Universiteit Leiden
The Netherlands

Faculty of Sciences

Databases and Data Mining in Astronomy

4303DBDMA

Final Project

Author:
Guadalupe Cañas Herrera

Student Number:
s1848151

January 26, 2018

Contents

1 About this project **2**

2 A database for a time-domain survey **3**
2.1 Introduction to the data . . . 3
2.2 Creation of the Database . . . 4
2.3 SQL queries . . . 9
2.3.1 R1 . . . 9
2.3.2 R2 . . . 11
2.3.3 R3 . . . 12
2.3.4 R4 . . . 14
2.3.5 R5 . . . 14
2.4 Estimation of random stars for Euclid mission . . . 17
2.4.1 Method 1: Kernel Density estimation using optimal bandwidth . . . 18
2.4.2 Method 2: Gaussian Mixture Model . . . 19
2.5 Conclusion . . . 23

3 Photometric redshifts of galaxies **24**
3.1 Introduction . . . 24
3.2 Feature extraction and data considerations . . . 25
3.3 Linear regression methods . . . 27
3.4 Non-linear regression methods . . . 29
3.4.1 K-Nearest Neighbours . . . 29
3.4.2 Random Forest . . . 32
3.4.3 Gaussian Processes . . . 34
3.5 Conclusion . . . 35

4 Code Documentation **37**
4.1 About . . . 37
4.2 Requisites . . . 37
4.3 Structure . . . 37

1 About this project

This project report represents the final stage of the course *Databases and Data Mining in Astronomy*, taught by Pr. Jarle Brinchmann, during the Fall Semester in 2017, at Leiden University. The project is based on the solutions found for this final assignment, where we need to cover several aspects of Databases and Data Mining. Mainly, it consists on two different parts: the first, related to the creation of a database and analysis of the data that it is contained in it, and the second, meant to study different Machine Learning Techniques to go through the analysis and regression processes of a dataset.

The project is divided into three different parts. The first part, in section 2, provides the main core of solutions related to the creation of a database for a time-domain survey, in which further analysis using SQL were performed and also a simple simulation of stars for Euclid mission. The second one, in section 3 of this report, shows several regression methods as well as feature extraction processes that were carried out in order to obtain a photometric redshift model as a function of magnitudes.

All the computational material (i.e. codes, graphical visualizations and database products) is publicly available and stored in `https://github.com/gcanasherrera/DDBDM2017_FP.git`, together with this report. Moreover, at the end of the report, in section 4, a basic code documentation mentioning some general characteristics of the codes can be found. For any problem derived from the use of them, the interested person can contact the author at the e-mail address `canasherrera@lorentz.leidenuniv.nl`. The use of git as a tool of controlling software versions was highly promoted during the whole course, so that I decided to make the code public available since the first moment (under other names apart from the chosen final one).

Among the most important results obtained during this project, it is worth remarking the database of the time-domain survey, as well as the estimation of the Kernel Density Distribution function of the stars used to simulate more for the Euclid mission. Furthermore, I highlight the regression process of the magnitudes to obtain the photometric redshift using Random Forest, which provided with an efficient way to approach the problem as well as the high speed of code execution and low training and generalization errors.

Along the writing process of this report, the author has paid attention to show, apart from the solutions to the problems addressed by the professor, other acquired techniques such as newly learned visualization techniques, regression methods, and use PYTHON PACKAGES.

From the personal point of view of the author, the project was challenging and highly educative, showing not only the improvement of computational and programming skills but also the kind of attitude that scientists should take regarding the analysis of real-life data: exploding personal resources in reading and searching for bibliographical information, getting initiative to discuss your outcomes with your colleagues, being transparent with your results and your achievements and, finally, keeping always a critical mind when analysing them.

2 A database for a time-domain survey

2.1 Introduction to the data

Time-domain surveys are those obtained in astronomy where astronomical objects are studied regarding how their properties change with time. Thus, instead of one-time long exposure of photographic plates that were registering one instance in the life of an object, series of quick and fast snapshots that can show the evolution with time of these objects are used [8].

Recently, time-domain astronomy will suffer a boost due to development of several projects (Zwicky Transient Factory, MASCARA or LSST). Therefore, it is important to plan in advance how the recorded data by these facilities would be allocated and organized. In this sense, databases play a fundamental role in how the data will be held and made accessible to the scientific community.

In order to get an insight of how this real work is performed, the problem asks to carry out the design of a database based on data obtained from The Vista Variables in the Vía Láctea Survey. The goal of the exercise is to create the database schema, ingest the data, and prove that it is efficient by means of extracting some required information back from it using queries.

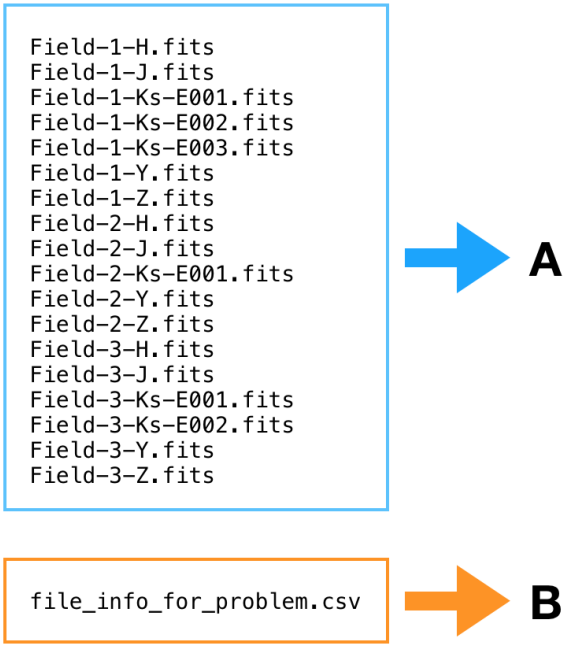


Figure 1: List of files provided for this problem. The classification into type A (.FITS files) or B (.CSV files) is meant to help the reader through the main text.

In principle, the survey, for which the database is created, is obtained by taking images of a patch of the sky at regular intervals. In our case, the patch of the sky is divided into three

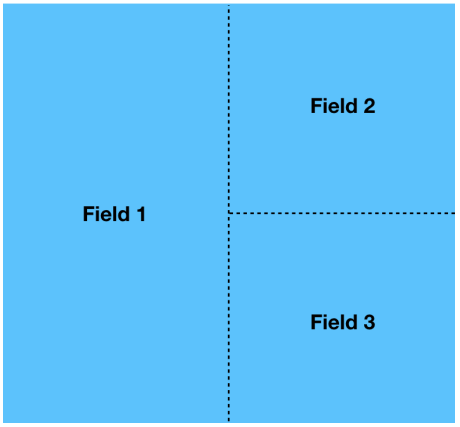


Figure 2: Schema of the sky patch divided in the 3 different fields of study. The size of each field is arbitrary and it is shown for indication only.

different fields (see figure 2). Every part of the survey area is imaged once using five filters (Z, Y, J, H, and Ks). After this, the survey continues to re-observe the same patch of sky in the Ks-band to attempt building up light-curves of stars. In spite of the fact that the number of observations of a particular sub-area varies from 50 to 300, we work with a reduction of this number of observations, that varies from 1 to 3. When an observation is done, regardless of the field, the astronomical objects are detected, classified and saved along with different calculated physical properties. The reader can observe a summary of these features and their description in table 1.

According to the problem, two different types of files containing all this information are provided, denominated by the author of this report, type A and type B (see figure 1). Firstly, the type A are 18 files (in FITS format, with the name, for instance, FIELD-1-Z.FITS) containing reduced images as well as a table with all physical properties described in table 1 for each filter. Secondly, the type B is an information table (in CSV format, called FILE_INFO_FOR_PROBLEM.CSV) containing a summary of the number of observations in each filter per field and their descriptions (see table 2).

2.2 Creation of the Database

Databases are created, handled, managed and maintained by database managers. They are expected to determine the best possible method of implementing it and organizing, accessing and storing the data. The type of database and data to deal with depends on the design of the experiment and how the data is measured.

The design of a database is not a straight-forward process. It is critical to know and understand the main purpose for which the database will be used for. Moreover, it is convenient that the database itself does not contain already processed or calculated extra data that can be obtained by each of the users to maintain the size of the database under control unless this information is highly demanded by the users.

Key	Description
RunningID	A counter for the objects detected in a given image.
Ra	The right ascension of the object in decimal degrees.
Dec	The declination of the object in decimal degrees.
X	The x location of the object in the image in pixels.
Y	The y location of the object in the image in pixels.
Flux1, Flux2, Flux3	The flux of the object in counts in apertures with 1", 2" and 3" diameter.
dFlux1, dFlux2, dFlux3	The uncertainty of flux in apertures with 1", 2" and 3" diameter.
Mag1, Mag2, Mag3	The calibrated magnitude of a star in the same apertures as flux.
dMag1, dMag2, dMag3	The uncertainty on magnitude.
StarID	An identifier of a particular star
Class	The type of object detected: -1 star, 0 noise, +1 non-stellar, -2 borderline stellar

Table 1: Summary of properties associated to the astronomical objects detected in the patch.

Key	Description
FieldID	Identifier of the field. It can be 1, 2 or 3.
Filter	Filter used for the image. It can be Z, Y, J, H and Ks.
MJD	Date in Julian Day when the picture was taken.
Airmass	path length for light from a source to pass through the atmosphere.
Exptime	Time of exposure.

Table 2: Summary of features described in the information table for each observation.

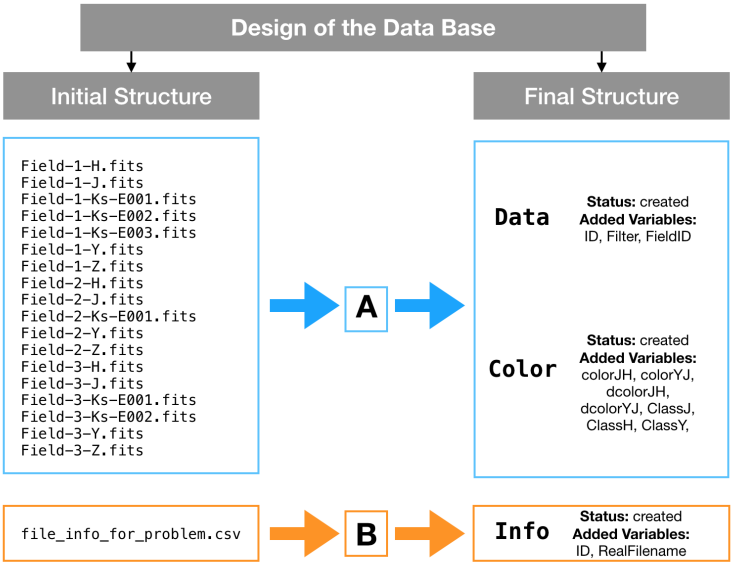


Figure 3: Graphical description of the design of the database along with the three attached tables.

ID	FieldID	Filter	MJD	Filename	RealFilename
1	1	Z	57267.167107	Z-ADP.2017-01-18T11:58:36.905.fits	Field-1-Z.fits
2	1	J	57257.050432	J-ADP.2017-01-18T11:58:35.781.fits	Field-1-J.fits
3	1	H	57257.044108	H-ADP.2017-01-18T11:58:35.780.fits	Field-1-H.fits
4	1	Ks	56788.346937	Ks-ADP.2016-05-25T15:33:39.546.fits	Field-1-Ks-E002.fits
5	1	Ks	56561.002016	Ks-ADP.2017-01-18T11:58:39.907.fits	Field-1-Ks-E001.fits
6	1	Ks	56829.039051	Ks-ADP.2016-05-25T15:33:43.377.fits	Field-1-Ks-E003.fits
7	1	Y	57267.159665	Y-ADP.2017-01-18T11:58:36.901.fits	Field-1-Y.fits
8	2	Z	57268.167107	Z-ADP.2017-01-18T11:58:36.905b.fits	Field-2-Z.fits
9	2	J	57258.050432	J-ADP.2017-01-18T11:58:35.781b.fits	Field-2-J.fits
10	2	H	57258.044108	H-ADP.2017-01-18T11:58:35.780b.fits	Field-2-H.fits
11	2	Ks	56789.346937	Ks-ADP.2016-05-25T15:33:39.546b.fits	Field-2-Ks-E001.fits
12	2	Y	57268.159665	Y-ADP.2017-01-18T11:58:36.901b.fits	Field-2-Y.fits
13	3	Z	57268.167107	Z-ADP.2017-01-18T11:58:36.905c.fits	Field-3-Z.fits
14	3	J	57258.050432	J-ADP.2017-01-18T11:58:35.781c.fits	Field-3-J.fits
15	3	H	57258.044108	H-ADP.2017-01-18T11:58:35.780c.fits	Field-3-H.fits
16	3	Ks	56789.346937	Ks-ADP.2016-05-25T15:33:39.546c.fits	Field-3-Ks-E002.fits
17	3	Ks	56562.002016	Ks-ADP.2017-01-18T11:58:39.907c.fits	Field-3-Ks-E001.fits
18	3	Y	57268.159665	Y-ADP.2017-01-18T11:58:36.901c.fits	Field-3-Y.fits

Table 3: Detail of the updated information table included in the database (name "Info"). The variables Airmass and Exptime have been omitted from this preview for convenience.

Length	RunningID	ID	FieldID	Filter	StarID
69999	12951	7	1	Y	9999
70000	225232	8	2	Z	170000
70001	225233	8	2	Z	170001

Table 4: Detail of the created data table included in the database (name "Data"). The variables associated to physical properties such as position, flux and magnitudes have been omitted from this preview for convenience as well as the total length of the table.

colorJH	colorYJ	StarID	FieldID	ClassH	ClassJ	ClassY
2.450781	-0.697231	0	1	-1	-2	1
2.190912	-0.245317	1	1	-1	1	1
1.946388	-0.305107	2	1	-1	-1	1

Table 5: Detail of the created colour table included in the database (name "Color"). The variables associated to the uncertainty of each color have been omitted from this preview for convenience as well as the total length of the table.

