuous variable, "age", measured in years).

ASSUMPTIONS OF MULTINOMIAL REGRESSION

- **Assumption 1:** Your dependent variable should be measured at the nominal level. Examples of nominal variables include ethnicity (e.g., with three categories: Caucasian, African American and Hispanic), transport type (e.g., with four categories: bus, car, tram and train), profession (e.g., with five groups: surgeon, doctor, nurse, dentist, therapist), and so forth.

- **Assumption 2:** Data can have one or more independent variables that are continuous, ordinal or nominal (including dichotomous variables). However, ordinal independent variables must be treated as being either continuous or categorical. They cannot be treated as ordinal variables when running a multinomial logistic regression.

- **Assumption 3:** You should have independence of observations and the dependent variable should have mutually exclusive and exhaustive categories.

- **Assumption 4:** There should be no multicollinearity. Multicollinearity occurs when you have two or more independent variables that are highly correlated with each other. This leads to problems with understanding which variable contributes to the explanation of the dependent variable and technical issues in calculating a multinomial logistic regression. Determining whether there is multicollinearity is an important step in multinomial logistic regression.

- **Assumption 5:** There needs to be a linear relationship between any continuous independent variables and the logit transformation of the dependent variable.

- **Assumption 6:** There should be no outliers, high leverage values or highly influential points.

MULTINOMIAL REGRESSION IN R

The multinomial regression model can be built using a couple of functions in R. In this project I am using the **multinom()** function from **{nnet}** package. When the logistic models are built we need to set one of the levels of the dependent variable as a baseline. This can be achieved by either using **relevel() function** or we can let the algorithm automate this process.

USAGE: multinom(formula, data)

MODEL

```
Coefficients:
     (Intercept)          g          r        i        z        plate
QSO   -46.840103 -0.3643057  -1.123704 1.053765 3.038037 0.0005811353
STAR    8.069391  2.0855107 -12.940019 0.464704 9.972495 0.0008082965

Std. Errors:
      (Intercept)            g            r            i            z        plate
QSO  2.862052e-05 0.0005250699 0.0005152871 0.0005090658 0.0005058540 1.333207e-05
STAR 2.306729e-05 0.0004024356 0.0003938349 0.0003892323 0.0003871995 1.040846e-05

Residual Deviance: 61458.56
AIC: 61482.56
```

SUMMARY

- The model summary contains a table for coefficients and a table for standard errors .

- The first row in the coefficients table represents coefficients of class quasar in comparison to the baseline which is galaxy and the second row represents coefficients of class star in comparison to the baseline, galaxy.

- The output coefficients are represented in the log of odds.

HYPOTHESIS

- A one unit increase in the variable ‘g’ is associated with the decrease in log odds of being a quasar vs. galaxy in the amount of 0.36.
- A one unit increase in the variable ‘r’ is associated with the decrease in log odds of being a quasar vs. galaxy in the amount of 1.12.
- A one unit increase in the variable ‘i’ is associated with the increase in log odds of being a quasar vs. galaxy in the amount of 1.05.
- A one unit increase in the variable ‘z’ is associated with the increase in log odds of being a quasar vs. galaxy in the amount of 3.03.
- A one unit increase in the variable ‘plate’ is associated with the increase in log odds of being a quasar vs. galaxy in the amount of 0.0006.
- A one unit increase in the variable ‘g’ is associated with the increase in log odds of being a star vs. galaxy in the amount of 2.08.
- A one unit increase in the variable ‘r’ is associated with the decrease in log odds of being a star vs. galaxy in the amount of 12.94.
- A one unit increase in the variable ‘i’ is associated with the increase in log odds of being a star vs. galaxy in the amount of 0.46.
- A one unit increase in the variable ‘z’ is associated with the increase in log odds of being a star vs. galaxy in the amount of 9.97.
- A one unit increase in the variable ‘plate’ is associated with the increase in log odds of being a star vs. galaxy in the amount of 0.0008.

CONFUSION MATRICES

CONFUSION MATRIX FOR TRAIN

- Total number of records in the train data set : 70000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICTED	GALAXY	33721	903	2212
	QSO	291	4991	586
	STAR	1912	1570	23814

MEASSURES	GALAXY	QUASAR	STAR
Specificity	90%	98%	91%
Precision	94%	67%	89%
Recall/ Sensitivity	93%	66%	89%
f1-score	93%	66%	89%
Model Accuracy	89%		

CONFUSION MATRIX FOR TEST

- Total number of records in the test data set : 30000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICTED	GALAXY	14291	126	769
	QSO	364	2219	697
	STAR	903	254	10377

MEASSURES	GALAXY	QUASAR	STAR
Specificity	91%	98%	92%
Precision	92%	85%	88%
Recall/ Sensitivity	94%	67%	89%
f1-score	93%	75%	88%
Model Accuracy	90%		

HYPOTHESIS

From the confusion matrices generated for both train and test, it can be hypothesized that the multinomial logistic model is a good fit since the accuracies and the class level measures for both train and test are similar but it will not be considered as the best model for this classification task as it did not produce good results for the Quasars class.

7.4 DECISION TREE

"Entities should not be multiplied unnecessarily."

-William of Occam
(Occam's Razor)

J.Ross Quinlan in 1975 published a book called Machine Learning vol 1 no 1, in which he presented an algorithm with its foundation based in Occam's Razor. It was called Iterative Dichotomiser 3 (ID3). This algorithm went on to become one of the most famous algorithms used in machine learning, which now a days goes by the name Decision Tree.

A decision tree is a decision support tool that uses a tree-like model for decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

INTRODUCTION TO DECISION TREE

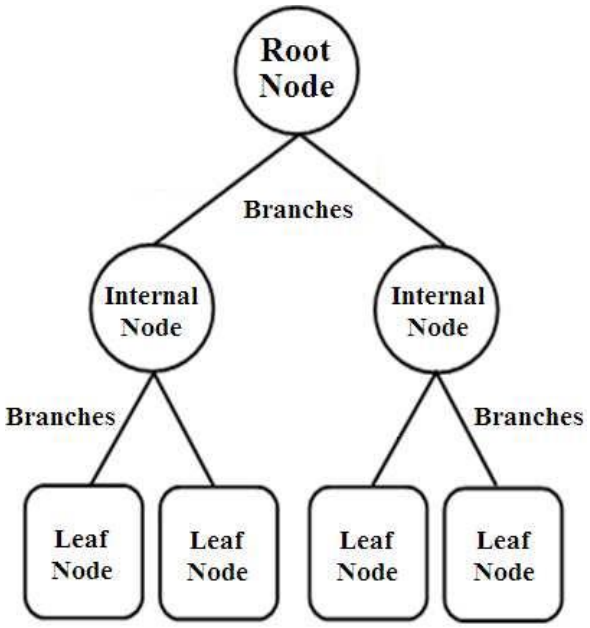
The Decision Tree algorithm is a supervised learning technique that incorporates a layered splitting process. At each layer, the algorithm tries to split the population or sample into two or more groups so that all the observations in the same group are similar to each other (homogeneity) and the groups are significantly different from each other (heterogeneity).

