PROJECT REPORT

ON

EDA AND CLASSIFICATION IN ASTRONOMY

Submitted to Vellore Institute of Technology

By

ANANTHU RAJ CM
(Roll No: 21MDT0020)

PARVINDER KAUR
(Roll No: 21MDT0131)

Under the guidance of

Dr. Rushi Kumar B
Associate Professor, SAS
VIT, Vellore

Course Code: CSE 5007
Course Title: Exploratory Data Analysis
J Component

SCHOOL OF ADVANCED SCIENCES

2021-2023

DECLARATION

We **Ananthu Raj C M** and **Parvinder Kaur,** hereby declare that the project work entitled "**EDA and Classification in Astronomy**" submitted to Vellore Institute of Technology, Vellore is a record of the original work done by us and this project work is presented in the partial fulfilment of the requirements for the award of the Degree of Master of Science in Data Science and is a record of J- component of project work carried out by us under the guidance of **Prof. Rushi Kumar B**. The results embodied in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

Place **: Vellore**

Date **:** 04/06/2022

Signature of Faculty :

Signature of Candidates

ANANTHU RAJ CM

PARVINDER KAUR

ACKNOWLEDGEMENT

We are deeply indebted to all those who helped us directly or indirectly. Our deepest thanks to **Dr. Rushi Kumar** , **Department of Mathematics, SAS, VIT** for guiding and correcting us. He has taken utmost care to go through the project and make necessary amendments as and when needed. We extend thanks to our classmates and professors of **VIT** for their continued support and inputs.

VELLORE. **ANANTHU RAJ CM**

PARVINDER KAUR

CONTENTS

ABSTRACT

To perform Exploratory Data Analysis on Sloan Digital Sky Survey (SDSS) Data Release 17 (DR17). This catalogue contains photometric data for all objects viewed through a telescope and spectroscopic data for a small part of these. Train ML classification models on the labelled Photometric data which is spectroscopically classified with labels. The EDA helps to choose the right models for the dataset. All the information gained from conducting EDA to help you choose a data model. We will use the trained models which have the highest accuracy on the training set , on new unclassified data. Various machine learning algorithms will be used to predict three classes - stars , galaxies , quasars. We will consider KNN , SVM , random forest algorithms for classification . Scaling of data will be done to get better results. The results will help astronomers and astrophysicists carry out further studies.

CHAPTER I
INTRODUCTION

INTRODUCTION

The classification scheme of galaxies, quasars, and stars is one of the most fundamental in Astronomy. **Quasars** are the brightest and most distant objects in the universe. The biggest **groups of stars** are called **galaxies**. The Dataset has around 500,000 observations (rows) of space. Every observation is described by 18 feature columns in which 1 column i.e. class column is a dependent variable or target variable. And we have to determine the outcome of the class column, i.e. whether it is a star, Galaxy or Quasar.

The SDSS gives a three-dimensional picture of the universe through a volume one hundred times larger than that explored to date. Data Release 17 (DR17) is the final data release of the fourth phase of the Sloan Digital Sky Survey (SDSS-IV). DR17 contains SDSS observations through January 2021. The SDSS records the distances to 100,000 quasars, the most distant objects known, giving us an unprecedented hint at the distribution of matter to the edge of the visible universe. The SDSS is the first large-area survey to use electronic light detectors, so the images it produces will be substantially more sensitive and accurate than earlier surveys, which relied on photographic plates. The results of the SDSS are electronically available to the scientific community and the general public, both as images and as precise catalogues of all objects discovered. By the end of the survey, the total quantity of information produced, about 15 terabytes.

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 dataset offers plenty of information about space to explore. Also, the class column is the perfect target for classification practices.

The classification models under consideration are :

1. k-Nearest Neighbours.
2. Decision Trees.

3. Naive Bayes.
4. Random Forest.
5. Gradient Boosting.
6. SVM

This study documents the importance of data and its interpretation in physics through familiarizing with state-of-the-art tools available in technology. The way we see data has changed over the centuries and yet the fundamentals of analysis remain more or less the same. Physics and science in general have always been a data driven field, from the early philosophers and astronomers looking up at the sky and tabulating the positions of planets and constellations.

Millions of petabytes of scientific data are being produced every single day and most of it is made available to the public through data sets and libraries.
Data which were inaccessible before the advent of internet is now within the reach of anyone with a decent internet connection.

One can access terabytes of astronomical data sitting in their bedroom. Apply your own algorithms and regression on that data set and find out patterns and anomalies.
It can even lead to the discovery of new galaxy clusters and planets.

Data in this century is like oil. It's the raw material that drives most of the cutting-edge research as well as technology, but unlike oil, data is in no shortage and its pretty certain that its amount will only increase exponentially. To utilize oil, we make use of drilling, refineries, transportation and eventual combustion in engines. Analogously we can think of tools for utilizing this surplus of scientific data.

For most of the human scientific journey Maths and statistics were the only tools at a physicist's disposal but the onset of transistor advancement has led to such a huge leap that programming surpasses Maths in most cases involving large data sets. Languages like Python and R are the modern equivalent of introduction of Calculus by Sir Isaac Newton. Having such versatile tools in one's arsenal will give further insight and ease of working with oceans of data that we can access now.

The aim of this study is to appreciate and comprehend some of the latest tools available in Machine Learning and Deep learning which can be used for analysing physical data in Physics. For this we select some publicly available datasets and run our analysis on them and through the process, learn and appreciate the superiority of combining and utilising technology in science.

The DataSet

The DataSets used this study serve as the raw material on which the work is done. Its from publicly available sources and in fact the creators of the data encourages the scientific as well as the non-scientific community to access, explore and apply their intuitions using Machine learning and statistical tools.

Sloan Digital Sky Survey

What is the Sloan Digital Sky Survey?

The survey will map one-quarter of the entire sky in detail, determining the positions and absolute brightness's of hundreds of millions of celestial objects. It will also measure the distances to more than a million galaxies and quasars.

The SDSS addresses fascinating, fundamental questions about the universe. With the survey, astronomers will be able to see the large-scale patterns of galaxies: sheets and voids through the whole universe. Scientists have many ideas about how the universe evolved, and different patterns of large-scale structure point to different theories. The Sloan Digital Sky Survey will tell us which theories are right - or whether we will have to come up with entirely new ideas.

Mapping the Universe

Making maps is an activity central to the step-by-step advance of human knowledge. The last decade has seen an explosion in the scale and diversity of the mapmaking enterprise, with fields as disparate as genetics, oceanography, neuroscience, and surface physics applying the power of computers to recording and understanding enormous and complex new territories. The ability to record and digest immense quantities of data in a timely way is changing the face of science. The Sloan Digital Sky Survey will bring this modern practice of comprehensive mapping to cosmography, the science of mapping and understanding the universe.

The SDSS will make the largest map in human history. It will give us a three-dimensional picture of the universe through a volume one hundred times larger than that explored to date. The SDSS will also record the distances to 100,000 quasars, the most distant objects known, giving us an unprecedented hint at the distribution of matter to the edge of the visible universe. The SDSS is the first large-area survey to use electronic light detectors, so the images it

produces will be substantially more sensitive and accurate than earlier surveys, which relied on photographic plates. The results of the SDSS are electronically available to the scientific community and the general public, both as images and as precise catalogues of all objects discovered. By the end of the survey, the total quantity of information produced, about 15 terabytes (trillion bytes), will rival the information content in all the books of the Library of Congress.

By systematically and sensitively observing a large fraction of the sky, the SDSS will have a significant impact on astronomical studies as diverse as the large-scale structure of the universe, the origin and evolution of galaxies, the relation between dark and luminous matter, the structure of our own Milky Way, and the properties and distribution of the dust from which stars like our sun were created. The SDSS will be a new reference point, a field guide to the universe that will be used by scientists for decades to come.

The Science of the SDSS

The universe today is filled with sheets of galaxies that curve through mostly empty space. Like soap bubbles in a sink, they form into dense filaments with voids between. Our best model for how the universe began, the Big Bang, gives us a picture of a universe filled with a hot, uniform soup of fundamental particles. Somehow, between the time the universe began and today, gravity has pulled together the matter into regions of high density, leaving behind voids. What triggered this change from uniformity to structure? Understanding the origin of the structure we see in the universe today is a crucial part of reconstructing our cosmic history.

Understanding the arrangement of matter in the universe is made more difficult because the luminous stars and galaxies that we see are only a small part of the total. More than 90% of the matter in the universe does not give off light. The nature, amount and distribution of this "dark matter" are among the most important questions in astrophysics. How has the gravity from dark matter influenced visible structures? Or, put another way, we can use careful mapping of the positions and motions of galaxies to reconstruct the distribution of mass, and from that, we can find clues about dark matter.

A Map of the Universe

One of the difficulties in studying the entire universe is getting enough information to make a picture. Astronomers designed the Sloan Digital Sky Survey to address this problem in

a direct and ambitious way: the SDSS gathers a body of data large enough and accurate enough to address a broad range of astronomical questions.

The SDSS will obtain high-resolution pictures of one quarter of the entire sky in five different colours. From these pictures, advanced image processing software will measure the shape, brightness, and colour of hundreds of millions of astronomical objects including stars, galaxies, quasars (compact but very bright objects thought to be powered by material falling into giant black holes), and an array of other celestial exotica. Selected galaxies, quasars, and stars will be observed using an instrument called a spectrograph to determine accurate distances to a million galaxies and 100,000 quasars, and to provide a wealth of information about the individual objects. These data will give the astronomical community one of the things it needs most: a comprehensive catalogue of the constituents of a representative part of the universe. SDSS's map will reveal how big the largest structures in our universe are, and what they look like. It will help us understand the mechanisms that converted a uniform "primordial soup" into a frothy network of galaxies.

An Intergalactic Census

The U.S. Census Bureau collects statistical information about how many people live in the U.S., where they live, their races, their family incomes, and other characteristics. The Census becomes a primary source of information for people trying to understand the nation. The Sloan Digital Sky Survey will conduct a sort of celestial census, gathering information about how many galaxies and quasars the universe contains, how they are distributed, their individual properties, and how bright they are. Astronomers will use this information to study questions such as why flat spiral galaxies are found in less dense regions of the universe than football-shaped elliptical galaxies, or how quasars have changed during the history of the universe.

The SDSS will also collect information about the Milky Way galaxy and even about our own solar system. The wide net cast by the SDSS telescope will sweep up as many stars as galaxies, and as many asteroids in our solar system as quasars in the universe. Knowledge of these objects will help us learn how stars are distributed in our galaxy, and where asteroids fit into the history of our solar system.

Needles in a Haystack, Lighthouses in the Fog

Rare objects, almost by definition, are scientifically interesting. By sifting through the several hundred million objects recorded by the SDSS, scientists will be able to construct entire catalogues of the most distant quasars, the rarest stars, and the most unusual galaxies. The most unusual objects in the catalogue will be about a hundred times rarer than the rarest objects now known.

For example, stars with a chemical composition very low in metals like iron are the oldest in the Milky Way. They can therefore tell us about the formation of our galaxy. However, such stars are also extremely rare, and only a wide-field deep sky survey can find enough of them to form a coherent picture.

Because they are so far away, quasars can serve as probes for intergalactic matter throughout the visible universe. In particular, astronomers can identify and study galaxies by the way they block certain wavelengths of light emitted by quasars behind them. Using the light from quasars, the SDSS will detect tens of thousands of galaxies in the initial stages of formation. These galaxies are typically too faint and too diffuse for their own light to be detected by even the largest of telescopes. Quasar probes will also allow scientists to study the evolution of the chemistry of the universe throughout its history.

The Telescope as a Time Machine

Peering into the universe with a telescope allows us to look not only out into space, but also back in time. Imagine intelligent beings in a planetary system around a star 20 light years away. Suppose these beings pick up a stray television transmission from Earth. They would see events 20 years after they occurred on Earth: for instance, a newscast covering Ronald Reagan's re-election (1984) would be seen 20 years later (2004). While today we have seen three new presidents, the beings would still see Reagan.

Light travels extremely fast, but the universe is a very big place. In fact, astronomers routinely look at quasars so far away that it takes billions of years for the light they produce to reach us. When we look at galaxies or quasars that are billions of light-years away, we are seeing them as they were billions of years ago.

By looking at galaxies and quasars at different distances, astronomers can see how their properties change with time. The SDSS will measure the distribution of nearby galaxies, allowing astronomers to compare them with more distant galaxies now being seen by the new instruments like the Hubble Space Telescope and the Keck Telescope. Because quasars are

very bright, the SDSS will allow astronomers to study their evolution through more than 90 percent of the history of the universe.

Measuring Distance and Time: Redshift

The universe is expanding like a loaf of raisin bread rising in an oven. Pick any raisin, and imagine that it's our own Milky Way galaxy. If you place yourself on that raisin, then no matter how you look at the loaf, as the bread rises, all the other raisins move away from you. The farther away another raisin is from you, the faster it moves away. In the same way, all the other galaxies are moving away from ours as the universe expands. And because the universe is uniformly expanding, the farther a galaxy is from Earth, the faster it is receding from us.

The light coming to us from these distant objects is shifted toward the red end of the electromagnetic spectrum, in much the same way the sound of a train whistle changes as a train leaves or approaches a station. The faster a distant object is moving, the more it is redshifted. Astronomers measure the amount of redshift in the spectrum of a galaxy to figure out how far away it is from us.

By measuring the redshifts of a million galaxies, the Sloan Digital Sky Survey will provide a three-dimensional picture of our local neighbourhood of the universe.

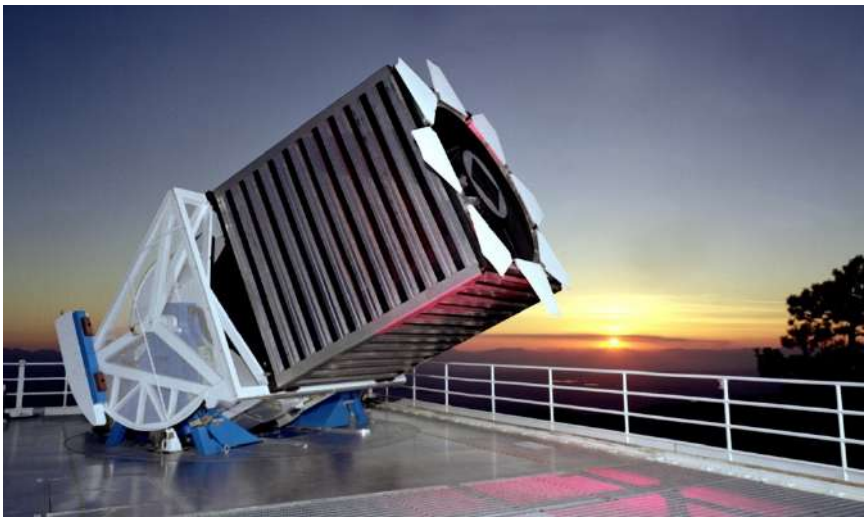
The SDSS Telescopes

Before astronomers can make a map of the sky, they need a telescope. Past surveys, such as the Palomar Sky Survey, were done with Schmidt telescopes with correcting lenses 48 inches (1.5 m) across. To map more distant, fainter objects, Sloan astronomers decided to build a brand new telescope with lenses 2.5 meters (100 inches) across.

Apache Point Observatory

The SDSS telescopes are located at Apache Point Observatory (APO) in Sunspot, New Mexico. The observatory is surrounded by the Lincoln National Forest in the Sacramento Mountains, and sits on a mountain 9,200 feet above sea level, where the atmosphere contains little water vapor and few pollutants. Because the site is so high and so far from major cities, the night sky seen from APO is among the darkest in the United States.

In addition to the SDSS telescopes, the APO also houses a 3.5-meter telescope and **New Mexico State University**'s 1.0-meter telescope.



The SDSS telescope outside its housing.

The SDSS Data

On a clear, dark night, light that has travelled through space for a billion years touches a mountaintop in southern New Mexico and enters the sophisticated instrumentation of the SDSS's 2.5-meter telescope. The light ceases to exist as photons, but the data within it lives on as digital images recorded on magnetic tape. Each image is composed of myriad pixels (picture elements); each pixel captures the brightness from each tiny point in the sky.

But the sky is not made of pixels. The task of data managers for the Sloan Digital Sky Survey is to take digitized data - the pixels electronically encoded on the mountaintop in New Mexico - and turn them into real information about real things. Astronomers process the data to produce information they can use to identify and measure properties of stars and galaxies. Astronomers must be able to find, distinguish, and measure the brightness of celestial objects, and then collect the stars, galaxies, and quasars into a catalogue.

Scientists at Fermilab have led the effort to develop what the SDSS calls data-processing pipelines. A pipeline is a computer program that processes digitized data automatically to extract certain types of information. The term "pipeline" connotes the automated nature of the data processing; the data "flow" through the pipelines with little human intervention. For example, the astrometric pipeline, built by computer scientists at the U.S. Naval Observatory, determines the precise absolute two-dimensional position of stars and galaxies in the sky. In this case, digitized data from photons reaching the 2.5-meter telescope go in one end of the astrometric pipeline, and object positions come out the other. In between, along the length of the pipeline, software changes pixels into real information.

The data pipelines are a collaborative effort. **Princeton University** scientists built the photometric pipeline, and **University of Chicago** scientists created the spectroscopic pipeline.

Fermilab's contributions include the monitor-telescope pipeline and the pipeline that selects candidates for the spectroscopic survey. Fermilab also coordinates the smooth operation of all the pipelines.

Information processing for the SDSS begins when the CCDs collect light. Charge "buckets" are converted to digitized signals and written to tape at the observatory. The tapes travel from **Apache Point** to Fermilab by express courier. The tapes go to Fermilab's Feynman Computing Center, where their data are read and sent into various pipelines: spectrographic data into the spectrographic pipeline, monitor telescope data into the monitor pipeline, and imaging data into the astrometric, photometric, target selection, and two other pipelines. Information about stars, galaxies and quasars comes out of the pipeline. This information is included in the Operations Database, written at Fermilab and at the Naval Observatory, which collects information needed to keep the Sky Survey running.

Eventually, experimenters will pass the information in the Operations Database to the science database developed by scientists at **Johns Hopkins University**. The science database will make the data readily available to scientists on the project.

The Databases

The processed data are stored in databases. The logical database design consists of photographic and spectrographic objects. They are organized into a pair of snowflake schemas. Sub setting views and many indices give convenient access to the conventional subsets (such as stars and galaxies). Procedures and indices are defined to make spatial lookups convenient and fast.

Database Physical Design

SkyServer initially took a simple approach to database design – and since that worked, we stopped there. The design counts on the SQL storage engine and query optimizer to make all the intelligent decisions about data layout and data access.

The total amount of data in the two databases is 818 GB, and the total number of rows exceeds 3.4 billion. The data tables are all created in several filegroups. The database files are spread across a single RAID0 volume. Each filegroup contains several database files that are limited to about 50Gb each. The log files and temporary database are also spread across these disks. SQL Server stripes the tables across all these files and hence across all these disks. It detects the sequential access, creates the parallel prefetch threads, and uses multiple processors to analyze the data as quickly as the disks can produce it. When reading or writing, this automatically gives the sum of the disk bandwidths (over 400 MBps peak, 180MBps typical) without any special user programming.

Filegroups	BESTDR1	TARGDR1
data	1	200
PhotoOther	18.1	
PhotoObjAll	165.4	
PhotoTag	78.1	73.7
PhotoTagIndex	53.6	
PhotoObjIndex	66.3	
PhotoObjProfile	80	
PhotoObjMask	22	17.2
SpecObj	6	
Neighbors	24.2	
Frame	30	30
Log	4.2	2
Total	495.3	322.9

Count of records and bytes in major tables. Indices approximately double the space.

