Classification of Stars

Massimo Iberti

2022-03-07

Introduction

This report was created as part of the assignment for the Capstone course of HarvardX's Data Science Professional Certificate.

The aim of this project is to built a classification model for astronomical objects based on the characteristics of the spectrum of their light.

A classification of object is a fundamental part in Astronomy. Many telescopes scattered around the world and in space provide large quantity of information about visible universe. Organizing the data by hand is not a fisible process. Here the idea to create a machine learning algorithm to help with the classification.

The dataset that we are taking into consideration can be found in https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17. It contains data released by the Sloan Digital Sky Survey (SDSS).

We will begin the analysis with the exploration of the dataset, containing 100k astronomical objects. Each of them is either a Star, a Galaxy or a Quasar. We will firstly study the position of the object in the sky, identifying the Milky Way and providing explanation about the coordinate system, and secondly we will dig into the exploration of the spectral characteristics.

Having built up a knowledge about the dataset, we will present two models capable of exploiting the distribution of the data. The first one is a combination of two generalized linear models, and the second one is a combination of a generalized linear model and a random forest model. We will then compare the result with a full random forest model and present our conclusions.

All the data used are released under public domain.

In this section we will download the libraries need to run the code.

The R code in the following sections is meant to run on R 4.0 or later version.

```
#
# load libraries needed in the calculations
#

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.7
```

```
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ----------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: knitr
```

```
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: corrplot
```

```
## corrplot 0.92 loaded
```

```
library(corrplot)
library(randomForest)
library(tidyverse)
library(caret)
library(data.table)
library(rpart)
library(gridExtra)
library(knitr)
```

Analysis and Methods

For convenience the dataset has been uploaded in github and we proceed to download it

```
# the code below downloads the zip archive from github and unzip it
# it populates the star_classification dataset

dl <- tempfile()
download.file("https://github.com/miberti/mlearning-stars/raw/main/archive_Stellar.zip", dl)
star_classification <- read.csv(unzip(dl, "star_classification.csv"))
rm(dl)
```

We will first start from the investigation of the dataset.

The dataset contains 100000 observations and 18 columns.

Each row of the dataset contains information about a stellar object, including its Class (if it is a Star, a Galaxy or a Quasar) the celestial coordinates of the observation, some characteristics of the spectrum of the emitted light and other id's relative to the measurement of the data were taken in the SDSS Database.

```
str(star_classification)
```

```
## 'data.frame':    100000 obs. of  18 variables:
##  $ obj_ID     : num  1.24e+18 1.24e+18 1.24e+18 1.24e+18 1.24e+18 ...
##  $ alpha      : num  136 145 142 339 345 ...
##  $ delta      : num  32.495 31.274 35.582 -0.403 21.184 ...
##  $ u          : num  23.9 24.8 25.3 22.1 19.4 ...
##  $ g          : num  22.3 22.8 22.7 23.8 17.6 ...
##  $ r          : num  20.4 22.6 20.6 21.6 16.5 ...
```

```
##  $ i          : num  19.2 21.2 19.3 20.5 16 ...
##  $ z          : num  18.8 21.6 18.9 19.3 15.5 ...
##  $ run_ID     : int  3606 4518 3606 4192 8102 8102 7773 7773 3716 5934 ...
##  $ rerun_ID   : int  301 301 301 301 301 301 301 301 301 301 ...
##  $ cam_col    : int  2 5 2 3 3 3 2 2 5 4 ...
##  $ field_ID   : int  79 119 120 214 137 110 462 346 108 122 ...
##  $ spec_obj_ID: num  6.54e+18 1.18e+19 5.15e+18 1.03e+19 6.89e+18 ...
##  $ class      : chr  "GALAXY" "GALAXY" "GALAXY" "GALAXY" ...
##  $ redshift   : num  0.635 0.779 0.644 0.932 0.116 ...
##  $ plate      : int  5812 10445 4576 9149 6121 5026 11069 6183 6625 2444 ...
##  $ MJD        : int  56354 58158 55592 58039 56187 55855 58456 56210 56386 54082 ...
##  $ fiber_ID   : int  171 427 299 775 842 741 113 15 719 232 ...
```

The columns are:

- obj_ID : Identifier of the Stellar object in the database
- alpha : Right Ascension angle (at J2000 epoch)
- delta : Declination angle (at J2000 epoch)
- u : Ultraviolet filter in the photometric system
- g : Green filter in the photometric system
- r : Red filter in the photometric system
- i : Near Infrared filter in the photometric system
- z : Infrared filter in the photometric system
- run_ID : Run Number used to identify the specific scan
- rerun_ID : Rerun Number to specify how the image was processed
- cam_col : Camera column to identify the scanline within the run
- field_ID : Field number to identify each field
- spec_obj_ID: Unique ID used for optical spectroscopic objects
- class : The Class of the Stellar Object
- redshift : Redshift value based on the increase in wavelength
- plate : Plate ID, identifies each plate in SDSS
- MJD : Modified Julian Date, used to indicate when a given piece of SDSS data was taken
- fiber_ID : fiber ID that identifies the fiber that pointed the light at the focal plane in each observation

In this analysis we are just going to use the information of the class, the position, the spectral features of the astronomical object, hHence we start reducing the amount of space of our dataset, converting the class in factor and selecting only the subset of the features we will work with. Our new (lighter) dataset will be

```
star_classification <- star_classification %>% mutate(class = as.factor(class)) %>%
select(obj_ID,class,alpha,delta,u,g,r,i,z,redshift)
str(star_classification)
```

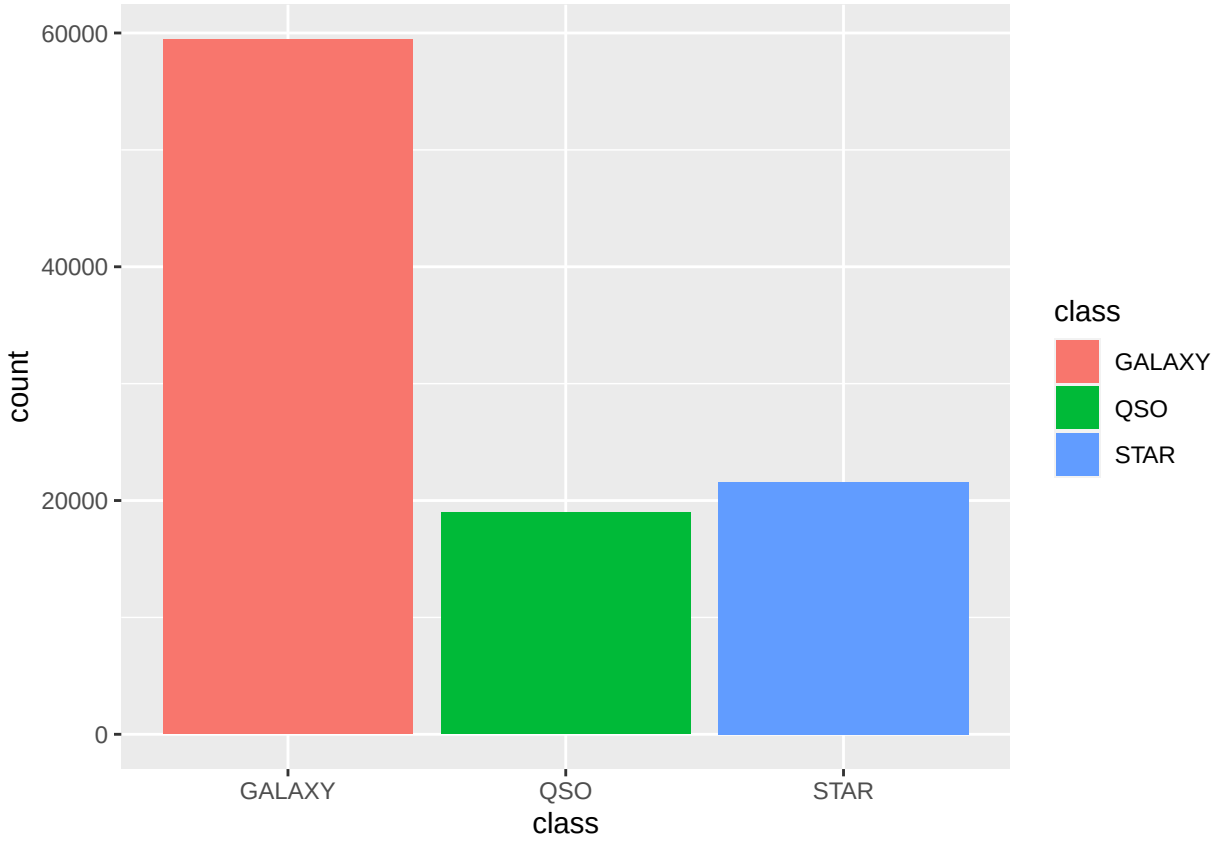
```
## 'data.frame':    100000 obs. of  10 variables:
##  $ obj_ID  : num  1.24e+18 1.24e+18 1.24e+18 1.24e+18 1.24e+18 ...
##  $ class   : Factor w/ 3 levels "GALAXY","QSO",..: 1 1 1 1 1 2 2 1 1 3 ...
##  $ alpha   : num  136 145 142 339 345 ...
##  $ delta   : num  32.495 31.274 35.582 -0.403 21.184 ...
##  $ u       : num  23.9 24.8 25.3 22.1 19.4 ...
##  $ g       : num  22.3 22.8 22.7 23.8 17.6 ...
##  $ r       : num  20.4 22.6 20.6 21.6 16.5 ...
##  $ i       : num  19.2 21.2 19.3 20.5 16 ...
##  $ z       : num  18.8 21.6 18.9 19.3 15.5 ...
##  $ redshift: num  0.635 0.779 0.644 0.932 0.116 ...
```

