

# Spatial Observation Classification

Nicolas Kirsch

26/09/2019

## Introduction

When observing space, determining what is observed is a complicated task to do manually. An observation is composed of many different features (wavelength emissions. . .). Researchers have to compare the observed values for each feature to the standard values for every spatial object. It is interesting to create a machine learning algorithm which automatically classifies observations according to the spatial object studied.

We tried to create a machine learning algorithm able to classify observations between stars, galaxies and quasars. A quasar is a highly energetic galaxy. To do so, we used the “*Sloan Digital Sky Survey DR14*” dataset. This dataset was composed of 10,000 observation of space. Every observation had 17 features columns and one class column. The class column specified whether the observed object was a star, a galaxy or a quasar. The features were : “objid”, “ra”, “dec”, “u”, “g”, “r”, “i”, “z”, “run”, “rerun”, “camcol”, “field”, “specobjid”, “redshift”, “plate”, “mjd”, “fiberID”.

“run”, “rerun”, “camcol”, “field”, “fiberID”, “plate” were specification on the tools used to make the observation. “ra” and “dec” were the astronomical coordinates of the object. “u”, “g”, “r”, “i”, “z” were the intensity of different wavelengths observed. They followed the Thuan-Gunn astronomic magnitude system. “mjd” corresponded to the date of the observation. “objID” was an identifier for the observation in the original databank. Finally, “redshift” was a physical quantity giving us insight on the distance separating the spatial object from earth. We loaded the dataset from a github repository.

```
sky <- read.csv("https://raw.githubusercontent.com/Nicolaskirsch2000/
StellarClassification/master/GalaxyStarsQuasars.csv")
```

We tried to use different machine learning algorithm to predict to which class an observation belonged. By doing this, we hoped to find out which one would give the best accuracy. Our objective was to obtain an accuracy of at least 0.99.

## Data Analysis and Model creation

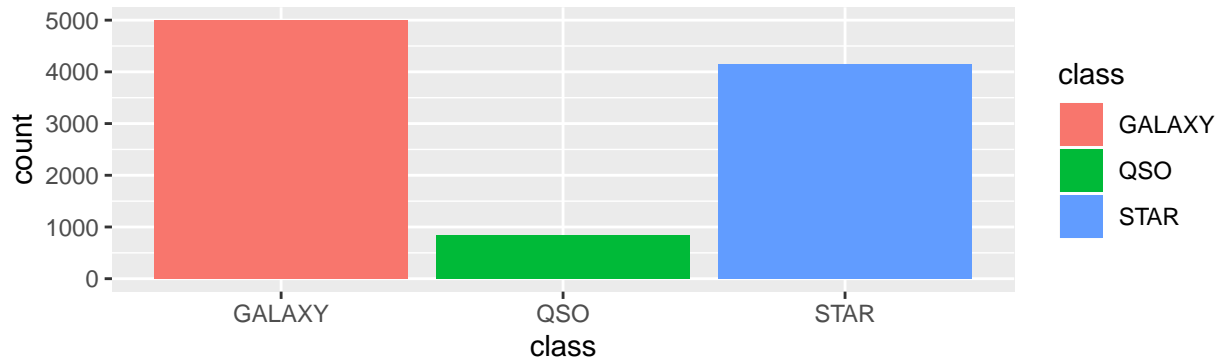
### Initital exploratory data analysis and filtering

The first thing we wanted to know was whether the dataset was complete or if some values were missing. To do so we checked for NA values in every columns.

```
colSums(is.na(sky))
```

```
##      objid      ra      dec      u      g      r      i
##         0         0         0         0         0         0         0
##         z      run      rerun      camcol      field specobjid      class
##         0         0         0         0         0         0         0
## redshift      plate      mjd      fiberid
##         0         0         0         0
```

There were no missing values in any column. We were able to proceed to data analysis and visualization. We also wanted to find out the repartition of observation between the three classes.



