Stellar Observation Classification

Nicolas Kirsch

26/09/2019

Introduction

Telling which stellar object an observation corresponds to is a complicated task to do manually because an observation is composed of many different features (wavelength emissions,). Researchers therefore have to compare the observed values for each feature to the standard values of every stellar object. It is therefore interesting to create a machine learning algorithm which would automatically classify observations according to the stellar object studied.

To a smaller scale, we tried to create a machine learning algorithm able to classify observations between stars, galaxies and quasars. To do so, we used the *"Sloan Digital Sky Survey DR14"* dataset. This dataset is composed of 10,000 observation of space. Every observation is composed of 17 features columns and one class column. The class column specifies whether the observed object is 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" are specification on the tools used to make the observation. "ra" and "dec" are the astronomical coordinates of the object. "u", "g", "r", "i", "z" are the intensity of different wavelengths observed. They follow the Thuan-Gunn astronomic magnitude system. "mjd" is the date of the observation. "objID" is an identifier for the observation in the original databank. Finally, "redshift" is a physical quantity giving us insight on the distance separating the stellar object from earth.

We tried to use different machine learning algorithm to predict the class from the 17 features of each observation 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

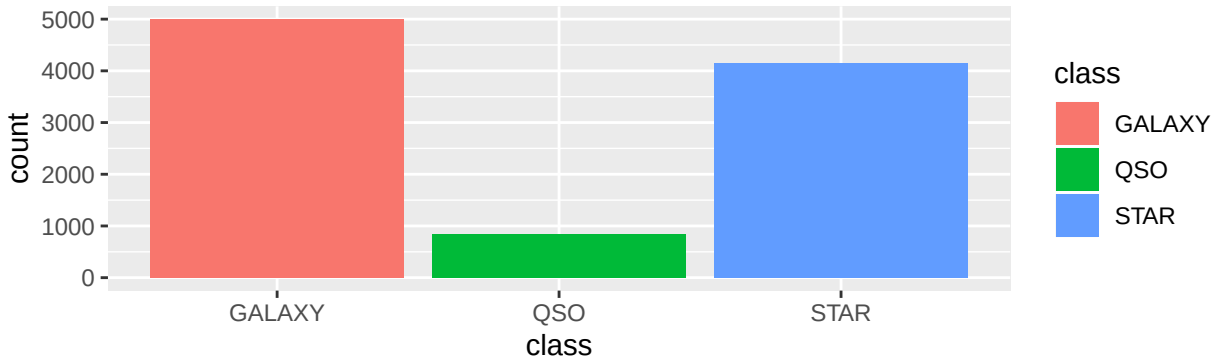
The first thing we wanted to know was whether the dataset was complete or if some values were missing. To do so we checked for NA values in every columns.

```
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 therefore were able to proceed to data analysis and visualization

We also wanted to find out the repartition of observation between the three classes.



The observations were mostly of galaxies and stars, with much less quasars observations.

A key part of our classification algorithm was the features which we were going to use to classify the observations. This corresponded to the columns of the dataset (except class which is what we try to predict).