Data Exploration

In order to provide a model, we first need to understand the information carried by the dataset.

We will start plotting the amount of records for each class of objects.

```
ggplot(star_classification, aes(x=class,fill=class)) + geom_histogram(stat="count")
```



From the histogram we see that there are only three possible classes and that there are more galaxies than any other objects. There are roughly 60k galaxies, 20k stars and 20k quasars in the dataset.

The coordinate system

We will now investigate the distribution of this objects in the sky (for the plot to be readable, and to give an idea of the distribution, we will only print 4000 observations).

```
star_classification[ sample(nrow(star_classification),4000),] %>%
  ggplot(aes(x= alpha ,y=delta,color=class)) + geom_point(size = 0.6 )
```

In order to measure the position of an astronomical object, an appropriate coordinate system is in order. For a distant object, we measure its position with two numbers: the *right ascension* and the *declination*. The Right Ascension is the equivalent of the terrestrial longitude, while the Declination is the equivalent of Latitude.

Suppose to extend in space the terrestrial equator, what we obtain is the *celestial equator* and it is taken as a reference for the declination angle. Hence the declination is the angular degree of the position of the star from the celestial equator.

In order to measure the right ascension angle, we need a point of reference on the celestial equator. This point of zero is given by the position on the celestial equator of the Sun at the March equinox.

We remark that this coordinate system does not provide the distance between the celestial object and the Earth.

Furthermore this coordinate system provides a uniform way to identify the direction of a star in the sky, regardless of the point on Earth where the measurement is taken. If the object is very distant, its direction is practically the same regardless of the point on Earth where the measurement is taken.

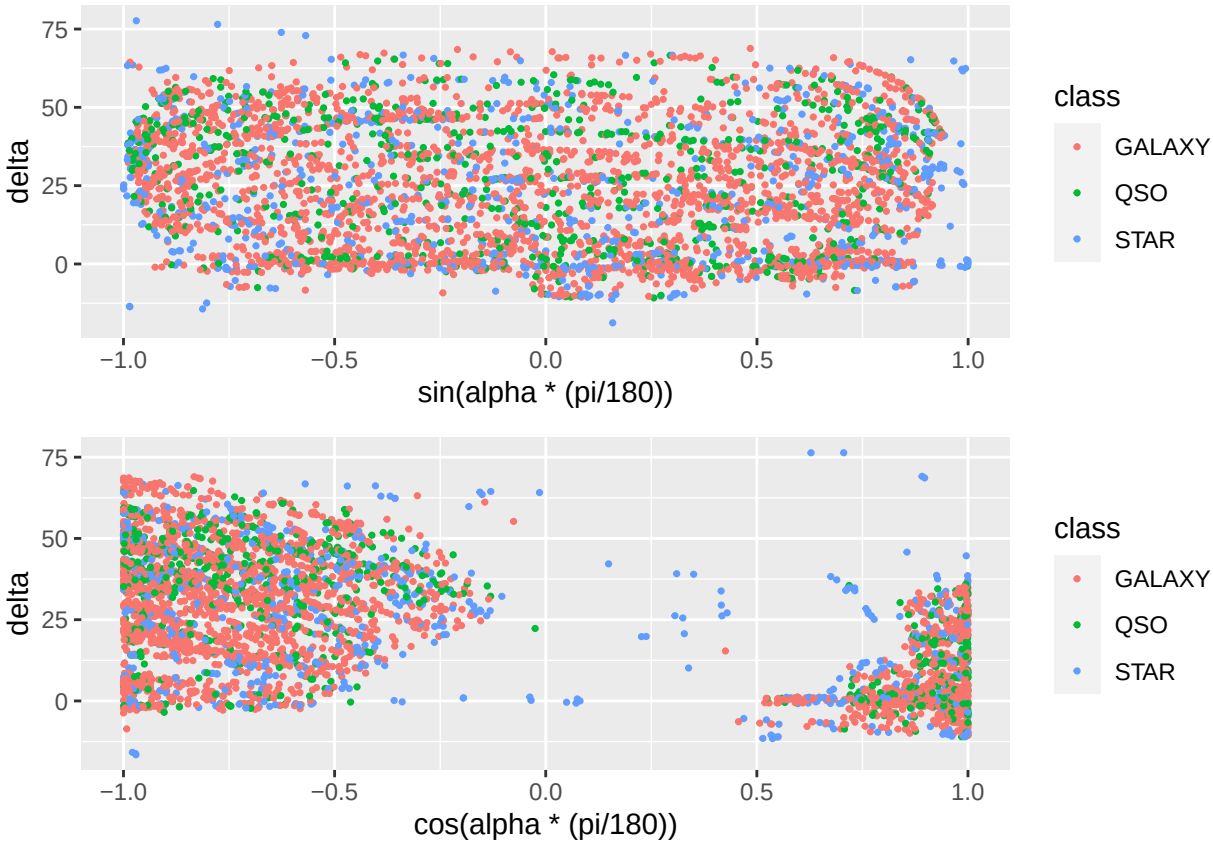
It is important to mention the fact that, due to the phenomenon of the precession of equinox, the reference point for the right ascension is rotating along the celestial equator, completing one circle every 26,000 years. For this reason it is also necessary to specify the year of the March equinox that we are using as a reference point. This year is called epoch, and the standard epoch astronomers are currently using is the *J2000*, which correspond to the year 2000 on the Julian calendar.

From the plot above we can see that the distribution of the stars differs slightly from the position of the galaxies and quasars. All of them are more easily found in a region of space in the direction of 0° and 180° right ascension, in what is known as the Milky Way. From the dataset we see that it is not uncommon to see stars outside of the Milky Way, while it is relatively rare to find quasars and galaxies out of it.

We can also notice from the plot a flaw of the method of representation. The right ascention coordinate, is a value between 0 and 360 that represents an angle. Objects having alpha value close to 360° are actually "close" to objects having alpha value of 0°. This tells us that we should be extra careful if we plan to apply linear models using this feature, as this feature is actually periodic.