Most of the observations were galaxies and stars, with much less quasars.

A key part of our classification algorithm to determine was the features used to classify the observations. These features were the columns of the dataset (except class which is what we try to predict).

```
colnames(sky)
```

```
## [1] "objid"      "ra"         "dec"        "u"          "g"
## [6] "r"          "i"          "z"          "run"        "rerun"
## [11] "camcol"     "field"      "specobjid"  "class"      "redshift"
## [16] "plate"      "mjd"        "fiberid"
```

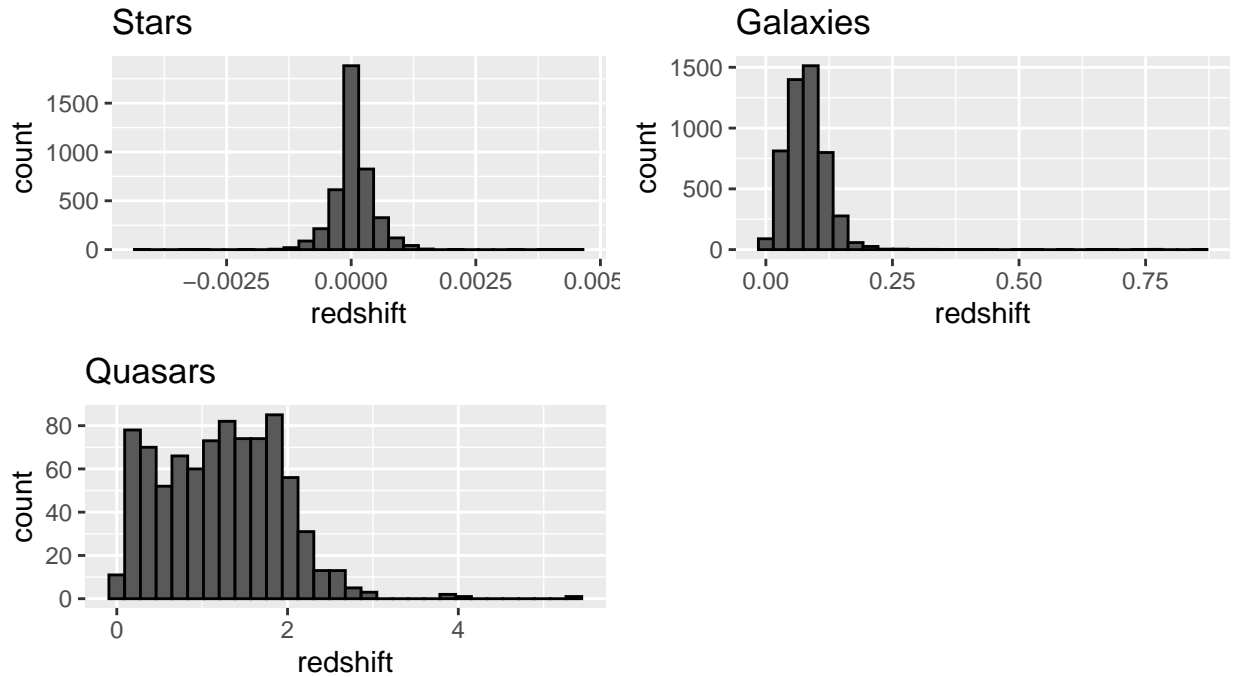
However, the columns “objid” and “specobjid” were not relevant for our algorithm as they were identifiers from the original databank. Furthermore, the columns “run”, “rerun”, “camcol” and “field” depended on the camera used to make the observation and not on the observed object. They were not giving useful informations on the observed object and would therefore not help to classify it. We decided to filter all those columns.

```
sky <- sky[, -c(1,9,10,11,12,13)]
colnames(sky)
```

```
## [1] "ra"         "dec"        "u"          "g"          "r"          "i"
## [7] "z"          "class"      "redshift"   "plate"      "mjd"        "fiberid"
```

