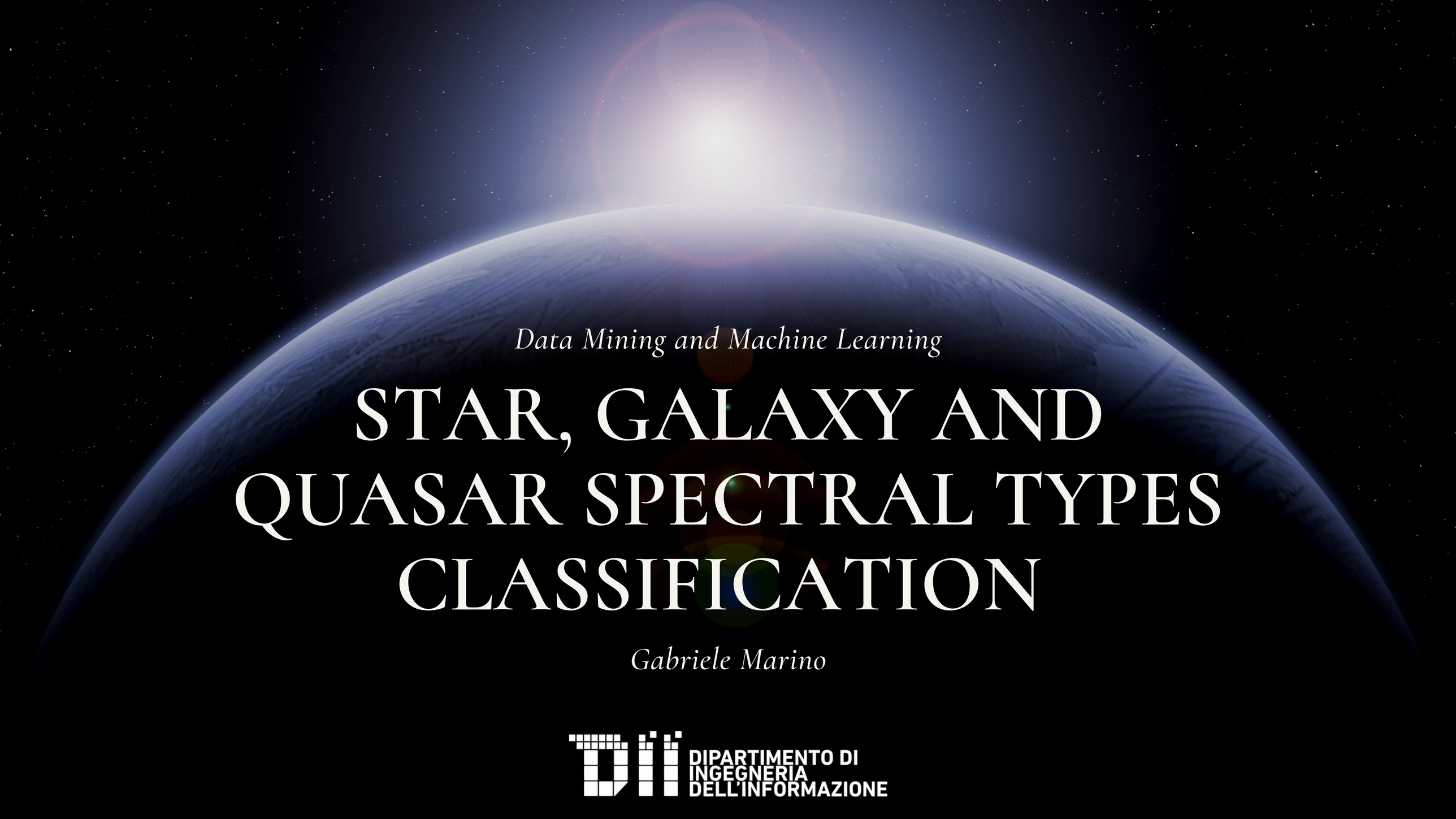


Data Mining and Machine Learning

STAR, GALAXY AND QUASAR SPECTRAL TYPES CLASSIFICATION

Gabriele Marino

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

— The Topic

Brief introduction

The classification of galaxies, quasars and stars is one of the most fundamental in astronomy. The first cataloging of stars and their distribution across the sky led to the understanding that they make up our galaxy.
This type of classification helped distinguish Andromeda from our own Milky Way.

Dataset

Context

The data consists of 100.000 observations of space taken by the SDSS (Sloan Digital Sky Survey). Every observation is described by 17 features and 1 class-labeled which identifies each object as a GALAXY, STAR or QSO.