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these

tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

A tree is built by splitting the source set, constituting the root node of the tree, into subsets - which constitute the successor children. The splitting is based on a set of splitting rules based on classification features. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees is an example of a *greedy algorithm*, and it is by far the most common strategy for learning decision trees from data.

VISUALISATION AND BASIC TERMINOLOGIES OF A DECISION TREE



- **Root Node (Top Decision Node):** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.

- **Splitting:** It is a process of dividing a node into two or more sub-nodes.

- **Decision/ Internal Node:** When a sub-node splits into further sub-nodes, then it is called a decision node.

- **Leaf/ Terminal Node:** Nodes with no children (no further split) is called Leaf or Terminal node.

- **Pruning:** When we reduce the size of decision trees by removing nodes (opposite of Splitting), the process is called pruning.

- **Branch / Sub-Tree:** A sub section of the decision tree is called branch or sub-tree.

- **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

DECISION TREE METRICS

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

GINI IMPURITY

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset. The Gini impurity can be computed by summing the probability P_i of an item with label i being chosen times the probability of a mistake, $P(k)$ in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

Probability of a mistake => $\sum_{k \neq i} P(k) = 1 - P(i)$

To compute Gini impurity for a set of items with J classes, suppose $i \in \{1,2,\ldots,J\}$, and let P_i be the fraction of items labelled with class i in the set.

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

INFORMATION GAIN

Information gain is based on the concept of entropy and information content from information theory.

Entropy is a basic quantity in information theory associated to any random variable, which can be interpreted as the average level of "information", "surprise", or "uncertainty" inherent in the variable's possible outcomes. The concept of information entropy was introduced by Claude Shannon in his 1948 paper "A Mathematical Theory of Communication". The entropy is the expected value of the self-information. The self-information quantifies the level of information or surprise associated with one particular outcome or event of a random variable, whereas the entropy quantifies how "informative" or "surprising" the entire random variable is, averaged on all its possible outcomes.

Mathematically Entropy is defined as

$$\mathrm{H}(T) = \mathrm{I}_E(p_1, p_2, \ldots, p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

Where ***P₁, P₂***… are fractions that add up to 1 and represent the percentage of each class present in the child node that results from a split in the tree.

$$\overbrace{IG(T,a)}^{\text{Information Gain}} = \overbrace{\mathrm{H}(T)}^{\text{Entropy (parent)}} - \overbrace{\mathrm{H}(T|a)}^{\text{Weighted Sum of Entropy (Children)}}$$

$$= -\sum_{i=1}^{J} p_i \log_2 p_i - \sum_{a} p(a) \sum_{i=1}^{J} -\Pr(i|a) \log_2 \Pr(i|a)$$

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes.

ADVANTAGES OF DECISION TREE ALGORITHM

- **Simple to understand and interpret**. People are able to understand decision tree models after a brief explanation. Trees can also be displayed graphically in a way that is easy for non-experts to interpret.

- **Able to handle both numerical and categorical data**. Other techniques are usually specialized in analysing datasets that have only one type of variable. (For example, relation rules can be used only with nominal variables while neural networks can be used only with numerical variables or categorical converted to 0-1 values.)

- **Requires little data preparation.** Other techniques often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables.

- **Performs well with large datasets**. Large amounts of data can be analysed using standard computing resources in reasonable time.

- **Mirrors human decision making more closely than other approaches**. This could be useful when modelling human decisions/behaviour.

LIMITATIONS OF DECISION TREE ALGORTHM

- Trees can be very non-robust. A small change in the training data can result in a large change in the tree and consequently the final predictions.

- Practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.

- Decision-tree learners can create over-complex trees that do not generalize well from the training data. (This is known as overfitting.) Mechanisms such as pruning are necessary to avoid this problem.

For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favour of attributes with more levels.

DECISION TREE IN PYTHON

In this project I am using DecisionTreeClassifier() function from the sklearn package to build the decision tree model.

I used entropy/information gain as a criteria for splitting and pruned the tree to a maximum depth of 5.

MODEL

- A subset of the original data (consisting of 70000 records) was used to train the model.
- Predictions were made on both train data (70000 records) as well as test data (30000 records).
- Confusion matrices were generated for both train and test data.
- Comparisons made of the measures generated by the confusion matrices, helped validate the model for over-fitting which is a common feature of the decision tree classifier.
- The visualisation generated of the decision tree is too large to fit in an A4 sheet, so the image was cropped into two, LHS (left hand side of the image) and RHS (right hand side of the image).

CONFUSION MATRICES

CONFUSION MATRIX FOR TRAIN

- Total number of records in the train data set : 70000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICTED	GALAXY	32927	671	6883
	QSO	314	5600	921
	STAR	2750	1168	18766

MEASSURES	GALAXY	QUASAR	STAR
Specificity	78%	98%	91%
Precision	91%	75%	71%
Recall/ Sensitivity	81%	82%	83%
f1-score	86%	78%	76%
Model Accuracy	82%		

CONFUSION MATRIX FOR TEST

- Total number of records in the test data set : 30000

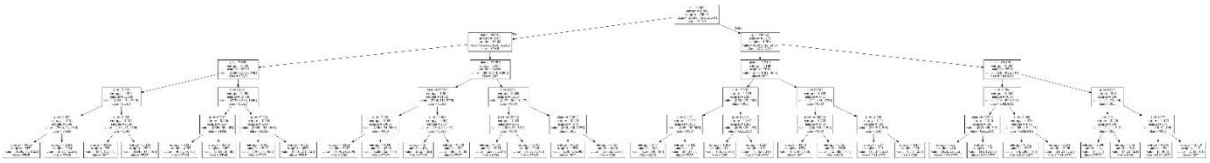
		ACTUAL		
		GALAXY	QSO	STAR
PREDICTED	GALAXY	14014	314	3015
	QSO	121	2349	439
	STAR	1197	479	8072

MEASSURES	GALAXY	QUASAR	STAR
Specificity	77%	98%	91%
Precision	91%	75%	70%
Recall/ Sensitivity	81%	81%	83%
f1-score	86%	78%	76%
Model Accuracy	81%		