In this sense, the design of the database has to fulfil some specific features. The general listed requisites for the creation of the database are the following:

- Design needs to have a predictable layout to ease future automatically processing. This implies that some new files may be created.
- The database has to keep track of the location of the reduced images and the catalogues of detections in each image.
- The database needs to keep track of the colours of the stars and the variability of the stars in the Ks-band.
- The database needs to meet the scientific requirements of the team by means of being able to retrieve some queries such as:
  - R1: Find all images observed between MJD=56800 and MJD=57300 and give me the number of stars detected with S/N $>$ 5 in each image.
  - R2: Find the objects that have J-H $>$ 1.5.
  - R3: Find the objects where Ks differs by more than 20 times the flux uncertainty from the mean flux.
  - R4: Find all catalogues that exist for a given field.
  - R5: For a given field I would like to retrieve the Y, Z, J, H and Ks magnitudes for all stars with S/N $>$ 30 in Y, Z, J, H, and Ks.

The provided data, as well as the SQL queries that should be taken into account to create the database, contain some ambiguities, listed below, together with the decisions taken to deal with them:

- The queries do not specify how we should deal with fluxes taken at different apertures. For this reason, the database manager of this report has decided that colours and signal-to-noise values are calculated using just values taken with aperture 1". For future improvements of the database, an analysis of how to deal with the data provided at different apertures could be performed.
- The MJD and the date in the file-names in the information file (FILE_INFO_FOR_PROBLEM.CSV), do not correspond. Thus, I use the explicit MJD values and ignore the date in the file-name.
- The names of the .FITS files do not agree to the filenames of the table in the provided file FILE_INFO_FOR_PROBLEM.CSV. Thus, I update this table with the real filename columns for simplicity.
- The different epochs of Ks observations are ordered such that E001 is before E002. This is crucial so that we are able to obtain the relation between the filenames and the real filenames for those in the Ks-band.

- In every case, we find a query that needs to be carried out for a given field. I choose Field 1, as it is the one that has more catalogues (3 observations in Ks).
- Queries R3 and R5 are ambiguous and their interpretation would depend on the design of the database itself. This would be explained in detail in the next subsection.

It is important to point out that the design of a database may also rely on the skills of the database manager. In this case, the database manager is the author of this report, who has hands-on experience in handling large data sets using PYTHON PACKAGES NUMPY and PANDAS, but whose practical skills in SQL are reduced to those developed during the course *Databases and Data Mining in Astronomy*. As a consequence, the database manager has decided to take advantage of her knowledge to carry out almost all needed transformations in PYTHON in order to reduce the amount of SQL lines needed to complete future SQL queries.

```
Info_schema = """
CREATE TABLE Info (ID INT, FieldID INT,
Filename VARCHAR(50), Filter VARCHAR(5), MJD DOUBLE,
Airmass DOUBLE, Exptime DOUBLE, RealFilename VARCHAR(50),
UNIQUE(ID),
PRIMARY KEY(ID, MJD),
FOREIGN KEY(FieldID))
"""

Color_schema = """
CREATE TABLE Color (colorJH DOUBLE, colorYJ DOUBLE, dcolorJH DOUBLE,
dcolorYJ DOUBLE, StarID INT, FieldID INT, ClassH, INT,
ClassJ INT,
ClassY INT,
UNIQUE(StarID),
PRIMARY KEY(StarID),
FOREIGN KEY(FieldID))
"""

Data_schema = """
CREATE TABLE Data (RunningID INT, X DOUBLE,
Y DOUBLE, Flux1 DOUBLE, dFlux1 DOUBLE, Flux2 DOUBLE,
dFlux2 DOUBLE, Flux3 DOUBLE, dFlux3 DOUBLE, Ra DOUBLE,
Dec DOUBLE, Class INT, Mag1 DOUBLE, dMag1 DOUBLE, Mag2 DOUBLE,
dMag2 DOUBLE, Mag3 DOUBLE, dMag3 DOUBLE, StarID INT,
ID INT, Filter VARCHAR(5), FieldID DOUBLE,
PRIMARY KEY(StarID),
FOREIGN KEY(FieldID, ID))
"""
```

Table 6: SQL Schemas of the three created tables in the database: INFO, COLOR, DATA.

Due to all previous reasons, the database is designed in the following way (see figure 3). First, the information table is updated including the real filename of the .FITS files for simplicity and every catalogue receives a unique ID that runs from 1 to 18 to point to only one catalogue. This updated information table receives the name of INFO can be seen in table 3. Second, the 18 .FITS files that contain the features described in table 1 will be compressed into one big table denominated DATA. Besides, some other variables are added to this table for better understanding and classification: the unique ID that is also contained in INFO, the FieldID, and the Filter. This new table DATA, which contains 180000 entries, can be partially seen in table 4 (only the new added variables). Finally, a new table is created meant to save the colors J-H and Y-J, which are asked in one SQL query as well as in the last part of this problem. This table, denominated COLOR and seen partially in table 5, contains, apart from these two variables, the uncertainties associated to the colours and the Class associated to each object in all catalogues with filter J, H and Y (because not all objects are classified equally in all bands). The SQL schemas used to create these tables and to be included in the database can be found in table 6.

The decision of creating only big table, DATA, containing all the physical features of the objects for all 18 catalogues will make easier carry out the queries in SQL, as it will avoid joining all the tables in some queries in order to extract the required information in general. Moreover, a table of size (180000, 22) is still within the SQL database standards, and the possibility of automatizing including future tables into the database is as simple as just attaching them to this big DATA table.

The database, which receives the name of DATABASE_S1848151.DB, has been created using PYTHON. For more information about this, the reader is kindly referred to section 4, where the corresponding IPYTHON NOTEBOOK is found.

2.3 SQL queries

The created database has been designed and the data provided for the problem has been ingested. In order to test the database, I tested 5 SQL queries. When the results are more than 20 entries, I have shown graphical representations.[1]

2.3.1 R1

The query R1 asks to find all images observed between MJD=56800 and MJD=57300 together with the number of stars detected with S/N $>$ 5 in each image. In this case, the S/N is calculated as,

$$S/N = Flux1/dFlux1 \tag{1}$$

[1]The author of this report has decided to use the SEABORN color palette for *color blind* people. The diagrams have been shown to a person with this sight problem to test the adequacy of this palette. The result was satisfactory. [7]

The final results can be obtained by means of performing the below SQL query:

```
"""SELECT filename, realfilename, mjd, numstars FROM
(SELECT inf.Filename as filename, inf.Realfilename as realfilename,
inf.MJD mjd, COUNT(dat.StarID) as numstars
FROM Info as inf, Data as dat
WHERE dat.ID == inf.ID
AND (dat.Flux1/dat.dFlux1) > 5
AND dat.Class=-1
GROUP BY inf.Filename)
WHERE mjd>56800 AND mjd<57300
ORDER BY mjd"""
```

The results of this query, being 13 different files, are the following ones:

```
1. FileName: Ks-ADP.2016-05-25T15:33:43.377.fits.
RealFileName: Field-1-Ks-E003.fits. MJD: 56829.0390512. Stars: 7888.
2. FileName: H-ADP.2017-01-18T11:58:35.780.fits.
RealFileName: Field-1-H.fits. MJD: 57257.044108. Stars: 7982.
3. FileName: J-ADP.2017-01-18T11:58:35.781.fits.
RealFileName: Field-1-J.fits. MJD: 57257.0504323. Stars: 7022.
4. FileName: H-ADP.2017-01-18T11:58:35.780b.fits.
RealFileName: Field-2-H.fits. MJD: 57258.044108. Stars: 7725.
5. FileName: H-ADP.2017-01-18T11:58:35.780c.fits.
RealFileName: Field-3-H.fits. MJD: 57258.044108. Stars: 8022.
6. FileName: J-ADP.2017-01-18T11:58:35.781b.fits.
RealFileName: Field-2-J.fits. MJD: 57258.0504323. Stars: 7354.
7. FileName: J-ADP.2017-01-18T11:58:35.781c.fits.
RealFileName: Field-3-J.fits. MJD: 57258.0504323. Stars: 7248.
8. FileName: Y-ADP.2017-01-18T11:58:36.901.fits.
RealFileName: Field-1-Y.fits. MJD: 57267.1596647. Stars: 6806.
9. FileName: Z-ADP.2017-01-18T11:58:36.905.fits.
RealFileName: Field-1-Z.fits. MJD: 57267.1671072. Stars: 6477.
10. FileName: Y-ADP.2017-01-18T11:58:36.901b.fits.
RealFileName: Field-2-Y.fits. MJD: 57268.1596647. Stars: 7215.
11. FileName: Y-ADP.2017-01-18T11:58:36.901c.fits.
RealFileName: Field-3-Y.fits. MJD: 57268.1596647. Stars: 7186.
12. FileName: Z-ADP.2017-01-18T11:58:36.905b.fits.
RealFileName: Field-2-Z.fits. MJD: 57268.1671072. Stars: 6929.
13. FileName: Z-ADP.2017-01-18T11:58:36.905c.fits.
RealFileName: Field-3-Z.fits. MJD: 57268.1671072. Stars: 6741.
Total number of stars: 94595
```

2.3.2 R2

The query R2 simply asks to find the objects that have J-H > 1.5. May the reader remember that this query was one of the reasons why the table COLOR in the database was created. The colour J-H was calculated:

$$J - H = (Mag1)_J - (Mag1)_H \tag{2}$$

where $(Mag1)$ is the magnitude for aperture 1" in both filters J and H. Furthermore, I calculated the uncertainties associated to these colours propagating errors according to [12]. Assuming that the errors associated to the magnitudes are randomly distributed and independent, the error associated to the colours are,

$$\Delta_{colour} = Colon\sqrt{\left(\frac{\Delta Mag1_{filter1}}{Mag1_{filter1}}\right)^2 + \left(\frac{\Delta Mag1_{filter2}}{Mag1_{filter2}}\right)^2} \tag{3}$$

where filter 1 and 2 may refer to filter J or H. These errors have resulted to be small, and do not provide much information.

The SQL query used to extract this information can be seen below.

```
""" SELECT Color.ColorJH, Color.FieldID FROM Color
WHERE Color.ColorJH >1.5"""
```

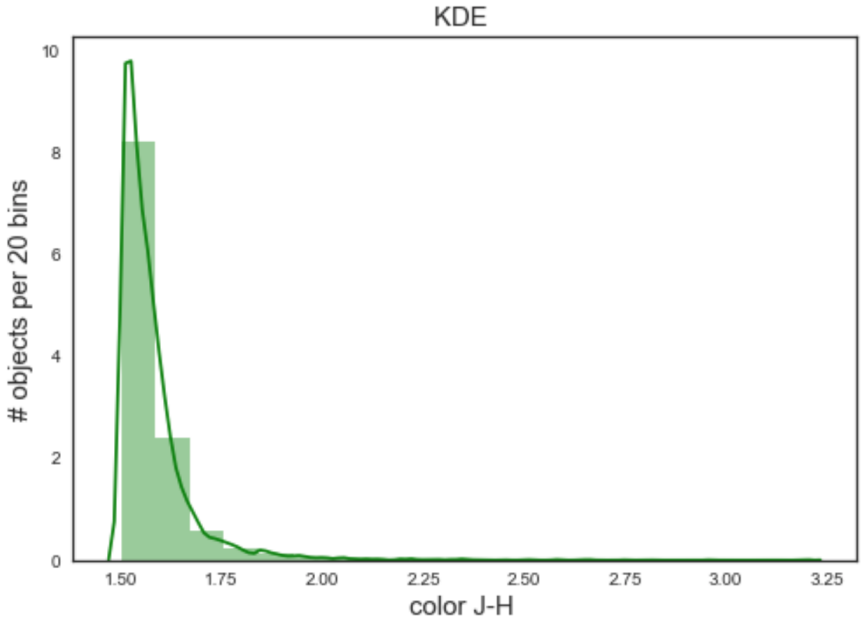


Figure 4: Histogram and KDE of the results obtained for query R2. Graphical representation of the number of objects as a function of the color J-H for 20 bins.

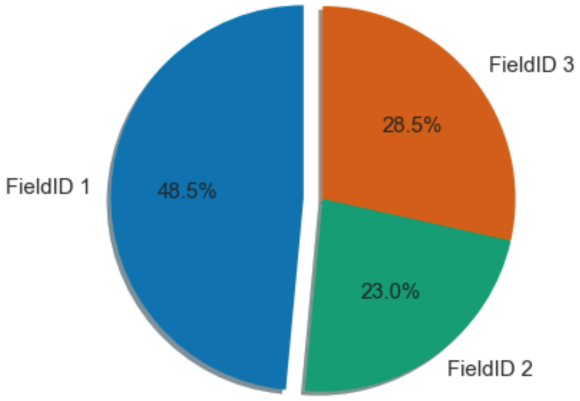


Figure 5: Pie chart of the results obtained for query R2. Graphical representation of the number of objects per Field.

The total number of objects that fulfils color J-H $\geq$ 1.5 is 7154, so that the results are more than 20 entries. Therefore, I decide to use a histogram and a pie plot in order to present the results. In figure 4, the number of objects detected as a function of the color J-H is plotted. We observe how the peak is around 1.6 in color.

In figure 5, the number of objects detected as a function of the FieldID is plotted. We observe that the vast majority of the objects that fulfil this condition correspond to FieldID 1.