Beyond this file group striping, SkyServer uses all the SQL Server default values. There is no special tuning. This is the hallmark of SQL Server – the system aims to have "no knobs" so that the out-of-the box performance is quite good. The SkyServer is a testimonial to that goal.

The focus of our study regarding SDSS

The above section gave a broad overview about the objectives and methods used in Sloan Digital Sky Survey. We use only a small subset of the main dataset and our algorithms and visualizations will be done in that subset. The data released by the SDSS is under public domain. It's taken from the current data release DR17.

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.

Inspiration: The dataset offers plenty of information about space to explore. Also, the class column is the perfect target for classification practices.

Feature Description

The table results from a query which joins two tables (actually views): "PhotoObj" which contains photometric data and "SpecObj" which contains spectral data.

The feature descriptions are as below:

View "PhotoObj"

- objid = Object Identifier
- ra = J2000 Right Ascension (r-band)
- dec = J2000 Declination (r-band)

Right ascension (abbreviated RA) is the angular distance measured eastward along the celestial equator from the Sun at the March equinox to the hour circle of the point above the earth in question. When paired with declination (abbreviated dec), these astronomical coordinates specify the direction of a point on the celestial sphere (traditionally called in English the skies or the sky) in the equatorial coordinate system.

Source: https://en.wikipedia.org/wiki/Right_ascension

- u = better of DeV/Exp magnitude fit
- g = better of DeV/Exp magnitude fit
- r = better of DeV/Exp magnitude fit
- i = better of DeV/Exp magnitude fit
- z = better of DeV/Exp magnitude fit

The Thuan-Gunn astronomic magnitude system. u, g, r, i, z represent the response of the 5 bands of the telescope.

Further education: https://www.astro.umd.edu/~ssm/ASTR620/mags.html

- run = Run Number
- rereun = Rerun Number
- camcol = Camera column
- field = Field number

Run, rerun, camcol and field are features which describe a field within an image taken by the SDSS. A field is basically a part of the entire image corresponding to 2048 by 1489 pixels. A field can be identified by:

- run number, which identifies the specific scan,
- the camera column, or "camcol," a number from 1 to 6, identifying the scanline within the run, and

- the field number. The field number typically starts at 11 (after an initial rampup time), and can be as large as 800 for particularly long runs.
- An additional number, rerun, specifies how the image was processed.

View "SpecObj"

- specobjid = Object Identifier
- class = object class (galaxy, star or quasar object)

The class identifies an object to be either a galaxy, star or quasar. This will be the response variable which we will be trying to predict.

- redshift = Final Redshift
- plate = plate number
- mjd = MJD of observation
- fiberid = fiber ID

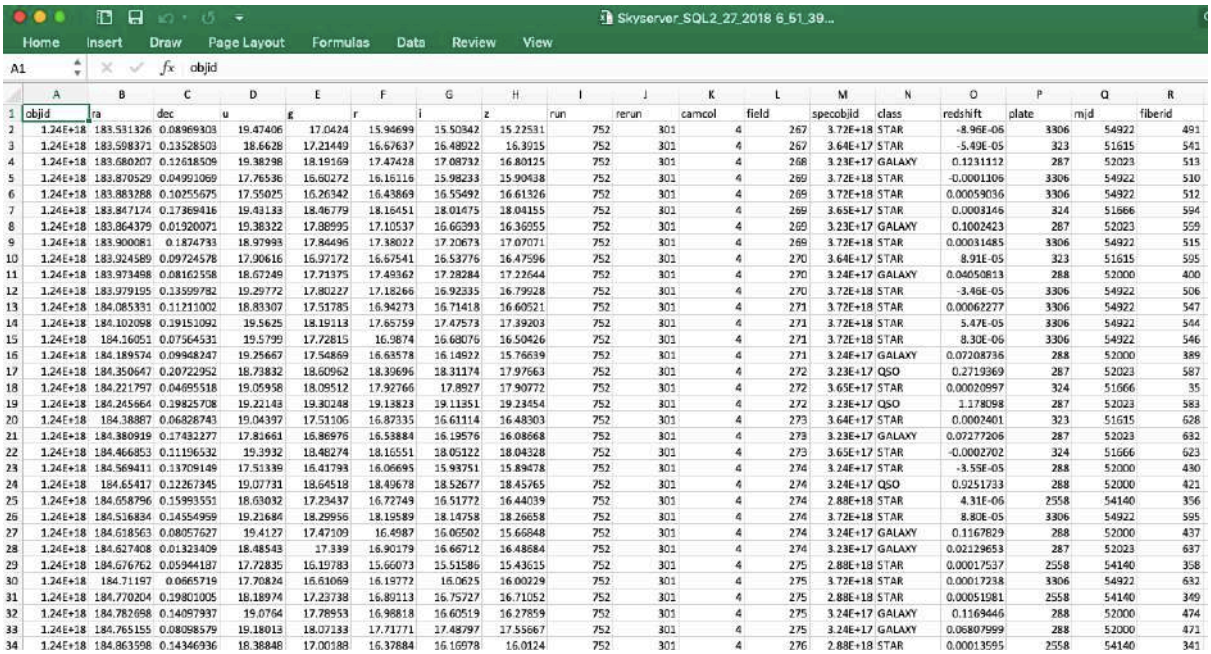
In physics, redshift happens when light or other electromagnetic radiation from an object is increased in wavelength, or shifted to the red end of the spectrum.

Each spectroscopic exposure employs a large, thin, circular metal plate that positions optical fibers via holes drilled at the locations of the images in the telescope focal plane. These fibers then feed into the spectrographs. Each plate has a unique serial number, which is called plate in views such as SpecObj in the CAS.

Modified Julian Date, used to indicate the date that a given piece of SDSS data (image or spectrum) was taken.

The SDSS spectrograph uses optical fibers to direct the light at the focal plane from individual objects to the slithead. Each object is assigned a corresponding fiberID.

The DataSet for our use can be downloaded as a CSV file which can be opened using excel. CSV stands for ‘comma separated values’ and it’s a format for storing tabular data. The DataSet will have 100,000 entries ad 17 feature columns and 1 class column. This can be seen while opening the file.



Skyserver_SQL2_27_2018 6_51_39...

Home Insert Draw Page Layout Formulas Data Review View

A1 objid

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	redshift	plate	mjd	fiberid
2	1.24E+18	183.531326	0.08969808	19.47406	17.0424	15.94699	15.50342	15.22531	752	301	4	267	3.72E+18	STAR	-8.96E-06	3306	54922	491
3	1.24E+18	183.598371	0.13528503	18.6628	17.21449	16.67637	16.48922	16.3015	752	301	4	267	3.64E+17	STAR	-5.49E-05	323	51615	541
4	1.24E+18	183.680207	0.12618509	19.38298	18.19169	17.47428	17.08732	16.80125	752	301	4	268	3.23E+17	GALAXY	0.1231112	287	52023	513
5	1.24E+18	183.870529	0.04991069	17.76536	16.60272	16.16116	15.98233	15.90438	752	301	4	269	3.72E+18	STAR	-0.0001106	3306	54922	510
6	1.24E+18	183.883288	0.10255675	17.55025	16.26342	16.43869	16.55492	16.61326	752	301	4	269	3.72E+18	STAR	0.00059036	3306	54922	512
7	1.24E+18	183.847174	0.17369416	19.43133	18.46779	18.16451	18.01475	18.04155	752	301	4	269	3.65E+17	STAR	0.0003146	324	51666	594
8	1.24E+18	183.864379	0.01920071	19.38322	17.88995	17.10537	16.66393	16.36955	752	301	4	269	3.23E+17	GALAXY	0.1002423	287	52023	559
9	1.24E+18	183.900081	0.1874733	18.97993	17.86496	17.38022	17.20673	17.07071	752	301	4	269	3.72E+18	STAR	0.00031485	3306	54922	515
10	1.24E+18	183.924589	0.09724578	17.90616	16.97172	16.67541	16.53776	16.47596	752	301	4	270	3.64E+17	STAR	8.91E-05	323	51615	595
11	1.24E+18	183.973498	0.08162558	18.67249	17.71375	17.49362	17.28284	17.22644	752	301	4	270	3.24E+17	GALAXY	0.04050813	288	52000	400
12	1.24E+18	183.979195	0.13599782	19.29772	17.80227	17.18266	16.92335	16.79928	752	301	4	270	3.72E+18	STAR	-3.46E-05	3306	54922	506
13	1.24E+18	184.085331	0.11211002	18.83307	17.51785	16.94273	16.71418	16.60521	752	301	4	271	3.72E+18	STAR	0.00062277	3306	54922	547
14	1.24E+18	184.102098	0.19151092	19.5625	18.19113	17.65759	17.47573	17.39203	752	301	4	271	3.72E+18	STAR	5.47E-05	3306	54922	544
15	1.24E+18	184.16051	0.07564531	19.5799	17.72815	16.9874	16.68076	16.50426	752	301	4	271	3.72E+18	STAR	8.30E-06	3306	54922	546
16	1.24E+18	184.189574	0.09948247	19.25667	17.54869	16.63578	16.14922	15.76639	752	301	4	271	3.24E+17	GALAXY	0.07208736	288	52000	389
17	1.24E+18	184.350647	0.20722952	18.73832	18.60962	18.39696	18.31174	17.97663	752	301	4	272	3.23E+17	QSO	0.2719369	287	52023	587
18	1.24E+18	184.221797	0.04695518	19.05958	18.09512	17.92766	17.8927	17.90772	752	301	4	272	3.65E+17	STAR	0.00020997	324	51666	35
19	1.24E+18	184.245664	0.19825708	19.22143	19.30248	19.13823	19.11351	19.23454	752	301	4	272	3.23E+17	QSO	1.178098	287	52023	583
20	1.24E+18	184.38887	0.06828743	19.04397	17.51106	16.87335	16.61114	16.48303	752	301	4	273	3.64E+17	STAR	0.0002401	323	51615	628
21	1.24E+18	184.380919	0.17432277	17.81661	16.86976	16.53884	16.19576	16.08668	752	301	4	273	3.23E+17	GALAXY	0.07277206	287	52023	632
22	1.24E+18	184.466853	0.11196532	19.3932	18.48274	18.16551	18.05122	18.04328	752	301	4	273	3.65E+17	STAR	-0.0002702	324	51666	623
23	1.24E+18	184.569411	0.13709149	17.51339	16.41793	16.06695	15.93751	15.89478	752	301	4	274	3.24E+17	STAR	-3.55E-05	288	52000	430
24	1.24E+18	184.65417	0.12267345	19.07731	18.64518	18.49678	18.52677	18.45765	752	301	4	274	3.24E+17	QSO	0.9251733	288	52000	421
25	1.24E+18	184.658796	0.15993551	18.63032	17.23437	16.72749	16.51772	16.44039	752	301	4	274	2.88E+18	STAR	4.31E-06	2558	54140	356
26	1.24E+18	184.516834	0.14554959	19.21684	18.29956	18.19589	18.14758	18.26658	752	301	4	274	3.72E+18	STAR	8.80E-05	3306	54922	595
27	1.24E+18	184.618563	0.08057627	19.4127	17.47109	16.4987	16.06502	15.66848	752	301	4	274	3.24E+17	GALAXY	0.1167829	288	52000	437
28	1.24E+18	184.627408	0.01323409	18.48543	17.339	16.90179	16.66712	16.48684	752	301	4	274	3.23E+17	GALAXY	0.02129653	287	52023	637
29	1.24E+18	184.676762	0.05944187	17.72835	16.19783	15.66073	15.51586	15.43615	752	301	4	275	2.88E+18	STAR	0.00017537	2558	54140	358
30	1.24E+18	184.71197	0.0665719	17.70824	16.61069	16.19772	16.0625	16.00229	752	301	4	275	3.72E+18	STAR	0.00017238	3306	54922	632
31	1.24E+18	184.770204	0.19801005	18.18974	17.23738	16.89113	16.75727	16.71052	752	301	4	275	2.88E+18	STAR	0.00051981	2558	54140	349
32	1.24E+18	184.782698	0.14097937	19.0764	17.78953	16.98818	16.60519	16.27859	752	301	4	275	3.24E+17	GALAXY	0.1169446	288	52000	474
33	1.24E+18	184.765155	0.08098579	19.18013	18.07133	17.71771	17.48797	17.55667	752	301	4	275	3.24E+17	GALAXY	0.06807999	288	52000	471
34	1.24E+18	184.863598	0.14346936	18.38848	17.00188	16.37884	16.16978	16.0124	752	301	4	276	2.88E+18	STAR	0.00013595	2558	54140	341

The excel presentation of the small subset of SDSS data that will be used for this study. Notice the columns with the features as described in the previous section. The class column identifies what type each entry is. It is the target class which will be the reference of our algorithms.

CHAPTER II
LITERATURE REVIEW

LITERATURE REVIEW

Acharya, Vishwanath, et al. "Classification of SDSS Photometric Data Using Machine Learning on a Cloud." *Current Science*, vol. 115, no. 2, Current Science Association, 2018, pp. 249–57, https://www.jstor.org/stable/26978189.

Abstract :

Astronomical datasets are typically very large, and manually classifying the data in them is effectively impossible. We use machine learning algorithms to provide classifications (as stars, quasars and galaxies) for more than one billion objects given photometrically in the Third Data Release of the Sloan Digital Sky Survey (SDSS III). Although it is possible to classify all the objects using spectroscopic data, it is impractical to obtain such data for each one of them. To classify such a big dataset on a single machine would be impractically slow, so have used the Spark cluster computing framework to implement a distributed computing environment over the cloud.

Gap and scope of improvement :

The above paper uses Data Release III, but we are going to use the latest and final addition to the SDSS Catalogues which is the DR17. This will be a superset of all the previous data releases giving a much more consistent and cleaner set of data points. Data Release 17 (DR17) is the final data release of the fourth phase of the Sloan Digital Sky Survey (SDSS-IV). DR17 contains SDSS observations through January 2021.

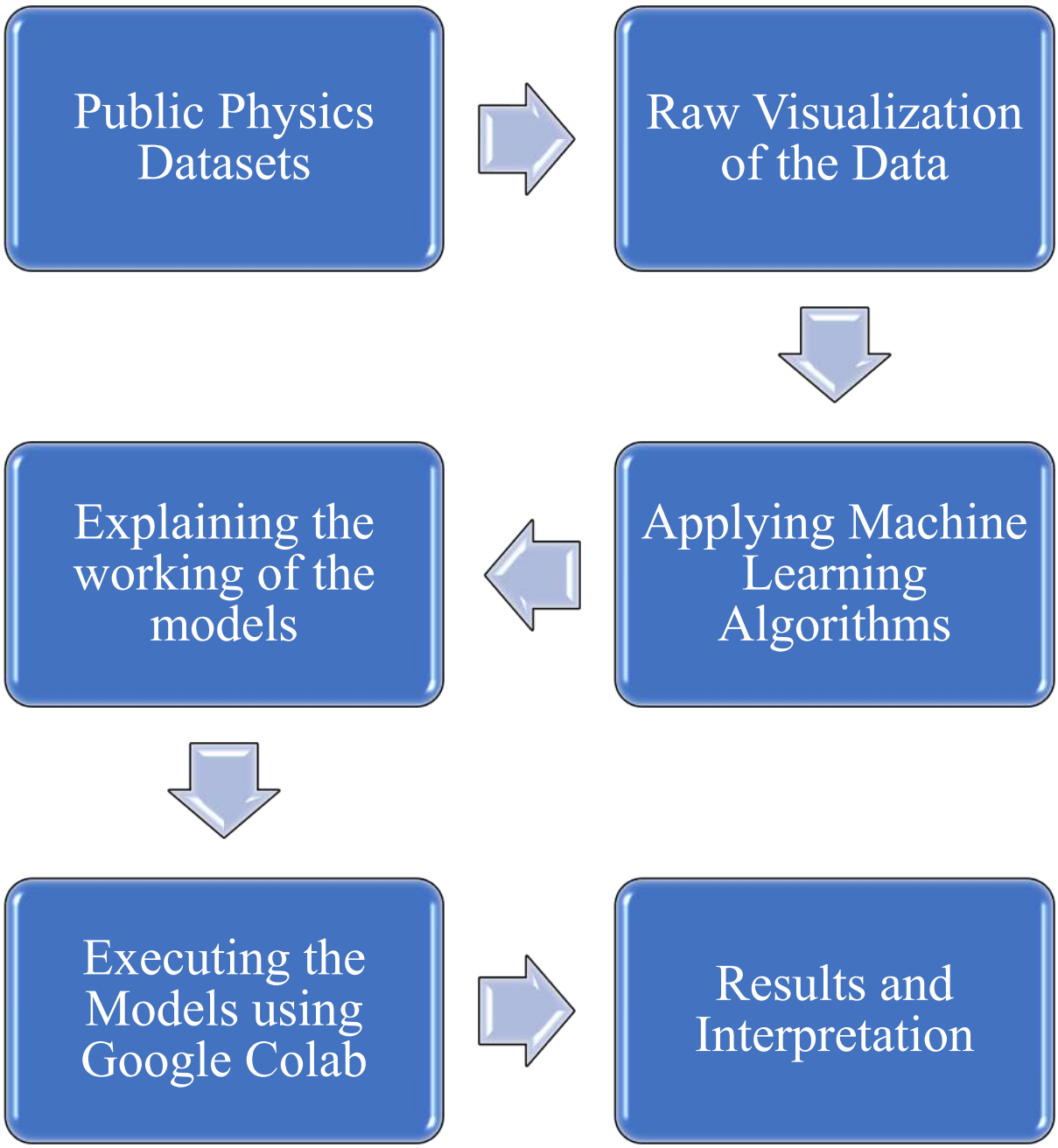
SDSS data releases are cumulative, so DR17 includes all the sky coverage of prior releases.

We will be using more fine-tuned ML classification models and find the one with the maximum accuracy.

CHAPTER III
PROBLEM FORMULATION

METHODOLOGY

The modus operandi is to use some publicly datasets and apply several machine learning models on them. The inner workings of those ML models are rooted in Mathematics and statistics whereas the data that we use is the nexus with Physics.



BASIC STRUCTURE OF ANALYSIS

1. **CONFIGURING THE ENVIRONMENT**
   - 1.1 Importing required libraries
2. **DATA PREPARATION**
   - 2.1 Loading the Data
   - 2.2 Dropping Features that are not significant
3. **HIGH LEVEL STATISTICS**
   - 3.1 Describing the Data
   - 3.2 Finding Duplicates
   - 3.3 Finding Unique Values
   - 3.4 Finding Null Values
4. **UNIVARIATE ANALYSIS**
   - 4.1 BoxPlot
   - 4.2 Frequency Polygon
   - 4.3 Histplot
   - 4.4 Probability Density Function (PDF)
   - 4.5 Violin Plot
   - 4.6 KDE Plot
5. **MULTIVARIATE ANALYSIS**
   - 5.1 LinePlot
   - 5.2 Correlation between variables
   - 5.3 3D Scatter Plot
   - 5.4 Pairplots
   - 5.5 Pandas Profiling
   - 5.6 Equatorial Coordinates Plot
6. **ML Algorithms**
   - 6.1 Feature Engineering
   - 6.2 Logistic Regression
   - 6.3 KNN
   - 6.4 Naïve Bayes
   - 6.5 Random Forest
   - 6.7 SVM
   - 6.8 Multi-Layer Perceptron Classifier (Neural Network)
   - 6.9 Decision Tree

CODE – PYTHON JUPYTER NOTEBOOK

The Analysis on SDSS:

From SDSS web page,

"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."