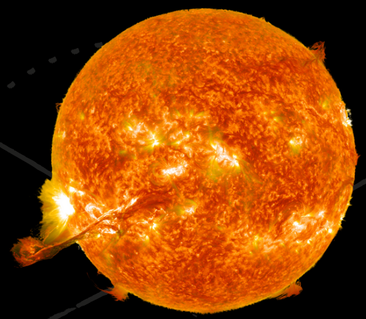
Attributes

- **objid** = Object Identifier, the unique value that identifies the object in the image catalog used by the CAS
- **ra** = Right Ascension angle (at J2000 epoch)
- **dec** = 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** = Run Number used to identify the specific scan
- **rereun** = Rerun Number to specify how the image was processed
- **camcol** = Camera column to identify the scanline within the run
- **field** = Field number to identify each field
- **specobjid** = Unique ID used for optical spectroscopic objects
- **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
- **fiberid** = fiber ID that identifies the fiber that pointed the light at the focal plane in each observation

Labeled Class

The labeled class has this kind of distribution among the three values:

Galaxies: 59%



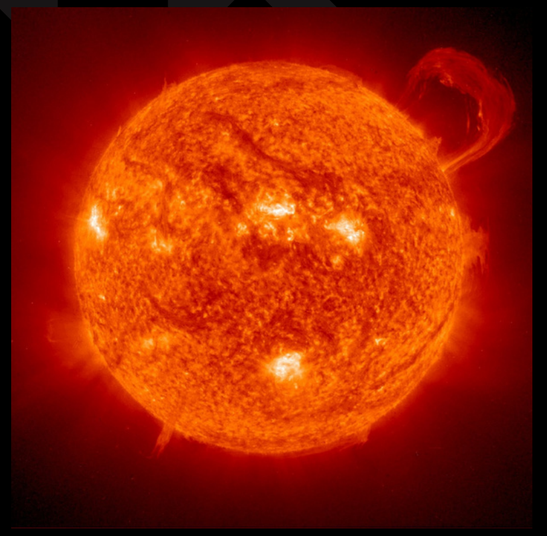
Stars: 22%

Quasars: 19%

I needed to balance the dataset

WorkFlow before Classification

- Initial Inspection
- Noise Correction
- Features Selection
- Rebalance



Initial inspection

01

There is no NaN or NULL value

02

There is no duplicate rows

Noise Correction

I noticed that there was some noise for the following attributes:

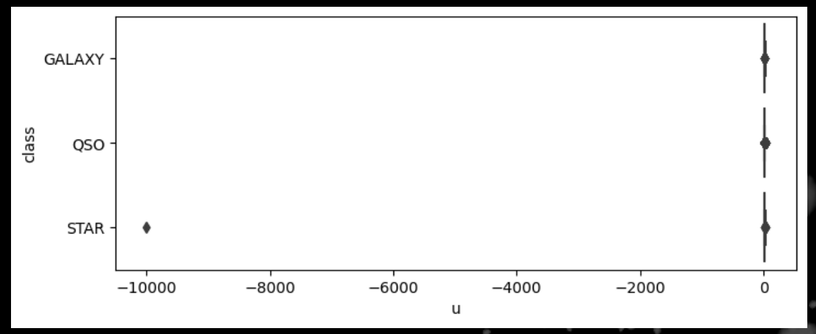
Considering that the noise is present for only one example, so I decided to delete it from the dataset.