2.3.3 R3

This SQL query asks to find the objects where Ks differs by more than 20 times the flux uncertainty from the mean flux. As mentioned before, the chosen flux is the one taken with aperture 1". The interpretation I have followed for this query is related to the fact that the database should keep track of the variability of stars in the Ks band. For this reason, I have decided to take the mean values of the Flux1 for the same star in different Ks observations in each Field, although I have not specify that the Class of the object has to be -1 (stars). This can be performed with the following SQL query:

```
"""SELECT␣total.StarID␣FROM␣Data␣as␣total␣WHERE
total.Filter="Ks"AND␣total.FieldID␣==␣1
AND␣ABS(␣(SELECT␣AVG(sub.Flux1)
FROM␣Data␣as␣sub␣WHERE␣sub.Filter="Ks"
AND␣sub.StarID␣==␣total.StarID␣GROUP␣BY␣sub.StarID))
>␣␣ABS(20*total.dFlux1)
GROUP␣BY␣total.StarID
ORDER␣BY␣total.StarID"""
```

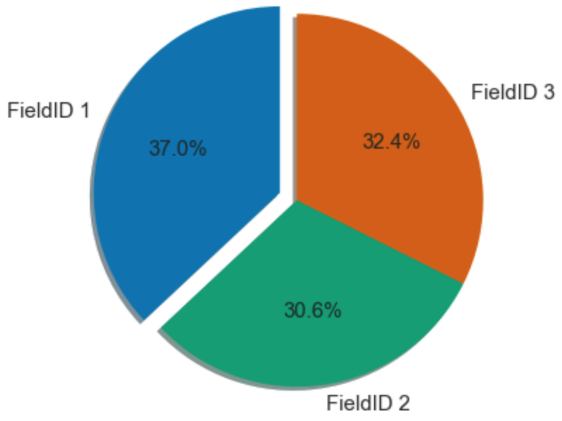


Figure 6: Pie chart of the results obtained for query R3. Graphical representation of the number of objects per Field.

May the reader realize that this interpretation asks to computation the average between Ks observations per field, although in Field 2 we only have one Ks observation. The number of objects found in FieldID 1, 2 and 3 are respectively 7884, 6529 and 6908. The percentages are represented in figure 6, and the relation with the corresponding Flux1 can be seen in figure 7. The output of this query is inconclusive. As the query itself is ambiguous, I am not in the position of exposing final conclusions about it. It is true that this query, in principle, should be related to the ability of the database to follow variable stars. Nevertheless, three points (as maximum, in Field 1), it is not enough to make a follow of this property.

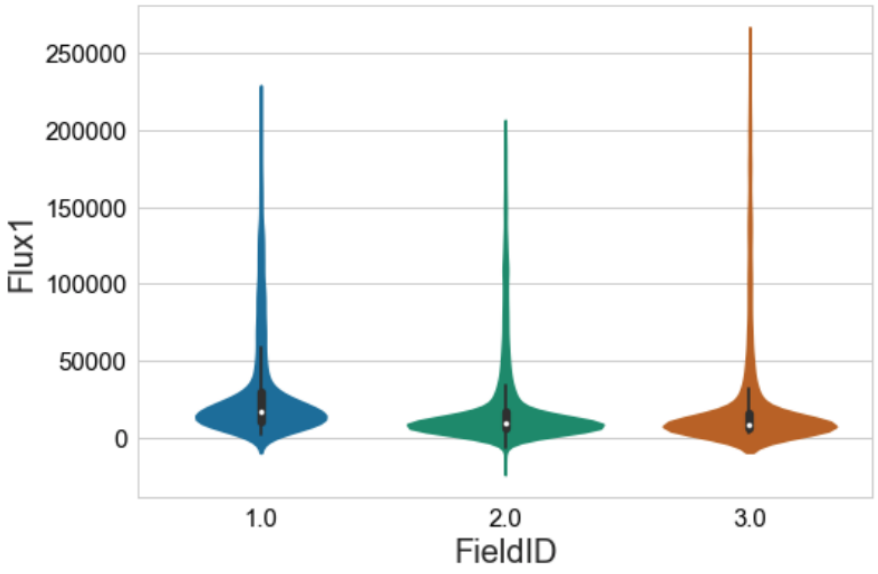


Figure 7: Violin representation of the results obtained for query R3. Graphical representation of the distribution of Flux1 per Field.

2.3.4 R4

The SQL query R4 asks to find all catalogues that exist for a given field. As mentioned before, when the query demands in particular to obtain results for a given field, FieldID 1 is chosen (although the rest of the fields were tested in the code as well). The results can be obtained from the database using the SQL query below:

```
""" SELECT fieldident , filename , realfilename , ident FROM
(SELECT inf.Filename as filename , inf.Realfilename as
realfilename , inf.FieldID as fieldident , inf.ID as ident
FROM Info as inf)
WHERE fieldident == 1
"""
```

The results consist of 7 different entries, observed below. I have printed the real filename that corresponds to the initially provided .FITS tables as well as the name that was part of the INFO table.

```
Field: 1
1. Filename: Z-ADP.2017-01-18T11:58:36.905.fits.
Realfilename: Field-1-Z.fits. ID: 1
2. Filename: J-ADP.2017-01-18T11:58:35.781.fits.
Realfilename: Field-1-J.fits. ID: 2
3. Filename: H-ADP.2017-01-18T11:58:35.780.fits.
Realfilename: Field-1-H.fits. ID: 3
4. Filename: Ks-ADP.2016-05-25T15:33:39.546.fits.
Realfilename: Field-1-Ks-E002.fits. ID: 4
5. Filename: Ks-ADP.2017-01-18T11:58:39.907.fits.
Realfilename: Field-1-Ks-E001.fits. ID: 5
6. Filename: Ks-ADP.2016-05-25T15:33:43.377.fits.
Realfilename: Field-1-Ks-E003.fits. ID: 6
7. Filename: Y-ADP.2017-01-18T11:58:36.901.fits.
Realfilename: Field-1-Y.fits. ID: 7
```

2.3.5 R5

The SQL query R5 asks to retrieve the Y, Z, J, H and Ks magnitudes for all stars with S/N $>$ 30 in Y, Z, J, H and Ks for a given field. This query shows some ambiguity. On the one hand, it may ask to obtain the stars that fulfil the S/N condition in each Y, Z, J, H and Ks. Or, on the other hand, it can be interpreted to retrieve the stars that fulfil the S/N condition in all filters at the same time. In this case, I have decided to carry out the query as if it were for each filter, not necessary in all at the same time. Once again, the FieldID chosen is 1, the S/N is calculated using equation (1) and the magnitude corresponds to the one with aperture 1".

The query to obtain the number of stars in each filter whose S/N>30 is the following:

```
"""SELECT inf.ID as ident, inf.FieldID as fieldident,
inf.Filter as filterident, dat.Mag1 as magnitude1,
dat.StarID as starident
FROM Info as inf, Data as dat
WHERE inf.FieldID == 1 AND
dat.Class=-1 AND
inf.ID == dat.ID
AND (dat.Flux1/dat.dFlux1) > 30
ORDER BY ident
"""
```

With this query, I recover all magnitudes of stars in each filter in just one table. As the result is more than 20 entries, I would like also to retrieve the total number of stars per field without being filtered by S/N in order to compare. The SQL query is as follows:

```
"""SELECT inf.ID as ident, inf.FieldID as fieldident,
inf.Filter as filterident, dat.Mag1 as magnitude1,
dat.StarID as starident
FROM Info as inf, Data as dat
WHERE inf.FieldID == 1
AND inf.ID == dat.ID
AND dat.Class=-1
ORDER BY ident
"""
```

The total number of stars that fulfil color S/N > 30, summing up those in every filter, is 34529, whereas the total number of stars, again summed up in every filter, is 50229. The graphical representations to show the results are histograms. To proceed to these representations, I need to extract the magnitudes per filter. This process can be done by PYTHON or, again, SQL. Both procedures give exactly the same result, as it has been tested, and the sample SQL query can be checked in table 7. The histograms represent the number of stars that fulfil and do not fulfil the condition in S/N per filter.

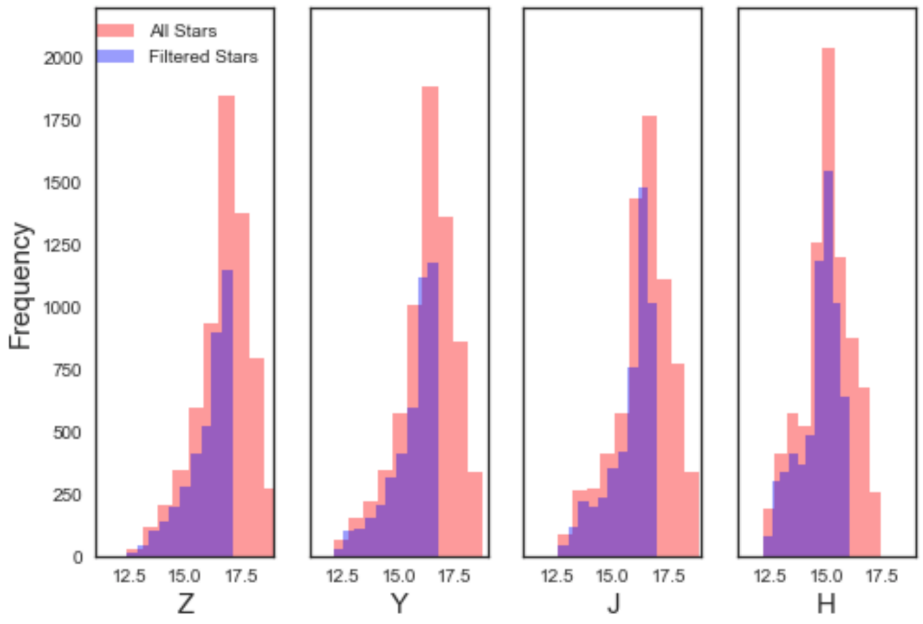


Figure 8: Histogram of the results obtained for query R5 for filters Z, Y, J and H In Field 1. Graphical representation of the number of objects as a function of the magnitude for different filters.

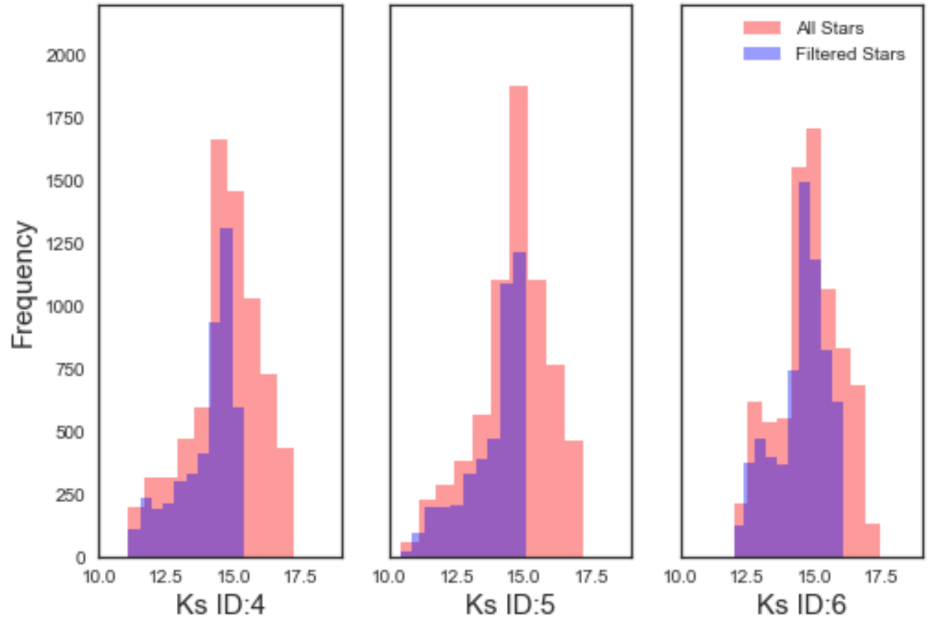


Figure 9: Histogram of the results obtained for query R5 for filters Ks in Field 1. Graphical representation of the number of objects as a function of the magnitude for different Ks.

```
"""SELECT inf.ID as ident, inf.FieldID as fieldident,
inf.Filter as filterident, dat.Mag1 as magnitude1,
dat.StarID as starident FROM Info as inf, Data as dat
WHERE inf.FieldID == 1 AND inf.Filter == "Z"
AND dat.Class=-1 AND inf.ID == dat.ID
AND (dat.Flux1/dat.dFlux1) > 30 ORDER BY ident"""
```

```
Number of stars per filter using SQL:
# stars that fulfil color S/N > 30 in Z: 3740
# stars that fulfil color S/N > 30 in Y: 4218
# stars that fulfil color S/N > 30 in J: 4829
# stars that fulfil color S/N > 30 in H: 6356
# stars that fulfil color S/N > 30 in Ks: 15386
# stars in Z: 6509
# stars in Y: 6810
# stars in J: 7022
# stars in H: 7982
# stars in Ks: 21906
```

Table 7: Top: sample SQL query to further filter stars that fulfil S/N > 30 for Filter Z. Bottom: number of stars that fulfil and do not fulfil the condition in S/N per Filter.

2.4 Estimation of random stars for Euclid mission

Euclid, an ESA astrophysics space mission, aims to understand why the Universe is suffering from an accelerated expansion and what is the nature of the source responsible for this acceleration which physicists refer to as dark energy, which represents around 75% of the energy content of the Universe today.