About the data:

From the Overview section,

"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 Analysis below, tries to study the different features of star, galaxy and quasar and uses Support Vector Machine(SVM) algorithm to predict the three classes *STAR, GALAXY OR QUASAR.*

Quasar- Also known as quasi-stellar object, it is an extremely luminous active galactic nucleus (AGN). The power radiated by quasars is enormous. The most powerful quasars have luminosities exceeding 1041 watts, thousands of times greater than an ordinary large galaxy such as the Milky Way.

Galaxy- A galaxy 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 supermassive black holes at their a

EDA AND CLASSIFICATION IN ASTRONOMY

1. Configuring the Environment

Importing required libraries

In [39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm.notebook import tqdm_notebook
import timeit
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import chi2, mutual_info_classif, SelectKBest, f_classif
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier, GradientBoostingClassifier

import warnings
warnings.filterwarnings('ignore')

#!pip install plotly
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

2. Data Preparation

2.1 Loading the data

In [2]:
```
%%time
df = pd.read_csv('1_lakh_entries.csv', skiprows = 0)
df
```

```
Wall time: 713 ms
```

Out[2]:

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	redshift	pl
0	1.237650e+18	210.016478	3.924162	19.58006	17.70593	16.74910	16.33185	15.97441	2190	301	2	187	9.637900e+17	GALAXY	0.075129	
1	1.237660e+18	137.468475	43.878875	19.29924	18.14330	17.71516	17.47766	17.43733	2777	301	3	207	9.368170e+17	GALAXY	0.093499	
2	1.237650e+18	14.123114	15.142815	19.28338	17.53712	16.72810	16.32265	16.04339	1904	301	4	242	4.740990e+17	GALAXY	0.078770	
3	1.237670e+18	266.621865	26.717041	18.74441	17.24021	16.67171	16.48060	16.37055	4828	301	6	116	2.457960e+18	STAR	-0.000124	2
4	1.237670e+18	159.771218	22.902207	18.18608	18.04931	17.75193	17.73256	17.78253	5137	301	4	308	7.235190e+18	QSO	1.088803	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
99995	1.237650e+18	235.711610	0.032597	19.58198	17.75248	16.82558	16.40437	16.04969	752	301	4	616	3.546800e+17	GALAXY	0.082767	
99996	1.237650e+18	237.082722	0.185409	19.38675	17.96262	17.14279	16.68619	16.36536	752	301	4	625	3.851970e+17	GALAXY	0.113793	
99997	1.237650e+18	240.853253	0.092452	19.18663	17.25995	16.27921	15.79979	15.42603	752	301	4	650	3.874260e+17	GALAXY	0.054825	
99998	1.237650e+18	241.534281	0.013727	19.00692	16.97655	15.93292	15.45320	15.01987	752	301	4	654	3.885430e+17	GALAXY	0.057930	
99999	1.237650e+18	241.737495	0.027826	19.43477	17.37000	16.39084	15.90887	15.56186	752	301	4	656	3.885440e+17	GALAXY	0.074140	

100000 rows × 18 columns

2.2 Dataset Overview

The table results from an SQL query which joins two tables:

"PhotoObj" which contains photometric data.
"SpecObj" which contains spectral data.
16 variables (double) and 1 additional variable (char) 'class'.
A class object can be predicted from the other 16 variables.

Variables description:

objid = Object Identifier **ra** = J2000 Right Ascension (r-band)
dec = J2000 Declination (r-band)
u = better of deV/Exp magnitude fit (u-band)
g = better of deV/Exp magnitude fit (g-band)
r = better of deV/Exp magnitude fit (r-band)
i = better of deV/Exp magnitude fit (i-band)
z = better of deV/Exp magnitude fit (z-band)
run = Run Number
rerun = Rerun Number
camcol = Camera column
field = Field number
specobjid = Object Identifier
class = object class (galaxy, star or quasar object)
redshift = Final Redshift
plate = plate number
mjd = MJD(Modified Julian Date) of observation
fiberid = fiberID

2.3 Basic information

In [3]:
```
df.shape
```

Out[3]:
```
(100000, 18)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 18 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   objid       100000 non-null  float64
 1   ra          100000 non-null  float64
 2   dec         100000 non-null  float64
 3   u           100000 non-null  float64
 4   g           100000 non-null  float64
 5   r           100000 non-null  float64
 6   i           100000 non-null  float64
 7   z           100000 non-null  float64
 8   run         100000 non-null  int64  
 9   rerun       100000 non-null  int64  
 10  camcol      100000 non-null  int64  
 11  field       100000 non-null  int64  
 12  specobjid   100000 non-null  float64
 13  class       100000 non-null  object 
 14  redshift    100000 non-null  float64
 15  plate       100000 non-null  int64  
 16  mjd         100000 non-null  int64  
 17  fiberid     100000 non-null  int64  
dtypes: float64(10), int64(7), object(1)
memory usage: 13.7+ MB
```

Obervations

1. There are 100000 data points and 17 features.
2. There is only one categorical column.

Dropping Features that are not significant

'objid' and 'specobjid' are only identifiers for getting to the rows back when they were put away in the original databank. Along these lines we won't need them for classification as they are not identified with the result.

Significantly more: The features 'run', 'rerun', 'camcol' and 'field' are values which describe portions of the camera right when mentioning the objective fact, for example 'run' speaks to the comparing check which caught the oject.

We'll drop these features.

In [5]:
```
df.drop(['run', 'rerun', 'camcol', 'field', 'objid', 'specobjid', 'fiberid'], axis = 1, inplace= True)
df.head(5)
```

Out[5]:

	ra	dec	u	g	r	i	z	class	redshift	plate	mjd
0	210.016478	3.924162	19.58006	17.70593	16.74910	16.33185	15.97441	GALAXY	0.075129	856	52339
1	137.468475	43.878875	19.29924	18.14330	17.71518	17.47768	17.43733	GALAXY	0.093499	832	52312
2	14.123114	15.142815	19.28338	17.53712	16.72810	16.32265	16.04339	GALAXY	0.078770	421	51821
3	266.621865	26.717041	18.74441	17.24021	16.67171	16.48060	16.37055	STAR	-0.000124	2183	53536
4	159.771218	22.902207	18.18608	18.04931	17.75193	17.73256	17.78253	QSO	1.088803	6426	56334

3. High Level Statistics

3.1 Describing the Data

In [6]:
```
df.describe()
```

Out[6]:

	ra	dec	u	g	r	i	z	redshift	plate	mjc
count	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.00000(
mean	177.517993	25.048529	18.637645	17.407114	16.881728	16.625299	16.466702	0.171041	2597.153180	53918.74686(
std	78.062282	20.544956	0.832263	0.985319	1.132708	1.207941	1.280648	0.439244	2225.962745	1553.44298(
min	0.011306	-19.495456	10.611810	9.668339	9.005167	8.848403	8.947795	-0.004268	266.000000	51608.00000(
25%	136.418824	6.732728	18.211495	16.852083	16.196345	15.864370	15.619682	0.000001	1186.000000	52734.00000(
50%	180.391592	23.962150	18.874240	17.515725	16.890090	16.598420	16.427275	0.045948	2087.000000	53726.00000(
75%	224.650548	40.344539	19.273580	18.056393	17.585740	17.345235	17.235225	0.095516	2911.000000	54589.00000(
max	359.999615	84.490494	19.599990	19.996050	31.990100	32.141470	29.383740	7.011245	12547.000000	58932.00000(

From the above table we can tell that are no missing values at all. This means: no imputing.

In [7]:
```
df.shape
```

Out[7]: (100000, 11)

Now we have only 10 features and 100000 data points since 6 features which were not significant to the class label were dropped.

3.2 Find the duplicates

In [8]:
```
df.duplicated().sum()
```

Out[8]: 0

Observations

There are no duplicates

3.4 Unique Values

In [9]:
```
# We can find the number of unique values in the particular column using unique() function in python.
df['class'].unique()
```

Out[9]:
```
array(['GALAXY', 'STAR', 'QSO'], dtype=object)
```

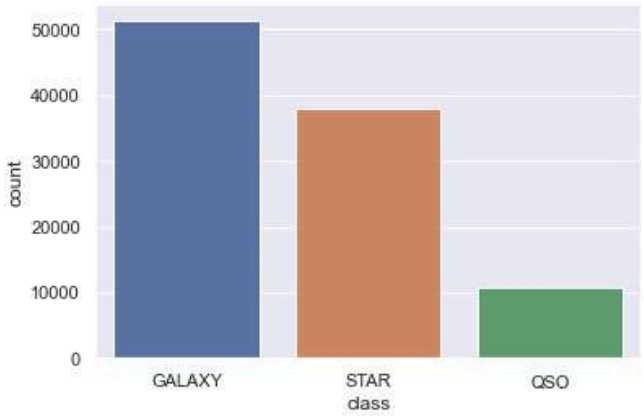
In [10]:
```
df['class'].value_counts()
```

Out[10]:
```
GALAXY    51304
STAR      38091
QSO       10605
Name: class, dtype: int64
```

3.4 Visualizing the Unique counts

In [11]:
```
sns.set_theme(style="darkgrid")
sns.countplot(df['class'])
```

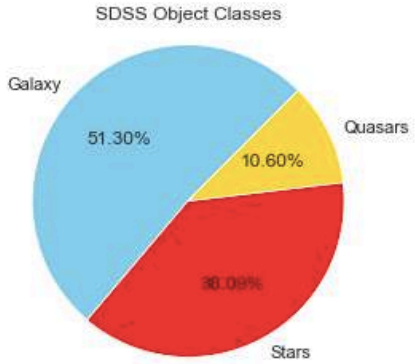
Out[11]:
```
<AxesSubplot:xlabel='class', ylabel='count'>
```



3.4 Observations

1. The column 'class' is the target variable and its a categorical variable. The unique values of the class label are 'GALAXY', 'QSO' and 'STAR'
2. The dataset is quite imbalanced with number of galaxies dominating.

```
    colors = ['skyblue', 'red', 'gold']
    fig1, ax1 = plt.subplots()
    ax1.pie(label_counts, labels=['Galaxy', 'Stars', 'Quasars'],
            autopct='%1.2f%%', startangle=45, colors=colors)
    ax1.axis('equal')
    plt.title('SDSS Object Classes')
    plt.show()
pieChart(df)
```



3.5 Find null values

```
In [13]: df.isnull().sum()
```

```
Out[13]: ra          0
         dec         0
         u           0
         g           0
         r           0
         i           0
         z           0
         class       0
         redshift    0
         plate       0
         mjd         0
         dtype: int64
```

There are no null values.

4. Univariate Analysis

Distribution plots are used to visually assess how the data points are distributed with respect to its frequency. Usually the data points are grouped into bins and the height of the bars representing each group increases with increase in the number of data points lie within that group. (histogram)

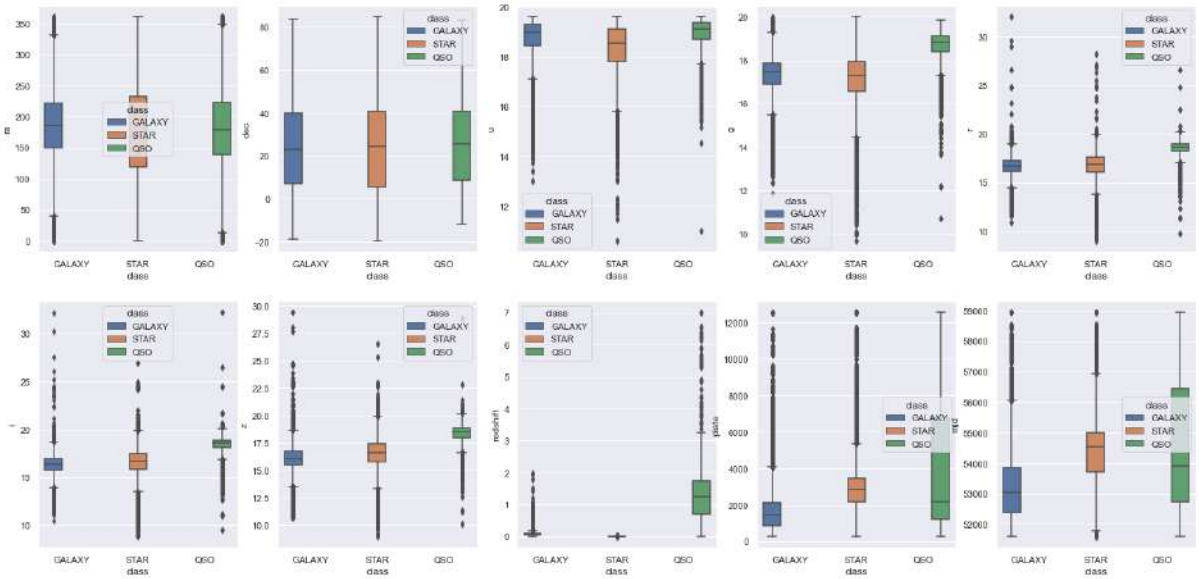
Probability Density Function (PDF) is the probability that the variable takes a value x. (smoothed version of the histogram)

4.1 Boxplot

In [14]:
```
df.columns
```

Out[14]:
```
Index(['ra', 'dec', 'u', 'g', 'r', 'i', 'z', 'class', 'redshift', 'plate',
       'mjd'],
      dtype='object')
```

In [15]:
```
fig, ((ax1, ax2, ax3, ax4, ax5), (ax6, ax7, ax8, ax9, ax10)) = plt.subplots(nrows = 2, ncols = 5,
                                                                             figsize = (25,12))


sns.boxplot(ax = ax1, x = 'class', y = 'ra', hue = 'class', data = df)
sns.boxplot(ax = ax2, x = 'class', y = 'dec', hue = 'class', data = df)
sns.boxplot(ax = ax3, x = 'class', y = 'u', hue = 'class', data = df)
sns.boxplot(ax = ax4, x = 'class', y = 'g', hue = 'class', data = df)
sns.boxplot(ax = ax5, x = 'class', y = 'r', hue = 'class', data = df)
sns.boxplot(ax = ax6, x = 'class', y = 'i', hue = 'class', data = df)
sns.boxplot(ax = ax7, x = 'class', y = 'z', hue = 'class', data = df)
sns.boxplot(ax = ax8, x = 'class', y = 'redshift', hue = 'class', data = df)
sns.boxplot(ax = ax9, x = 'class', y = 'plate', hue = 'class', data = df)
sns.boxplot(ax = ax10, x = 'class', y = 'mjd', hue = 'class', data = df)
```

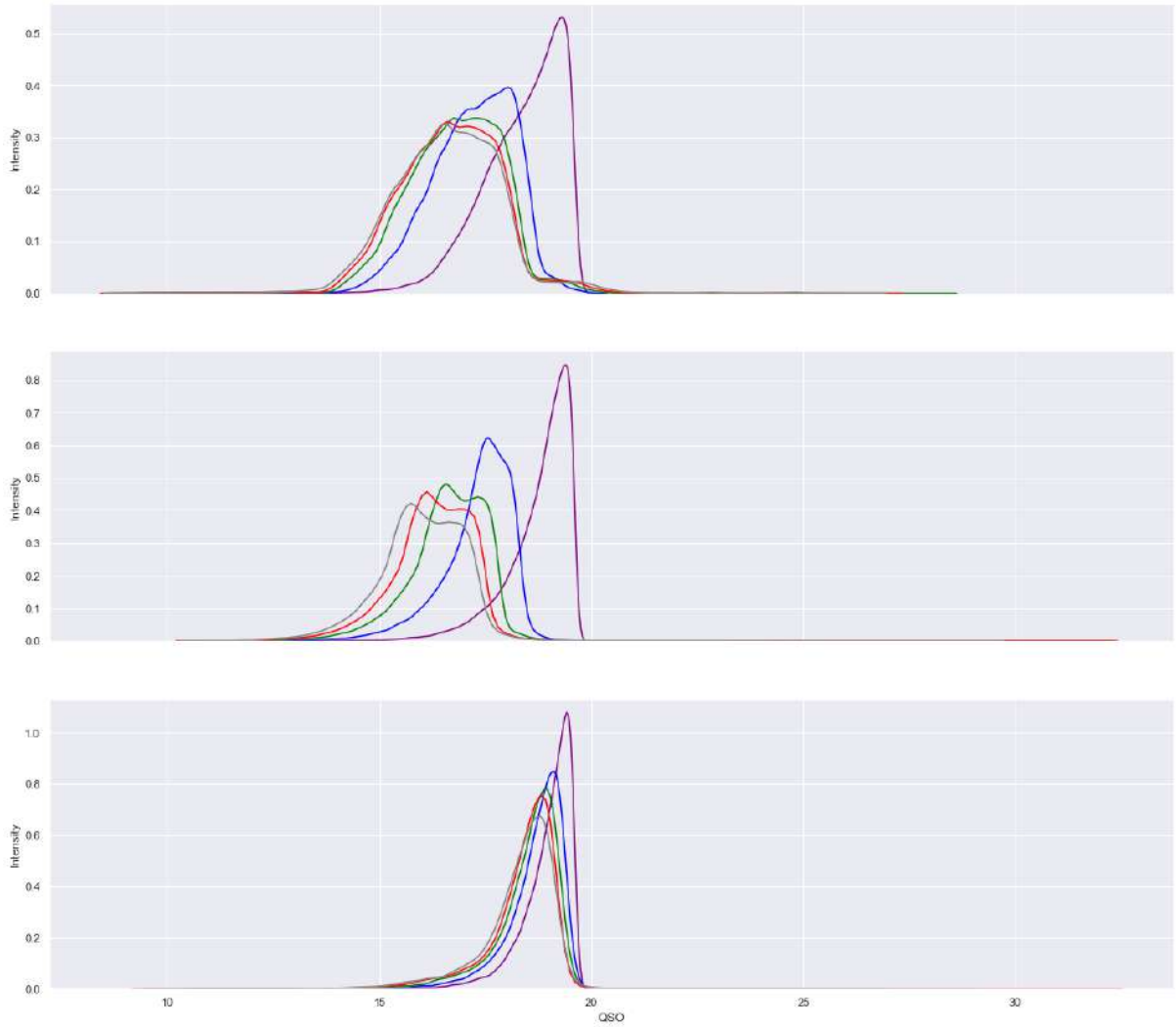


Observations

As we can see from the boxplot there are a large number of outliers.