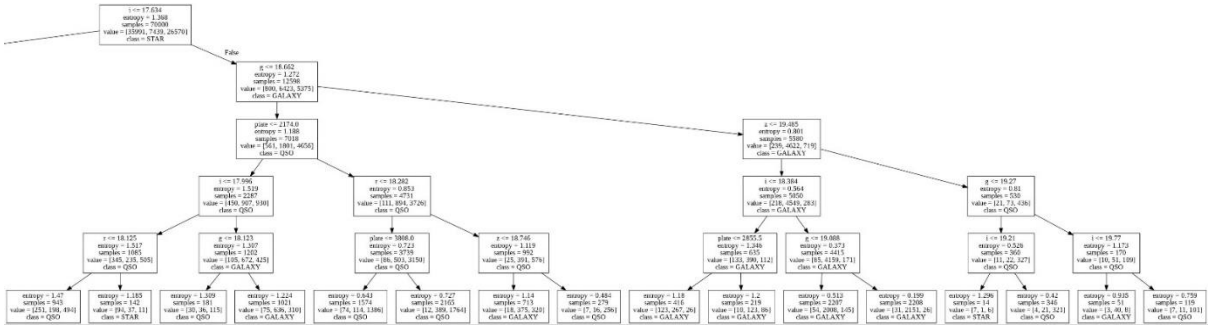
HYPOTHESIS

From the confusion matrices generated for both train and test, it can be hypothesized that the Decision tree model is a good fit as the accuracies and the class level measures for both train and test are similar.

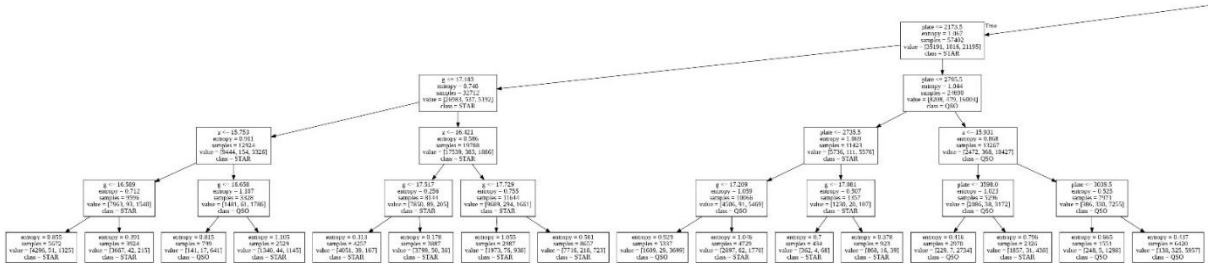
VISUALISATION OF THE DECISION TREE



RIGHT HAND SIDE



LEFT HAND SIDE



7.5 RANDOM FOREST

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The first algorithm for random decision forests was created by Tin Kam Ho. Ho established that forests of trees splitting with oblique hyperplanes can gain accuracy as they grow without suffering from overtraining, as long as the forests are randomly restricted to be sensitive to only selected feature dimensions. The early development of Breiman's notion of random forests was influenced by the work of Amit and Geman who introduced the idea of searching over a random subset of the available decisions when splitting a node, in the context of growing a single tree. The idea of random subspace selection from Ho was also influential in the design of random forests. In this method a forest of trees is grown, and variation among the trees is introduced by projecting the training data into a randomly chosen subspace before fitting each tree or each node.

INTRODUCTION TO RANDOM FOREST

Random forest is a classifier that evolves from decision trees. It actually consists of many decision trees. To classify a new instance, each decision tree provides a classification for input data; random forest collects the classifications and chooses the most voted prediction as the result. The input of each tree is sampled data from the original dataset. In addition, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, random forest enables a large number of weak or weakly-correlated classifiers to form a strong classifier.

These Decision Trees are trained by the bagging method, based on the

idea that the combination of learning models improves the result. This differs from the standard trees because each node is divided using the best combination of all the variables. As the decision trees grow, an extra randomness is brought in for the random forest model and it searches for the best feature within a random subset of features. It does not make the algorithm more complicated to use since it requires only few hyperparameters and these generally produce a good classification results. The classifier uses GINI index as an attribute selection measure to measure the impurity of an attribute with respect to the classes.

BAGGING

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1,.., x_n$ with responses $Y = y_1,., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree fb on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the

same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set.

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

ADVANTAGES OF RANDOM FOREST ALGORITHM

- Reduces overfitting problem in decision trees and also reduces the variance and therefore improves the accuracy.
- Random Forest works well with both categorical and continuous variables.
- Random Forest can automatically handle missing values.
- **No feature scaling required:** No feature scaling (standardization and normalization) required in case of Random Forest as it uses rule based approach instead of distance calculation.

- **Handles non-linear parameters efficiently**: Non-linear parameters don't affect the performance of a Random Forest unlike curve based algorithms. So, if there is high non-linearity between the independent variables, Random Forest may outperform other curve based algorithms.

- Random Forest is usually robust to outliers and can handle them automatically.

- **Random Forest algorithm is very stable**. Even if a new data point is introduced in the dataset, the overall algorithm is not affected much since the new data may impact one tree, but it is very hard for it to impact all the trees.