Another approach to overcome this, is to use the value of some trigonometric functions of alpha, instead of the right ascension itself. We might be tempted to use both sin and cosine, to preserve the amount of information, however, from the plot below we might actually want to just keep the cosine of the right ascension, as it seem capable to capture the separation in the original plot.

```
p1<-star_classification[ sample(nrow(star_classification),4000),] %>%
  ggplot(aes(x= sin(alpha*(pi/180)) ,y=delta,color=class)) + geom_point(size = 0.6 )
p2<-star_classification[ sample(nrow(star_classification),4000),] %>%
  ggplot(aes(x= cos(alpha*(pi/180)) ,y=delta,color=class)) + geom_point(size = 0.6 )
grid.arrange(p1, p2, nrow = 2)
```

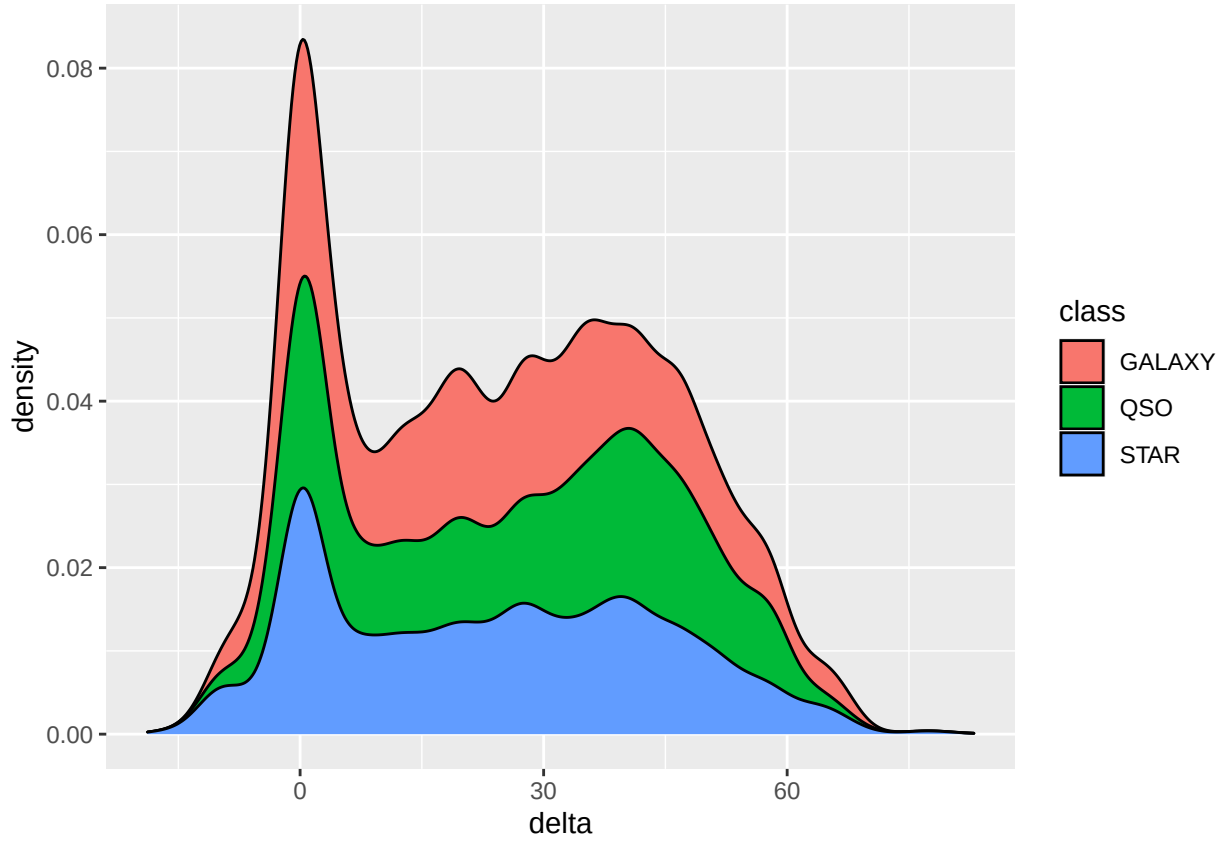


We will not follow this choice here, mainly because of the information loss in just keeping the value of the cosine and the fact that we are going to focus on the spectral features of the dataset.

In contrast, the declination feature is not periodic, but it doesn't seem to provide much information alone.

The majority of the astronomical objects can be found in the northern celestial hemisphere, with similar proportions among the categories.

```
ggplot(star_classification, aes(x=delta,fill=class)) + geom_density( position = "stack")
```



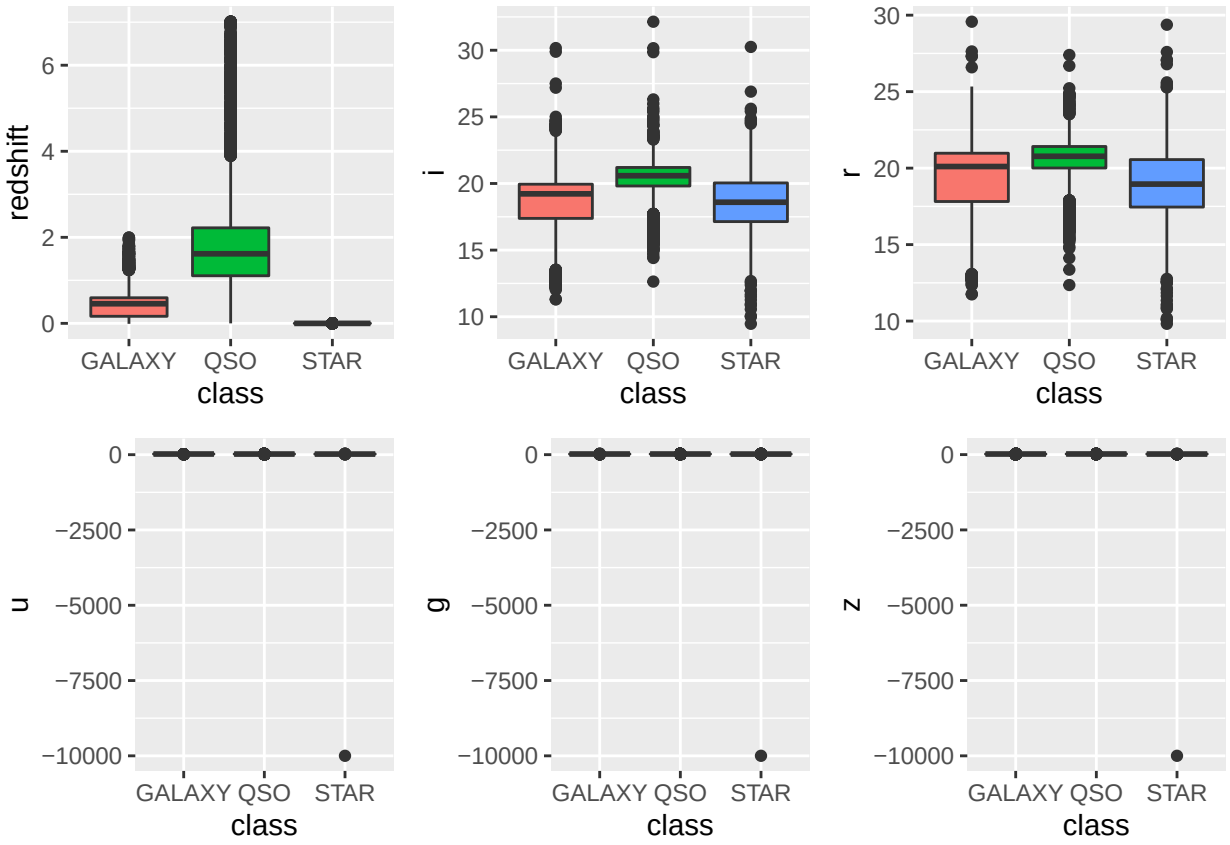
The spectral features

We are now looking at the spectral features of the data. They capture the frequency of the light registered by the telescope. As an electromagnetic wave, a photon of light has its own frequency and its own wavelength. As the telescope register the presence of many photons of light, it can measure their average frequency. Furthermore, applying a filter to the light, we can measure the amount of light in a particular band of frequency. The dataset contains information about the following filter

- u : Ultraviolet filter
- g : Green filter
- r : Red filter
- i : Near Infrared filter
- z : Infrared filter

in addition to the previous spectral features, the dataset contains the value of the redshift which is the increase of the wavelength due to the increase of the relative distance between the astral object and the Earth. The redshift is a consequence of the Doppler effect for electromagnetic waves and the fact that, due to the expansion of the universe, each object is slowly getting more distant from the others. The value of the redshift is usually subtracted from the frequency, as it is measured on Earth, in order to obtain its true value, regardless of the expansion of the universe.