Euclid will be equipped with a 1.2 m diameter Silicon Carbide (SiC) mirror feeding 2 instruments, VIS and NISP. VIS will be a high quality panoramic visible imager and NISP, a near infrared 3-filter (Y, J, and H) photometer. With these instruments, physicists will probe the expansion history of the Universe and the evolution of cosmic structures by measuring the modification of shapes of galaxies induced by gravitational lensing effects. [3]

The database I have created gives the possibility of simulating 100000 stars for Euclid Consortium, as we have already had a sample of stars in the colours Y-J and J-H, which will be the ones Euclid will be operating in.

First, we need to get the distribution of the original stars in Y-J, J-H space from our database. This is straight-forward due to the fact that this simulation was one of the reasons why the table Color in the database was created. I extract those colours from the database, and then I select the objects that are classified as stars in all catalogues for which the colours where calculated. The reason why I restrict myself to objects that were classified as stars in catalogues

with Filters Y, J and H is to be sure that these objects have a larger possibility of being, in fact, stars. This criterion would reduce systematic errors in further steps of the simulation. The SQL query used to extract the colours Y-J, J-H of stars from the database is the following:

```
""" SELECT colorJH , colorYJ FROM Color
WHERE Color.ClassH==-1.0
AND Color.ClassJ==-1.0
AND Color.ClassY==-1.0
AND colorJH IS NOT NULL
AND colorYJ IS NOT NULL"""
```

In this SQL query, I have specified that I am not interested in retrieving values of colours that have a "NotANumber" value (there exist a couple of these values in the catalogues). The total amount of original stars to work with is 18428, and can be observed in figure 10.

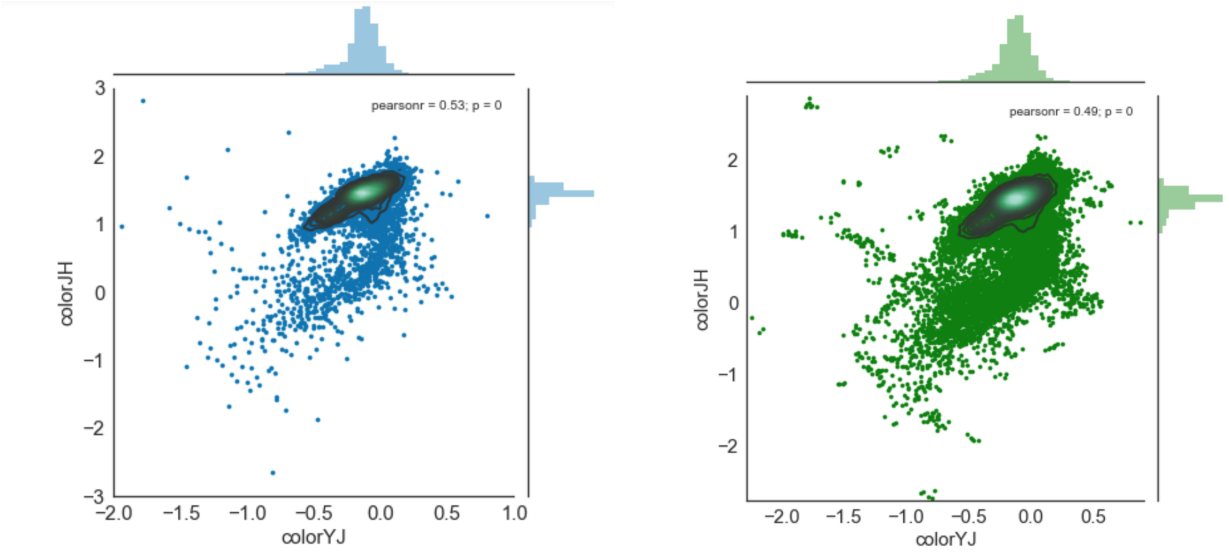


Figure 10: Graphical representation of the original distribution of stars (left) and simulated stars (right) as a function of the colours Y-J and J-H, together with the KDE distribution from seaborn.

From the original distribution of stars, we can simulate X number of stars following the two methods learned during the course, and explored below:

2.4.1 Method 1: Kernel Density estimation using optimal bandwidth

It is possible to calculate the Kernel Density distribution function using a Gaussian Model using the PYTHON PACKAGE SKLEARN KDE. For that, we need to provide with the optimal bandwidth. I have obtained the optimal bandwidth for this distribution using mainly two

Method	Value for bandwidth
Cross-validation	0.044
Mean-shift algorithm	0.046

Table 8: Results for the bandwidth using the two different methods.

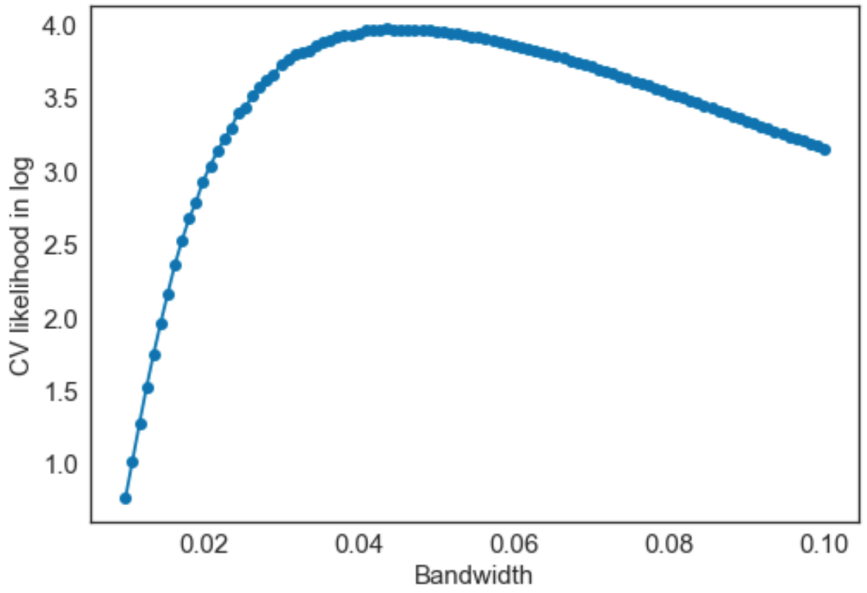


Figure 11: Graphical representation of likelihood in logarithmic scale as a function of the bandwidth for the cross-validation method.

different methods: **cross-validation** (provided by the lecturer Jarle Brinchmann, and observed in figure 11) and **mean-shift algorithm** (provided at SKLEARN). The results of these two methods are shown in table 8. The discrepancy between the two values is approximately 5%.

Using the bandwidth of the cross-validation method, the KDE distribution function with a Gaussian Model is created. The representation of such as distribution function can be seen in figure 12, together with the original set of stars (left, blue) as well as the 100000 simulated stars obtained from the method SAMPLE of KDE (right, green). Moreover, it is possible to observe as well how the simulated stars distribution compare to the original set of stars in figure 13. In general, both distributions follow the same tendency except for the tails, which correspond likely to those stars whose colours do not sit within the more dense part of the KDE.

2.4.2 Method 2: Gaussian Mixture Model

Another possibility of simulating 100000 stars is to fit the 2D colour distribution of the set of original stars by means of overlapping Gaussian distributions with different mean and standard deviation. This can be done using SKLEARN MIXTURE method, imposing a Gaussian model.

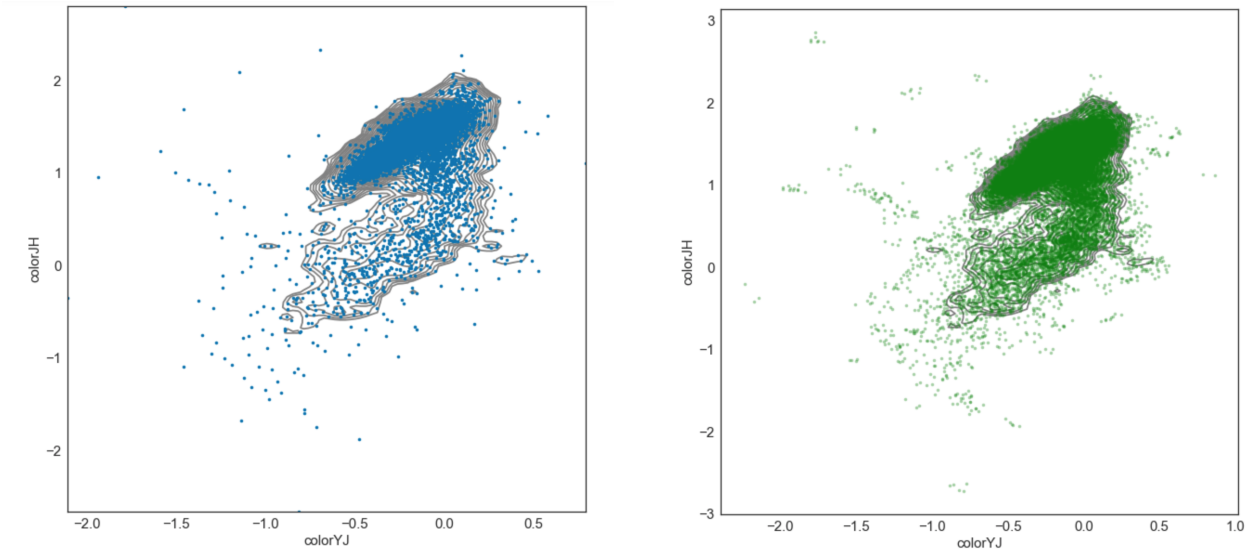


Figure 12: Graphical representation of the original distribution of stars (left) and simulated stars (right) as a function of the colours Y-J and J-H, together with the KDE distribution function obtained using the optimal bandwidth for the Gaussian Model.

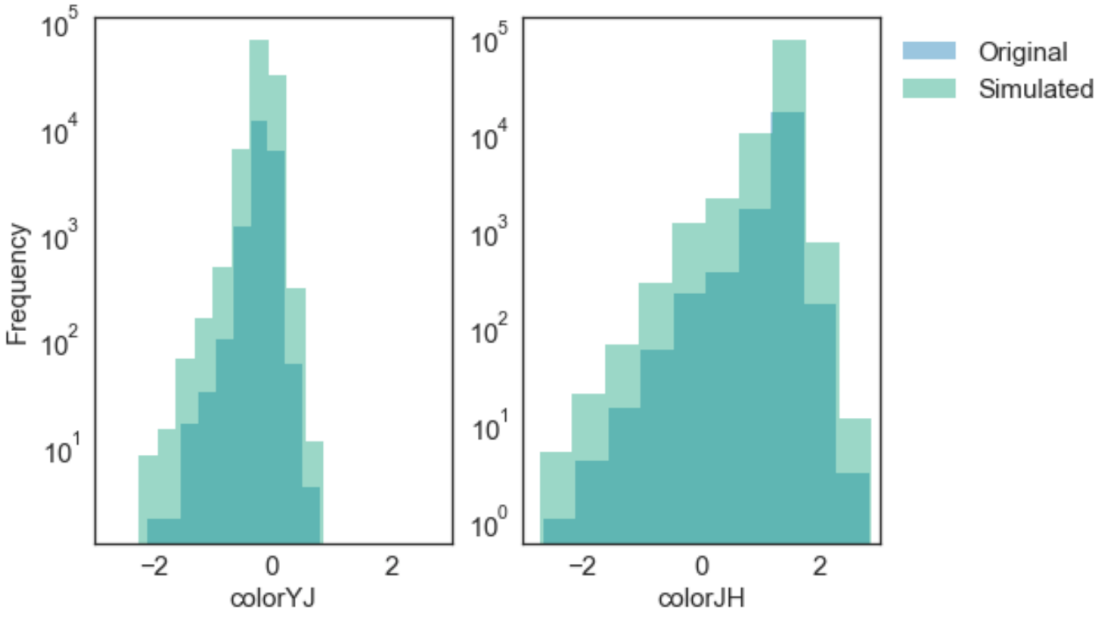


Figure 13: Graphical representation of the number of original and simulated stars as a function of colours Y-J and J-H in logarithmic scale.

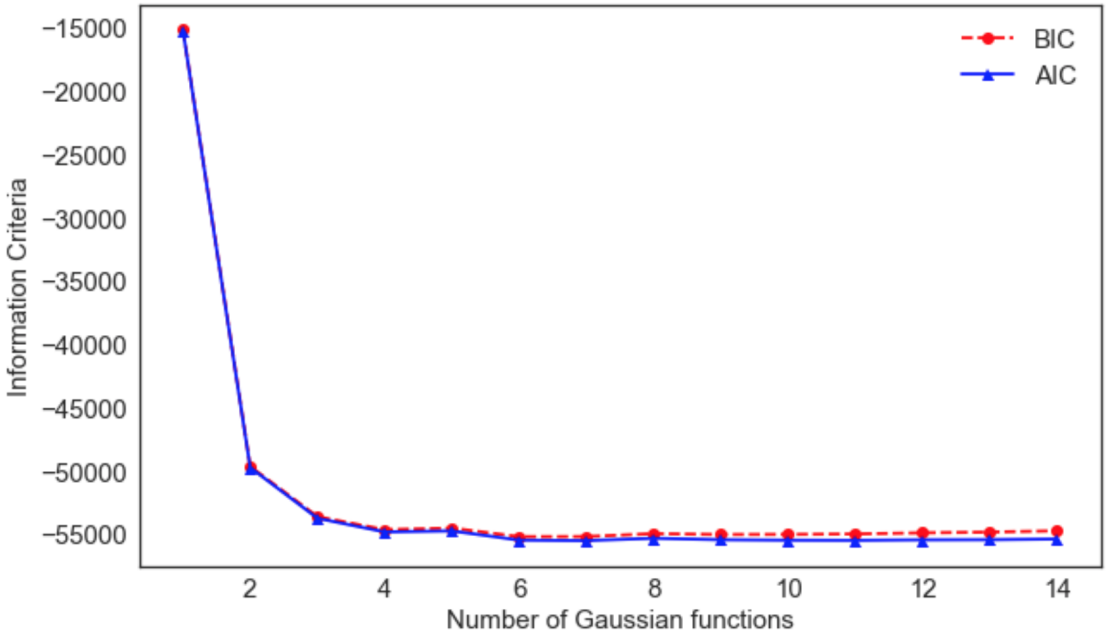


Figure 14: Graphical representation of different Information Criteria (BIC in red with circles and AIC in blue with triangles) as a function of the number of Gaussian functions.