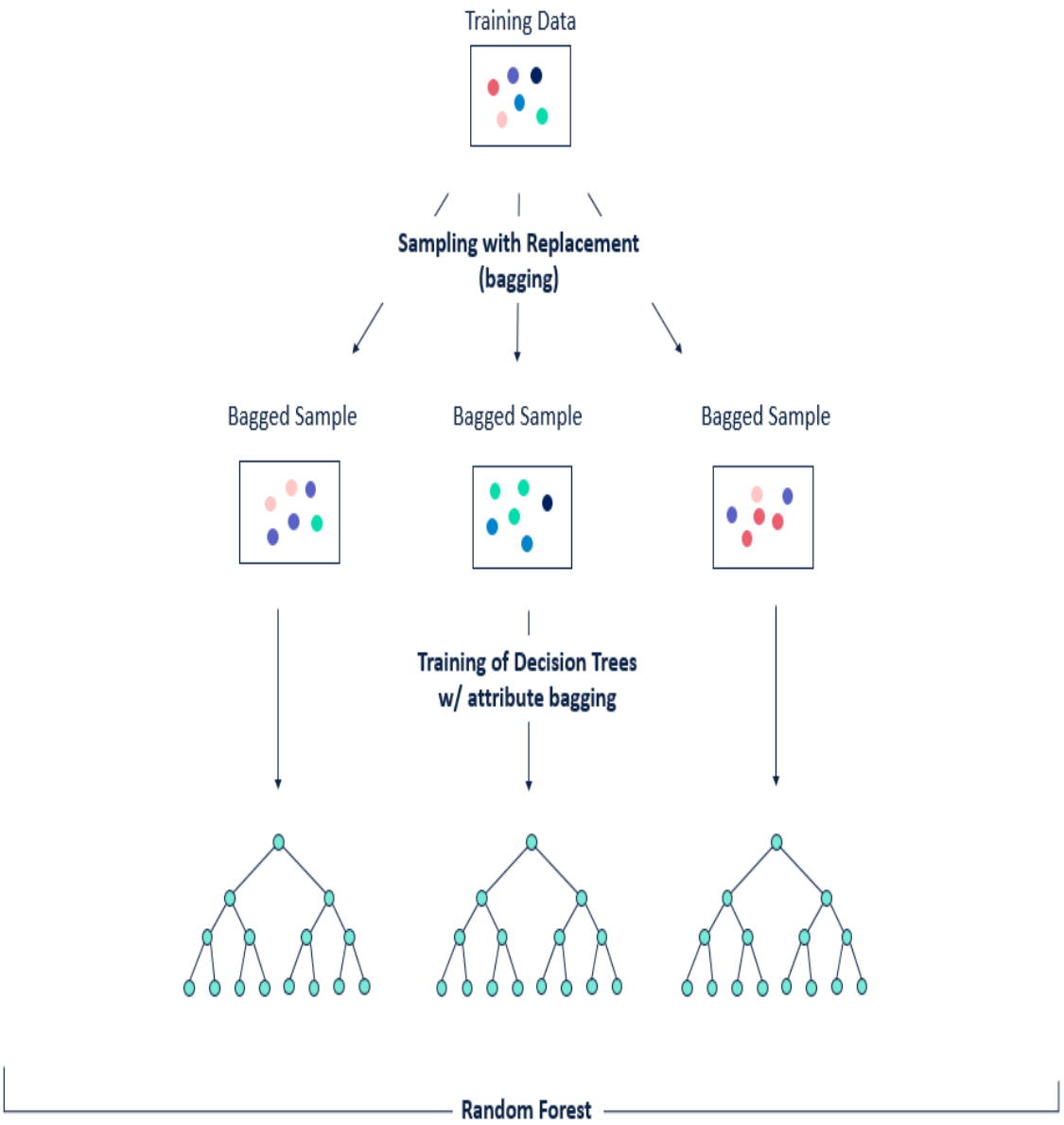
- Random Forest is comparatively less impacted by noise.

DISADVANTAGES OF RANDOM FOREST ALGORITHM

- **Complexity:** Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.

- **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

GENERAL VISUALISATION OF A RANDOM FOREST ALGORITHM



RANDOM FOREST IN PYTHON

In this project I am using RandomForestClassifier() function from the sklearn package to build the random forest model.

10 trees/estimators were used for this model.

MODEL

- A subset of the original data (consisting of 70000 records) was used to train the model.
- Predictions were made on both train data (70000 records) as well as test data (30000 records).
- Confusion matrices were generated for both train and test data.
- Comparisons made of the measures generated by the confusion matrices, helped validate if the model was a good fit or not.

CONFUSION MATRICES

CONFUSION MATRIX FOR TRAIN

- Total number of records in the train data set : 70000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICED	GALAXY	35933	122	104
	QSO	14	7275	29
	STAR	44	42	26437

MEASSURES	GALAXY	QUASAR	STAR
Specificity	99%	100%	98%
Precision	99%	99%	100%
Recall/ Sensitivity	100%	98%	99%
f1-score	100%	99%	100%
Model Accuracy	99%		

CONFUSION MATRIX FOR TEST

- Total number of records in the test data set : 30000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICTED	GALAXY	14765	107	460
	QSO	311	2541	290
	STAR	563	191	10772

MEASSURES	GALAXY	QUASAR	STAR
Specificity	94%	99%	83%
Precision	94%	90%	93%
Recall/ Sensitivity	96%	81%	93%
f1-score	95%	85%	93%
Model Accuracy	94%		

HYPOTHESIS

- From the confusion matrices generated for both train and test, it can be hypothesized that the Random Forest model is a good fit as the accuracies and the class level measures for both train and test are similar.

- The Random Forest classifier generated better results than the Decision Tree Classifier.

7.6 XGBoost CLASSIFIER

"The name XGBoost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use XGBoost." – Tianqi Chen"

XGBoost stands for Extreme Gradient Boosting. It was initially started as a research project by Tianqi Chen and Carlos Guestrin as a part of the Distributed Machine Learning Community (DMLC) group. They presented their paper at a conference and caught the machine learning world by fire. Since its introduction, this algorithm has been credited with winning numerous Kaggle competitions. It is also used in the industry for developing several cutting edge applications. The algorithm has received two awards, John Chambers Award (2016) and High Energy Physics meets Machine Learning award (HEP meets ML) (2016).

Tianqi Chen and Tong He have authored a paper that proposes to solve the Higgs Boson Challenge using XGBoost Classifier. The challenge was to extract signals of the Higgs Boson from Background noises. Their final solutions gave remarkable results, making them the top 2% in the Higgs Boson Challenge.

INTRODUCTION TO XGBoost CLASSIFIER

XGBoost is a **decision tree** based ensemble Machine Learning Algorithm that uses a **Gradient Boosting** framework.

So, What is Gradient Boosting?

To understand Gradient boosting we need to first understand the meaning of the term Boosting. In simple terms Boosting is the process of converting weak learners into strong learners. In boosting, each new tree is fit on a modified version of the original data set.

The most common type of boosting used is the AdaBoost (Adaptive boosting) algorithm. The AdaBoost algorithm trains Decision trees in

a way that each observation is assigned an equal weight. After evaluating the first tree it increases the weights of those observations that are difficult to classify and lowers the weights for those that are easy to classify. The second tree is then grown on the weighted data. Here , the idea is to improve the predictions of the first tree. The algorithm then computes the classification error of this two-tree ensemble model and grows a third tree to be used for prediction. This process is carried on for specified number of iterations. Subsequent trees help classify observations that are not well classified by previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous models.

Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient boost is how the two algorithms identify the deficiency of weak learners. While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function (y=ax+b+e , e is the error term). The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data.

XGBoost is a specific implementation of the Gradient Boosting method which delivers more accurate approximations by using the strengths of second order derivative of *the loss function, L1 and L2 regularisation and lucl computing.*

ADVANTAGES OF XGBoost CLASSIFIER

- **Regularization**: XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XGBoost is also called regularized form of Gradient Boosting Machine.

- **Parallel Processing**: XGBoost utilizes the power of parallel processing and that is why it is much faster than Gradient Boosting Machine. It uses multiple CPU cores to execute the model.

- **Handling Missing Values:** XGBoost has an in-built capability to handle missing values. When XGBoost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

- **Cross Validation:** XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike Gradient Boosting Machine where we have to run a grid-search and only limited values can be tested.

- **Effective Tree Pruning:** A Gradient Boosting Machine would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XGBoost on the other hand make splits upto the max_depth specified and then starts pruning the tree backwards and removes splits beyond which there is no positive gain.