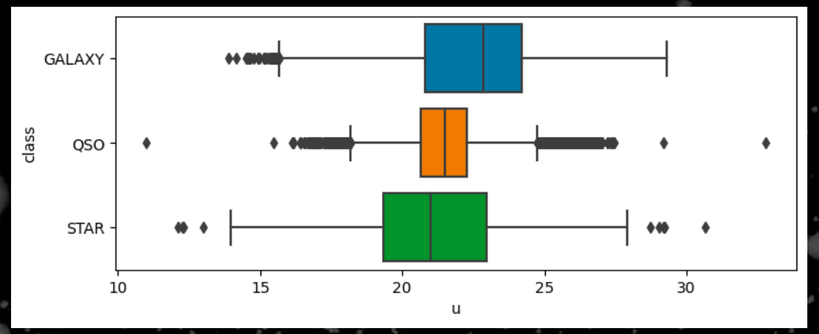
Noise Correction

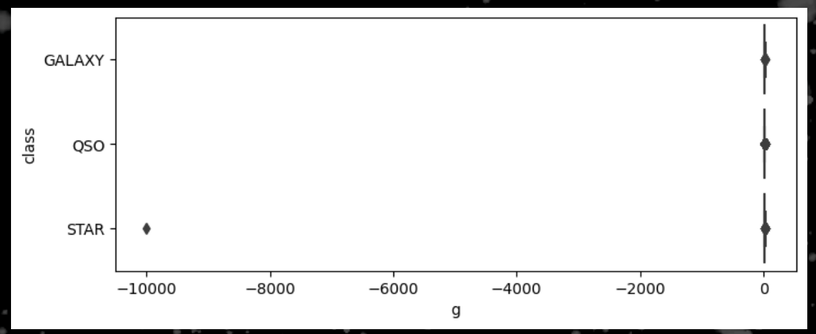
u

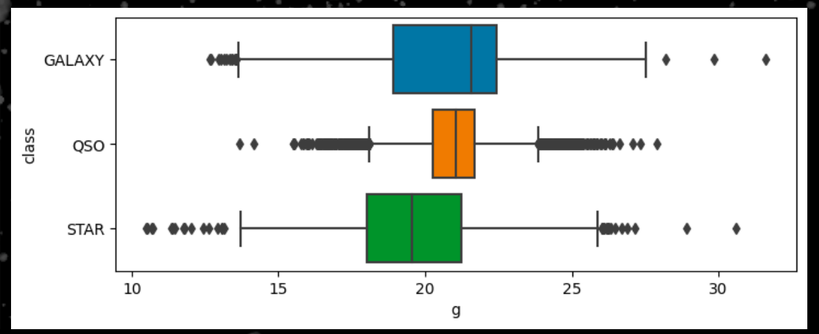
g

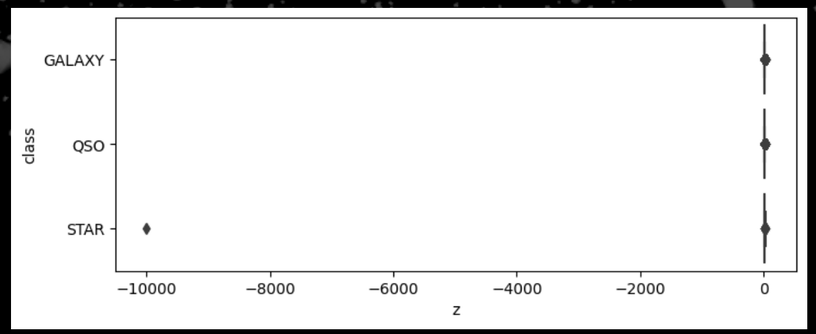
z

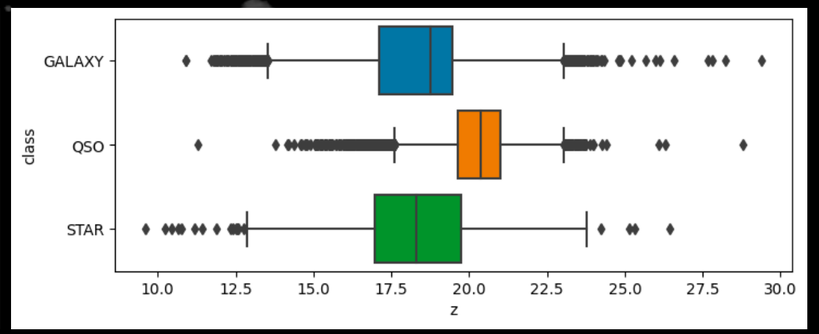












Before Noise Correction

After Noise Correction

Cross-Validation

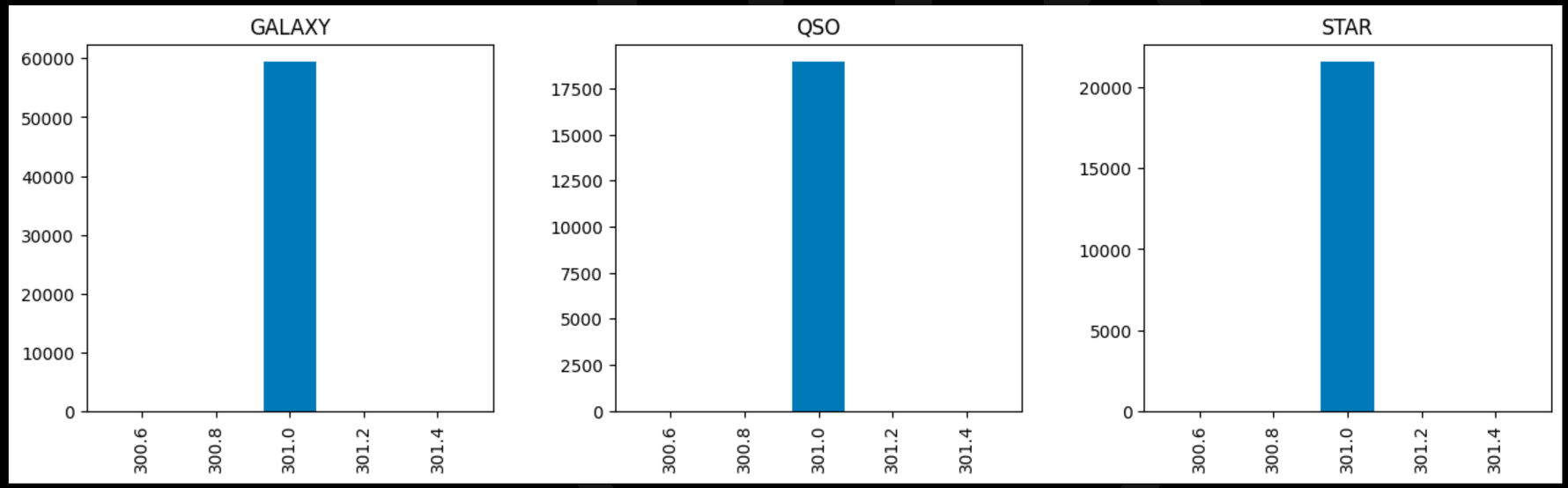
Each Classifier has been tested using the 10-fold cross-validation. For each iteration, I performed:

- Features Selection
- Classes Rebalance

Features Selection

Not all the features are really useful for this classification.
For instance the attribute ***rerun*** is constant for all the rows, whereas the attribute ***objid*** varies.
I decided to non use this two features for the classification, because are not rilevant.

rerun distribution:



Features Selection

I used the sklearn's class ***SelectKBest*** with ***mutual_info_classif*** as a score function, inside the cross-validation, and the mean ranking that I found is the follow:

I decided to use ***mutual_info_classif*** because of non-linear dependence of this features to the labeled-class

		Score
1	redshift	0.801910
2	specobjid	0.302147
3	plate	0.274762
4	mjd	0.193337
5	z	0.145171
6	run	0.143389
7	g	0.119731
8	i	0.109557

		Score
9	u	0.100488
10	r	0.075636
11	fiberid	0.048376
12	dec	0.044506
13	ra	0.040434
14	field	0.003996
15	camcol	0.003361

Features Selection

I chose the feature: *fiberid*, *dec*, *ra*, *field* and *camcol*, because they have a score closer to zero.

1. redshift
2. specobjid
3. plate
4. mjd
5. z
6. run
7. g
8. i
9. u
10. r

Rebalance

To balance all the labeled classes I used the ***SMOTE*** class from the ***imblearn.over_sampling*** library.
In this way I brought the two minority classes to the same size as the majority class.
I chose to use the over sampling methodology, because otherwise most of the data would have been lost.