We now display a box plot of the features, to better understand their distributions.



From the boxplot above we notice an a possible outlier for the ultraviolet, green and infrared filter.

Outiliers

Before analyzing the distributions of each feature, it is important to look for outliers in the dataset and remove them. The the previous section, we saw that there might be points with an extremely negative value for the g, u and z variable. Let's investigate further.

```
star_classification[star_classification$g < -1000, ] %>% select(obj_ID, class, u, g,z) %>%
  format(  scientific = FALSE) %>% kable()
```

	obj_ID	class	u	g	z
79544	1237648703521095936	STAR	-9999	-9999	-9999

The values of u,g and z for the star with obj_ID = 1237648703521095936 are suspicious and very different from their respective means.
A good test to see if this point is outlier in our dataset is the analysis of the standard deviation and the median absolute deviation of the distribution. The median absolute deviation (MAD) is the median of the absolute value of the difference between each observation and the mean, rescaled by an appropriate factor. We normally don't expect their value to differ.

In case of an outlier, however, the value of the standard deviation will be affected by it, but the MAD will not.

```
df <- summarise(star_classification,Feature = "Ultraviolet Filter", Column = "(u)" ,
  Mean= mean(u) ,"StandardDev" = sd(u) ,Median = median(u), "MedianAbsDev" =  mad(u))
df <- rbind(df,summarise(star_classification,Feature = "Green", Column = "(g)" ,
  Mean= mean(g) ,"StandardDev" = sd(g) ,Median = median(g), "MedianAbsDev" =  mad(g)))
df <- rbind(df,summarise(star_classification,Feature = "Infrared", Column = "(z)" ,
  Mean= mean(z) ,"StandardDev" = sd(z) ,Median = median(z), "MedianAbsDev" =  mad(z)))
kable(df)
```

Feature	Column	Mean	StandardDev	Median	MedianAbsDev
Ultraviolet Filter	(u)	21.98047	31.76929	22.17913	2.471279
Green	(g)	20.53139	31.75029	21.09983	1.975564
Infrared	(z)	18.66881	31.72815	19.00459	1.737014

From the difference between the standard deviation and the MAD, we see that we are in presence of an outlier. For each of the three features taken into consideration, calculations show that the value -9999 lies more than 300 times the standard deviation away from the mean of the distribution, well beyond Tukey's definition of an "far out" outlier.

Henceforth we consider the value an outlier and remove it from the dataset. Statistically speaking, we refuse the null hypothesis that the point belongs to the distribution.

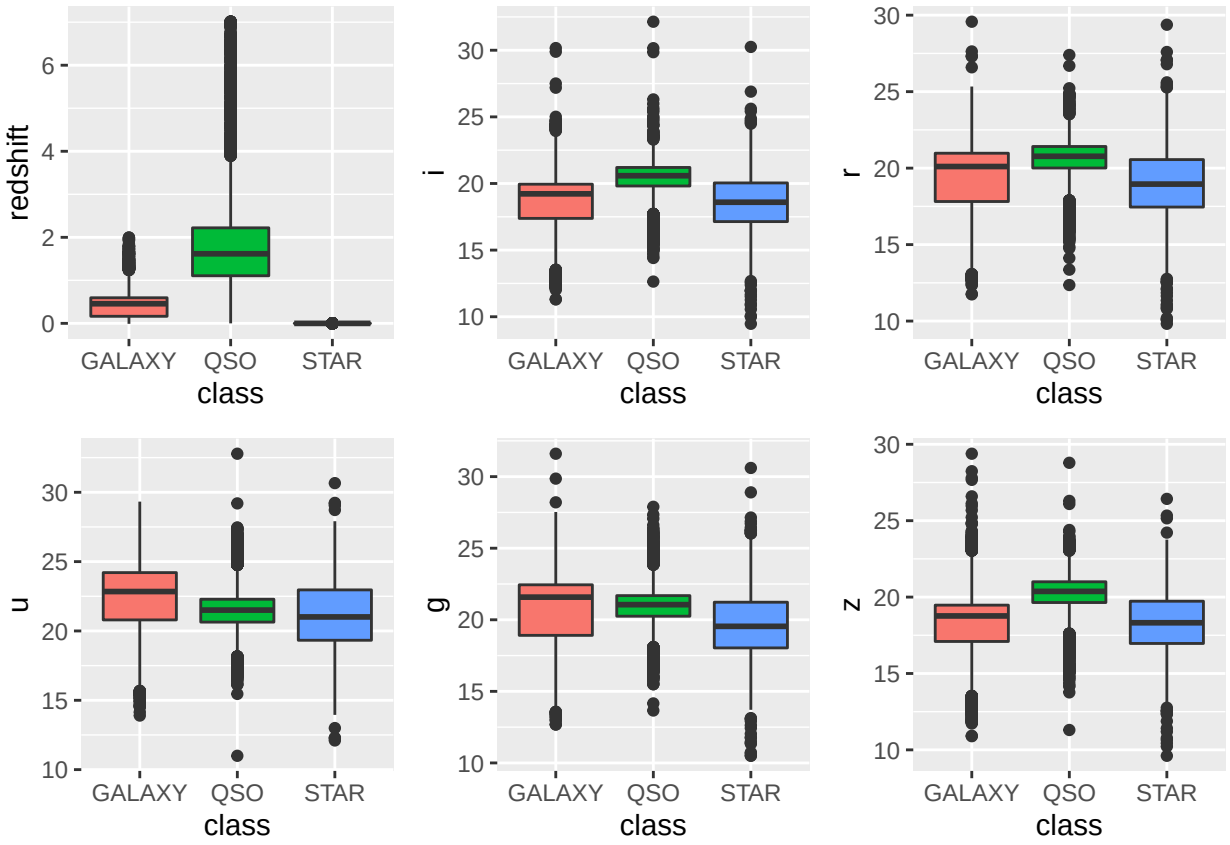
```
star_classification <-
  star_classification[star_classification$obj_ID != 1237648703521095936, ]
```

Let's see how that affected the mean and the standard deviation

```
df <- summarise(star_classification,Feature = "Ultraviolet Filter", Column = "(u)" ,
  Mean= mean(u) ,"StandardDev" = sd(u) ,Median = median(u), "MedianAbsDev" =  mad(u))
df <- rbind(df,summarise(star_classification,Feature = "Green", Column = "(g)" ,
  Mean= mean(g) ,"StandardDev" = sd(g) ,Median = median(g), "MedianAbsDev" =  mad(g)))
df <- rbind(df,summarise(star_classification,Feature = "Infrared", Column = "(z)" ,
  Mean= mean(z) ,"StandardDev" = sd(z) ,Median = median(z), "MedianAbsDev" =  mad(z)))
kable(df)
```

Feature	Column	Mean	StandardDev	Median	MedianAbsDev
Ultraviolet Filter	(u)	22.08068	2.251068	22.17914	2.471272
Green	(g)	20.63158	2.037384	21.09993	1.975446
Infrared	(z)	18.76899	1.765982	19.00460	1.736999