DISADVANTAGES OF XGBoost CLASSIFIER

- **Sensitive to outliers:** Every classifier is obliged to fix the errors in the predecessors; thus, the method is too dependent on outliers.

XGBoost IN PYTHON

In this project I am using the package XGBoost available in python to build the model. The data was converted into a format called the DMatrix . DMatrix is an internal data structure that used by XGBoost which is optimized for both memory efficiency and training speed.

MODEL

- A subset of the original data (consisting of 70000 records) was used to train the model.

- Predictions were made on both train data (70000 records) as well as test data (30000 records).

- Confusion matrices were generated for both train and test data.

- Comparisons made of the measures generated by the confusion matrices, helped validate the model.

TREE BOOSTER PARAMETERS USED

- "eta" : [default=0.3, alias: learning rate]

  - Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.
  - range: [0,1].
  - Default value of "eta" was used.

- 'max depth' : [default=6]
  - Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.
  - range: [0,∞]
  - 'max depth' was set to 3.

The 'softprob' function was used as the objective function and the number of classes was set to three.

CONFUSION MATRICES

CONFUSION MATRIX FOR TRAIN

- Total number of records in the train data set : 70000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICED	GALAXY	33964	601	2699
	QSO	344	5486	544
	STAR	1683	1352	23327

MEASSURES	GALAXY	QUASAR	STAR
Specificity	94%	97%	73%
Precision	91%	86%	88%
Recall/ Sensitivity	94%	74%	88%
f1-score	93%	79%	88%
Model Accuracy	90%		

CONFUSION MATRIX FOR TEST

- Total number of records in the test data set : 30000

		ACTUAL		
		GALAXY	QSO	STAR
PREDICTED	GALAXY	14450	274	1280
	QSO	129	2275	248
	STAR	753	593	9998

MEASSURES	GALAXY	QUASAR	STAR
Specificity	94%	97%	71%
Precision	90%	86%	88%
Recall/ Sensitivity	94%	72%	87%
f1-score	92%	79%	87%
Model Accuracy	89%		

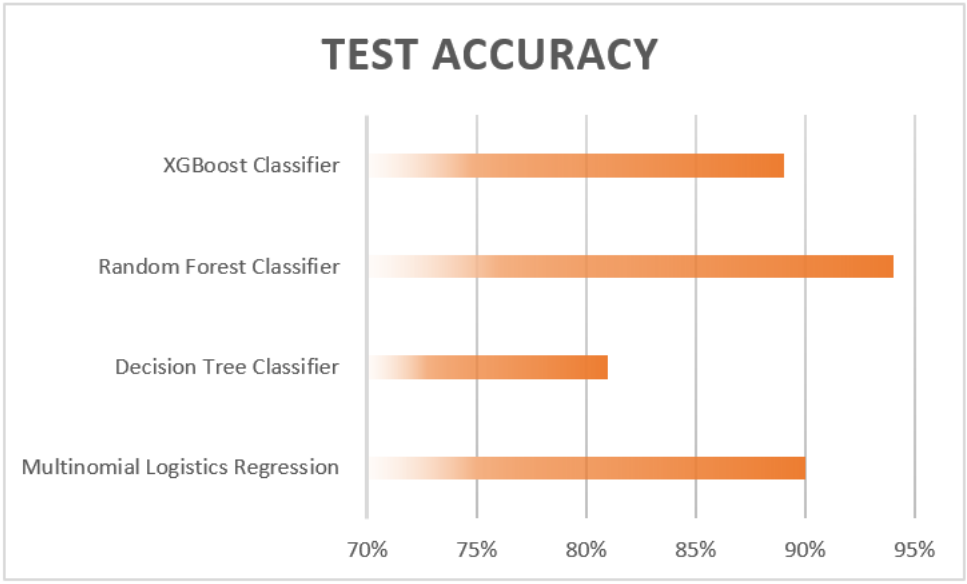
HYPOTHESIS

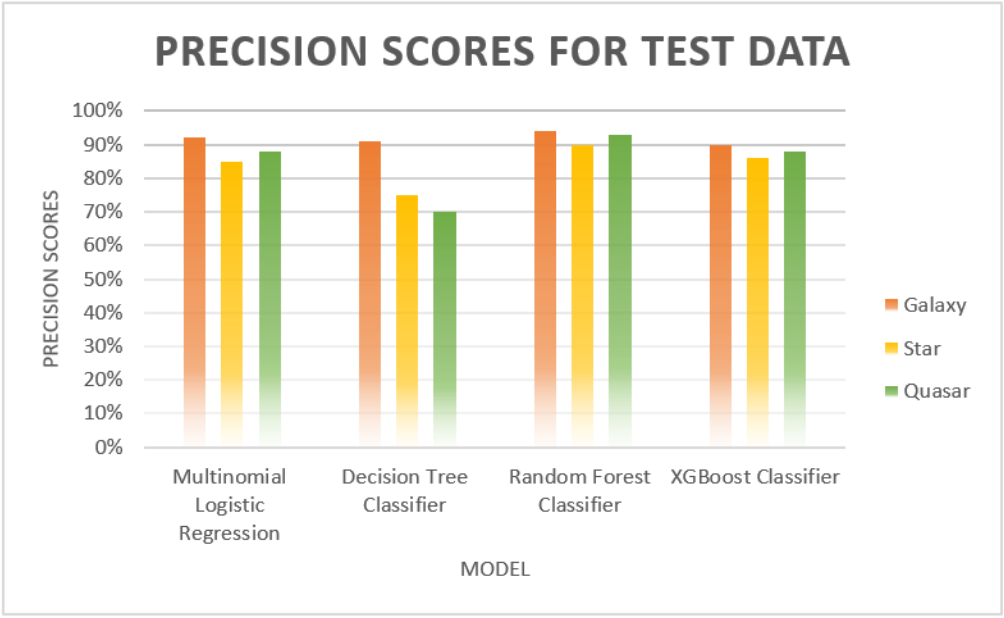
From the confusion matrices generated for both train and test, it can be hypothesized that the XGBoost Model is a good fit as the accuracies and the class level measures for both train and test are similar.