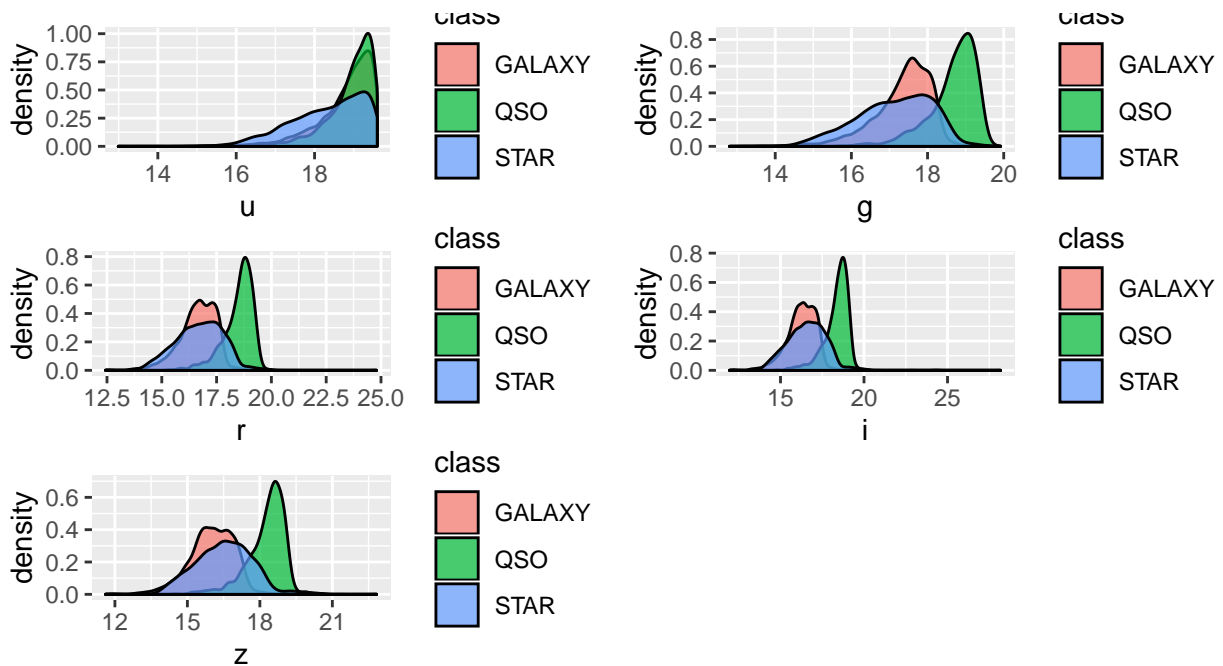
As only significant features were now remaining, we evaluated their distribution for each class. We started with the redshift (a value giving us information on the distance of the object)

## Redshift distribution for Stars , Galaxies and Quasars



We saw that the magnitude of the redshift varied significantly from one class to the other (Stars around 0.001, Galaxies 0.01 and Quasars 1). We deduced that there were significant variation in the distance to Earth between these observed objects, with stars being much closer than galaxies themselves closer than quasars. These variations also led us to think that redshift would be a useful predictor.

We then looked at the different wavelengths used to capture the observations (u, g, r, i, z)



For these features we could also see significant variability in the distribution between classes and therefore decided to keep them as predictors. We had a set of useful predictors and we were ready to separate the dataset into training and test set to start creating our models.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = sky$class, times = 1, p = 0.1, list = FALSE)
train_set <- sky[-test_index,]
test_set <- sky[test_index,]
```

## Random Classification

The easiest model we could create was a model randomly assigning an observation to a class, regardless of the predictors. To do so, we created a vector of the same length as the test set, randomly assigning one of the three classes.