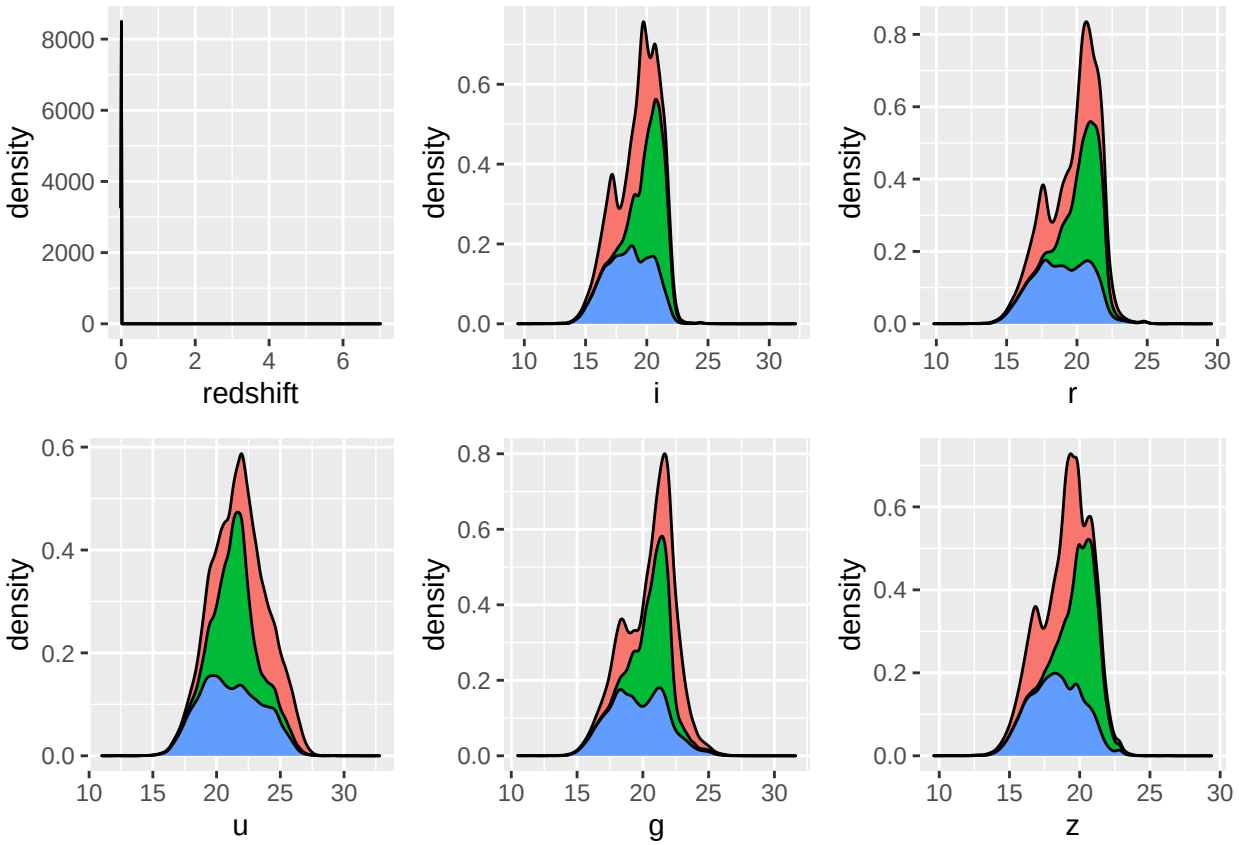
The standard variation and the MAD are much more similar now.
And this is the boxplot of the distribution of each single feature, without the outlier



Density

We can now investigate the distribution of each feature using a histogram

```
hrs  <- ggplot(star_classification, aes(redshift,fill=class)) + geom_density( position = "stack")+
  theme(legend.position = "none")
hi <- ggplot(star_classification, aes(i,fill=class)) + geom_density( position = "stack")+
  theme(legend.position = "none")
hr  <- ggplot(star_classification, aes(r,fill=class)) + geom_density( position = "stack")+
  theme(legend.position = "none")
hu  <- ggplot(star_classification, aes(u,fill=class)) + geom_density( position = "stack")+
  theme(legend.position = "none")
hg  <- ggplot(star_classification, aes(g,fill=class)) + geom_density( position = "stack")+
  theme(legend.position = "none")
hz  <- ggplot(star_classification, aes(z,fill=class)) + geom_density( position = "stack")+
  theme(legend.position = "none")
grid.arrange(hrs,hi,hr,hu, hg , hz, nrow = 2)
```

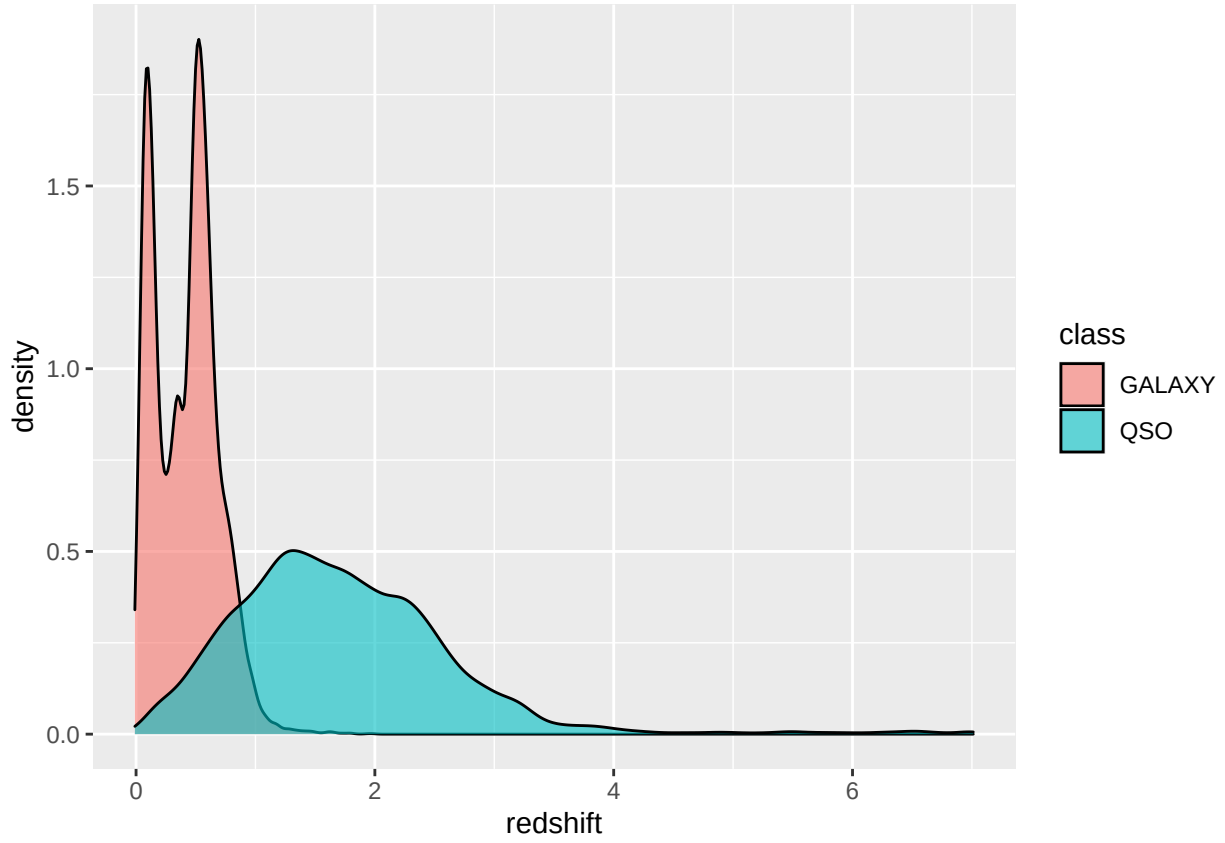


From the above graph, we can see that indeed the values of g,z and u lie between 10 and 30, and it is positive for all the other points. All the features but the redshift, are more or less concentrated around their mean, with the exception of the galaxies that seem to have two peaks in their distributions, the first on before the value 20 and the second one after the value 20.

We couldn't see that with a boxblot.