To estimate the number of Gaussian functions required to fit the original distribution of stars as a function of the two different colours, I use two Information Criteria: BIC and AIC. The value of these Information Criteria as a function of the number of Gaussian functions needed to perform the mixture model can be found in figure 14. In this figure, it is observed how these two criteria suffer a decrease from numbers 1 to 3. The AIC remains constant for the rest of values, whereas the BIC tends to increase starting around the number 7. This is due to the fact that the BIC suffers a penalization because of the number of components of the model so that it usually indicates overfitting. Due to this reason, I have decided to use 6 as the number of Gaussian functions.

This value was also estimated with the method BayesianGaussianMixture, that calculates itself by Bayesian inference the suitable number of Gaussian components. In this case, it resulted to be 13. The difference between the information provided by BIC and AIC, and the result obtained from BayesianGaussianMixture, probably relies on the fact that BayesianGaussianMixture also uses Gaussian functions to fit those stars that sit on the tails of the distributions. A simulation using this method was also performed, but the visual analysis of the results do not considerably differ. For this reason, this visualizations have not been included in this report, although it can be seen in the corresponding IPython Notebook.

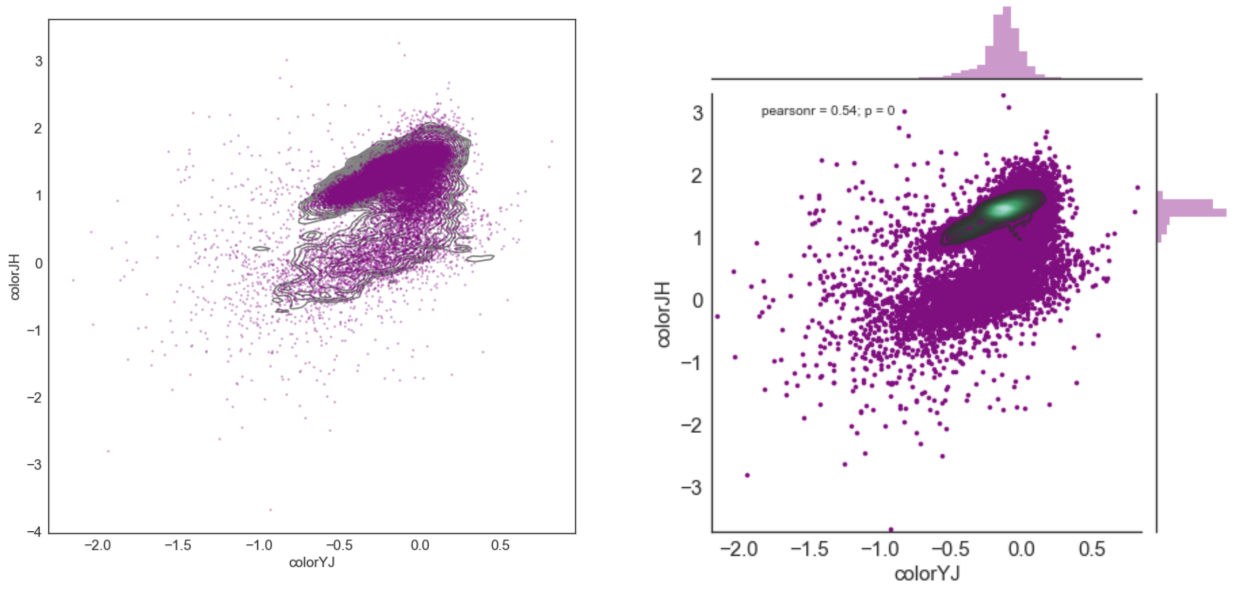


Figure 15: Graphical representation of the simulated distribution of stars using Gaussian Mixture Model as a function of the colours Y-J and J-H, together with the KDE distribution function obtained using the optimal bandwidth for the Gaussian Model (left) and using seaborn KDE (right).

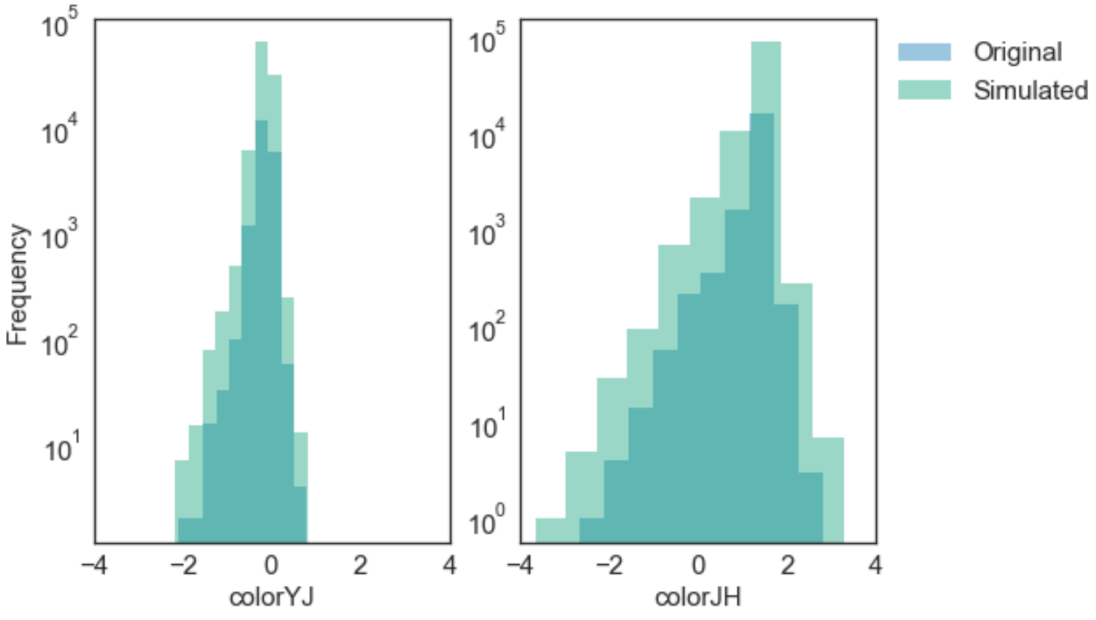


Figure 16: Graphical representation of the number of original and simulated stars using Gaussian Mixture Model as a function of colours Y-J and J-H in logarithmic scale.

The result of the 100000 simulated stars using the Gaussian Mixture Model can be seen both in figures 15 and 16. Comparing these figures with the corresponding ones obtained using the

estiation of the Kernel Density using the optimal bandwidth, we observe how the Gaussian Mixture Model is not able to fit with precision the tails of the distributions (see in particular histograms in figure 16). Therefore, we encounter a larger number of simulated stars outside of the KDE estimated using the optimal bandwidth. Nevertheless, the implementation of the Gaussian Mixture Model is faster, as well as it requires less time of computation, even if we increase the number of Gaussian functions until 30. Taking into account that the estimation of the optimal bandwidth using cross-validation needed around 40 minutes of computation for a computer with 16GB of RAM and intel processor i7 with two cores, I conclude that the results of the Gaussian Mixture Model are precise enough to achieve a starting point about future results for this simulation.

2.5 Conclusion

I have carried out the design of a database, realizing that when the designer has not experience on dealing with databases, the process is mostly dominated by testing and trying different designs and analyzing the provided feedback. The creation of just one DATA table makes very inefficient to look for concrete objects line per line (this is the case of query R3, which requires at least 1 hours and a half to be finished).

Moreover, the interpretation of the queries may change during the process itself, putting the question of the design of the database back in discussion. Finally, I would like to point out that, although it contains data that can be easily calculated using PYTHON, included the COLOR tables was an advantage as it allows to work efficiently retrieving the information quickly for the Euclid simulation. Something similar can be proposed to deal with varying stars in the Ks-bands.

3 Photometric redshifts of galaxies

3.1 Introduction

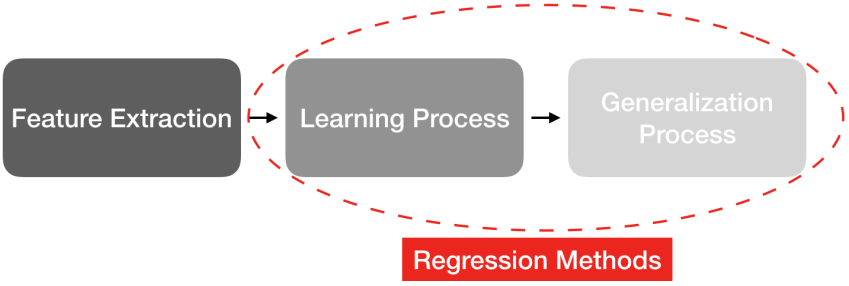


Figure 17: Machine Learning schema followed in the photometric redshift estimation.

Most of the astronomy research relies upon distance information. For extra-galactic objects, the true distance cannot be measured and we need to use the redshift as a measurement of the distance, which is close to the true distance if the object is very far away. Spectroscopic redshifts are more accurate than photometric redshifts, but their acquisition is time-consuming and limited to only the brightest objects. An alternative approach is the use of multiple filter images to derive photometric redshifts, which depends on colour changes. Since photometric redshifts rely only on accurate colours, it is possible to acquire much larger samples of photometric redshifts than spectroscopic ones. [5]

A way of determining the photometric redshifts is learning a function $f(\theta)$ that approximately predicts the redshifts of a galaxy when given parameters θ. In this case, the parameters θ would be those associated with the measurement of the colours of the galaxy, that can be the colours or magnitudes. The simplest and most empirical photometric redshift model is that assumed to be a linear function of the magnitudes m_i of the galaxies, such as,

$$f(\theta) = z_{phot} = a_0 + \sum_i b_i m_i \tag{4}$$

where a_0 and b_i are the fitting coefficients. The precision of the learning process is measured as the discrepancy between photometric and spectroscopic redshifts as,

$$E(\theta) = median\left(|z_{spec} - f(\theta)/(1 + z_{spec})|\right) \tag{5}$$

where the requirements of large future cosmological surveys are $E(\theta) < 0.01$. [4] [11]

The procedure of this part would be as follows (see figure 17): the data would be analyze according to feature extraction, and then the model will be learned and generalized according to several regression methods.

3.2 Feature extraction and data considerations

The problem presents two different tables, A and B, in Votable format, containing the magnitude r, the colours $(u-g)$, $(g-r)$, $(r-i)$, $(i-z)$ and the spectroscopic redshift z_{spec} for galaxies. I will denominate the the sample A, training data, and the sample B, generalization data. Moreover, I calculate the magnitudes corresponding to the above mentioned colours for tables A and B. The result are 5 magnitude variables: m_r, m_g, m_u, m_i, and m_z.

The resulting magnitude data will be 5-dimensional. Therefore, we should consider using feature extraction to obtain relevant information from the data. This process will ease further required steps such as fitting processes based on training and generalization methods. In this case, it is convenient to think on using feature extraction as a form of obtaining dimensionality reduction. This technique is meant to reduce the number of random variables under some considerations to obtain a set of principal variables. Thus, I perform Principal Component Analysis (PCA) to find out whether there are variables that are correlated.

PCA is a process of linear data transformation. It can be used to transform the possible correlated variables of the data into a set of values of linearly uncorrelated variables that will be denominated Principal Components (PC). The PC, in which the first component will be that whose variance is the largest, will be the eigenvectors of the covariance matrix. It is important to remark that the covariance matrix gives an insight of how closely a dataset can be fit by a linear regression in N dimensions. Thus, knowing that the relation of the photometric redshift and the magnitudes is linear according to (3), the use of PCA would facilitate future linear regression processes, as we will see below.

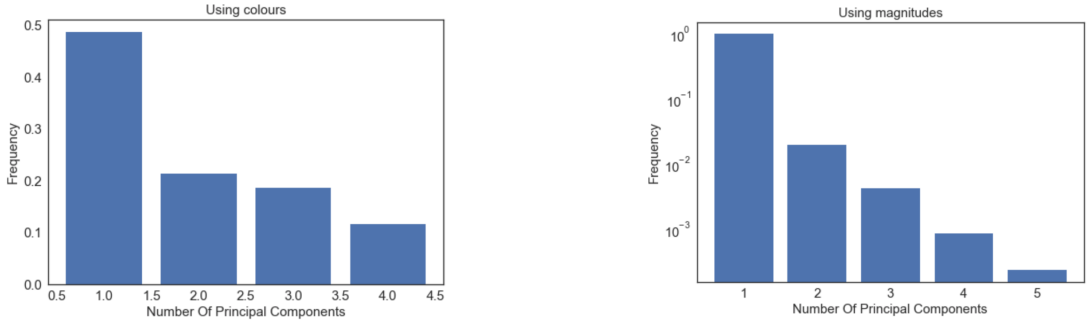


Figure 18: Graphical representation of the number of principal components using colours (left), with 4 main components ($(u-g)$, $(g-r)$, $(r-i)$, $(i-z)$), and using magnitudes (right), with 5 main components $(m_r, m_g, m_u, m_i, m_z)$, and shown in semi-logarithmic scale.