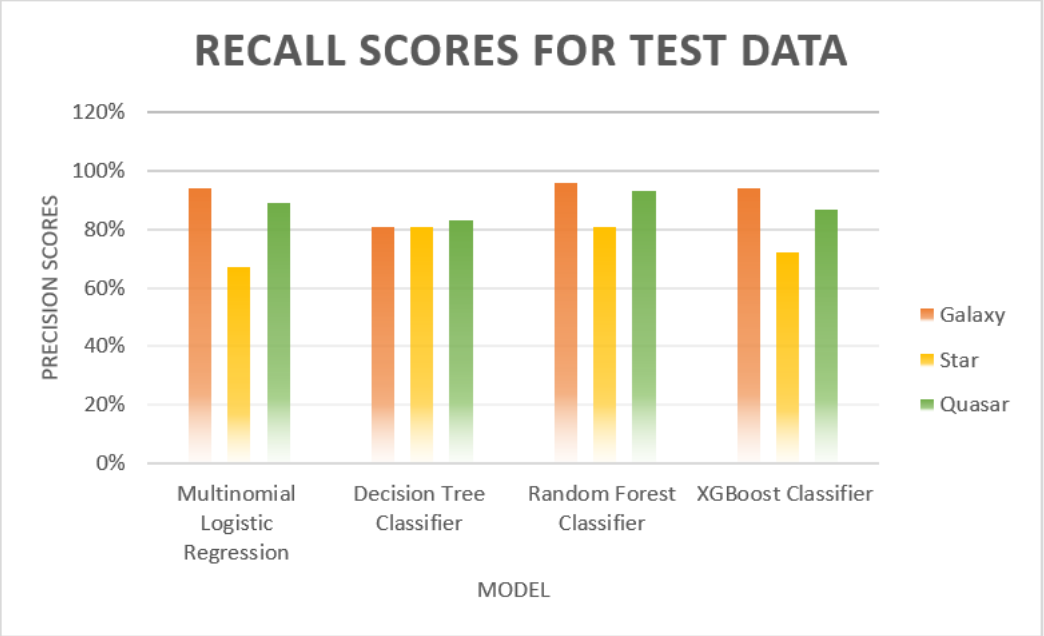
8. SUMMARY

TRAIN				TEST			
Multinomial Regression				Multinomial Regression			
Model Accuracy	89%			Model Accuracy	90%		
	GALAXY	STAR	QUASAR		GALAXY	STAR	QUASAR
Specificity	90%	98%	91%	Specificity	91%	98%	92%
Precision	94%	67%	89%	Precision	92%	85%	88%
Recall/ Sensitivity	93%	66%	89%	Recall/ Sensitivity	94%	67%	89%
f1-score	93%	66%	89%	f1-score	93%	75%	88%
Decision Tree Classifier				Decision Tree Classifier			
Model Accuracy	82%			Model Accuracy	81%		
	GALAXY	STAR	QUASAR		GALAXY	STAR	QUASAR
Specificity	78%	98%	91%	Specificity	77%	98%	91%
Precision	91%	75%	71%	Precision	91%	75%	70%
Recall/ Sensitivity	81%	82%	83%	Recall/ Sensitivity	81%	81%	83%
f1-score	86%	78%	76%	f1-score	86%	78%	76%
Random Forest Classifier				Random Forest Classifier			
Model Accuracy	99%			Model Accuracy	94%		
	GALAXY	STAR	QUASAR		GALAXY	STAR	QUASAR
Specificity	99%	100%	98%	Specificity	94%	99%	83%
Precision	99%	99%	100%	Precision	94%	90%	93%
Recall/ Sensitivity	100%	98%	99%	Recall/ Sensitivity	96%	81%	93%
f1-score	100%	99%	100%	f1-score	95%	85%	93%
XGBoost Classifier				XGBoost Classifier			
Model Accuracy	90%			Model Accuracy	89%		
	GALAXY	STAR	QUASAR		GALAXY	STAR	QUASAR
Specificity	94%	97%	73%	Specificity	94%	97%	71%
Precision	91%	86%	88%	Precision	90%	86%	88%
Recall/ Sensitivity	94%	74%	88%	Recall/ Sensitivity	94%	72%	87%
f1-score	93%	79%	88%	f1-score	92%	79%	87%

9. CONCLUSION







HYPOTHESIS

The above visualisations are a clear indication that the Random Forest classifier not only has good accuracy scores but also has excellent precision and recall scores. It can therefore be hypothesised that the Random Forest classifier is the best model to classify galaxies, Stars and quasars.

10. IMAGES

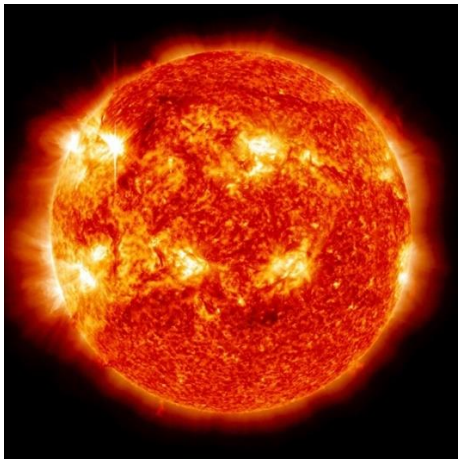


Fig 1 Image of the sun, the closest star to the Earth

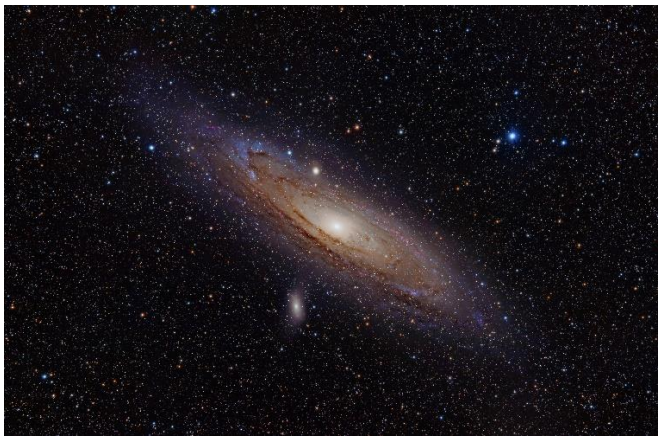


Fig 2 Image of a Galaxy

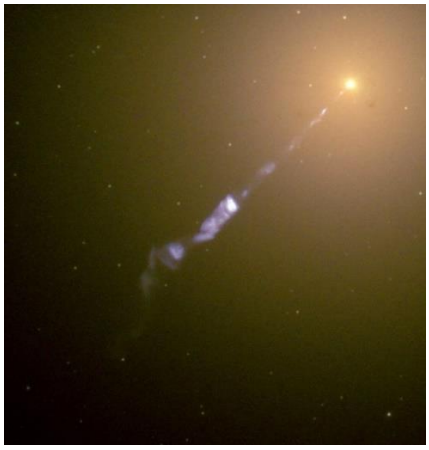


Fig 3 The brightest quasar, photographed by the Hubble Space Telescope's Advanced Camera for Surveys.