The boxplot graph is however much more clearer to represent the redshift. The plot of the density is not very informative due to the fact that basically all the stars have a value of the redshift close to zero. With this in mind, we try to see the distribution of the redshift just between the galaxies and the quasars

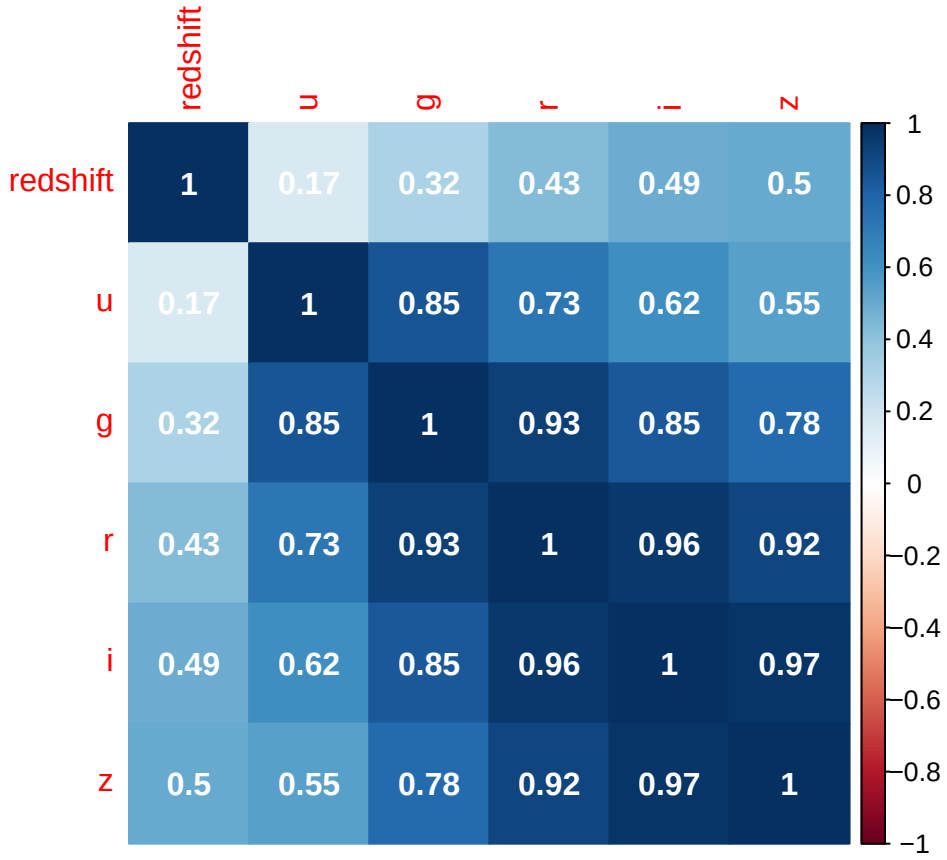
```
star_classification %>% filter(as.character(class) !="STAR") %>%
  ggplot(aes(redshift,fill=class)) + geom_density(alpha = 0.6)
```



Correlation

In the previous section, we have studied the distribution of each spectral features. As their distributions show some similarities, we ask ourself if there is any correlation between them.

```
star_classification %>% select(redshift,u,g,r,i,z) %>% cor  %>%
  corrplot( method = "color",addCoef.col = "white")
```



As we can see from the correlation matrix, all the variables seem to be correlated. The correlation is stronger among the pure measurement of the intensity of the light, and weak between the redshift and the other.

As we go from ultraviolet, green, red, near-infrared and infrared, we see that the correlation is higher, the closer is their wavelength. This is not surprising.

The redshift doesn't seem to be strongly correlated with the other spectral variables, but it is again more correlated to measurements of longer wavelength.

In the next sections we are going to present a model capable of exploiting this facts.

The models

As we are knowledgeable about the dataset, we are ready to find a method to classify the astronomical objects.

For the sake of clarity, we define extra binary variables, one for each value of the class.

```
star_classification <- star_classification %>%
mutate(isSTAR = as.factor(ifelse( as.character(class) =="STAR",1,0)),
       isQSO = as.factor(ifelse( as.character(class) =="QSO",1,0)),
       isGALAXY = as.factor(ifelse( as.character(class) =="GALAXY",1,0)))
```

We will start dividing the set into two parts, the train set and the test set. The train set will contain 90% of

the records of the dataset, randomly chosen, and the rest of the data will work as our test set.

```
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use set.seed(1)
```

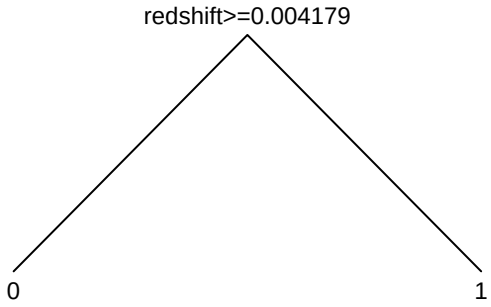
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <-
  createDataPartition(y = star_classification$class, times = 1, p = 0.1, list = FALSE)
train_set <- star_classification[-test_index,]
test_set <- star_classification[test_index,]
```

Separating Stars

From the boxplot of the previous section we can see that the redshift alone is an informative quantity when separating stars from other astronomical objects. For this reason, only for identification of stars, it might be sufficient to use a decision tree

```
fit_star_dtree <- rpart(isSTAR ~ redshift , data = train_set)
plot(fit_star_dtree, margin = 0.5 ,branch = 0 )
text(fit_star_dtree, cex = 0.75 )
```




```
#grid.arrange(brs,dtree_plot, ncol = 2)
```

and test this simple classification model, valid only for stars, on the test set. We then produce a confusion matrix to see the accuracy of our prediction

```
pred_star_dtree <- predict(fit_star_dtree,newdata = test_set,type = "class")
confusionMatrix(pred_star_dtree ,test_set$isSTAR)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7792    0
##          1   50 2160
##
##                Accuracy : 0.995
##                  95% CI : (0.9934, 0.9963)
##     No Information Rate : 0.784
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9854
##
##  Mcnemar's Test P-Value : 4.219e-12
##
##             Sensitivity : 0.9936
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9774
##              Prevalence : 0.7840
##          Detection Rate : 0.7790
##    Detection Prevalence : 0.7790
##       Balanced Accuracy : 0.9968
##
##        'Positive' Class : 0
##
```

We see that this simple decision model is sufficient to achieve an accuracy above 99%.

Because of the simplicity of the separation, we should be able to obtain a similar estimate using logistic regression. Let train a logistic model to identify stars.

```
fit_star_glm <-
  glm(  isSTAR ~ redshift + i + r + u + g + z , data = train_set , family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
y_star <- predict(fit_star_glm, test_set ,type = "response")
y_star <-  as.factor( ifelse(y_star >= 0.5,1,0) ) # declare a star if the probability is above .5
```

And then we calculate the accuracy of the logistic model

```
 confusionMatrix( y_star , test_set$isSTAR)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9947011
```

The accuracy we obtain is comparable the one obtained with the decision tree. It is interesting to look at the coefficients produced by the linear model.