I carry out this process using the PCA PYTHON SKLEARN PACKAGE. I perform PCA for both magnitudes and colours, and I also standardize features by removing the mean and scaling to unit variance. Results are shown in figure 18. It is possible to see how it seems that there is one important PC in both plots. However, this noticeable principal component is due to systematics. This strong correlation means that the magnitude decays as the distance increases. Therefore, it is not convenient to perform PCA again projecting the data to this main component, as it

would induce a lack of information, but to include the rest of component in our PCA method. The representation of the magnitudes and the PCA transformation can be seen in figures 18 and 19.

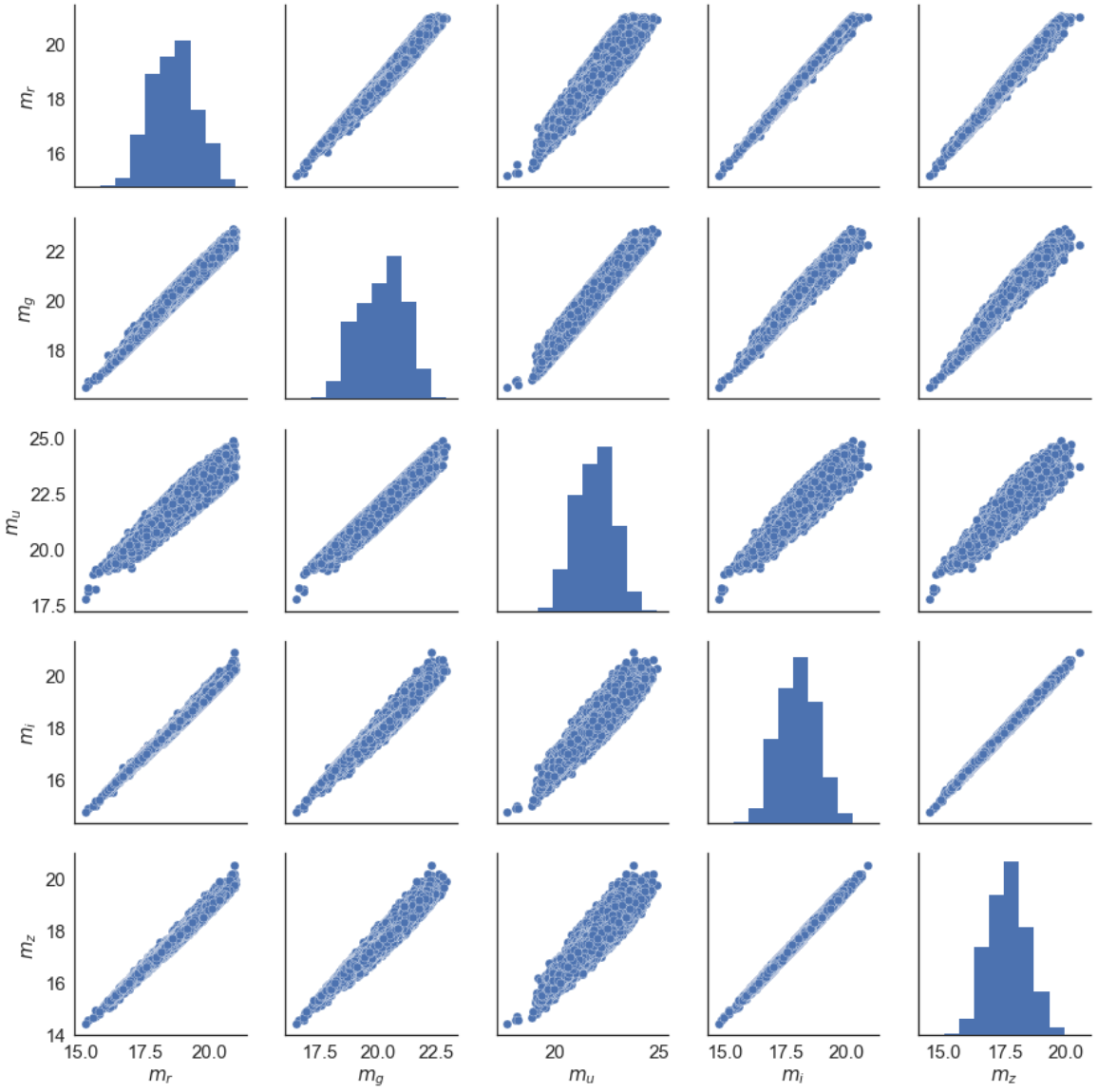


Figure 19: Triangle plot of the magnitudes in table A. We observe that there are certain linear correlations in the data.

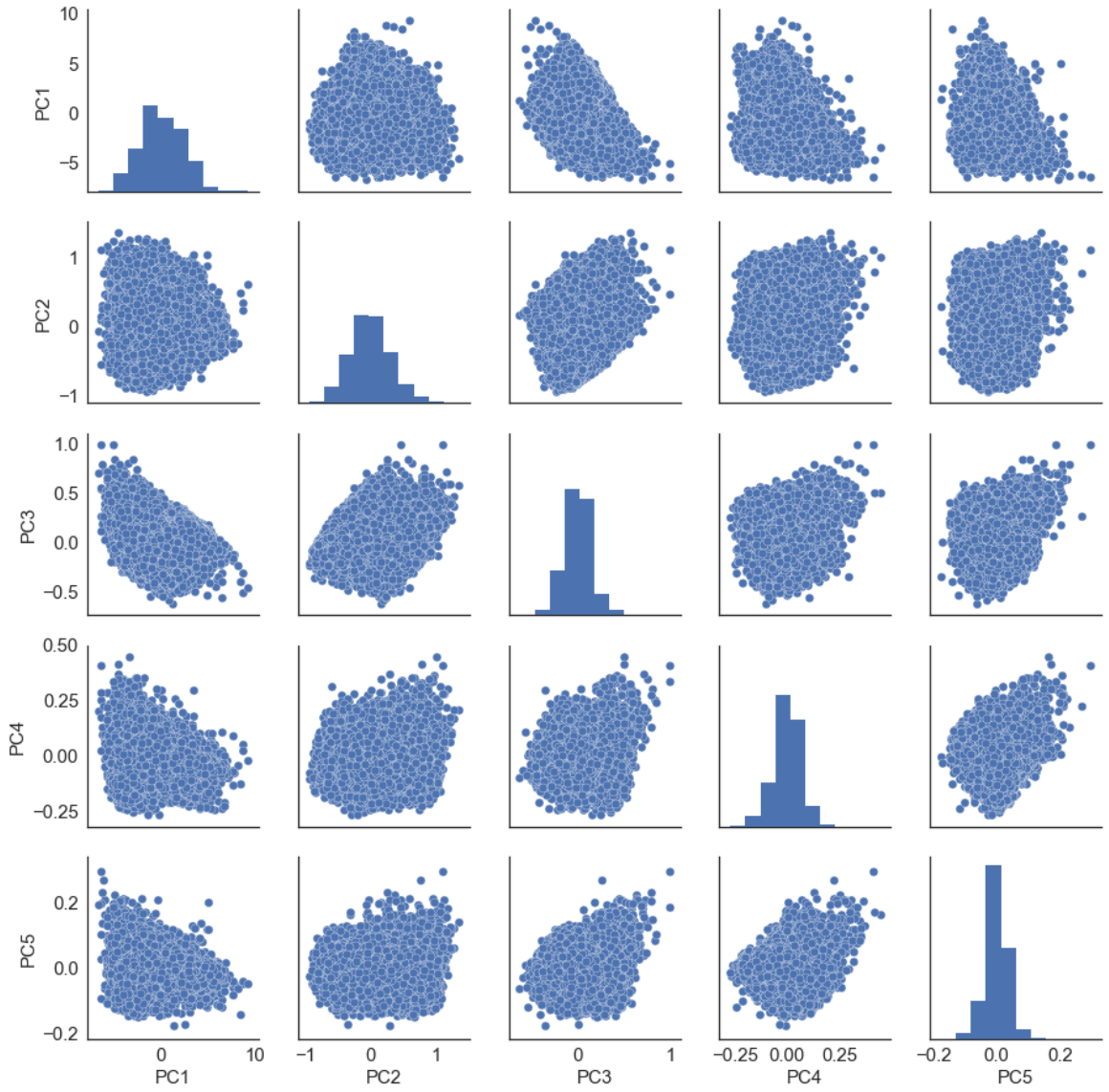


Figure 20: Triangle plot of the PCA transformation of magnitudes in table A.

3.3 Linear regression methods

I perform the fitting of the function $f(\theta)$ as well as the estimation of $E(\theta)$ using the linear regression methods available in PYTHON SKLEARN PACKAGE: LinearRegression, Lasso and Ridge. The results are shown in table 10. The graphical representation of the result of the fitting performed using LinearRegression can be seen in 21.

The models are trained using the PCA training data A. The training error $E(\theta)_t$, defined according to eq. (4), is the error obtained when the trained model is run on the training data. On the contrary, the generalization error $E(\theta)_g$, calculated using eq. (4) as well, is the one obtained when the trained model is run on the generalization data B that has gone through

Method	$E(\theta)_t$	$E(\theta)_g$
LinearRegression	0.01433	0.01447
Ridge	0.01456	0.01462
Lasso	0.01456	0.01462

Table 9: Results for the training error $E(\theta)_t$ and generalization error $E(\theta)_g$ for the three different linear methods.

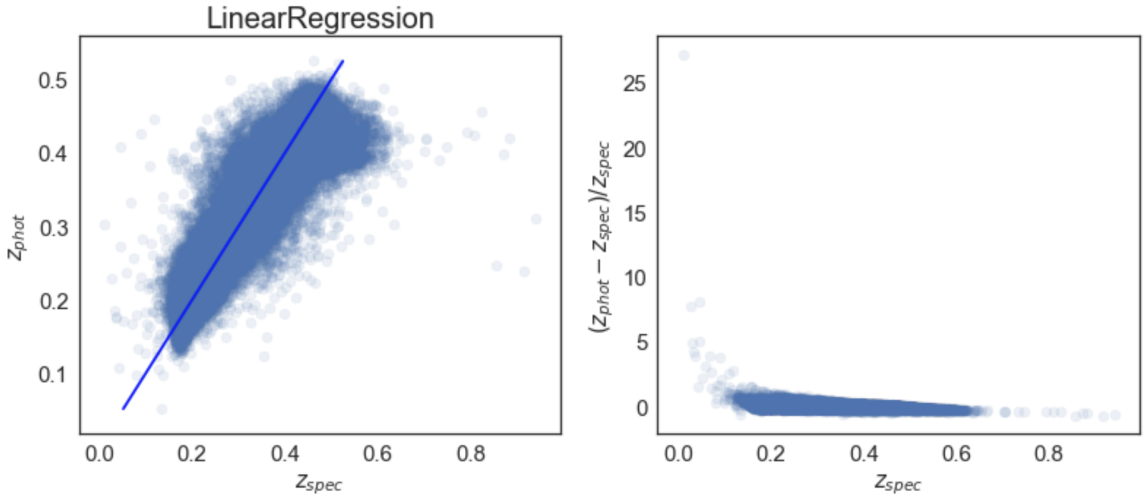


Figure 21: Graphical representation of the trained photometric redshift (left) and relative residuals (right) as a function of the spectroscopic redshift, for the Linear Regression, using training data A. Very similar results were found for Lasso and Ridge.

PCA with the fitting of the PCA of data A. The value of these errors are shown in table 10. In order to obtain the lowest value in the training data using Lasso and Ridge, it was necessary to set the regression element α to zero. This means that these linear regression methods behave as the normal LinearRegression.

The training error could not be reached to be lower than 0.01, in spite of the fact that the data went through PCA analysis. Furthermore, the training error cannot be trusted. This is due to the fact that the model has been trained using the training data, and this does not necessarily imply that the trained model will show a satisfactory performance when it is applied to another data (for instance, the testing or generalization data) which do not need, in principle, to resemble the training data. The generalization error gives then an estimation of how good your model fits other datasets.

3.4 Non-linear regression methods

As mentioned before, it is known that the relation between the photometric redshift and the magnitude can be modelled as linear [11]. However, it is convenient to explore other regression methods in order to reduce the training and generalization errors.

3.4.1 K-Nearest Neighbours

Until now, the linear regression methods used above were *parametric* approaches, because it assumes a linear function as eq. (3). The easiest generalization is to move to a non-parametric method such as K-Nearest Neighbours (KNN).

According to the documentation of SKLEARN, the algorithm of this method consists of assuming or taking a value for the number of nearest neighbours K and a prediction of an initial point x_0. Then, KNN estimates the number of training values N closest to the prediction point x_0 by calculating the distance to them, and finally, it estimates the function $f(x_0)$ as the average value of all the N values found before. This is the reason why KNN is known as a local regression method.

KNN regression shows some advantages with respect to the linear methods shown above. It does not need a specific function to be fitted to, providing a more flexible approach. For instance, if the data does not show a very evident linear relationship, KNN will work better than linear regressions. Taking into consideration the power of KNN given its easy implementation, it also shows possible simple ways of tuning the algorithm by means of assign weights to the neighbours, which makes it a very adaptive methods. Finally, it does not require much computational time and effort if the KNN algorithm is looking for the closest neighbours using KD-Trees or balls.

On the other side, KNN regression also presents some disadvantages with respect to linear regressions. For instance, the interpretation of the algorithm, as well as the results of the regression are not as intuitive as those from linear regressions. It usually estimates a higher number of coefficients than parametric methods and does not learn anything from the training data and just use the training data for classification. Furthermore, an estimation of the optimal number of neighbours should be carried out.

The estimation of the optimal number of neighbours is important when computing the regression itself. In most of the cases, the optimal value for the number of neighbours depends on the bias-variance tradeoff. On the one hand, an small value for the number of closest neighbours provides the most flexible fit, showing a low bias but a high variance because the prediction in a given region will depend on few observations. This model will be consistent, but inaccurate on average. On the other hand, for higher values of the number of neighbours, the variance will be reduced and the fit will be smoother. In this case, the model will be accurate on average, but inconsistent. For a visualization of the meaning of the bias-variance tradeoff problem, the reader is referred to [1].