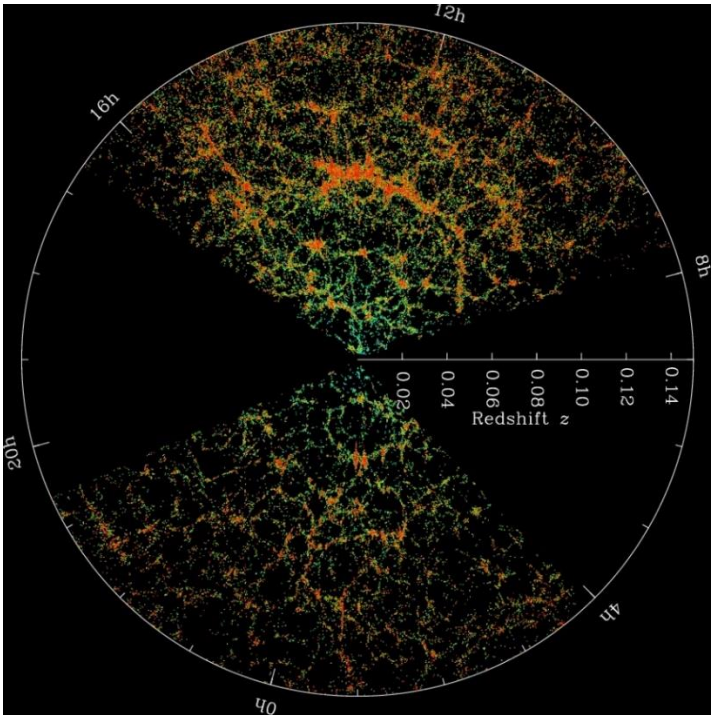


Fig 4 Slices through the SDSS 3-dimensional map of the distribution of galaxies. Earth is at the centre, and each point represents a galaxy, typically containing about 100 billion stars. Galaxies are coloured according to the ages of their stars, with the redder, more strongly clustered points showing galaxies that are made of older stars. The outer circle is at a distance of two billion light years. The region between the wedges was not mapped by the SDSS because dust in our own Galaxy obscures the view of the distant universe in these directions. Both slices contain all galaxies within -1.25 and 1.25 degrees declination.

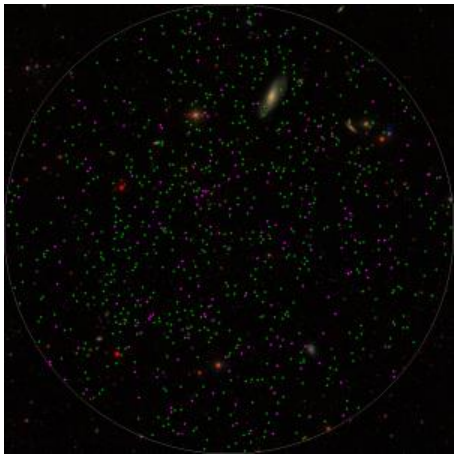


Fig 5 Image of a plate used in SDSS

11. GLOSSARY

1. Charged Coupled Devices : A charge-coupled device is a device for the movement of electrical charge, usually from within the device to an area where the charge can be manipulated, such as conversion into a digital value. This is achieved by "shifting" the signals between stages within the device one at a time.

2. Greedy Algorithm : A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution.

3. Independence from irrelevant alternatives : The independence of irrelevant alternatives (IIA), also known as binary independence or the independence axiom, is an axiom of decision theory and various social sciences.

The social preferences between alternatives x and y depend only on the individual preferences between x and y.

Another expression of the principle:

If A is preferred to B out of the choice set {A,B}, introducing a third option X, expanding the choice set to {A,B,X}, must not make B preferable to A.

4. Luce's Choice Axiom : In probability theory, Luce's choice axiom, formulated by R. Duncan Luce (1959), states that the probability of selecting one item over another from a pool of many items is not affected by the presence or absence of other items in the pool.

5. L1 and L2 regularisation : In mathematics, statistics, and computer science, particularly in machine learning and inverse problems, regularization is the process of adding information in order to solve an ill-posed problem or to prevent overfitting.

A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression.

The key difference between these two is the penalty term.

Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function.

Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function. Lasso shrinks the less important feature’s coefficient to zero thus, removing some feature altogether.

6. Parallel Computing : Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time.

7. Recursive Partitioning : Recursive partitioning is a statistical method for multivariable analysis. Recursive partitioning creates a decision tree that strives to correctly classify members of the population by splitting it into sub-populations based on several dichotomous independent variables. The process is termed recursive because each sub-population may in turn be split an indefinite number of times until the splitting process terminates after a particular stopping criterion is reached.

8. Theory of discrete choice : In economics, discrete choice models, or qualitative choice models, describe, explain, and predict choices between two or more discrete alternatives, such as entering or not entering the labour market, or choosing between modes of transport.

12. APENDIX

12.1 REFERENCES

MULTINOMIAL LOGISTIC REGRESSION

https://statistics.laerd.com/spss-tutorials/multinomial-logistic-regression-using-spss-statistics.php, 15th march 202

https://en.wikipedia.org/wiki/Multinomial_logistic_regression, 15th march 2020

https://en.wikipedia.org/wiki/Logistic_regression, 15th march 2020

DECISION TREE CLASSIFIER

https://towardsdatascience.com/the-indecisive-decision-tree-story-of-an-emotional-algorithm-1-2-8611eea7e397, 16TH March 2020

https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052; 16th march 2020

https://en.wikipedia.org/wiki/Decision_tree_learning; 16th march 2020

https://towardsdatascience.com/machine-learning-basics-descision-tree-from-scratch-part-i-4251bfa1b45c; 16thmarch 2020

RANDOM FOREST CLASSIFIER

https://en.wikipedia.org/wiki/Random_forest; 19TH march 2020

https://www.sciencedirect.com/topics/engineering/random-forest; 19th march 2020

http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html;19thmarch 2020

XGBoost CLASSIFIER

http://proceedings.mlr.press/v42/chen14.pdf : 5th march 2020

https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab ; 8th march 2020

theprofessionalspoint.blogspot.com/2019/02/difference-between-gbm-gradient.html ;8th march 2020

http://theprofessionalspoint.blogspot.com/2019/03/advantages-of-xgboost-algorithm-in.html , 19TH march 2020

INFORMATION VALUE AND WEIGHT OF EVIDENCE

https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html; 21ST march 2020

WIKIPEDIA

https://www.wikipedia.org/

GREEDY ALGORITHM AND LI & L2 REGULARIZATION

https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/ 29th march 2020

https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c - 29th march 2020

CONFUSION MATRIX

https://dev.to/overridedeveloper/understanding-the-confusion-matrix-264i 29TH march 2020