```
fit_star_glm$coefficients
```

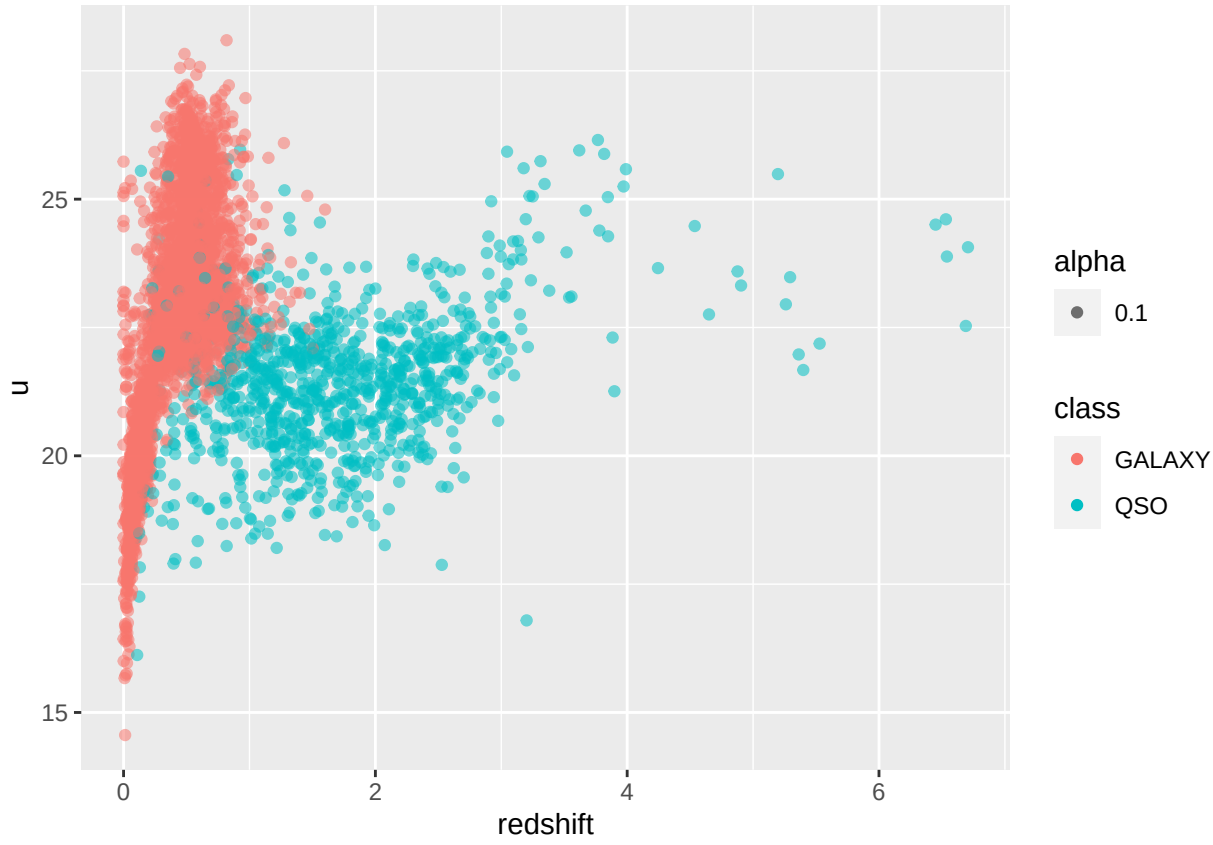
```
##  (Intercept)     redshift            i            r            u            g
##    7.8370346 -842.7105086   -0.2053485    0.2987490    0.1782072   -0.4645783
##            z
##   -0.0280998
```

As we expected, the most significant one is the coefficient of the redshift, as the separation along this variable is quite sharp.

We then focus our attention to the separation between galaxies and quasars. To this purpouse let us filter the dataset.

```
train_set_filtered <- train_set %>% filter(as.character(class) !="STAR")
test_set_filtered <- test_set %>% filter(as.character(class) !="STAR")
```

In this case we expect other features to play a significant role in the classification. Among them, we decide to plot the graph of the redshift against the light in the ultraviolet spectrum u, since it is the variable with less correlation compared to the redshift, as we saw in the previous section. For the readability of the graph, we are just going to plot 4000 points



We want to see how effective a linear classification model between galaxies and quasars can be. Hence we train a logistic regression model, but only on the train set without stars. This should give us a more precise model, than what we would have obtained with the whole set.

```
start_time <- Sys.time()
fit_galaxy_glm <- glm( isGALAXY ~ redshift + i + r + u + g + z ,
                       data = train_set_filtered, family = "binomial"  )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
end_time <- Sys.time()
comp_time_m1 <- end_time - start_time
```

We then apply the model to the test set without stars.

```
hat_galaxy <- predict(fit_galaxy_glm, test_set_filtered, type = "response")
hat_galaxy <- as.factor(ifelse(hat_galaxy >= 0.5,1,0))
confusionMatrix( hat_galaxy, test_set_filtered$isGALAXY)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1678  104
##          1  219 5841
##
##                Accuracy : 0.9588
##                  95% CI : (0.9542, 0.9631)
##     No Information Rate : 0.7581
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8853
##
##  Mcnemar's Test P-Value : 2.251e-10
##
##             Sensitivity : 0.8846
##             Specificity : 0.9825
##          Pos Pred Value : 0.9416
##          Neg Pred Value : 0.9639
##              Prevalence : 0.2419
##          Detection Rate : 0.2140
##    Detection Prevalence : 0.2272
##       Balanced Accuracy : 0.9335
##
##        'Positive' Class : 0
##
```

We see that this classification is not as accurate the previous one. Despite being a simple model, it is already classifying correctly almost 96% of the test dataset.

Two-step Generalized Linear Model

Let's summarize this considerations in this subsection into a two-step linear model to classify the stars. The steps are

- On the whole train set, train a logistic model A to classify the stars
- On the train set without the stars, train a logistic model B to classify galaxies
- Then classify a record as a star if the model A classify it as such, otherwise classify it as predicted by the outcome of model B

We have already trained the models, we just need to put them together

```
y_model1  <- test_set %>%
  mutate(p_star = predict(fit_star_glm, newdata = . ,type = "response"))  %>%
  mutate(p_galaxy = predict(fit_galaxy_glm,newdata = .  , type = "response")) %>%
  mutate(pred=ifelse(p_star>=0.5,"STAR",ifelse(p_galaxy>=0.5,"GALAXY","QSO")))%>%
  mutate(pred = as.factor(pred) )
```

Let's measure the accuracy of our model

```
cm_m1<-confusionMatrix( y_model1$pred, y_model1$class)
cm_m1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction GALAXY  QSO STAR
##     GALAXY   5789  219    1
##     QSO       104 1678    0
##     STAR       52    0 2159
##
## Overall Statistics
##
##                Accuracy : 0.9624
##                  95% CI : (0.9585, 0.966)
##     No Information Rate : 0.5944
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.933
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: GALAXY Class: QSO Class: STAR
## Sensitivity                 0.9738     0.8846      0.9995
## Specificity                 0.9458     0.9872      0.9934
## Pos Pred Value              0.9634     0.9416      0.9765
## Neg Pred Value              0.9609     0.9734      0.9999
## Prevalence                  0.5944     0.1897      0.2160
## Detection Rate              0.5788     0.1678      0.2159
## Detection Prevalence        0.6008     0.1782      0.2211
## Balanced Accuracy           0.9598     0.9359      0.9965
```

As we can see from the confusion matrix, the overall accuracy of the model, on the test set is above 96%.

We would like to exploit this idea of the two step classification model and improve our prediction using a Random Forest Model.

Two-Step Random Forest Model

We saw in the previous section that a decision tree was able to separate a star from the other astronomical object. A natural extention of decisions trees is given by the random forest model.

The random forest model is a powerful tool capable of modeling the non linearity of the data in the train set. The model is built taking a random subset of the train set and applying a decision tree on this subset. The process is repeated over many subset sampled randomly and then the final prediction is obtained averaging the values of all the predictions.

The main difficulty with the use of random forest model is that they suffer from what is known as "curse of dimensionality". The approach is then most effective when we can find a way to reduce the dimensionality of our data. But we saw in the previous sections that many of the spectral features are highly correlated, so we are going to perform a Principal Component analysis before applying the model. The aim of the PCA is to rotate the feature space, performing a change of coordinate, and highlight the most important directions the

data can be described with.

The second model we are about to present is also a two-step model. The first step (separation of stars) is identical to the first step in the previous model. Since we already trained the model, we are not going to repeat the training. We can focus on the second step, the separation between the galaxies and the quasars.

We are now going to find the principal components of the dataset. This process has to be done using only the train set. Since we want to focus on the difference between galaxies and quasars, we decide not to take into consideration the contribution of the stars for this change of coordinates. This is important, since we can tailor even more the principal component decomposition. Moreover the redshift proved to be a valuable information the the classification of stars.