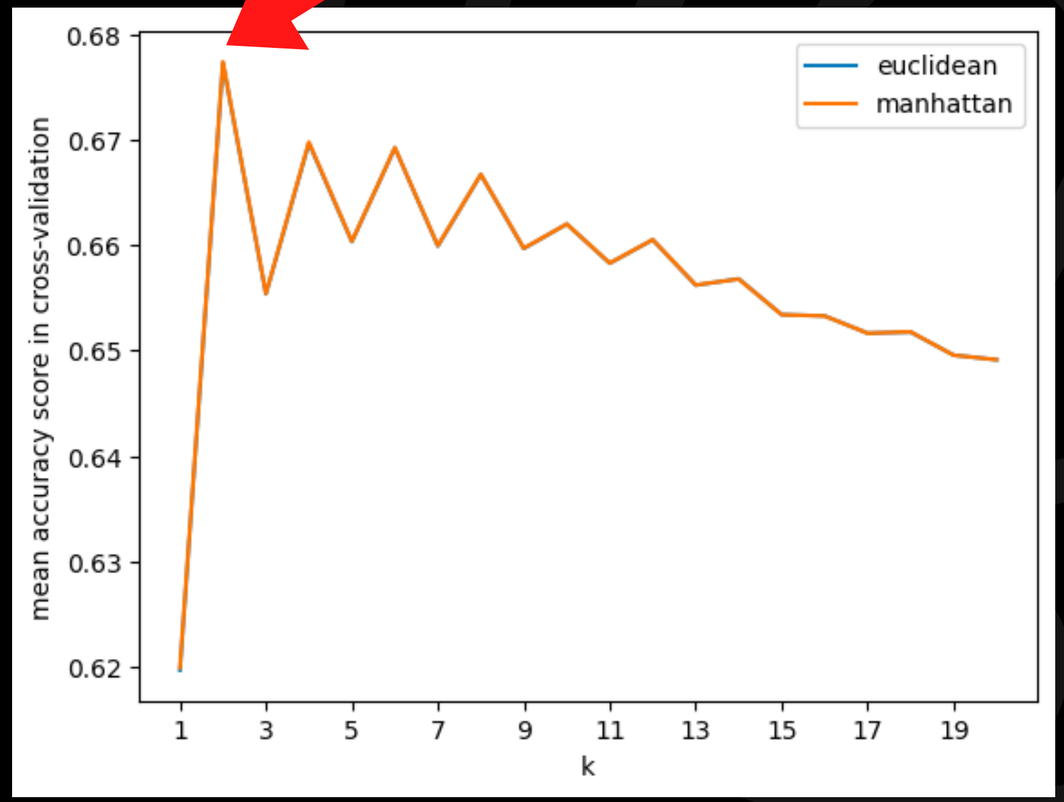
```
pipe = imbpipeline(steps = [
    ['kBest', SelectKBest(score_func=mutual_info_classif, k=10)],
    ['smote', SMOTE(random_state=42)],
    ['classifier', clf]]
)
```

Classification

K-Nearest Neighbors, Decision Tree, Random Forest, Bayesian Classifier

K-Nearest Neighbors

I tried to perform KNN with different k, for understand which is the best.
I decided to use the euclidean and the manhattan distance, for make a comparison



Euclidean Distance

- K = 2
- Accuracy = 0.67730

Manhattan Distance

- K = 2
- Accuracy = 0.67741

K-Nearest Neighbors

	GALAXY	STAR	QSO
GALAXY	49039	7501	2905
STAR	8714	8598	1649
QSO	7396	4159	10038

Classification Report:

	Precision	Recall	F1-Score
GALAXY	0.75	0.82	0.79
STAR	0.42	0.45	0.44
SQO	0.69	0.46	0.55

Cross-validation accuracy scores:

0.6827	0.6750
0.6731	0.6722
0.6811	0.6737
0.6751	0.6712
0.6808	0.68267

Mean Value: **68%**

Decision Tree

	GALAXY	STAR	QSO
GALAXY	56899	2437	109
STAR	1276	17682	3
QSO	98	3	21492

Classification Report:

	Precision	Recall	F1-Score
GALAXY	0.98	0.96	0.97
STAR	0.88	0.93	0.90
SQO	0.99	1.00	1.00

Cross-validation accuracy scores:

0.9625	0.9600
0.9568	0.9596
0.9594	0.9629
0.9610	0.9577
0.9621	0.9639

Mean Value: **96%**

Random Forest

	GALAXY	STAR	QSO
GALAXY	58258	1033	154
STAR	1043	17915	3
QSO	7	0	21586

Classification Report:

	Precision	Recall	F1-Score
GALAXY	0.98	0.98	0.98
STAR	0.95	0.94	0.95
SQO	0.99	1.00	1.00

Cross-validation accuracy scores:

0.9768	0.9780
0.9767	0.9762
0.9779	0.9766
0.9764	0.9769
0.9795	0.9802

Mean Value: **98%**

Bayesian Classifier

	GALAXY	STAR	QSO
GALAXY	2289	22923	34233
STAR	648	12978	5335
QSO	515	6216	14863

Classification Report:

	Precision	Recall	F1-Score
GALAXY	0.66	0.04	0.07
STAR	0.31	0.68	0.42
SQO	0.27	0.69	0.39

Cross-validation accuracy scores:

0.3031	0.2990
0.3026	0.3047
0.3015	0.2961
0.3009	0.3031
0.3066	0.2978

Mean Value: **30%**

Conclusion

The conclusion is that the best classifier for this problem is Random Forest, because they has the best metrics in terms of Precision, Recall, Accuracy and Cross-Validation scores. And if we looking at the ROC curve, Random Forest have the greatest value.

User Application

I made a simple user application. I exported the model via the **Pickle** library, which allows you to save the model in a file