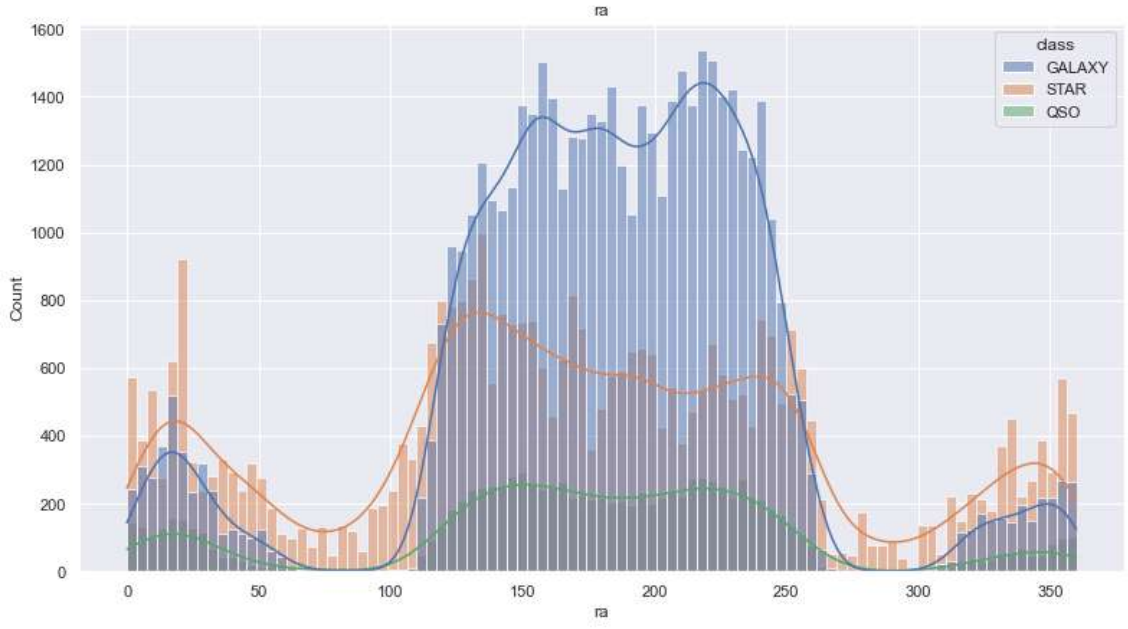
4.2 Frequency polygon

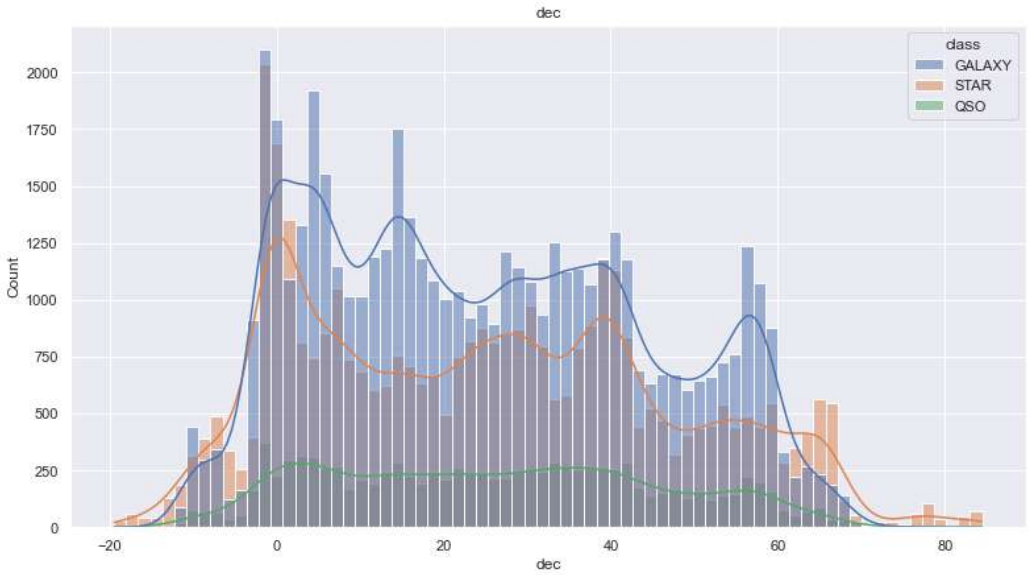
In [16]:
```
f, axes = plt.subplots(3, 1, figsize=(20, 18), sharex=True)
c = ['STAR', 'GALAXY', 'QSO']

for ax_id in range(3):
    sns.distplot(df.loc[df['class']==c[ax_id],'u'], hist=False, color='purple', ax=axes[ax_id], label='u')
    sns.distplot(df.loc[df['class']==c[ax_id],'g'], hist=False, color='blue', ax=axes[ax_id], label='g')
    sns.distplot(df.loc[df['class']==c[ax_id],'r'], hist=False, color='green', ax=axes[ax_id], label='r')
    sns.distplot(df.loc[df['class']==c[ax_id],'i'], hist=False, color='red', ax=axes[ax_id], label='i')
    sns.distplot(df.loc[df['class']==c[ax_id],'z'], hist=False, color='grey', ax=axes[ax_id], label='z')
    axes[ax_id].set(xlabel=c[ax_id], ylabel='Intensity')
```

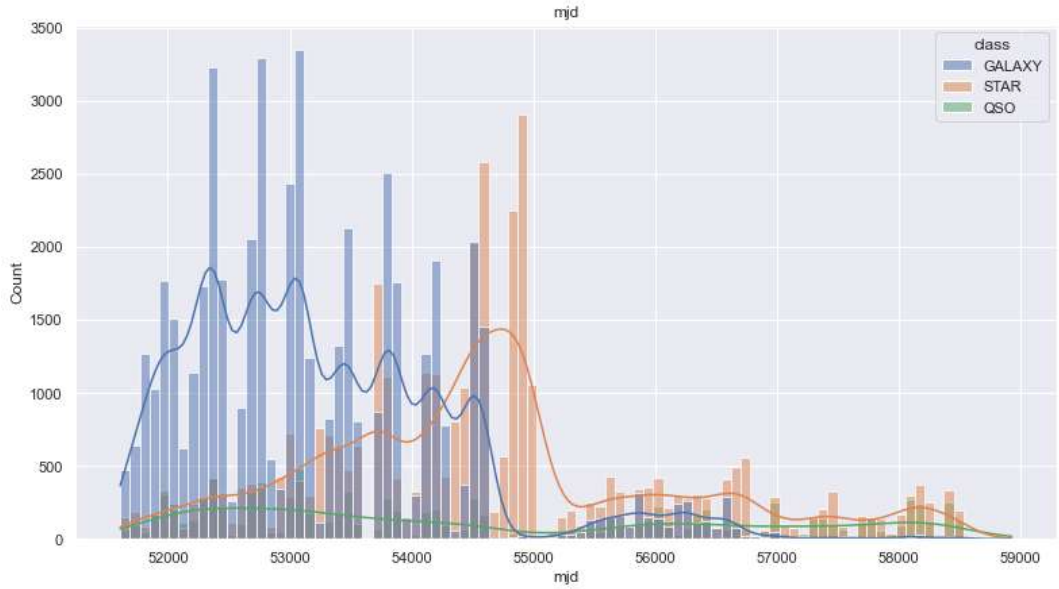


4.3 Histplot

```
In [17]: import seaborn as sb
         for i in ['ra', 'dec', 'redshift', 'plate', 'mjd']:
             plt.figure(figsize=(13,7))
             sb.histplot(data=df, x=i, kde=True, hue="class")
             plt.title(i)
```




dec
class
GALAXY
STAR
QSO
Count
dec


mjd
class
GALAXY
STAR
QSO
Count
mjd

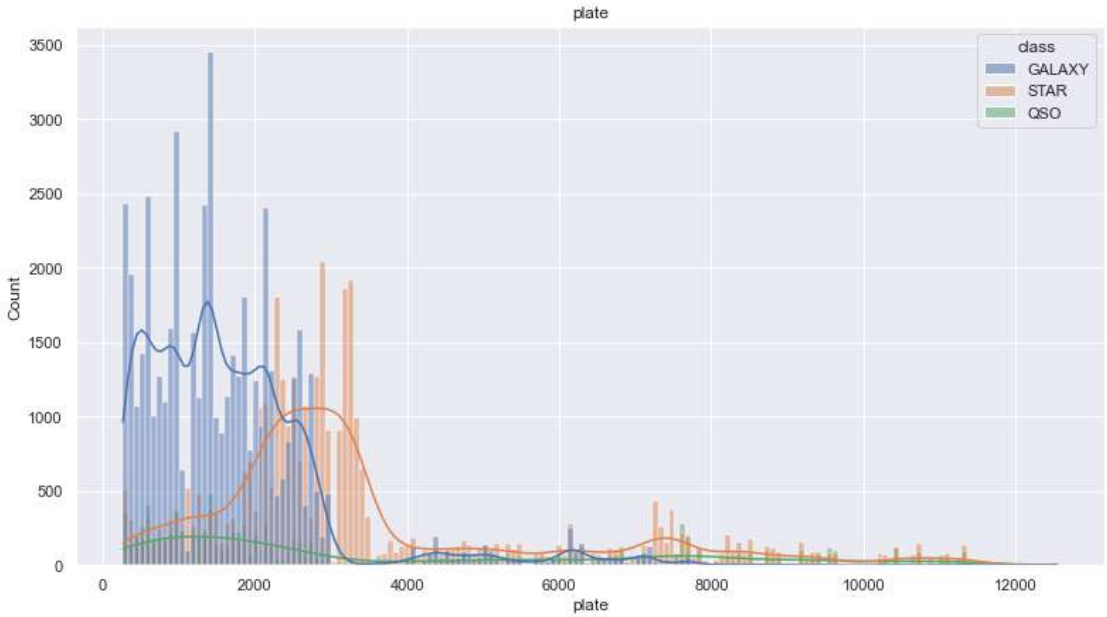


plate
class
GALAXY
STAR
QSO
Count
plate

4.4 Probability Density Function (PDF)

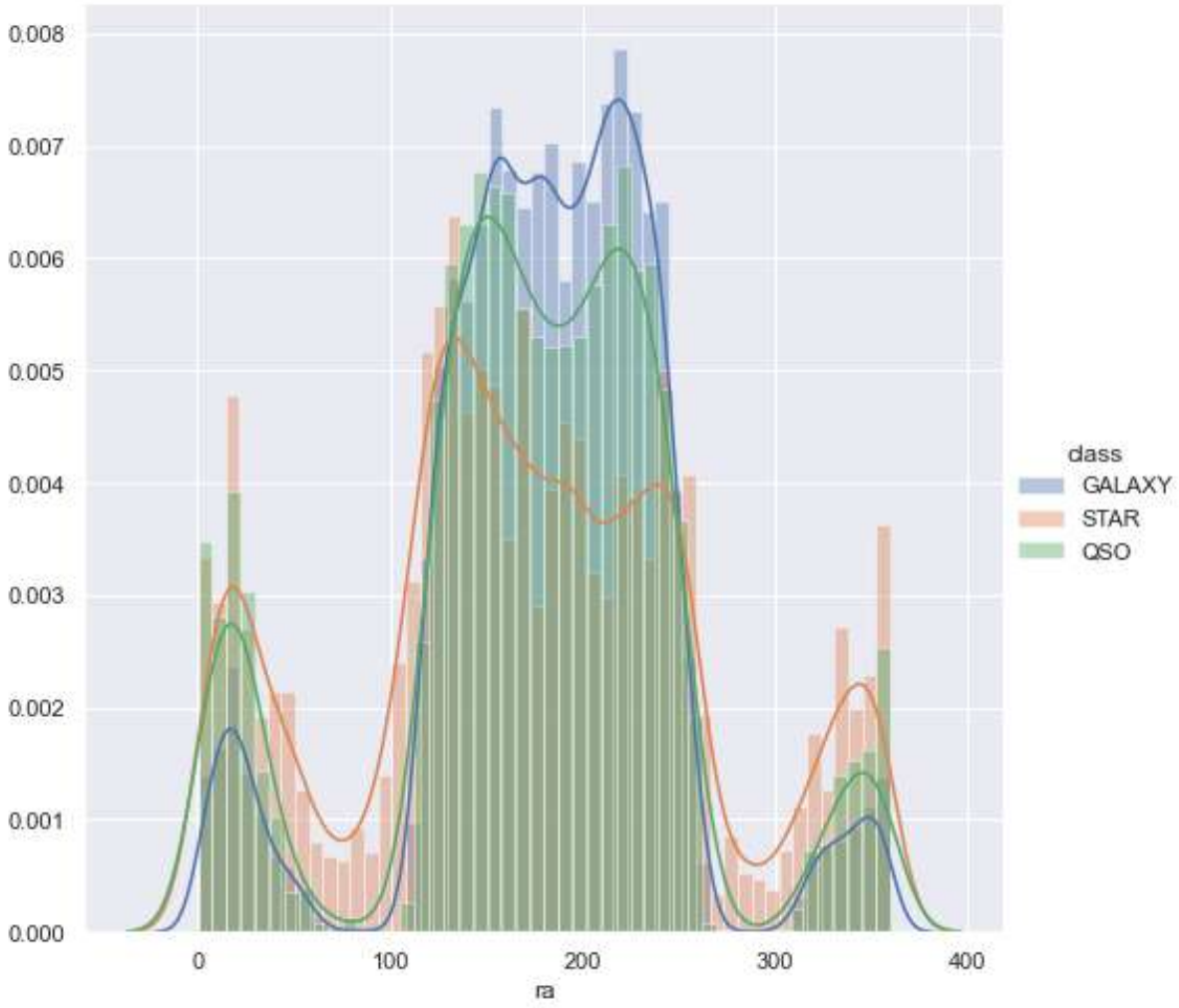
In [18]:
```
df.columns
```

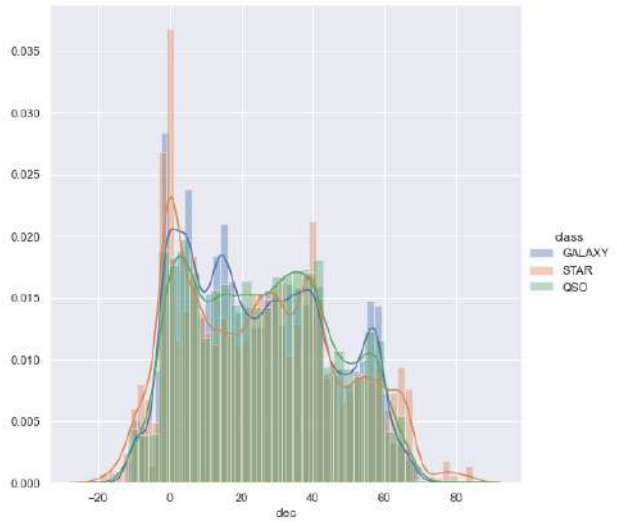
Out[18]:
```
Index(['ra', 'dec', 'u', 'g', 'r', 'i', 'z', 'class', 'redshift', 'plate',
       'mjd'],
      dtype='object')
```

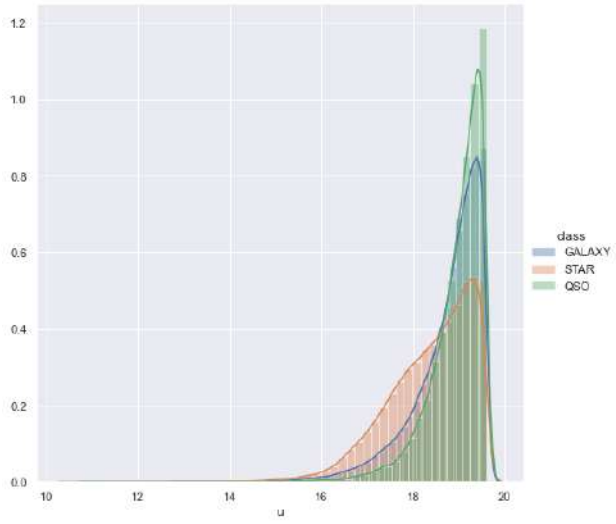
In [19]:
```
col = list(df.columns)
col.pop(7)
col
```

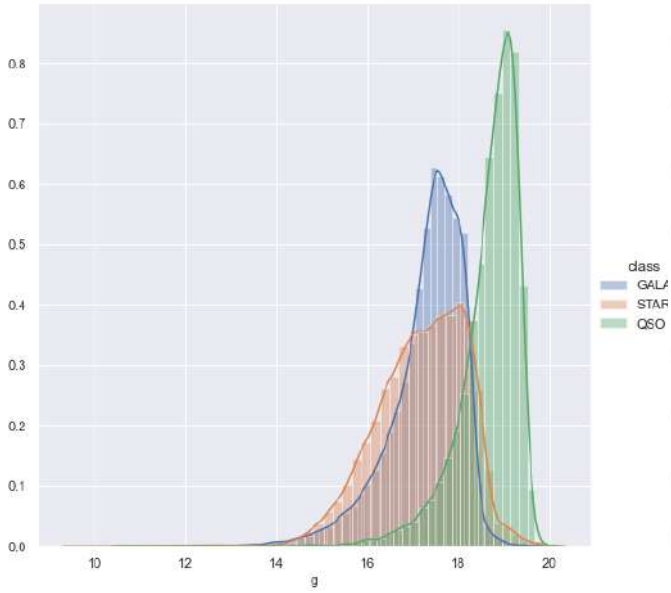
Out[19]:
```
['ra', 'dec', 'u', 'g', 'r', 'i', 'z', 'redshift', 'plate', 'mjd']
```

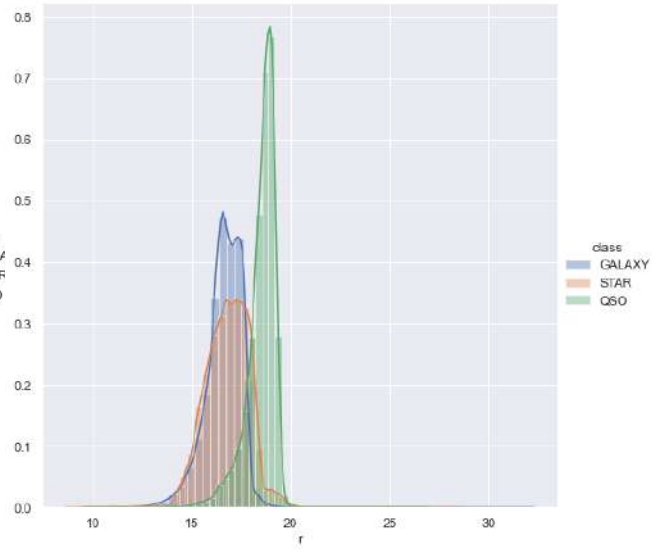
In [20]:
```
%%time
for idx, feature in enumerate(col):
    fg = sns.FacetGrid(df, hue='class', height=7)
    fg.map(sns.distplot, feature).add_legend()
    plt.show()
```