```
colnames(sky)
```

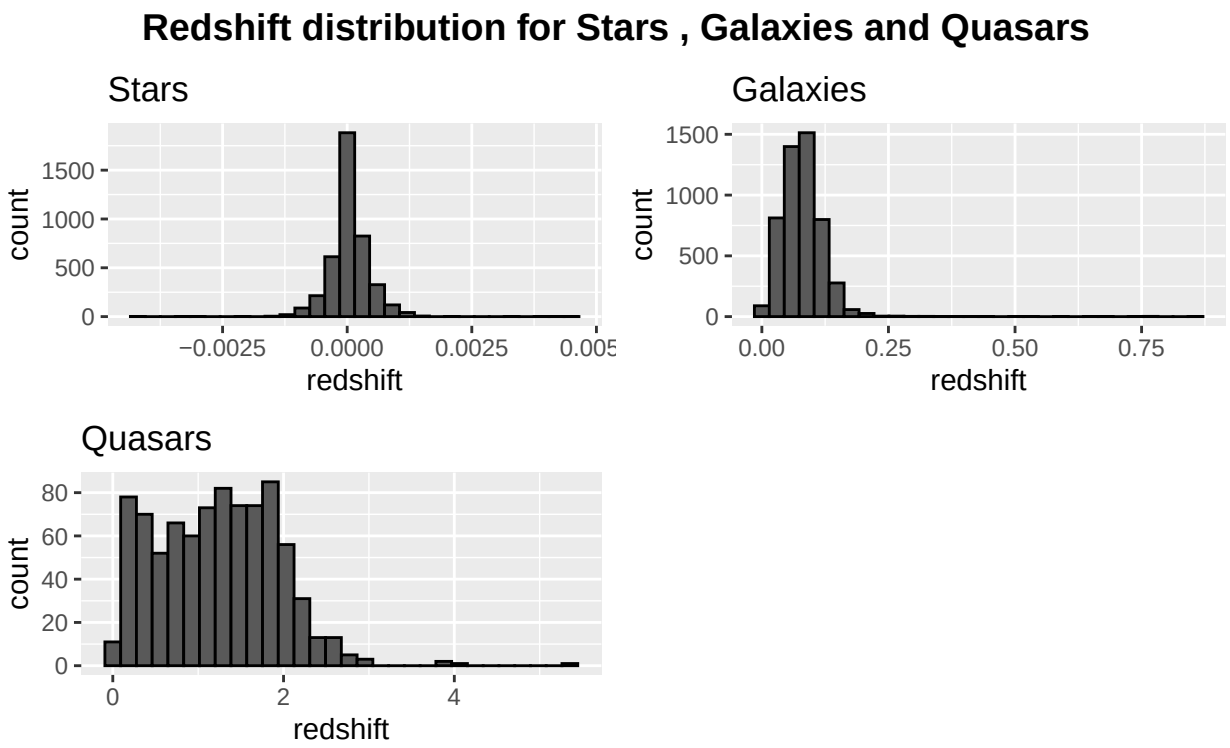
```
##  [1] "objid"     "ra"        "dec"       "u"         "g"
##  [6] "r"         "i"         "z"         "run"       "rerun"
## [11] "camcol"    "field"     "specobjid" "class"     "redshift"
## [16] "plate"     "mjd"       "fiberid"
```

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

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

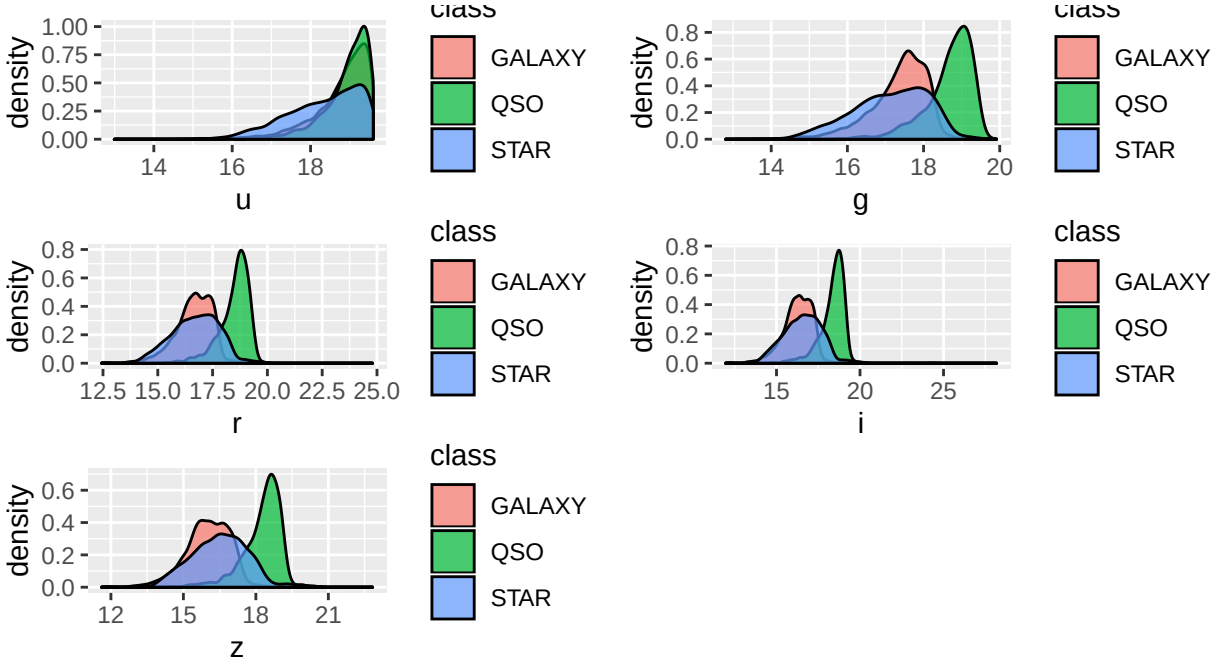
```
##         ra        dec        u        g        r        i        z  class
## 1 183.5313 0.08969303 19.47406 17.04240 15.94699 15.50342 15.22531   STAR
## 2 183.5984 0.13528503 18.66280 17.21449 16.67637 16.48922 16.39150   STAR
## 3 183.6802 0.12618509 19.38298 18.19169 17.47428 17.08732 16.80125 GALAXY
## 4 183.8705 0.04991069 17.76536 16.60272 16.16116 15.98233 15.90438   STAR
## 5 183.8833 0.10255675 17.55025 16.26342 16.43869 16.55492 16.61326   STAR
## 6 183.8472 0.17369416 19.43133 18.46779 18.16451 18.01475 18.04155   STAR
##       redshift plate   mjd fiberid
## 1 -0.000008960  3306 54922     491
## 2 -0.000054900   323 51615     541
## 3  0.123111200   287 52023     513
## 4 -0.000110616  3306 54922     510
## 5  0.000590357  3306 54922     512
## 6  0.000314603   324 51666     594
```

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



We saw that the magnitude of the redshift varied significantly from one class to the other (Stars around to 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 when classifying the observations between Stars, Galaxies and Quasars.

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 creat 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 is close to 1/3, which makes sense as there is three preditors. 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 trainset 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 of **0.803**

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

Simply using a KNN algorithm to classify the observations instead of randomly classifying them has a very strong impact on the accuracy as it increased 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 were able to create 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 is 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, "RandomForest". Afterwards, we followed the same protocol, training the model on the train set and then predicting the classes of the test set using it. We then 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 were able to say that the most accurate model was random forest, with a nearly perfect accuracy of 0.990.

Conclusion

In conclusion, our objective was to determinate which stellar object (stars, galaxies or quasars) corresponded to an observation. We wanted to classify the observations between these stellar 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 stellar objects (supernovas, asteroids...) to see if the accuracy is still as high.