```
pca <- train_set_filtered %>%
  prcomp( ~ redshift + i + r + u + g + z , data =.)
pca
```

```
## Standard deviations (1, .., p=6):
## [1] 3.7980836 1.6400844 0.6751489 0.5053512 0.2508348 0.1957434
##
## Rotation (n x k) = (6 x 6):
##                 PC1        PC2         PC3         PC4         PC5          PC6
## redshift 0.08206865 -0.2278817  0.66164028  0.70862716  0.02695945 -0.026221228
## i        0.41976121 -0.3497621  0.06303841 -0.17951556 -0.34311101  0.739960894
## r        0.45600041 -0.1857557 -0.19570534  0.07121164 -0.60297350 -0.592123055
## u        0.45528955  0.7594042  0.41829194 -0.19749986 -0.04503591 -0.003753058
## g        0.48639665  0.1652476 -0.54887407  0.49981425  0.40084491  0.156070109
## z        0.40652013 -0.4327004  0.20910961 -0.41440334  0.59604204 -0.277107644
```

```
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     3.7981 1.6401 0.67515 0.50535 0.25083 0.19574
## Proportion of Variance 0.8046 0.1500 0.02543 0.01424 0.00351 0.00214
## Cumulative Proportion  0.8046 0.9547 0.98011 0.99435 0.99786 1.00000
```

The output of the procedure is a $6x6$ matrix representing a change of coordinate.

From the summary above we can see that the first two components are responsible for more than 95% of the variability in the data and the the first three are responsible for 98%.

Let us plot a subset of or data along using first two principal components

```
train_set_filtered <- cbind(train_set_filtered,pca$x[,1:6])

 train_set_filtered[ sample(nrow(train_set_filtered),4000),] %>%
   ggplot(aes(PC1,PC2 )) +geom_point(aes(color = isGALAXY,alpha = 0.5))
```

We are going to train a random forest model for classification of quasar and galaxies. As the first three principal components describe the variability of the data

```
# the following code might take several minutes to complete
start_time <- Sys.time()
fit_galaxy_rf <- randomForest( isGALAXY ~ PC1 + PC2 + PC3, data = train_set_filtered)
end_time <- Sys.time()
comp_time_m2 <- end_time - start_time
```

Our second model, again a two step model, consists in:

- On the whole train set, train a logistic model A to classify the stars
- On the train set without the stars, train a random forest model C to classify the galaxies
- classify a record as a star if the model A classify it as such, otherwise classify it as predicted by the outcome of model C

In order to apply the random forest model, we first add to the test set the new coordinates, using the rotation matrix.

```
test_set_newcoords <- test_set[,c("redshift","i","r","u","g","z")] %>% as.matrix(.)%>%
  sweep( . , 2,pca$center) %*% pca$rotation %>% as.data.frame(.)
test_set <- cbind(test_set,test_set_newcoords)
```

and then we find the prediction

```
y_model2 <- test_set %>%
  mutate(p_star = predict(fit_star_glm, newdata = . ,type = "response")) %>%
  mutate(p_galaxy = predict(fit_galaxy_rf,newdata = . )) %>%
  mutate(pred=ifelse(p_star>=0.5,"STAR",
```

```
                  ifelse(as.character(p_galaxy)=="1","GALAXY","QSO")))%>%
  mutate(pred = as.factor(pred) )
cm_m2 <- confusionMatrix(y_model2$pred, y_model2$class)
cm_m2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction GALAXY  QSO STAR
##     GALAXY   5800  165    1
##     QSO        93 1732    0
##     STAR       52    0 2159
##
## Overall Statistics
##
##                Accuracy : 0.9689
##                  95% CI : (0.9653, 0.9722)
##     No Information Rate : 0.5944
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9448
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: GALAXY Class: QSO Class: STAR
## Sensitivity                 0.9756     0.9130      0.9995
## Specificity                 0.9591     0.9885      0.9934
## Pos Pred Value              0.9722     0.9490      0.9765
## Neg Pred Value              0.9641     0.9798      0.9999
## Prevalence                  0.5944     0.1897      0.2160
## Detection Rate              0.5799     0.1732      0.2159
## Detection Prevalence        0.5965     0.1825      0.2211
## Balanced Accuracy           0.9673     0.9508      0.9965
```

With the help of this technique, we were able to achieve an accuracy of almost 97%.

Full Random Forest Model

Of course one could wonder which accuracy can we get if we trained the random forest model using the set of all the initial features.

```
# the following code might take several minutes to complete
start_time <- Sys.time()
fit_rf_test <- randomForest( class ~ redshift + i+r+u+g+z , data = train_set )
end_time <- Sys.time()
comp_time_rf <- end_time - start_time
```

And then we produce the confusion matrix for the full random forest model

```
y_hat_rf_all <- predict(fit_rf_test,newdata = test_set  )
cm_rf <-confusionMatrix(y_hat_rf_all, test_set$class)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction GALAXY  QSO STAR
##     GALAXY   5860  120    0
##     QSO        75 1777    0
##     STAR       10    0 2160
##
## Overall Statistics
##
##                Accuracy : 0.9795
##                  95% CI : (0.9765, 0.9822)
##     No Information Rate : 0.5944
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9636
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: GALAXY Class: QSO Class: STAR
## Sensitivity                 0.9857     0.9367      1.0000
## Specificity                 0.9704     0.9907      0.9987
## Pos Pred Value              0.9799     0.9595      0.9954
## Neg Pred Value              0.9789     0.9853      1.0000
## Prevalence                  0.5944     0.1897      0.2160
## Detection Rate              0.5859     0.1777      0.2160
## Detection Prevalence        0.5979     0.1852      0.2170
## Balanced Accuracy           0.9781     0.9637      0.9994
```

As we can see, with the full random forest model we were able to achieve an accuracy close to 98%. We summarise the result in the next section.

Results

Here we report the considerations on the models.

```
models <- c("Two-step linear model",
            "Two-step linear and random forest model",
            "Random forest model")
accuracy <- c(cm_m1$overall["Accuracy"],cm_m2$overall["Accuracy"],cm_rf$overall["Accuracy"])
comptime <- c(comp_time_m1,comp_time_m2,comp_time_rf)
s <- data.frame("Model"=models,"Accuracy"=accuracy,"Computation time"=comptime)
kable(s)
```

Model	Accuracy	Computation.time
Two-step linear model	0.9624075	0.6841741 secs
Two-step linear and random forest model	0.9689062	50.2836869 secs
Random forest model	0.9795041	88.9589341 secs

As we can see the model with the most accuracy is the full random forest model. However, this comes with the requirement of an higher computational cost in terms of time.

In this particular case the difference between the last two computational times is relatively small, but in general for a larger number of features could become a problem (we remark that the computational times are dependent on the machine in use).

This examples show that, we can achieve similar accuracy using a model with a less features and less computational time, provided that we are able to tailor the model on the specific problem we are trying to solve.

Conclusion

We analyzed a dataset containing 100k observations of astronomical objects: stars, galaxies and quasars.
With the insights gained from looking at the data, we decided to build a two-step model, separating at first the stars with a first simple algorithm, and the the other two classes with a different model.
we deliberately didn't use the position of the object in our decision algorithms. Partly because we were interested in finding a decision model only based on the spectral characteristics of the object, and partly because the quasars and the galaxies pretty much occupy the same region in the astronomical sphere, and the stars are very much separated by the spectral features already. As a possible future work we could look at the position as well.
We could improve our model with the use a parameter for the random forest model, and then with the help of the cross-validation to find the best parameter. We didn't pursue this path, as the random forest model used is already quite accurate.
Another way we could improve the models provided is by assigning different weights to each observation when we train the models. As a matter of fact in the first chapter we observed an higher number of galaxies in our data. This means that in our training set, the galaxies are over represented.

Bibliography

- Abdurro'uf et al., The Seventeenth data release of the Sloan Digital Sky Surveys: Complete Release of MaNGA, MaStar and APOGEE-2 DATA (Abdurro'uf et al. submitted to ApJS)
  arXiv preprint: arXiv:2112.02026 .

- Astronomical Coordinate Systems https://en.wikipedia.org/wiki/Astronomical_coordinate_systems
- Photometric Systems https://en.wikipedia.org/wiki/Photometric_system