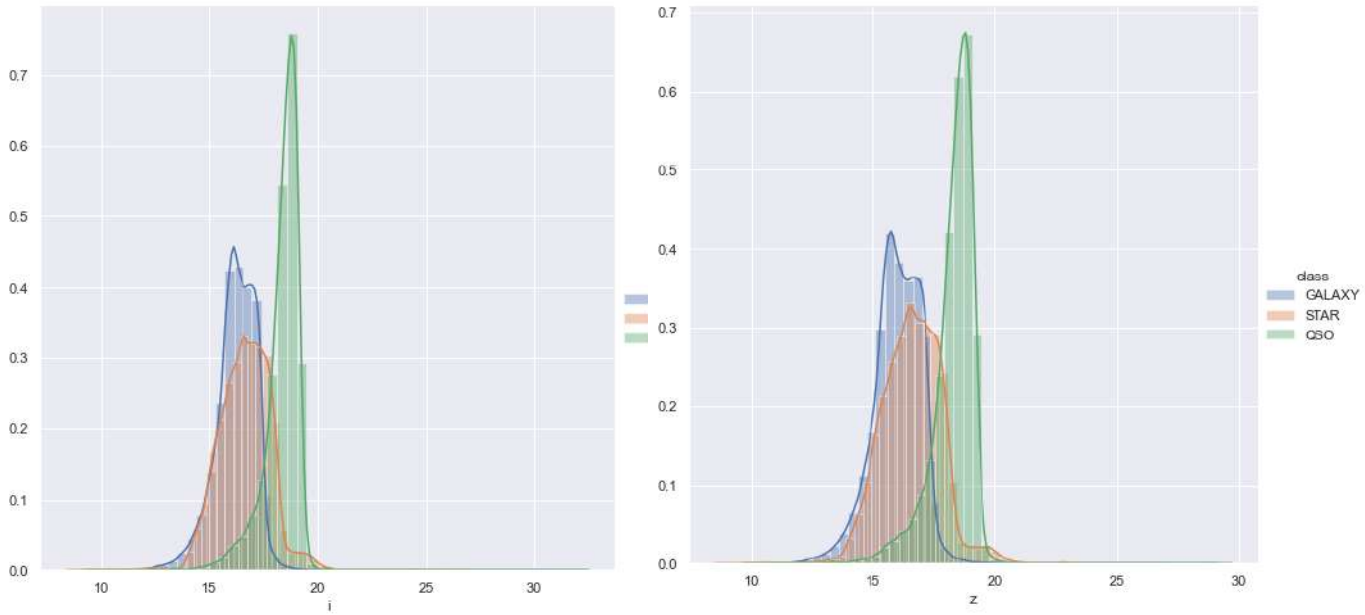


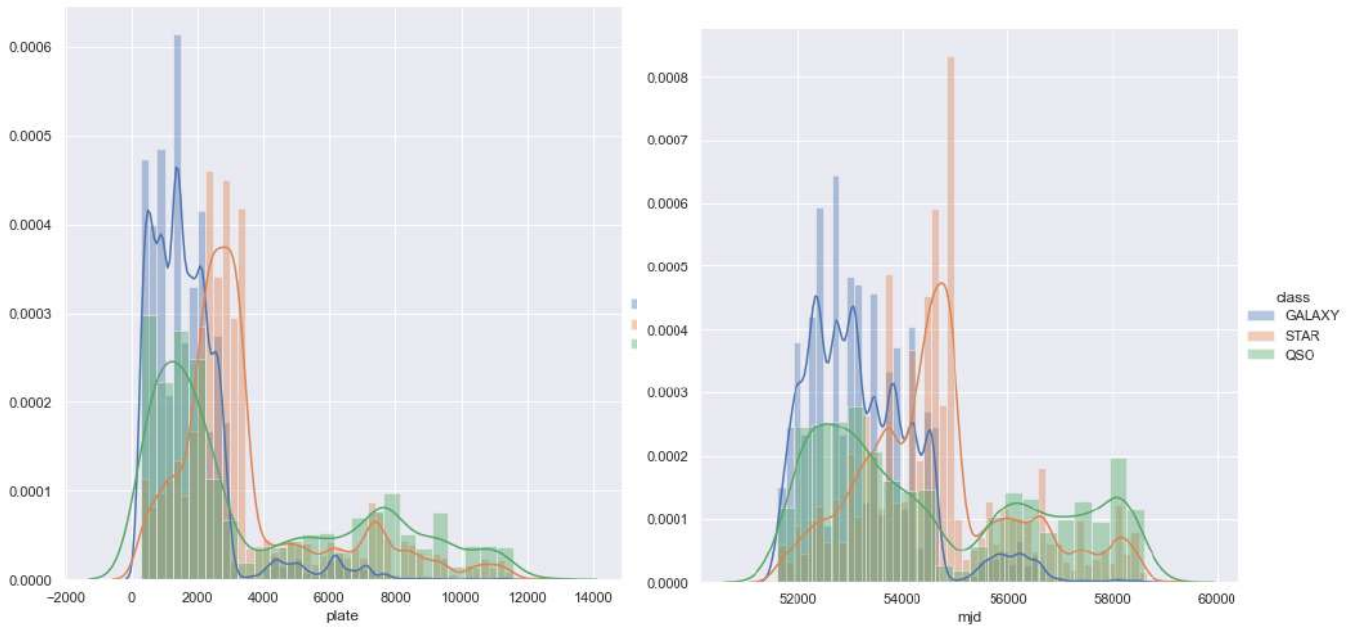

0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−20
0
20
40
60
80
dec
class
GALAXY
STAR
QSO


1.2
1.0
0.8
0.6
0.4
0.2
0.0
10
12
14
16
18
20
u
class
GALAXY
STAR
QSO


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
10
12
14
16
18
20
g
class
GALA
STAR
QSO

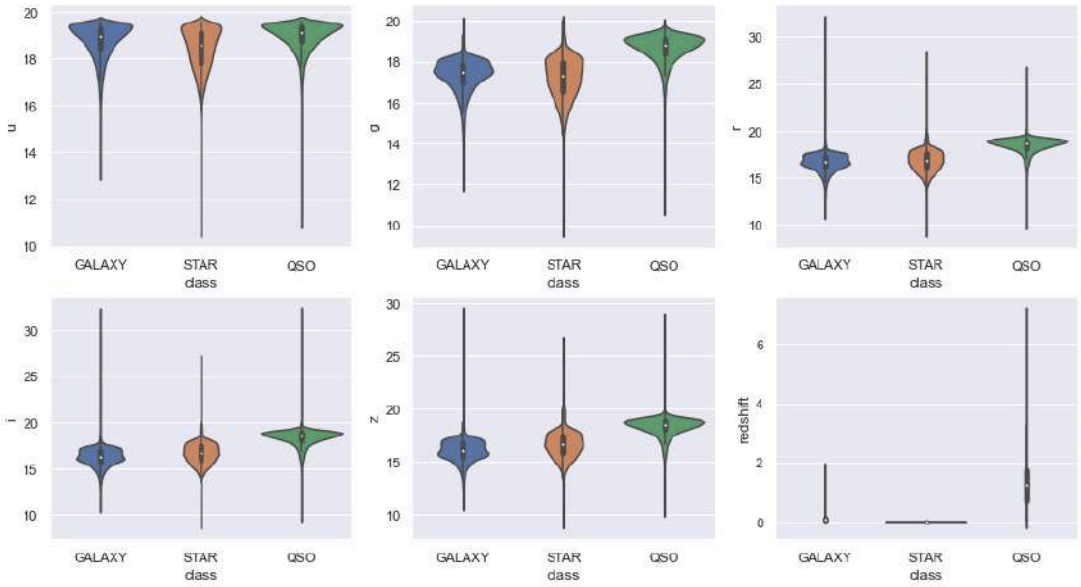

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
10
15
20
25
30
r
class
GALAXY
STAR
QSO


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
10
15
20
25
30
i
z
class
GALAXY
STAR
QSO


0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
−2000
0
2000
4000
6000
8000
10000
12000
14000
plate
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
52000
54000
56000
58000
60000
mjd
class
GALAXY
STAR
QSO

4.5 Violinplot

In [21]:
```
fig, axs = plt.subplots(2, 3, figsize=(15, 8))

sns.violinplot(x="class", y="u", data=df, ax=axs[0, 0])
sns.violinplot(x="class", y="g", data=df, ax=axs[0, 1])
sns.violinplot(x="class", y="r", data=df, ax=axs[0, 2])
sns.violinplot(x="class", y="i", data=df, ax=axs[1, 0])
sns.violinplot(x="class", y="z", data=df, ax=axs[1, 1])
sns.violinplot(x="class", y="redshift", data=df, ax=axs[1, 2])
plt.show()
```

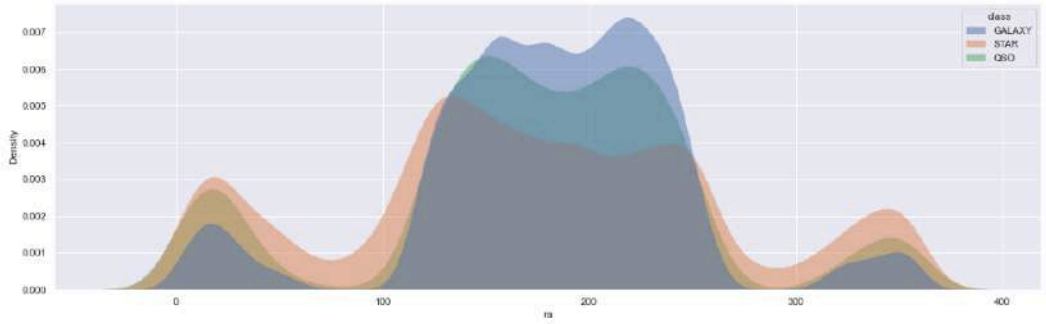


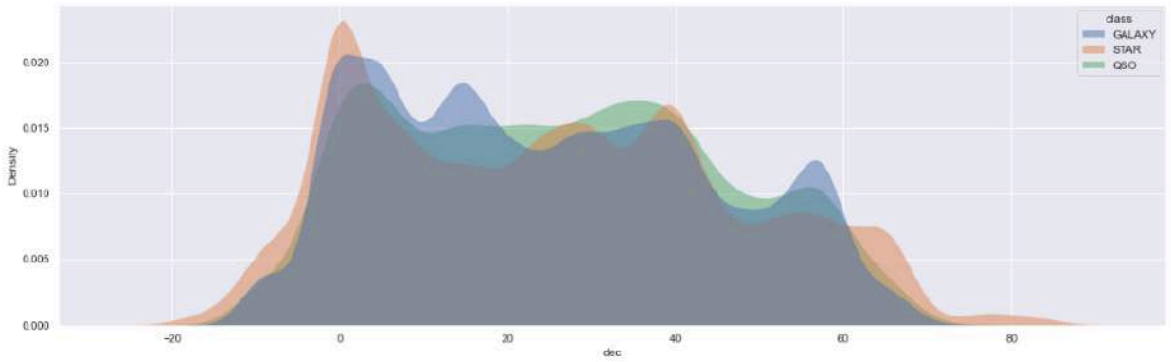
4.6 KDE Plot

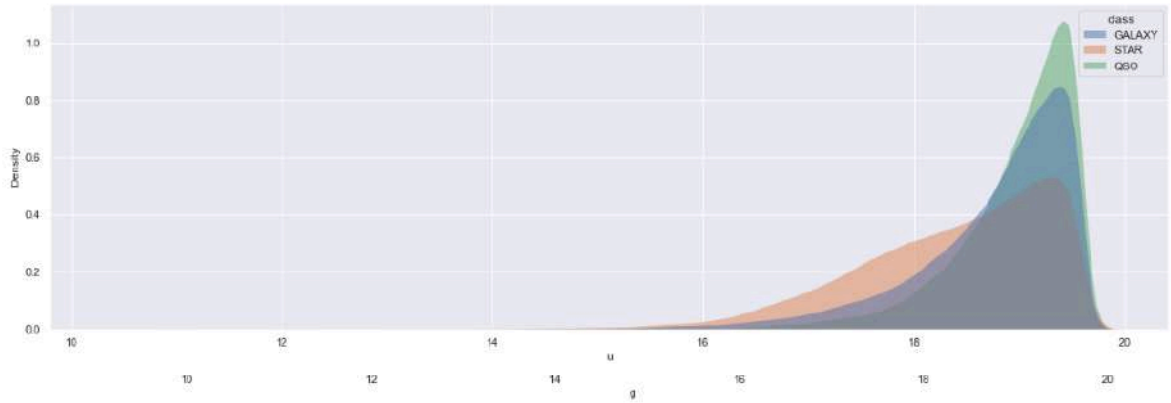
Plot univariate or bivariate distributions using kernel density estimation. A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analagous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

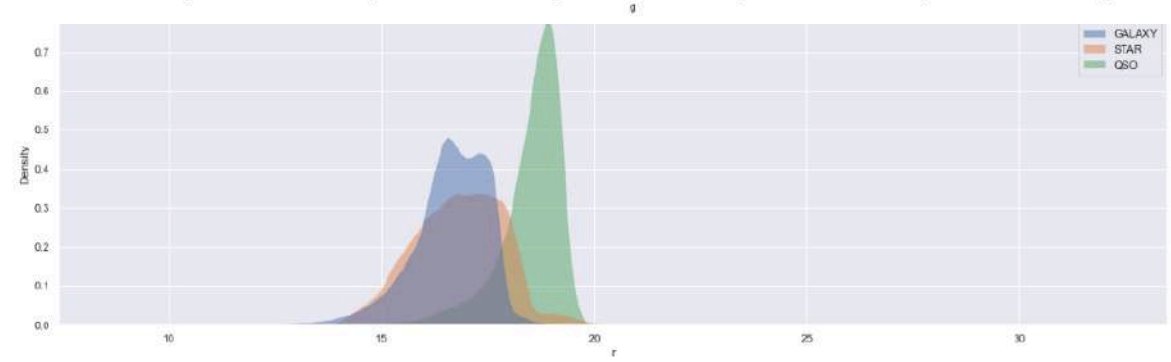
A histogram puts all samples between the boundaries of each bin will fall into the bin. It doesn't differentiate whether the value falls close the left, to the right or the center of the bin. A kde plot, on the other hand, takes each individual sample value and draws a small gaussian bell curve over it.

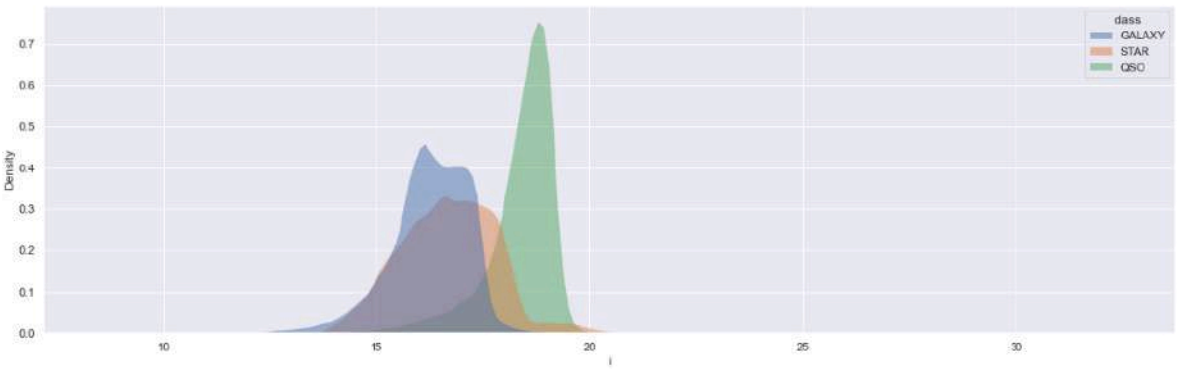
In [22]:
```
fig, axes = plt.subplots(nrows=10, ncols=1,figsize=(20, 70))
ax = sns.kdeplot(data=df, x='ra', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[0])
ax = sns.kdeplot(data=df, x='dec', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[1])
ax = sns.kdeplot(data=df, x='u', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[2])
ax = sns.kdeplot(data=df, x='g', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[3])
ax = sns.kdeplot(data=df, x='r', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[4])
ax = sns.kdeplot(data=df, x='i', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[5])
ax = sns.kdeplot(data=df, x='z', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[6])
ax = sns.kdeplot(data=df, x='redshift', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[7])
ax = sns.kdeplot(data=df, x='plate', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[8])
ax = sns.kdeplot(data=df, x='mjd', hue="class", fill=True, common_norm=False, alpha=.5, linewidth=0,ax = axes[9])
```

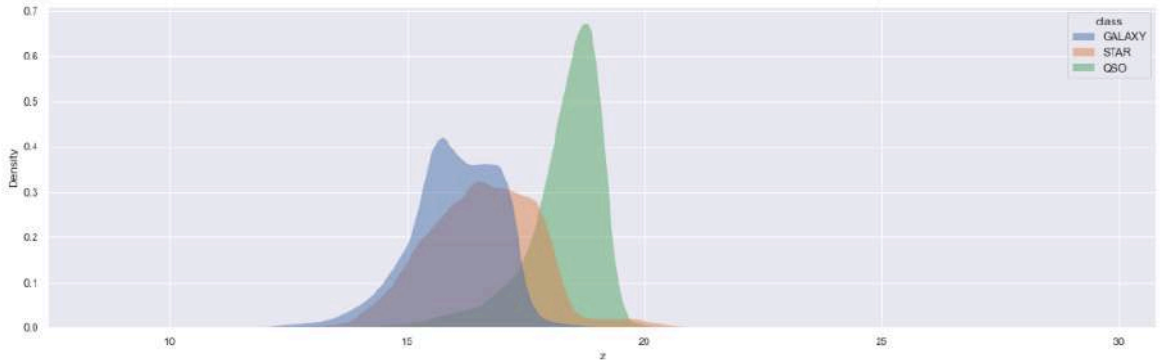


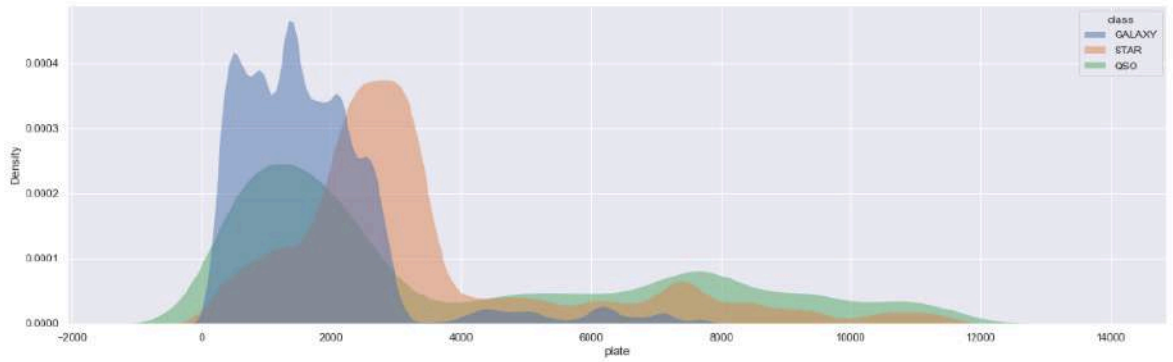

class
GALAXY
STAR
QSO
Density
0.020
0.015
0.010
0.005
0.000
−20
0
20
40
60
80
dec

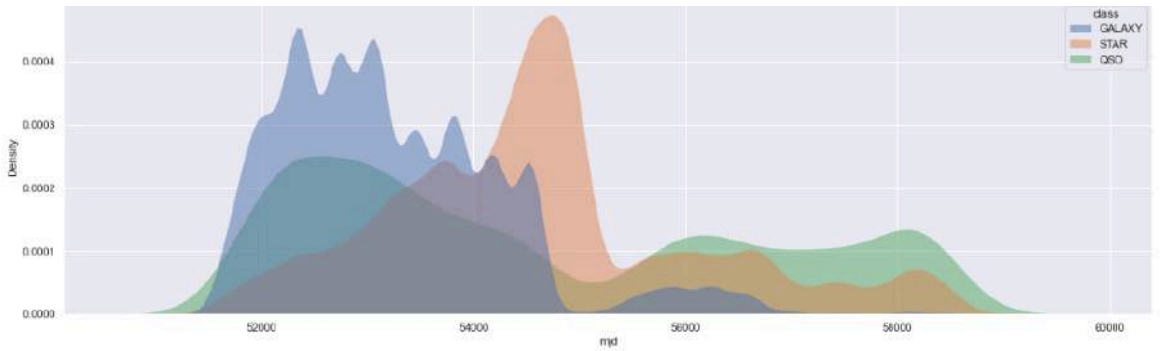

class
GALAXY
STAR
QSO
Density
1.0
0.8
0.6
0.4
0.2
0.0
10
12
14
16
18
20
u


GALAXY
STAR
QSO
Density
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
10
15
20
25
30
r