IMAGES

https://astronomy.com/news/2018/11/astronomers-find-a-solar-twin--a-star-that-looks-almost-exactly-like-our-sun?https://www.Freedatacheat.com 12th April 2020

https://en.wikipedia.org/wiki/Andromeda_Galaxy 12th April 2020

https://spacetelescope.org/science/black_holes/ 12th April 2020

12.2 CODES USED IN THE PROJECT

➢ INFORMATION VALUE USING R

```
#clubbing qso and star
data$change_class <- ifelse(data$class=="QSO","STAR",ifelse(data$class=='STAR','STAR','GALAXY'))
table(data$change_class)

#converting to 1 and 0
data$change_class <- as.factor(ifelse(data$change_class=='GALAXY',1,0))
# 1-galaxy , 0-star
table(data$change_class)
data$change_class <- as.numeric(ifelse(data$change_class=='1',1,0))
table(data$change_class)

#computing the information value
install.packages("Information")
install.packages("woe")
library(Information)
library(woe)
#data2 <- data[,-c(1,14)]
info1 <- create_infotables(data = data,y="change_class")
info1$Summary
```

➢ MULTINOMIAL REGRESSION USING R

```
#installing all important packages
install.packages('nnet')
install.packages('dplyr')
install.packages("caret")
library(nnet)
library(dplyr)
library(caret)

#preparing the data for modeling, converting the dependent variable "class" to factor
mdata$class <- as.factor(mdata$class)

#splitting data into train and test
train <- sample_frac(mdata,0.7)
sample_id=as.numeric(rownames(train))
test <- mdata[-sample_id,]
head(test)

#extracting all the important features
fe_x_train <- subset(train,select = -c(1,2,3,8,9,10,12))
fe_x_test <- subset(test,select = -c(1,2,3,8,9,10,12))

#building model
train.fit <- multinom(class~.,data = fe_x_train)

#predicting on train
pred_train <- predict(train.fit,newdata = fe_x_train,'class')

#Generating a confusion matrix
cm_train <- confusionMatrix(data = pred_train,reference = fe_x_train$class)
print(cm_train)


#predicting on test
pred_test <- predict(train.fit,newdata = fe_x_test,'class')

#confusion matrix for test
cm_test <- confusionMatrix(data = pred_test, reference = fe_x_test$class)
print(cm_test)
summary(train.fit)
```

PYTHON PACKAGES USED FOR MODEL BUILDING

```
#importing basic packages
import pandas as pd
import numpy as np
import statistics
import matplotlib.pyplot as plt
import seaborn as sns

#confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

#for train test split
from sklearn import model_selection
from sklearn.model_selection import train_test_split

#data preprocessing
from sklearn.preprocessing import label_binarize

#for z scores
from scipy.stats import zscore
```

➢ DECISION TREE CLASSIFIER USING PYTHON

CODE FOR SPLITTING THE DATA INTO TRAIN AND TEST SET.

```
for feature in mdata.columns:
  if mdata[feature].dtype == 'object':
    mdata[feature]=pd.Categorical(mdata[feature]).codes

#data prepareation
x=mdata.drop(['class'],axis=1)
y=mdata.pop('class')
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.30, random_state=1)
#feature extracted data set (based on iv value)
fe_x_train=x_train.drop(['ra','run','u','dec','field','camcol','redshift'],axis=1)
fe_x_test=x_test.drop(['ra','run','u','dec','field','camcol','redshift'],axis=1)
```

CODE FOR BUILDING THE DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt_model_o=DecisionTreeClassifier(criterion="entropy",max_depth=5)
dt_model_o.fit(fe_x_train,y_train)
```

CODE FOR PREDICTING ON TEST SET

```
pred_test_o=dt_model_o.predict(fe_x_test)
```

CODE FOR GENERATION CONFUSION MATRIX

```
cm_test=confusion_matrix(pred_test_o,y_test)
cm_test
```

➢ RANDOM FOREST CLASSIFIER USING PYTHON

CODE FOR BUILDING RANDOM FOREST CLASSIFIER IN PYTHON

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 10)
rf_train = rf.fit(fe_x_train, y_train)
```

CODE FOR PREDICTING ON TEST SET

```
rf_pred_test=rf.predict(fe_x_test)
```

CODE FOR GENERATING CONFUSION MATRIX

```
cm_test=confusion_matrix(y_test,rf_pred_test)
cm_test
```

➢ XGBoost CLASSIFIER USING PYTHON

DATA PRE-PROCESSING

```
#In order for XGBoost to be able to use our data,
#we'll need to transform it into a specific format that XGBoost can handle.
#That format is called DMatrix.
#It's a very simple one-linear to transform a numpy array of data to DMatrix format
d_train = xgb.DMatrix(x_train, label=y_train)
d_test = xgb.DMatrix(x_test, label=y_test)
fe_d_train = xgb.DMatrix(fe_x_train, label=y_train)
fe_d_test = xgb.DMatrix(fe_x_test, label=y_test)
```

CODE FOR SETTING THE PARAMETERS OF THE MODEL

```
param = {'eta': 0.3, 'max_depth': 3,  'objective': 'multi:softprob',   'num_class': 3}
steps = 20  # The number of training iterations
```

CODE FOR BUILDING THE XGBoost CLASSIFIER

```
mod4=xgb.XGBClassifier(objective="multi:softprob")
```

```
mod4.fit(fe_x_train,y_train)
```

CODE FOR PREDICTING ON TEST SET

```
pred_mod4_test=mod4.predict(fe_x_test)
```

CODE FOR GENERATING CONFUSION MATRIX

```
cm_mod4_test=confusion_matrix(y_test,pred_mod4_test)
cm_mod4_test
```

➢ EXPLORATORY DATA ANALYSIS USING PYTHON

CODE FOR UNIVARIATE ANALYSIS

```
#univariate analysis
idata.describe().T
```

CODE FOR CLASS DISTRIBUTION PLOT

```
#distribution plot
total=float(len(idata))
ax = sns.countplot(x="class", data=idata)
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{0:.0%}'.format(height/total),
            ha="center")
```

CODE FOR BOXPLOT

```
sns.boxplot(x='class',y='ra',data=idata)
```

CODE FOR DENSITY PLOT

```
#density plots
#https://towardsdatascience.com/predicting-stars-galaxies-quasars-with-random-forest-classifiers-in-python-edb127878e43
featuredf = idata.drop(['class'], axis=1)
featurecols = list(featuredf)
astrObjs = idata['class'].unique()
colours = ['coral', 'magenta', 'lime']
plt.figure(figsize=(15,10))
for i in range(len(featurecols)):
    plt.subplot(4, 4, i+1)
    for j in range(len(astrObjs)):
        sns.distplot(idata[idata['class']==astrObjs[j]][featurecols[i]], hist = False, kde = True, color = colours[j],
                     kde_kws = {'shade': True, 'linewidth': 3}, label = astrObjs[j])
    plt.legend()
    plt.title('Density Plot')
    plt.xlabel(featurecols[i])
    plt.ylabel('Density')
plt.tight_layout()
```

CODE FOR CORR PLOT

```
# corr plot
f, ax = plt.subplots(figsize=(20, 10))
corr = idata.corr("pearson")
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax,annot=True)
```