To find the suitable number of neighbours, I use cross-validation using PYTHON SKLEARN, following the process described in [2], and the result is shown in figure 23. Although the result

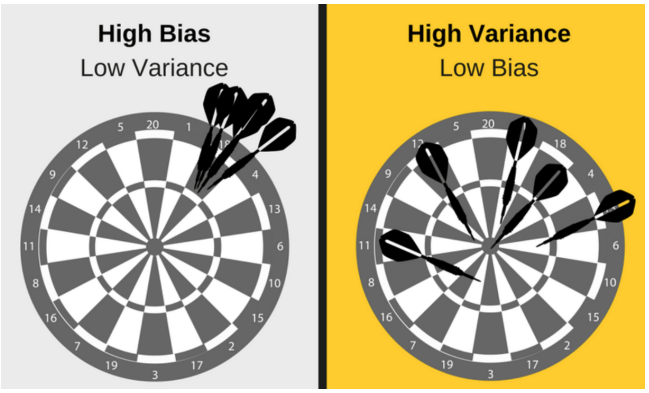


Figure 22: Schematic visualization of the difference between high/low bias and high/low variance. Obtained from [1].

of cross-validation indicates that the most suitable number of neighbours is equal to 3, the difference with respect to other values is very small. In figure 24, the fitting process using KNN with 3 as the number of nearest neighbours is shown. The values for the training error is $E(\theta)_t = 0.00978$ and for the generalization error is $E(\theta)_g = 0.01407$.

I have also carried out a search of the changes in the $E(\theta)_t$ and $E(\theta)_g$ as a function to the number of neighbours (see figure 25). We can observe how the dependency shows an asymptotic behaviour between $E(\theta)_t$ and $E(\theta)_g$ starting in the number of neighbours around 20: the $E(\theta)_t$ increases as the number of neighbours also increases whereas the $E(\theta)_g$ decreases. The same results were found regardless of the use of the original or the PCA data we used for the linear regressions. Therefore, this implies that there is not much space for improvement regarding this method.

Still, I also carried out a joint analysis of KNN and AdaBoost, obtained from SKLEARN too. AdaBoost is the short name of Adaptive Boosting. It is a general ensemble method that creates a strong classifier from a number of weak classifiers, such as for instance KNN. Basically, KNN builds the model from the training data; then, AdaBoost creates a second model that attempts to correct the errors from the first one. Models are added until the training set is predicted perfectly or a maximum number of models are added.

The main advantage of AdaBoost is that it gives the possibility of improving weak method whose implementation are less computational time-consuming, although it is true that it adds extra computational time to the regression process. However, as it attempts to correct misclassification in the training data, this data needs to be of very high-quality. In this sense, noisy data, specifically noise in the output variable can be problematic.

I have carried out an AdaBoosting on the KNN regressor using 20 as the number of estimators and keeping 3 as the number of closest neighbours. The results were very similar, not showing any improvement even increasing the number of estimators until 100 (see figure 26, where the asymptotic behaviour takes place again). The values of the training and generalization errors are respectively $E(\theta)_t = 0.00963$ and for the generalization error is $E(\theta)_g = 0.01435$. We observe how the training error has slightly decreased, whereas the generalization error is now

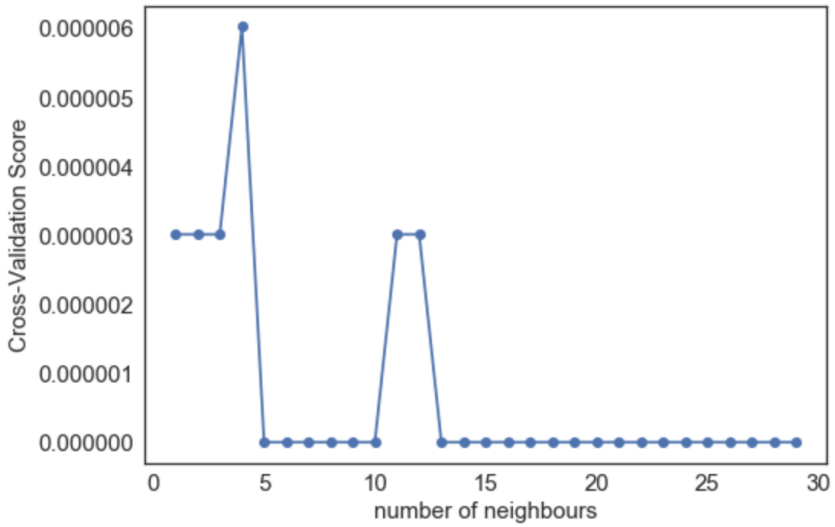


Figure 23: Graphical representation of the cross-validation score as a function of the number of neighbours. The suitable value for the closest neighbours should be 3.

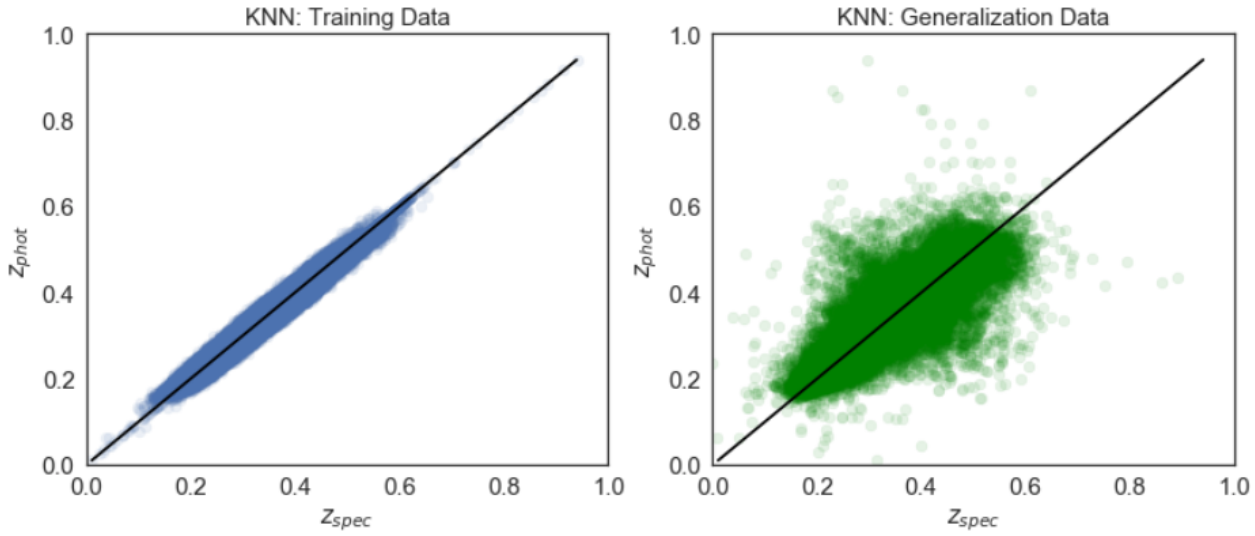


Figure 24: Graphical representation of the training process of the photometric redshift (left) and generalization process of the photometric redshift (right) as a function of the spectroscopic redshift, for the K-Nearest Neighbour Regression, using 3 as the number of neighbours.

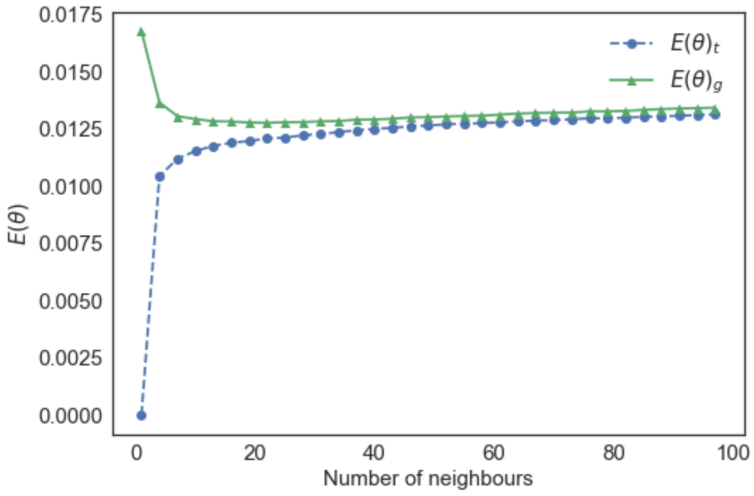


Figure 25: Graphical representation of the training error $E(\theta)_t$ (blue circle, dashed lines) and generalization error $E(\theta)_g$ (green triangles, solid line) as a function of the number of neighbours in the KNN regression process.

higher.

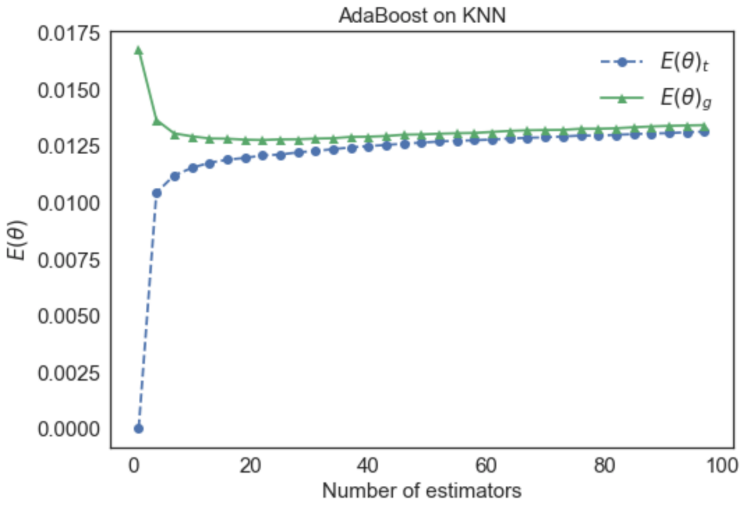


Figure 26: Graphical representation of the training error $E(\theta)_t$ (blue circle, dashed lines) and generalization error $E(\theta)_g$ (green triangles, solid line) as a function of the number of estimators in the AdaBoost regression process, maintaining 3 as the number of neighbours in KNN.

3.4.2 Random Forest

A Random Forest (RFs) is a meta estimator that fits a collection of classifying decision trees on various sub-samples of the training dataset that is split randomly. According to the docu-

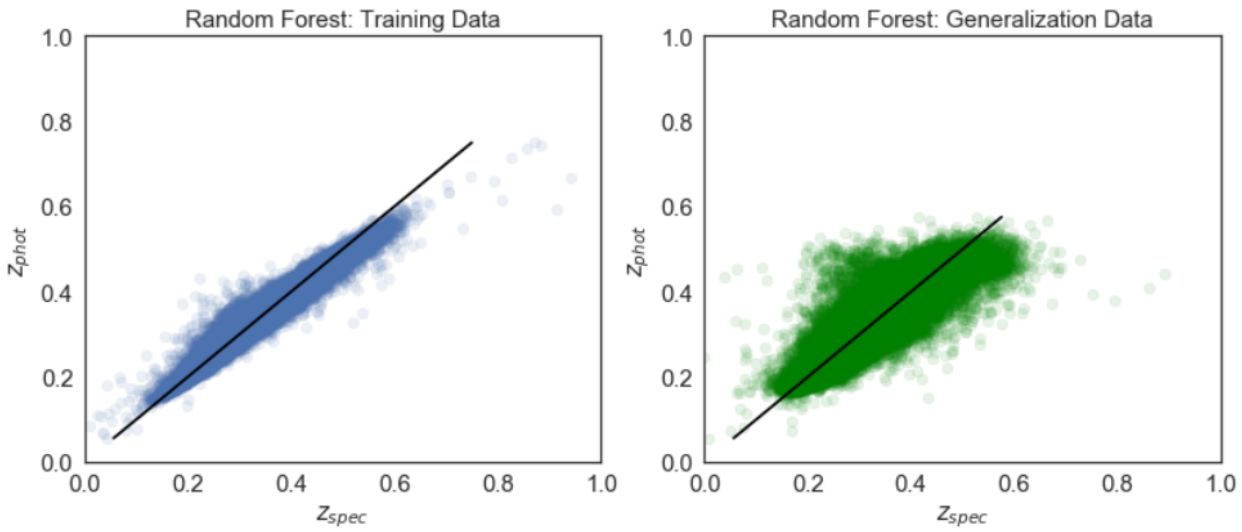


Figure 27: Graphical representation of the training process of the photometric redshift (left) and generalization process of the photometric redshift (right) as a function of the spectroscopic redshift, for the Random Forest Regression, using 20 as the number of estimators.

mentation of SKLEARN, the working mechanism is very similar to decision trees. [6]

Random Forest offers several important positive aspects with respect to other methods. For example, the bias remains almost constant as that of a single decision tree, meanwhile, the variance decreases so that the possibilities of overfitting get low. However, there are also negative points to take into account. To start with, RFs do not offer a good training process on small datasets as it is not able to understand the pattern. Moreover, it is difficult to interpret the relationship between the response and the independent variable. This is due to the fact that it is a predictive tool instead of a descriptive tool (in this sense, linear regressions provide with a description of the model). Besides, the time required to train RFs could be on some occasions very high because of the existence of multiple decision trees. Finally, RFs are unable to take values outside of the range of the training data, whereas this is something possible by linear regression methods.

RF regression method has been carried out using the whole set of training data with 20 as the number of estimators, which would basically correspond to the number of trees in the RFs. Later, I have obtained the generalization error using the whole set of testing data B. The values of the training and generalization errors are respectively $E(\theta)_t = 0.00455$ and $E(\theta)_g = 0.01256$. The results of the training and testing processes are shown in figure 27.