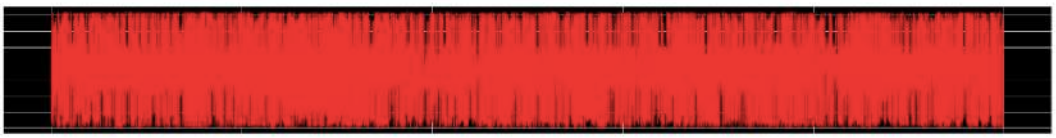

class
GALAXY
STAR
QSO
Density
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
10
15
20
25
30
i



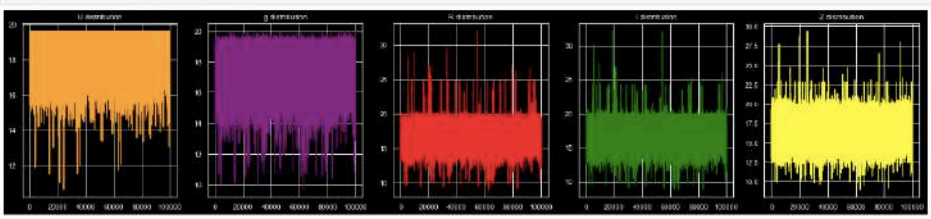




In [23]:
```
plt.figure(figsize = (40,5))
plt.style.use('dark_background')
plt.plot(df['ra'],color = 'red');
plt.title('ra distribution');
```



In [24]:
```
fig,ax = plt.subplots(1,5,figsize = (25,5))
ax[0].plot(df['u'],color = 'orange');
ax[0].set_title('U distribution');
ax[1].plot(df['g'],color = 'purple');
ax[1].set_title('g distribution');
ax[2].plot(df['r'],color = 'red');
ax[2].set_title('R distribution');
ax[3].plot(df['i'],color = 'green');
ax[3].set_title('I distribution');
ax[4].plot(df['z'],color = 'yellow');
ax[4].set_title('Z distribution');
```



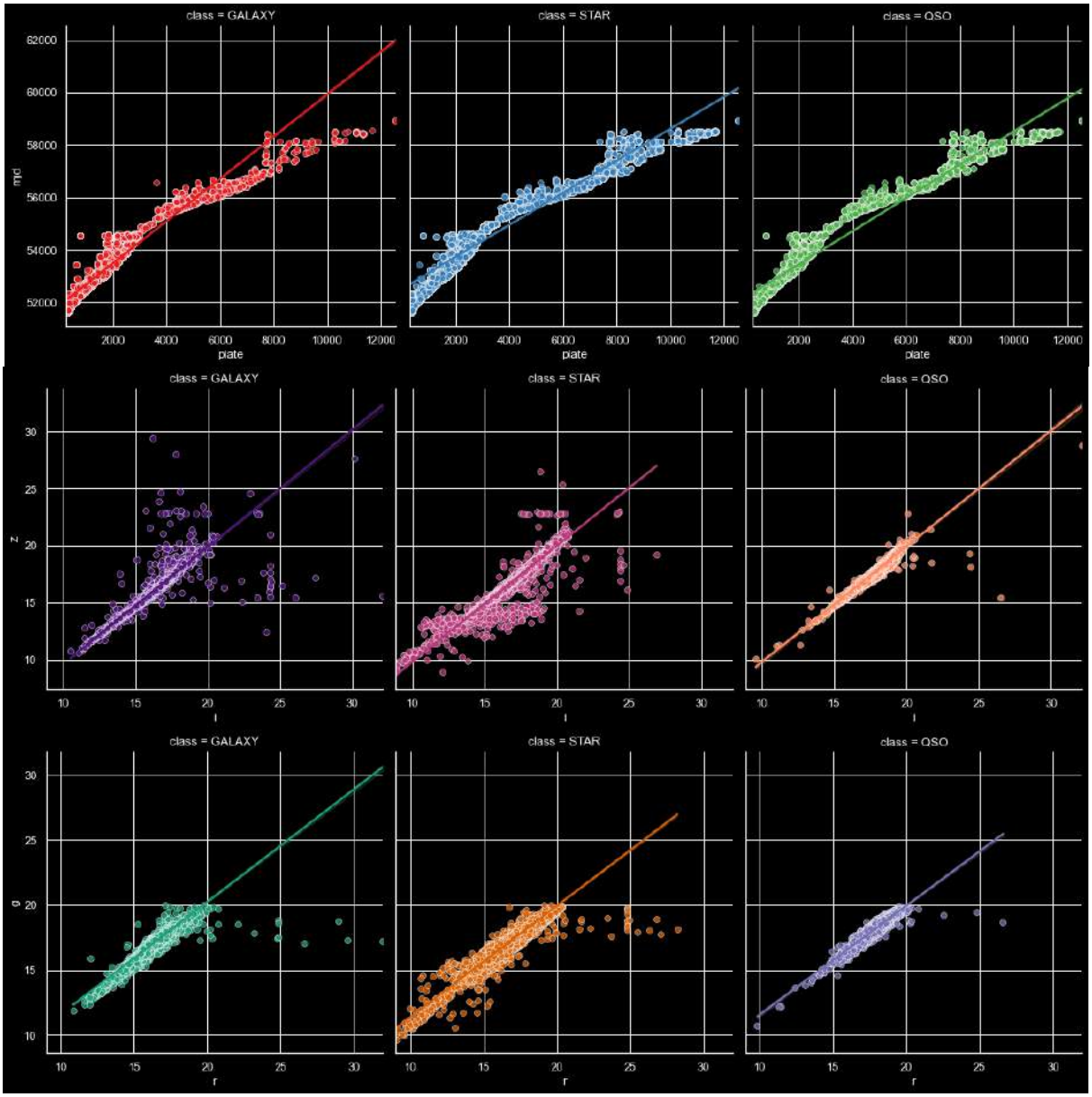
5 . Multivariate Analysis

5.1 Lineplot

In [25]:
```
%%time
sns.lmplot(x = 'plate', y='mjd', data = df, hue='class', col = 'class', palette='Set1', scatter_kws= {'edgecolor':'whi
sns.lmplot(x = 'i', y='z', data = df, hue='class', col = 'class', palette='magma', scatter_kws= {'edgecolor':'white',
sns.lmplot(x = 'r', y='g', data = df, hue='class', col = 'class', palette='Dark2', scatter_kws= {'edgecolor':'white',
```

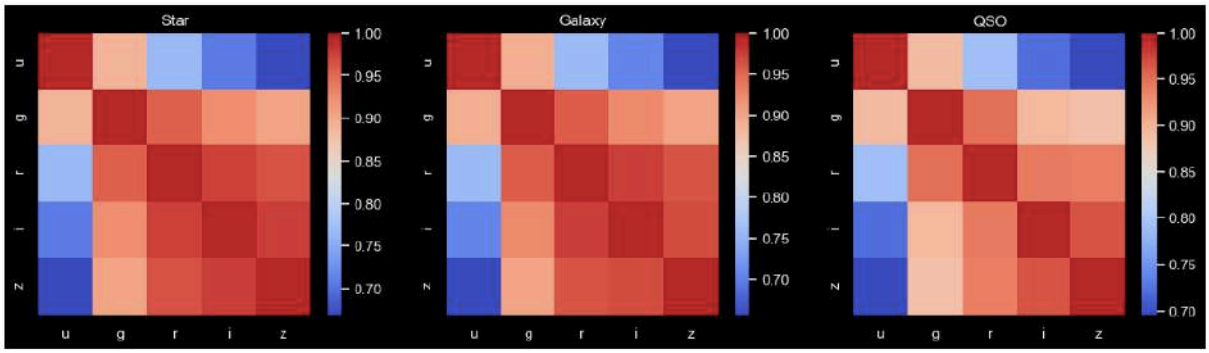
```
Wall time: 48.4 s
```

Out[25]: <seaborn.axisgrid.FacetGrid at 0x288001ea370>



5.3 Correlation between different variables

In [26]:
```
fig, axes = plt.subplots(nrows=1, ncols=3,figsize=(16, 4))
fig.set_dpi(100)
ax = sns.heatmap(df[df['class']=='STAR'][['u', 'g', 'r', 'i', 'z']].corr(), ax = axes[0], cmap='coolwarm')
ax.set_title('Star')
ax = sns.heatmap(df[df['class']=='GALAXY'][['u', 'g', 'r', 'i', 'z']].corr(), ax = axes[1], cmap='coolwarm')
ax.set_title('Galaxy')
ax = sns.heatmap(df[df['class']=='QSO'][['u', 'g', 'r', 'i', 'z']].corr(), ax = axes[2], cmap='coolwarm')
ax = ax.set_title('QSO')
```



Removing Outliers

In [8]:
```
#Sometimes due to outliers, we can't do good analysis of our data therefore, we decided to remove them as our current

def rem_outliers():
    s1 = df.shape

    for i in df.select_dtypes(include = 'number').columns:
        qt1 = df[i].quantile(0.25)
        qt3 = df[i].quantile(0.75)
        iqr =  qt3 - qt1
        lower = qt1-(1.5*iqr)
        upper = qt3+(1.5*iqr)
        min_in = df[df[i]<lower].index
        max_in = df[df[i]>upper].index
        df.drop(min_in, inplace = True)
        df.drop(max_in, inplace = True)

    s2 = df.shape
    outliers = s1[0] - s2[0]
    return outliers
```

In [9]:
```
print("Number of outliers deleted are : ", rem_outliers())
```

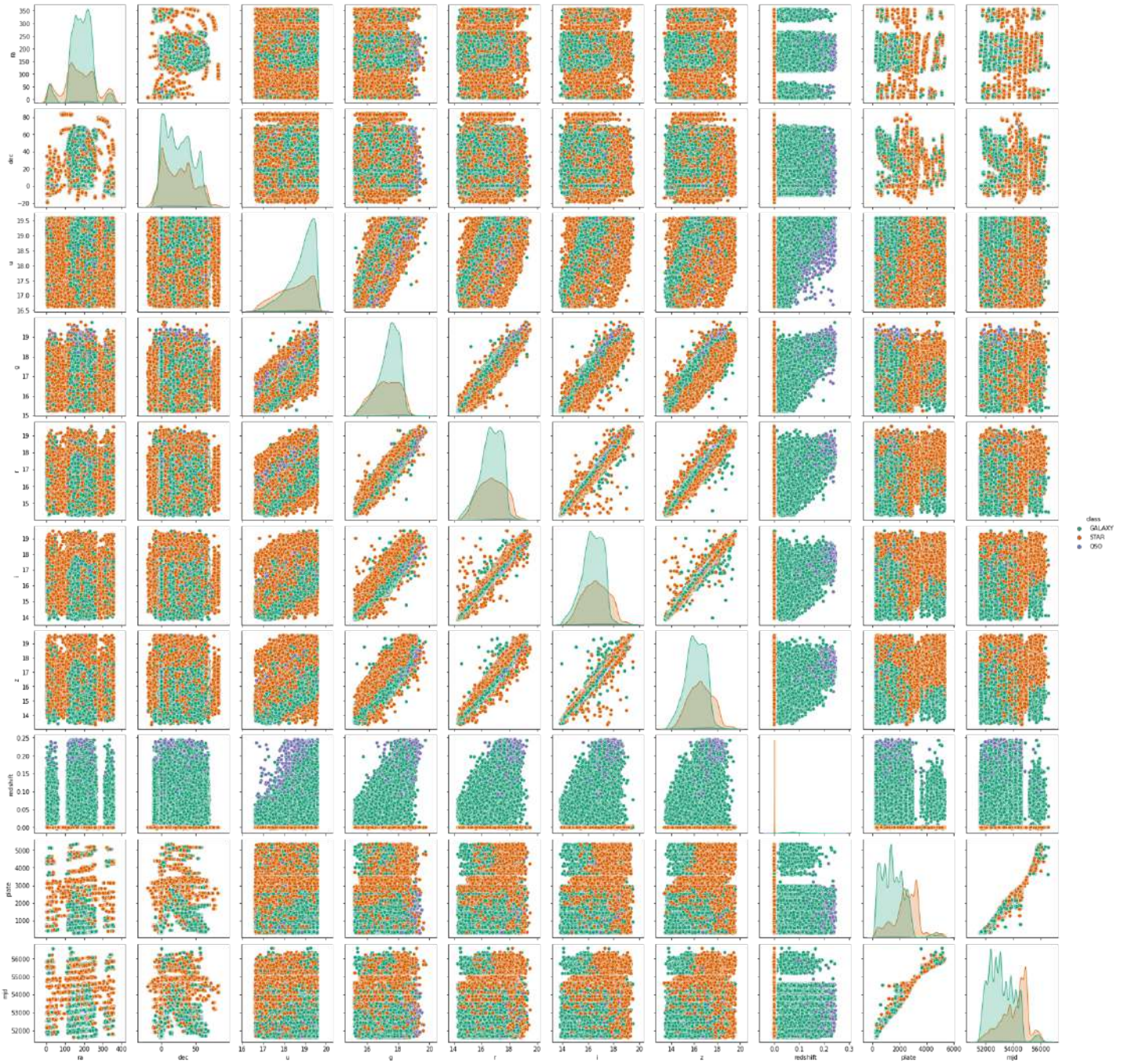
```
Number of outliers deleted are :  22790
```

In [10]:
```
df.shape
```

Out[10]:
```
(77210, 11)
```

5.4 Pairplot

In [18]:
```
sns.pairplot(df,palette = 'Dark2',hue = 'class')
```



5.5 Correlation Matrix

In [16]:
```
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(), annot=True)
df.corr()
```

Out[16]:

	ra	dec	u	g	r	i	z	redshift	plate	mjd
ra	1.000000	0.054652	0.016463	0.019506	0.013206	0.006705	-0.000617	0.049057	0.003047	0.041765
dec	0.054652	1.000000	-0.020849	-0.023028	-0.019869	-0.019009	-0.014103	0.004714	0.025852	0.025079
u	0.016463	-0.020849	1.000000	0.859561	0.685888	0.595883	0.516801	0.397221	-0.068822	-0.070772
g	0.019506	-0.023028	0.859561	1.000000	0.949588	0.895153	0.843922	0.308837	-0.022102	-0.027656
r	0.013206	-0.019869	0.685888	0.949588	1.000000	0.985407	0.963497	0.129829	0.031882	0.024378
i	0.006705	-0.019009	0.595883	0.895153	0.985407	1.000000	0.990258	0.014931	0.078392	0.069568
z	-0.000617	-0.014103	0.516801	0.843922	0.963497	0.990258	1.000000	-0.061164	0.110117	0.100618
redshift	0.049057	0.004714	0.397221	0.308837	0.129829	0.014931	-0.061164	1.000000	-0.361682	-0.357039
plate	0.003047	0.025852	-0.068822	-0.022102	0.031882	0.078392	0.110117	-0.361682	1.000000	0.967004
mjd	0.041765	0.025079	-0.070772	-0.027656	0.024378	0.069568	0.100618	-0.357039	0.967004	1.000000

5.6 Pandas Profiling

```
In [17]: import pandas_profiling
profile = pandas_profiling.ProfileReport(df, title='SDSS dataset Report')
profile.to_file(output_file='SDSS_with_pandas_profiling.html')
```

Summarize dataset: 100% 146/146 [00:44<00:00, 4.09it/s, Completed]

Generate report structure: 100% 1/1 [00:07<00:00, 7.93s/it]

Render HTML: 100% 1/1 [00:06<00:00, 6.04s/it]

Export report to file: 100% 1/1 [00:00<00:00, 6.45it/s]

Generates profile reports from a pandas DataFrame.

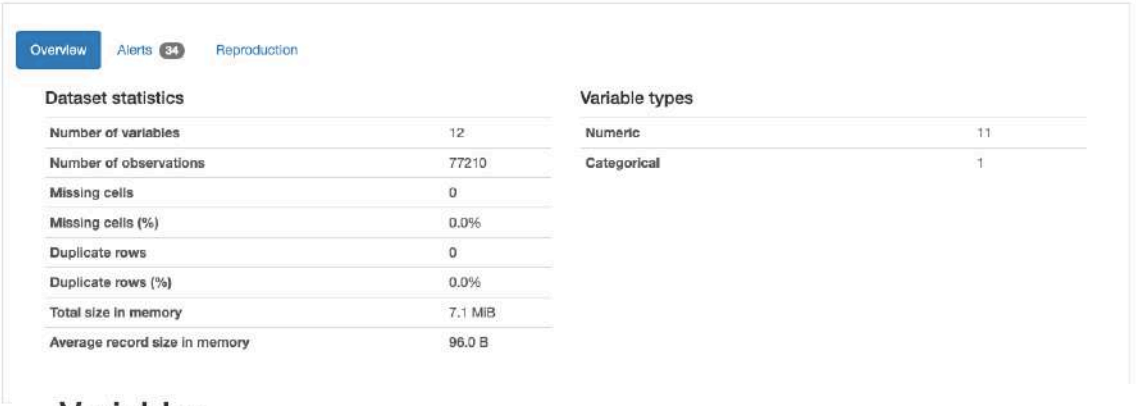
The pandas df.describe() function is great but a little basic for serious exploratory data analysis. pandas_profiling extends the pandas DataFrame with df.profile_report() for quick data analysis.

For each column the following statistics - if relevant for the column type - are presented in an interactive HTML report:

- **Type inference**: detect the types of columns in a dataframe.
- **Essentials**: type, unique values, missing values
- **Quantile statistics** like minimum value, Q1, median, Q3, maximum, range, interquartile range
- **Descriptive statistics** like mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness
- **Most frequent values**

- **Histogram**
- **Correlations** highlighting of highly correlated variables, Spearman, Pearson and Kendall matrices
- **Missing values** matrix, count, heatmap and dendrogram of missing values
- **Text analysis** learn about categories (Uppercase, Space), scripts (Latin, Cyrillic) and blocks (ASCII) of text data.
- **File and Image analysis** extract file sizes, creation dates and dimensions and scan for truncated images or those containing EXIF information.