```
random_classification <- as.factor(sample(c("GALAXY", "STAR", "QSO"), nrow(test_set), replace=TRUE))
```

Then, using a confusion matrix between this vector and the actual test set classes, we were able to obtain the accuracy of our model. The model's accuracy was **0.345**. This value was close to 1/3, which made sense as there were three predictors. As one could have thought, randomly classifying the observations gave us a very poor accuracy, far from our objective. We stored this value in a table in which we would add the accuracies of the different models.

```
accuracy_random
```

```
## Accuracy
##    0.3237
```

## Classifying using KNN

We then tried to ameliorate our model's accuracy using a KNN algorithm. As we saw during data exploration that the predictors varied strongly from one class to another but not as much inside a class, it made sense to use a model based on the similarity between the closest "neighbors". As KNN algorithm uses a tuning parameter  $k$ , we trained our algorithm on the training set to get the  $k$  value maximising the accuracy of the model.

```
knn_model <- train(class ~ .,
  method='knn',
  data = train_set,
  tuneGrid=data.frame(k=seq(0,200,20)))
```

We then used the optimised model on the test set to predict its classes. Creating a confusion matrix between the predicted and the actual classes, we obtained the accuracy of this model. This value was **0.803**

```
## Accuracy
##    0.8032
```

Simply using a KNN algorithm instead of randomly classifying them has a very strong impact on the accuracy as it increased it by more than 100%. However it still was not the desired accuracy, so we continued testing other models.

## QDA Classification

We also saw by visualizing the different predictors that their distributions for each classes were somewhat normal. Therefore by making the assumption that all the predictors were multivariate normal, we created a classification model based on Quadratic Discriminant Analysis. As for the KNN model, our first step was to train the model and the trainset

```
train_qda <- train(class ~ ., method = "qda", data = train_set)
```

We used this model to predict the test set classes and obtain its accuracy from a confusion matrix between the predicted and actual classes of the test set. We obtained an accuracy of **0.981**.

```
## Accuracy
##      0.981
```

This model was much more accurate, nearly attaining the 0.99 accuracy threshold.

## Decision tree Classification

As our objective was to create a classification algorithm, it made sense to use a decision tree, also called classification tree. This algorithm uses a tuning parameter called cp, which makes its flexibility vary. To get the optimal version of the model, we trained it using the train set.

```
train_dt <- train(class ~ ., method = "rpart",
                  tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),
                  data = train_set)
```

Using the same methodology as the previous models, we found out that using classification trees made the accuracy of the model rise up to **0.989**.

```
## Accuracy
##      0.989
```

We were nearly there, but still needed to ameliorate our model.

## Random Forest Classification

As we had just used decision trees, the logical next step was to use random forests. To do so we had to load a new package called “RandomForest”. Afterwards, we followed the same protocol, training the model on the train set and then predicting the classes of the test set using the new model. We recuperated the accuracy from the confusion matrix between actual and predicted values. This model gave us an accuracy of **0.990**

```
## Accuracy
##      0.99
```

The random forest model finally was sufficiently accurate to meet our accuracy threshold of 0.99.

## Results

```
## # A tibble: 5 x 2
##   method      Accuracy
##   <chr>      <dbl>
## 1 Random Classification 0.345
## 2 KNN model            0.803
## 3 QDA model            0.981
## 4 Decision tree model   0.989
## 5 Random forest model   0.99
```

From the table comparing the accuracy of the different models, we concluded that the most accurate model was random forest, with a nearly perfect accuracy of 0.990.

## Conclusion

In conclusion, our objective was to determine which spatial object (stars, galaxies or quasars) corresponded to an observation. We wanted to classify the observations between these spatial objects. To do so our model first randomly classified the observation, but as the accuracy was very low, we decided to use K nearest neighbors and Quadratic Discriminant Analysis. These methods gave us much better results, but still not the accuracy we were looking for. We finally decided to use decision trees and random forests, resulting in the targetted accuracy of 0.99.

Further to this project, it could be interesting to use similar models on larger datasets with more classes of spatial objects (supernovas, asteroids...) to see if the accuracy is still as high.