Overview



Overview | Alerts 34 | Reproduction

Dataset statistics

Number of variables	12
Number of observations	77210
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	7.1 MiB
Average record size in memory	96.0 B

Variable types

Numeric	11
Categorical	1

Variables

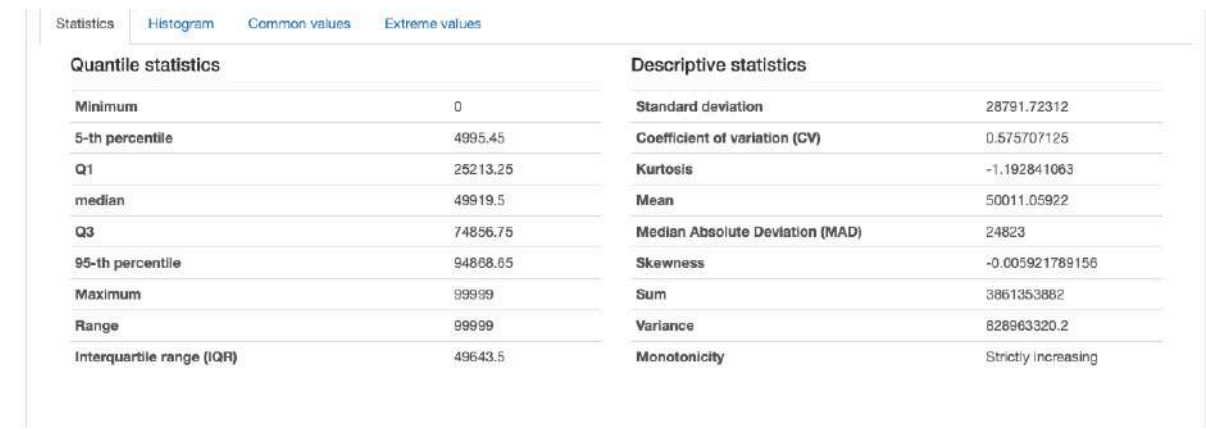
df_index

Real number ($\mathbb{R}_{\geq 0}$)

UNIQUE

Distinct	77210	Minimum	0
Distinct (%)	100.0%	Maximum	99999
Missing	0	Zeros	1
Missing (%)	0.0%	Zeros (%)	< 0.1%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	50011.05922	Memory size	603.3 KiB

Toggle details



Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	4995.45
Q1	25213.25
median	49919.5
Q3	74856.75
95-th percentile	94868.65
Maximum	99999
Range	99999
Interquartile range (IQR)	49643.5

Descriptive statistics

Standard deviation	28791.72312
Coefficient of variation (CV)	0.575707125
Kurtosis	-1.192841063
Mean	50011.05922
Median Absolute Deviation (MAD)	24823
Skewness	-0.005921789156
Sum	3861353882
Variance	828963320.2
Monotonicity	Strictly increasing

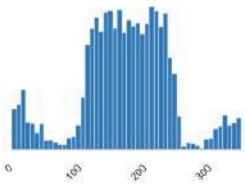
ra

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	77209
Distinct (%)	> 99.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	179.8863146

Minimum	4.075876283
Maximum	356.9973742
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	4.075876283
5-th percentile	27.04068059
Q1	140.117425
median	182.0929392
Q3	224.5100275
95-th percentile	317.8335278
Maximum	356.9973742
Range	352.9214979
Interquartile range (IQR)	84.39260247

Descriptive statistics

Standard deviation	71.19769028
Coefficient of variation (CV)	0.3957927007
Kurtosis	0.5265448431
Mean	179.8863146
Median Absolute Deviation (MAD)	42.19733045
Skewness	-0.1893861167
Sum	13889022.35
Variance	5069.111101
Monotonicity	Not monotonic

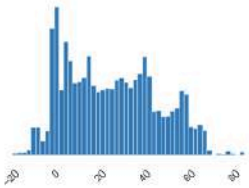
dec

Real number (ℝ)

HIGH CORRELATION
UNIQUE

Distinct	77210
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	24.98834372

Minimum	-19.49545595
Maximum	84.49049355
Zeros	0
Zeros (%)	0.0%
Negative	9112
Negative (%)	11.8%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	-19.49545595
5-th percentile	-2.287426727
Q1	6.513522364
median	23.58689543
Q3	40.2904455
95-th percentile	59.74369334
Maximum	84.49049355
Range	103.9859495
Interquartile range (IQR)	33.77692314

Descriptive statistics

Standard deviation	20.66364958
Coefficient of variation (CV)	0.8269315412
Kurtosis	-0.9101641366
Mean	24.98834372
Median Absolute Deviation (MAD)	16.87985535
Skewness	0.2865957951
Sum	1929350.019
Variance	426.9864142
Monotonicity	Not monotonic

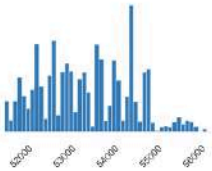
mjd

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	1408
Distinct (%)	1.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	53494.56065

Minimum	51608
Maximum	56577
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	51608
5-th percentile	51924
Q1	52652
median	53460
Q3	54379
95-th percentile	54999
Maximum	56577
Range	4969
Interquartile range (IQR)	1727

Descriptive statistics

Standard deviation	1041.719608
Coefficient of variation (CV)	0.0194733744
Kurtosis	-0.8015650223
Mean	53494.56065
Median Absolute Deviation (MAD)	816
Skewness	0.2067457324
Sum	4130315028
Variance	1085179.742
Monotonicity	Not monotonic

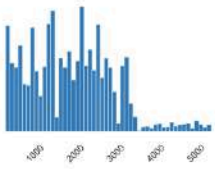
plate

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	3642
Distinct (%)	4.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	1898.125346

Minimum	266
Maximum	5362
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	266
5-th percentile	378
Q1	1008.25
median	1869
Q3	2577
95-th percentile	3454
Maximum	5362
Range	5096
Interquartile range (IQR)	1568.75

Descriptive statistics

Standard deviation	1046.145564
Coefficient of variation (CV)	0.5511467226
Kurtosis	0.2096301874
Mean	1898.125346
Median Absolute Deviation (MAD)	740
Skewness	0.5732396421
Sum	146554258
Variance	1094420.541
Monotonicity	Not monotonic

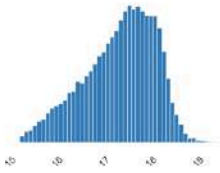
g

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	66895
Distinct (%)	86.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	17.31761957

Minimum	15.20344
Maximum	19.76544
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	15.20344
5-th percentile	15.8821805
Q1	16.821545
median	17.431315
Q3	17.90095
95-th percentile	18.3558275
Maximum	19.76544
Range	4.562
Interquartile range (IQR)	1.079405

Descriptive statistics

Standard deviation	0.7576569181
Coefficient of variation (CV)	0.04375063876
Kurtosis	-0.3511356254
Mean	17.31761957
Median Absolute Deviation (MAD)	0.525735
Skewness	-0.4926563981
Sum	1337093.407
Variance	0.5740440055
Monotonicity	Not monotonic

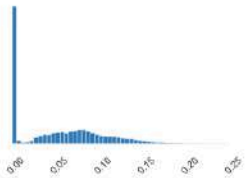
redshift

Real number (ℝ)

HIGH CORRELATION

Distinct	70091
Distinct (%)	90.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.0499027864

Minimum	-0.004136078
Maximum	0.2448289
Zeros	311
Zeros (%)	0.4%
Negative	17993
Negative (%)	23.3%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	-0.004136078
5-th percentile	-0.0004546614
Q1	2.22×10^{-5}
median	0.045500435
Q3	0.08360148
95-th percentile	0.13879242
Maximum	0.2448289
Range	0.248964978
Interquartile range (IQR)	0.08357928

Descriptive statistics

Standard deviation	0.04971749587
Coefficient of variation (CV)	0.9962869703
Kurtosis	-0.125526736
Mean	0.0499027864
Median Absolute Deviation (MAD)	0.0453928805
Skewness	0.7196152027
Sum	3852.994138
Variance	0.002471829396
Monotonicity	Not monotonic

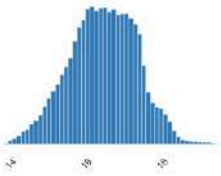
i

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	68636
Distinct (%)	88.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	16.42789801

Minimum	13.80943
Maximum	19.49216
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	13.80943
5-th percentile	14.9135325
Q1	15.81786
median	16.446025
Q3	17.0768675
95-th percentile	17.8595455
Maximum	19.49216
Range	5.68273
Interquartile range (IQR)	1.2590075

Descriptive statistics

Standard deviation	0.8926061077
Coefficient of variation (CV)	0.05433477291
Kurtosis	-0.2057738419
Mean	16.42789801
Median Absolute Deviation (MAD)	0.629665
Skewness	-0.06841238392
Sum	1268398.005
Variance	0.7967456636
Monotonicity	Not monotonic

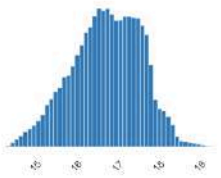
r

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	68180
Distinct (%)	88.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	16.72188917

Minimum	14.27426
Maximum	19.5365
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	14.27426
5-th percentile	15.2500845
Q1	16.15262
median	16.759695
Q3	17.36099
95-th percentile	17.9946765
Maximum	19.5365
Range	5.26224
Interquartile range (IQR)	1.20837

Descriptive statistics

Standard deviation	0.8411251913
Coefficient of variation (CV)	0.05030084716
Kurtosis	-0.3541941969
Mean	16.72188917
Median Absolute Deviation (MAD)	0.603645
Skewness	-0.215484275
Sum	1291097.063
Variance	0.7074915874
Monotonicity	Not monotonic

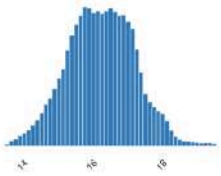
z

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	69244
Distinct (%)	89.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	16.24766649

Minimum	13.36443
Maximum	19.56497
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	13.36443
5-th percentile	14.6512615
Q1	15.57816
median	16.249795
Q3	16.927865
95-th percentile	17.834821
Maximum	19.56497
Range	6.20054
Interquartile range (IQR)	1.349705

Descriptive statistics

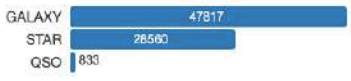
Standard deviation	0.9638690294
Coefficient of variation (CV)	0.05932353609
Kurtosis	-0.1076149923
Mean	16.24766649
Median Absolute Deviation (MAD)	0.67527
Skewness	0.04227799027
Sum	1254482.33
Variance	0.9290435058
Monotonicity	Not monotonic

class

Categorical

HIGH CORRELATION

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	603.3 KiB



Toggle details

Overview | Categories | Words | Characters

Length

Max length	6
Median length	6
Mean length	5.227833182
Min length	3

Characters and Unicode

Total characters	0
Distinct characters	0
Distinct categories	0
Distinct scripts	0
Distinct blocks	0

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Unique

Unique	0
Unique (%)	0.0%

Sample

1st row	GALAXY
2nd row	GALAXY
3rd row	GALAXY
4th row	STAR
5th row	STAR

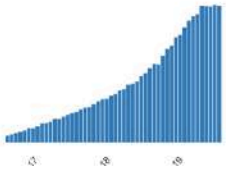
u

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	64338
Distinct (%)	83.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	18.68922552

Minimum	16.61704
Maximum	19.59999
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	603.3 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

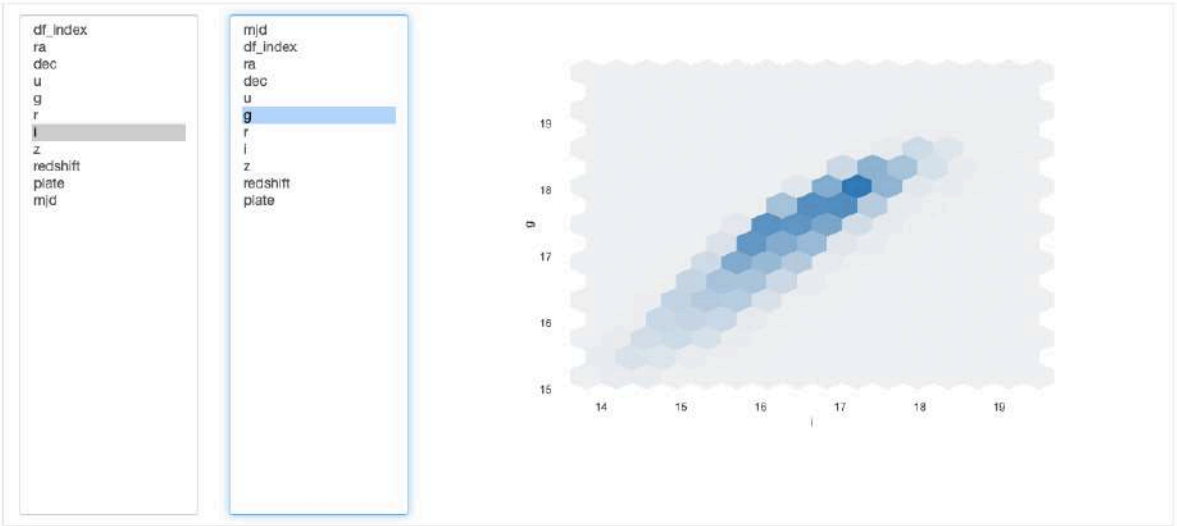
Quantile statistics

Minimum	16.61704
5-th percentile	17.256437
Q1	18.2387525
median	18.87246
Q3	19.2710825
95-th percentile	19.5344765
Maximum	19.59999
Range	2.98295
Interquartile range (IQR)	1.03233

Descriptive statistics

Standard deviation	0.7120167169
Coefficient of variation (CV)	0.03809771123
Kurtosis	-0.1684168912
Mean	18.68922552
Median Absolute Deviation (MAD)	0.468355
Skewness	-0.8318198113
Sum	1442995.102
Variance	0.5069678051
Monotonicity	Not monotonic

Interactions

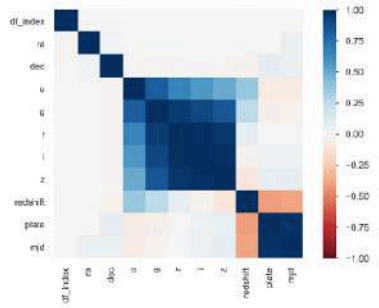


In the above interactive window, we can select the variable pair and observe how they interact with each other.

Different Correlations displayed in the HTML File

Correlations

Spearman's ρ | Pearson's r | Kendall's τ | Phik (φk) | Toggle correlation descriptions

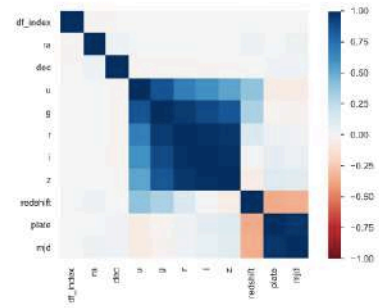


Spearman's ρ

The Spearman's rank correlation coefficient (ρ) is a measure of monotonic correlation between two variables, and is therefore better in catching nonlinear monotonic correlations than Pearson's r. It's value lies between -1 and +1, -1 indicating total negative monotonic correlation, 0 indicating no monotonic correlation and 1 indicating total positive monotonic correlation.

To calculate ρ for two variables *X* and *Y*, one divides the covariance of the rank variables of *X* and *Y* by the product of their standard deviations.

Spearman's ρ | Pearson's r | Kendall's τ | Phik (φk) | Toggle correlation descriptions

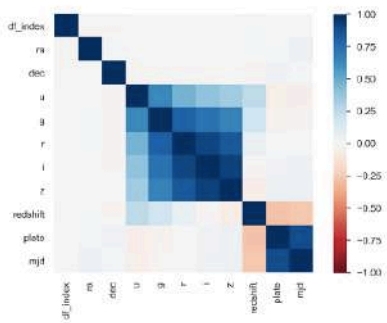


Pearson's r

The Pearson's correlation coefficient (r) is a measure of linear correlation between two variables. It's value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation. Furthermore, *r* is invariant under separate changes in location and scale of the two variables, implying that for a linear function the angle to the x-axis does not affect *r*.

To calculate *r* for two variables *X* and *Y*, one divides the covariance of *X* and *Y* by the product of their standard deviations.

Spearman's ρ | Pearson's r | Kendall's τ | Phik (φk) | Toggle correlation descriptions

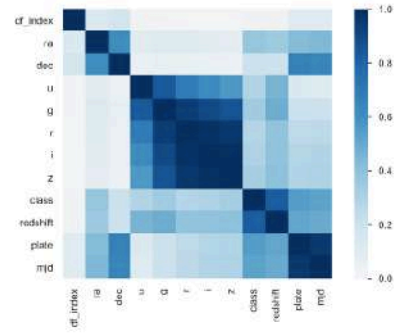


Kendall's τ

Similarly to Spearman's rank correlation coefficient, the Kendall rank correlation coefficient (τ) measures ordinal association between two variables. It's value lies between -1 and +1, -1 indicating total negative correlation, 0 indicating no correlation and 1 indicating total positive correlation.

To calculate τ for two variables *X* and *Y*, one determines the number of concordant and discordant pairs of observations. τ is given by the number of concordant pairs minus the discordant pairs divided by the total number of pairs.

Spearman's ρ | Pearson's r | Kendall's τ | Phik (φk) | Toggle correlation descriptions



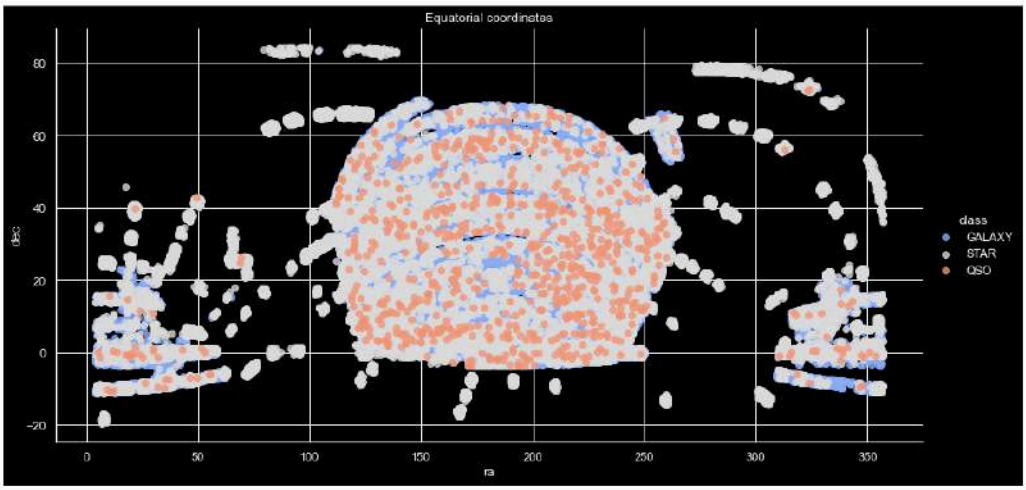
Phik (φk)

Phik (φk) is a new and practical correlation coefficient that works consistently between categorical, ordinal and interval variables, captures non-linear dependency and reverts to the Pearson correlation coefficient in case of a bivariate normal input distribution. There is extensive documentation available here.

5.7 Ra Vs Dec : Equatorial Coordinates Plot

In [38]:
```
sns.lmplot(x='ra', y='dec', data= df, hue='class', fit_reg=False, palette='coolwarm', size=6, aspect=2)
plt.title('Equatorial coordinates')
```

Out[38]:
```
Text(0.5, 1.0, 'Equatorial coordinates')
```



6. ML Algorithms

Encoding class labels For some cases, we cannot simply provide categorical values (just strings). Instead, we can convert them to numerical values. For example, since we have 3 classes, we able to assign to each class some values, so that:

0 is for GALAXY

1 is for QSO

2 is for STAR

In [5]:
```
from sklearn.model_selection import train_test_split, cross_val_predict, cross_val_score
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import confusion_matrix, precision_score
import time
```

In [6]:
```
# Some algorithms don't support categorical classes so we'll have to replace them with numbers
d=pd.DataFrame(df)


class_num=pd.DataFrame(LabelEncoder().fit_transform(d['class']), columns=['class'])
d.drop(['class'], axis=1, inplace=True)
names=list(d)
```

In [7]:
```
#Data are normalized for better conditioning of the problem

scaler = MinMaxScaler()
d=pd.DataFrame(scaler.fit_transform(d), columns=names)


d=pd.concat([d, class_num], axis=1)

d.head(3)
```

Out[7]:

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	redshift	plate	mjd
0	0.000000	0.583531	0.225219	0.993319	0.548551	0.470302	0.443875	0.421129	0.259930	0.0	0.2	0.365145	0.128650	0.327491	0.128652	0.148096
1	0.333333	0.377967	0.609451	0.899177	0.644423	0.653889	0.645508	0.657176	0.333250	0.0	0.4	0.406639	0.123426	0.403386	0.123419	0.142626
2	0.000000	0.028469	0.333105	0.893860	0.511548	0.466311	0.442256	0.432259	0.224207	0.0	0.6	0.479253	0.033815	0.342533	0.033799	0.043152

In [8]:
```
#A cross validation will be performed to ensure the reliability of the results.

#In addition, an isolated training will serve to measure the times and extract a matrix of confusion than will give us

x=d.drop('class',axis=1);
y=d['class']

x_train, x_test, y_train, y_test = train_test_split(d.drop('class',axis=1), d['class'], test_size=0.4)
```

In [31]:
```
x_train
```

Out[31]:

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	redshift	plate	
71218	0.333333	0.919399	0.855501	0.967281	0.788925	0.350607	0.339249	0.372009	0.499814	0.0	0.0	0.174047	0.185576	0.000590	0.185571	0.314
95031	0.333333	0.335836	0.683758	0.954985	0.814232	0.379381	0.375460	0.421421	0.509003	0.0	0.8	0.120494	0.123286	0.000593	0.123280	0.202
62971	0.333333	0.468526	0.725262	0.880621	0.706220	0.313473	0.297589	0.318141	0.337390	0.0	0.2	0.229660	0.052277	0.007926	0.052276	0.104
57848	0.000000	0.402742	0.788083	0.940818	0.766394	0.348213	0.341528	0.379700	0.161803	0.0	0.8	0.043254	0.172388	0.000609	0.172380	0.299
81033	0.333333	0.687345	0.477307	0.791314	0.665635	0.321894	0.322945	0.362118	0.477959	0.0	1.0	0.112255	0.115547	0.000588	0.115544	0.253
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
77167	0.333333	0.653191	0.272384	0.991566	0.842422	0.388668	0.378355	0.420520	0.482677	0.0	0.0	0.201854	0.118804	0.015052	0.118801	0.362
59105	0.333333	0.407884	0.536072	0.973284	0.924197	0.441232	0.433283	0.488601	0.446417	0.0	0.6	0.094748	0.108224	0.226655	0.108216	0.189
62365	0.333333	0.644309	0.164427	0.887819	0.835740	0.369772	0.365160	0.432084	0.281883	0.0	0.2	0.145211	0.053666	0.186284	0.053660	0.109
7394	0.000000	0.463475	0.829872	0.985027	0.803263	0.357956	0.340593	0.372143	0.161803	0.0	0.6	0.114315	0.018251	0.016522	0.018240	0.043
59962	0.333333	0.545515	0.758408	0.938795	0.788980	0.348181	0.334530	0.360459	0.357631	0.0	0.4	0.071061	0.056352	0.021224	0.056347	0.109

60000 rows x 17 columns

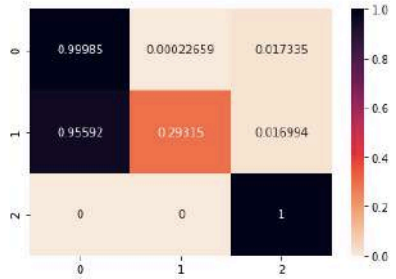
In [32]:
```
y_test
```

Out[32]:
```
79732    0
1890     2
75586    0
26394    2
20325    1
        ..
32436    2
51031    1
46523    2
57171    0
33508    0
Name: class, Length: 40000, dtype: int32
```

6.1 LOGISTIC REGRESSION

In [9]:
```
from sklearn import linear_model, datasets


lr = linear_model.LogisticRegression()

training_start = time.perf_counter()
lr.fit(x_train, y_train)#Training
training_end = time.perf_counter()

predict_start = time.perf_counter()
preds = lr.predict(x_test)#Prediction
predict_end = time.perf_counter()
acc_lreg = (preds == y_test).sum().astype(float) / len(preds)*100

print("The first iteration of the Logistic Regression gives an accuracy of the %3.2f %%" % (acc_lreg))

from numpy import linalg as LA
mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g',);

print("[0=Galaxy 1=Quasar 2=Star]")
```

```
The first iteration of the Logistic Regression gives an accuracy of the 98.14 %
[0=Galaxy 1=Quasar 2=Star]
```



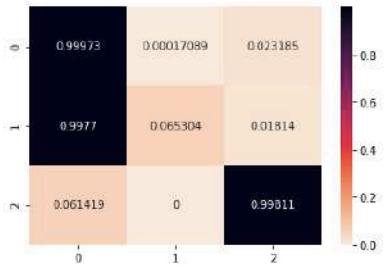
In [10]:
```
lr_train_t=training_end-training_start;
lr_predict_t=predict_end-predict_start;

scores = cross_val_score(lr, x, y, cv=10, scoring = "accuracy")
score_lr=scores.mean()
print("The 10 cross validations of Logistic Regression have had an average success rate of  %3.2f %%" %(score_lr*100))
std_lr=scores.std()
print("..and a standar deviation of %8.5f" %(std_lr))
```

```
The 10 cross validations of Logistic Regression have had an average success rate of  98.15 %
..and a standar deviation of  0.00128
```

6.2 KNN Neighbors

In [11]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

training_start = time.perf_counter()
knn.fit(x_train, y_train)
training_end = time.perf_counter()

predict_start = time.perf_counter()
preds = knn.predict(x_test)
predict_end = time.perf_counter()
acc_knn = (preds == y_test).sum().astype(float) / len(preds)*100

print("The first iteration of the K-Nearest Neighbours gives an accuracy of the %3.2f %%" % (acc_knn))



mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g')


print("[0=Galaxy 1=Quasar 2=Star]")
```

```
The first iteration of the K-Nearest Neighbours gives an accuracy of the 95.47 %
[0=Galaxy 1=Quasar 2=Star]
```



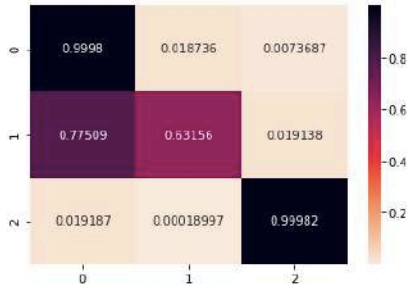
In [12]:
```
knn_train_t=training_end-training_start;
knn_predict_t=predict_end-predict_start;

scores = cross_val_score(knn, x, y, cv=10, scoring = "accuracy")
score_knn=scores.mean()
print("The 10 cross validations of K- Nearest Neighbours have had an average success rate of %3.2f %%" %(score_knn*100)
std_knn=scores.std()
print("..and a standar deviation of %8.5f" %(std_knn))
```

```
The 10 cross validations of K- Nearest Neighbours have had an average success rate of 95.25 %
..and a standar deviation of  0.00342
```

6.3 NAIVE BAYES

In [13]:
```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

training_start = time.perf_counter()
gnb.fit(x_train, y_train)
training_end = time.perf_counter()

predict_start=time.perf_counter()
preds = gnb.predict(x_test)
predict_end = time.perf_counter()
acc_gnb = (preds == y_test).sum().astype(float) / len(preds)*100


print("The first iteration of the naive Bayes gives an accuracy of the %3.2f %%" % (acc_gnb))


mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g');
```

```
The first iteration of the naive Bayes gives an accuracy of the 97.15 %
```



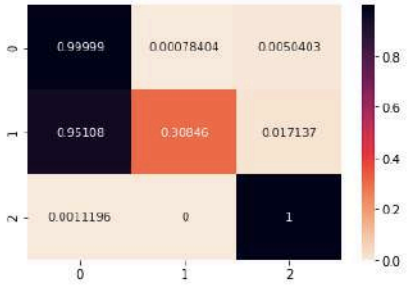
In [14]:
```
gnb_train_t=training_end-training_start;
gnb_predict_t=predict_end-predict_start;

scores = cross_val_score(gnb, x, y, cv=10, scoring = "accuracy")
score_gnb=scores.mean()
print("The 10 cross validations of naive Bayes have had an average success rate of %3.2f %%" %(score_gnb*100))
std_gnb=scores.std()
print("..and a standar deviation of %8.6f" %(std_gnb))
```

```
The 10 cross validations of naive Bayes have had an average success rate of 97.00 %
..and a standar deviation of 0.001918
```

6.4 RANDOM FOREST

In [15]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=10)

training_start = time.perf_counter()
rfc.fit(x_train, y_train)
training_end = time.perf_counter()

predict_start=time.perf_counter()
preds = rfc.predict(x_test)
predict_end = time.perf_counter()

acc_rfc = (preds == y_test).sum().astype(float) / len(preds)*100


print("The first iteration of the Random Forest gives an accuracy of the %3.2f %%" % (acc_rfc))


mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g');
```

```
The first iteration of the Random Forest gives an accuracy of the 98.82 %
```



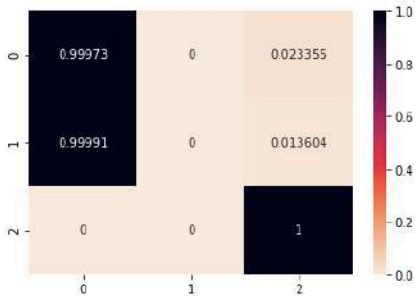
In [16]:
```
rfc_train_t=training_end-training_start;
rfc_predict_t=predict_end-predict_start;

scores = cross_val_score(rfc, x, y, cv=10, scoring = "accuracy")
score_rfc=scores.mean()
print("The 10 cross validations of Random Forest have had an average success rate of %3.2f" %(score_rfc*100))
std_rfc=scores.std()
print("..and a standar deviation of %8.6f" %(std_rfc))
```

```
The 10 cross validations of Random Forest have had an average success rate of 98.96
..and a standar deviation of 0.000890
```

6.5 SUPPORT VECTOR MACHINES

In [17]:
```
from sklearn.svm import SVC

svm = SVC(kernel='sigmoid', gamma='auto')

training_start = time.perf_counter()
svm.fit(x_train, y_train)
training_end = time.perf_counter()

predict_start = time.perf_counter()
preds = svm.predict(x_test)
predict_end = time.perf_counter()

acc_svm = (preds == y_test).sum().astype(float) / len(preds)*100


print("The first iteration of the SVM gives an accuracy of the %3.2f %%" % (acc_svm))


mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g');
```

```
The first iteration of the SVM gives an accuracy of the 97.56 %
```



In [18]:
```
svm_train_t=training_end-training_start;
svm_predict_t=predict_end-predict_start;

scores = cross_val_score(svm, x, y, cv=10, scoring = "accuracy")
score_svm=scores.mean()
print("The 10 cross validations of SVM have had an average success rate of %3.2f %%" %(score_svm*100))
std_svm=scores.std()
print("..and a standar deviation of %8.6f" %(std_svm))
```

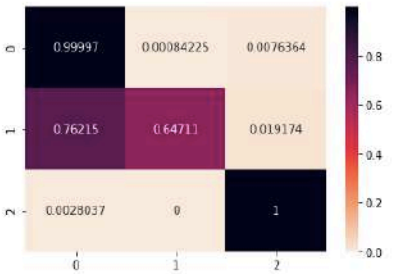
```
The 10 cross validations of SVM have had an average success rate of 97.53 %
..and a standar deviation of 0.001306
```

6.6 NEURAL NETWORKS

Multi-layer Perceptron classifier (MLPC)

In [19]:
```
from sklearn.neural_network import MLPClassifier

nnc = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(4, 3), random_state=1)

training_start = time.perf_counter()
nnc.fit(x_train, y_train)
training_end = time.perf_counter()

predict_start = time.perf_counter()
preds=nnc.predict(x_test)
predict_end=time.perf_counter()

acc_nnc = (preds == y_test).sum().astype(float) / len(preds)*100
print("The first iteration of the Neural Networks gives an accuracy of the %3.2f %%" % (acc_nnc))
mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g');
```

```
The first iteration of the Neural Networks gives an accuracy of the 98.81 %
```



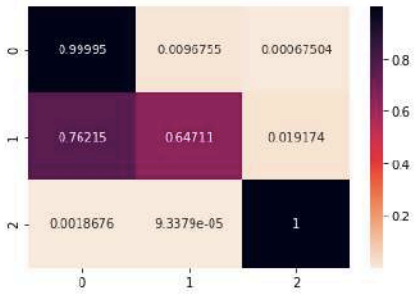
In [20]:
```
nnc_train_t=training_end-training_start;
nnc_predict_t=predict_end-predict_start;

scores = cross_val_score(nnc, x, y, cv=10, scoring = "accuracy")
score_nnc=scores.mean()
print("The 10 cross validations of Neural Networks have had an average success rate of %3.2f %%" %(score_nnc*100))
std_nnc=scores.std()
print("..and a standar deviation of %8.6f" %(std_nnc))
```

```
The 10 cross validations of Neural Networks have had an average success rate of 98.60 %
..and a standar deviation of 0.002011
```

6.7 DECISION TREE

In [23]:
```
from sklearn.tree import DecisionTreeClassifier

nnc = DecisionTreeClassifier(criterion='entropy', random_state=0)

training_start = time.perf_counter()
nnc.fit(x_train, y_train)
training_end = time.perf_counter()

predict_start = time.perf_counter()
preds=nnc.predict(x_test)
predict_end=time.perf_counter()

acc_nnc = (preds == y_test).sum().astype(float) / len(preds)*100
print("The first iteration of the Decision tree gives an accuracy of the %3.2f %%" % (acc_nnc))
mc=confusion_matrix(y_test, preds)
mc_norm = mc / np.linalg.norm(mc, axis=1, keepdims=True)
sns.heatmap(pd.DataFrame(mc_norm), cmap=sns.cm.rocket_r, annot=True, fmt='.5g');
```

```
The first iteration of the Decision tree gives an accuracy of the 98.73 %
```



In [24]:
```
nnc_train_t=training_end-training_start;
nnc_predict_t=predict_end-predict_start;

scores = cross_val_score(nnc, x, y, cv=10, scoring = "accuracy")
score_nnc=scores.mean()
print("The 10 cross validations of Decision tree have had an average success rate of %3.2f %%" %(score_nnc*100))
std_nnc=scores.std()
print("..and a standar deviation of %8.6f" %(std_nnc))
```

```
The 10 cross validations of Decision tree have had an average success rate of 98.74 %
..and a standar deviation of 0.000517
```

CHAPTER IV

RESULTS AND DISCUSSION

EDA Section

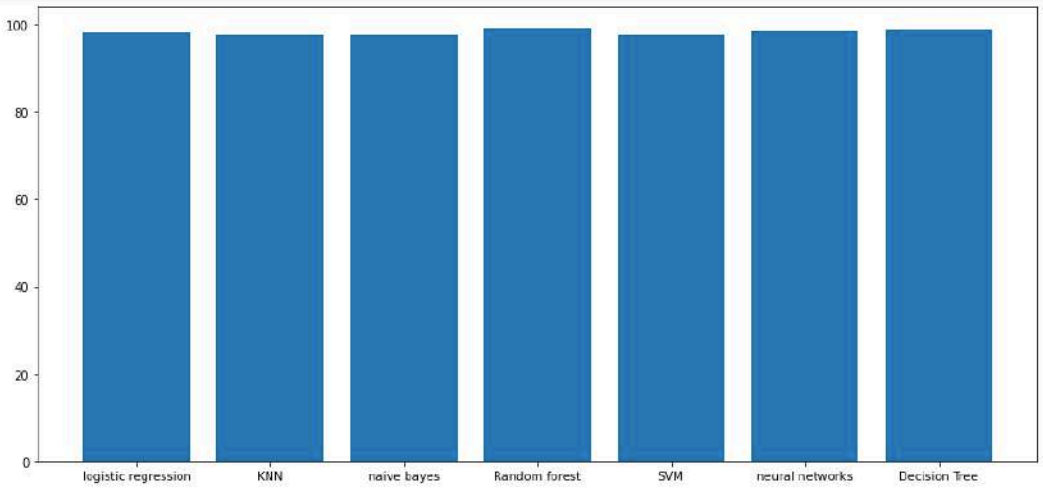
1. Stars have the lowest average redshift, followed by Galaxies and then Quasars.
2. u,g,r,i,z correlation looks in accordance with expected physical behaviour - Hotter objects emit more of every wavelength.
3. all wavelength radiations are strongly correlated except u as u has low correlation.
4. Redshift gives us the distance between us and the object. Redshift value based on the increase in wavelength.
5. There is not much difference in distribution according to class, but I can see that there are some characteristics to distinguish star class.
6. From the violin plot it can be said that, value of obj_ID doesn't contribute in classifying Star or Galaxy.
7. From the violin plot it can be concluded that values of alpha doesn't contribute in classifying Star and Galaxy. So, classification of star and galaxy doesn't depend on the right ascension angle
8. In the 'redshift' feature, if the value is negative(blueshift), the observation is more likely to be a Star. If the value is positive(redshift), the observation is more likely to be Galaxy.
9. Pandas profiling was done and elaborate EDA results were compiled as an HTML file.

ML Section

1. Random Forest scores the best accuracy and the minimum deviation, so we can consider it as the best classifier for this problem.
2. Galaxies are easier to separate than stars and quasars could have done.
3. Although quasars and stars shows parallel quantities, they have some distinctive statistical distributions to filters to help to decide which one it is.

COMPARISON OF MODEL ACCURACIES

In [96]:
```
data = {'logistic regression':score_lr,'KNN':score_knn,'naive bayes':score_gnb,
        'Random forest':score_rfc,'SVM':score_svm,'neural networks':score_nnc,'Decision Tree':score_nnd}
val1 = list(data.keys())
val2 = list(data.values())
per = [element * 100 for element in val2]

for index, value in enumerate(per):
    plt.text(value, index,
             str(value))

fig = plt.figure(figsize = (15,7))
plt.bar(val1,per)


plt.show()
```



In [93]:
```
per
```

Out[93]:
```
[98.16085999222899,
 97.63113586323013,
 97.43297500323791,
 99.08820101023184,
 97.53011267970471,
 98.40046626084705,
 98.79160730475327]
```

CHAPTER V

CONCLUSIONS

CONCLUSIONS

We have always been presented a colourful and cheesy version of Astrophysics and Cosmology in Tv shows and Science books but in reality its mostly about staring at data tables, graphs and equations for majority of your time. I presume this study serves as a culmination of my revelations about science and the real work that scientists do. This study has partially been successful at providing a glimpse in to the tools and methodology that researchers apply on real world problems. This is a humble attempt at representing a cross section of the real scientific analysis that happens all around the world even as this is written.

- **The application of EDA on Astronomy DataSet from SDSS has given insights into how photometric parameters (FEATURES) contribute to class labels(STAR< GALAXY, QUASAR)**
- **EDA with Statistical summaries as well as extensive visualizations provided the guidelines to transform the data as input for ML models.**
- **A comparison of the accuracy and performance of the different ML models were done.**

REFERENCES

1. The data released by the SDSS is under public domain. It's taken from the current data release RD14.
   More information about the license:
   http://www.sdss.org/science/image-gallery/

2. It was acquired by querying the CasJobs database which contains all the data published by the SDSS. The exact query can be found at:
   http://skyserver.sdss.org/CasJobs/ (Free account is required)

3. There are also other ways to get data from the SDSS catalogue. They can be found under:
   http://www.sdss.org/dr14/

4. Scikit-Learn Implementation of SVM:
   https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

5. https://medium.com/@suvigya2001/the-gaussian-rbf-kernel-in-non-linear-svm-2fb1c822aae0
6. https://engineering.papercup.com/posts/kernel-methods/
7. https://machinelearningmastery.com/support-vector-machines-for-machine-learning/
8. http://skyserver.sdss.org/dr2/en/proj/advanced/color/sdssfilters.asp
9. https://www.celestron.com/blogs/knowledgebase/what-are-ra-and-dec
10. Data Release 2 of S-PLUS: Accurate template-fitting based photometry covering 1000 deg2 in 12 optical filters.
11. miniJPAS survey: star-galaxy classification using machine learning.