In order to obtain an interpretation of how many number of estimators would be necessary to reduce the training and generalization errors, I have computed several RFs regressions varying the number of estimators calculating each time both kind of errors. The results are visible in figure 28. In this image, we can observe how the relation between the errors and the number of estimators is almost constant starting at around the number of estimators equal to 10. Therefore, the testing and training error do not depend on the number of trees.

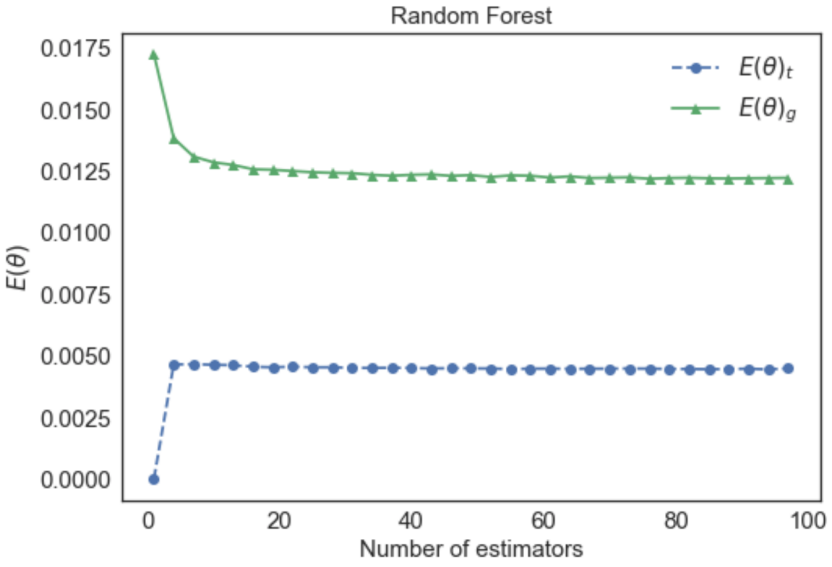


Figure 28: Graphical representation of the training error $E(\theta)_t$ (blue circle, dashed lines) and generalization error $E(\theta)_g$ (green triangles, solid line) as a function of the number of estimators used in the Random Forest regression. The Minimum value of $E(\theta)_t = 0.0044$ was obtained for the number of estimators equal to 31. Minimum value of $E(\theta)_g = 0.0122$ was found for the number of estimators equal to 25.

3.4.3 Gaussian Processes

The decision of using Gaussian Processes (GPs) as a non-linear method of regression is mainly funded for two different reasons: some astronomers have already used this method precisely to address this problem [10], and Gaussian Processes is a Machine Learning technique that will be used in future inflation research analysis described in [9], in which the author of this report will actively collaborate. Therefore, learning how to use of Gaussian Processes is a very valued initial process.

According to the documentation of SKLEARN, Gaussian Processes are generic supervised learning method designed to solve regression and probabilistic classification problems. They are also known for being a collection of processes based on random variables indexed by time or space, such that every finite collection of those random variables has a multivariate normal distribution. They are widely used for regression purposes. Using this python package, the hyperparameters of the kernel are optimized during fitting of GaussianProcessRegressor by maximizing the log-marginal-likelihood (LML) based on the passed optimizer. As the LML may have multiple local optima, the optimizer can be started repeatedly by specifying $n_restarts_optimizer$. In this case, the $n_restarts_optimizer$ was set to 3. The first run is always conducted starting from the initial hyperparameter values of the kernel. The Kernels are a crucial ingredient of GPs which determine the shape of prior and posterior of the GPs. In this case, a simple "squared exponential" kernel (RBF) was used. The results can be seen in figure 29.

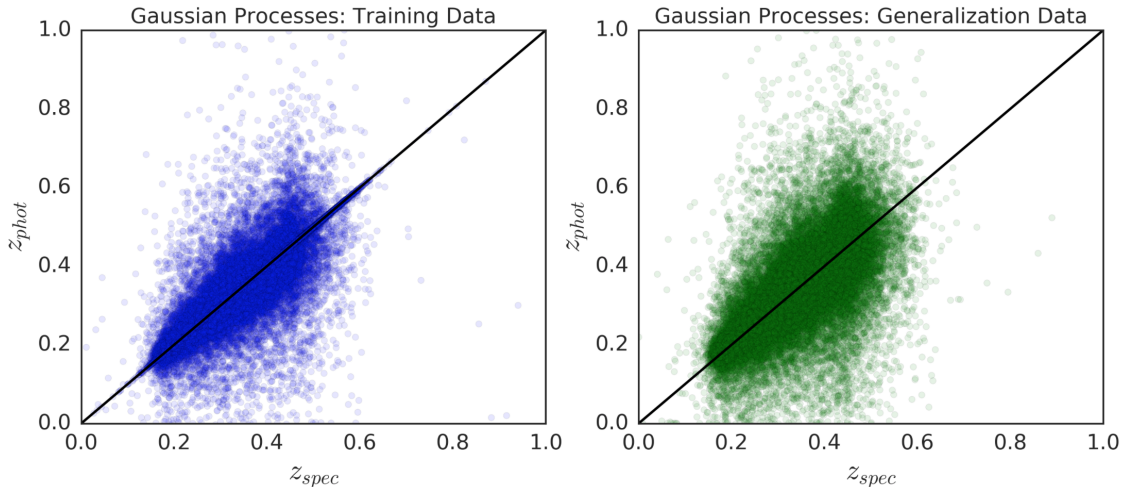


Figure 29: Graphical representation of the training process of the photometric redshift (left) and generalization process of the photometric redshift (right) as a function of the spectroscopic redshift, for the Gaussian Processes Regression, using 50% of the training data.

Mostly, the advantages of GPs rely on the fact that the prediction of the fitting is probabilistic (common statistical in confidence intervals can be computed) and interpolates the observations for regular kernels. They are also very versatile because of the several kernel possibilities. On the contrary, they require to use the whole samples/features information to perform the prediction. They lose efficiency in high dimensional spaces (when the number of features exceeds a few dozens).

Particularly because of this reason and from the user experience, although GPs seem very powerful, the main disadvantage of this method has resulted to be the high computational cost: time as well as the cost of memory it required for the regression process. Even though the process was carried out using the Lorentz Institute computational cluster NOVAMARIS, the complete set of training data could not be used for the fitting and prediction. Using 50% of the training data to make the fit and the prediction, and 100% of the data of the generalization data, the process took for approximated 2 hours. The value for the training error was $E(\theta)_t = 0.0083$ and for the generalization error $E(\theta)_g = 0.0193$.

We perform more tests on the amount of data that the Gaussian Processes Regressor could handle without showing memory problems; however, at 75% the python kernel always died independently of the required amount of time, which was considerably high. Thus, the errors shown above cannot be trusted and a further analysis of how to deal with time and memory issues should be carried out for future improvement.

3.5 Conclusion

After pursuing an exhaustive analysis on different regression methods, linear and non-linear, if we rely on the training and generalization errors, Random Forests is the most efficient regressor

Method	$E(\theta)_t$	$E(\theta)_g$
LinearRegression	0.01433	0.01447
Ridge	0.01456	0.01462
Lasso	0.01456	0.01462
KNN	0.00978	0.01407
AdaBoost on KNN	0.00963	0.01435
Random Forest	0.00455	0.01256

Table 10: Results for the training error $E(\theta)_t$ and generalization error $E(\theta)_g$ for some of the regression methods used.

so far (see table 10). It has shown the lowest errors compared to other methods, apart from being a quick algorithm that does not consume much computational time and memory. Being interested in predictions (our main goal was to reduce the generalization error using non-linear methods), Random Forests has been proved to be a convenient method as it does not require any assumption about the model or linearity in the provided data.

Nevertheless, someone would also expect to obtain good results with linear regressions, as we knew a priori that according to some bibliographical resources the relation between the photometric redshift and the magnitudes can be modelled to be linear. In spite of the fact that the feature extraction carried out by means of performing Principal Component Analysis would in principle allow finding new minima in the likelihood during the regression process, this could not be confirmed with this dataset, as similar (or equal) results were found using the original variables and the principal component descriptions. Still, it is believed that if PCA analysis worked, linear regression methods would give similar results than KNN, for instance, as it is probable that the training error could have been further reduced. In case that true, linear methods would offer us speed, feasibility and direct understanding of a simple model.

During the whole process of data mining, some other regressors corresponding to other packages were tested such as linear regressors based in different minimization algorithms from ASTROPY, the linear regressors of SKLEARN using weights in the y component, and even boosting in Random Forest. However, all these tries did not provide with significant results. With all this experience, I conclude that the method chosen to carry out the regression process should depend on the characteristics of the data, as well as the computational needs or availability of the scientist.

4 Code Documentation

4.1 About

The codes written to carry out the whole tasks of the project are publicly available at the Github Repository `https://github.com/gcanasherrera/DDBDM2017_FP.git`, including figures, and some other derived products such as the database created in the first part of the assignment and the cross-validation estimation of the bandwidth data used to calculate the Kernel Density Distribution for the estimation of the 100000 stars for the Euclid Mission.

4.2 Requisites

All computation were written in PYTHON 2.7 using IPYTHON NOTEBOOK. Moreover, all programs were run in a laptop MacBook Pro (with MacOS High Sierra) with processor Intel Core i7 at speed 3 GHz with 2 cores and 1 processor.

To run the IPYTHON NOTEBOOKS, the following python packages need to be installed:

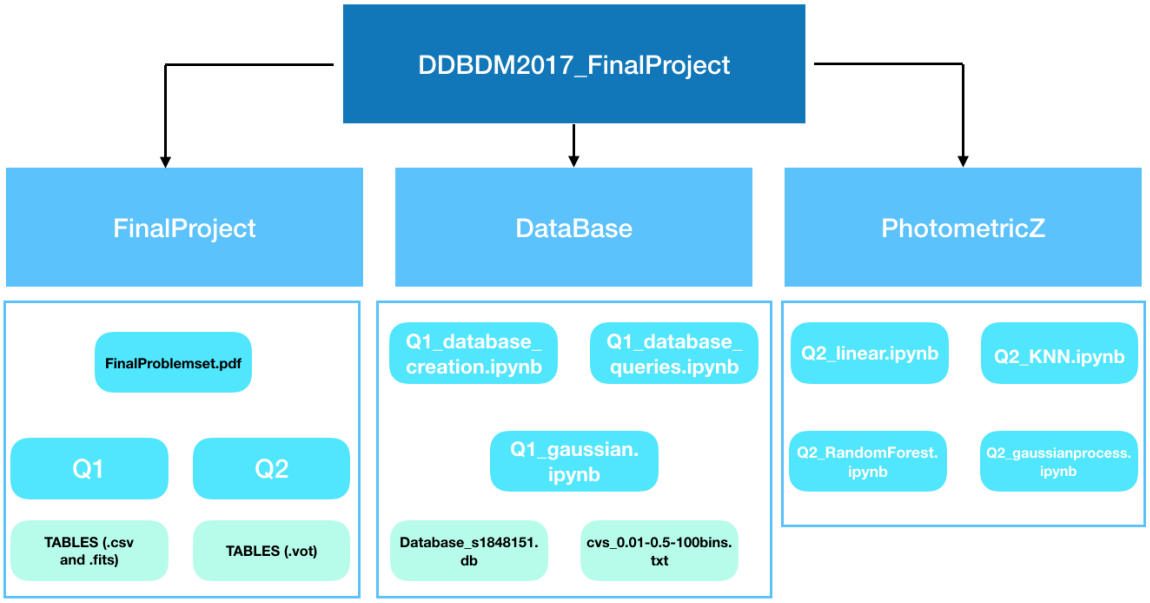
1. MATPLOTLIB (version 2.1.1)
2. NUMPY (version 1.13.3)
3. ASTROPY (version 2.0.2)
4. SKLEARN (version 0.19.1)
5. SEABORN (version 0.6.0)
6. SQLITE3 (version 2.6.0)
7. PANDAS (version 0.16.2)

4.3 Structure

AS shown in the schema below, the repository contains three different folders:

- **FinalProject**: contains provided data by Jarle Brinchmann in form of tables. The original question sheet is also present.
- **DataBase**: contains all codes used to answer questions corresponding to the first part of the assignment in databases. There are three IPYTHON NOTEBOOKS corresponding to the creation of the database, the SQL queries and their representations of the results, and the simulation of 100000 stars of the Euclid mission ($Q1\_gaussian.ipynb$). A simple python file called AUXILIARY.PY is also included containing already coded routined by Jarle Brinchmann to calculate the CV Bandwidth.

- **PhotometricZ**: contains provided data to carry out the second part of the report. The *Q2_linear* contains the linear regression methods, and the other notebooks contain each one non-linear regressor: KNN, RFs and GPs.



Acknowledges

First at all, I would like to say thanks to my classmates Charlotte, Jorge, Felipe, Louis and Esmee for the useful discussions we have maintained during the whole project. I want also to show my gratefulness to Juan Claramunt, for being the "color blind Guinea Pig" as well as a friendly statistician always willing to help. Finally, I appreciated very much having Pablo de Castro interested in answering random questions about the package SKLEARN.

References

[1] Bias-variance-tradeoff. `https://elitedatascience.com/bias-variance-tradeoff`. Accessed: 19-01-2018.

[2] A complete guide to k-nearest-neighbors with applications in python and r. `https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/`. Accessed: 19-01-2018.

[3] Euclid consortium: A space mission to map the dark universe. `https://www.euclid-ec.org`. Accessed: 19-01-2018.

[4] Photometric redshift. `https://ned.ipac.caltech.edu/level5/Glossary/Essay_photredshifts.html`. Accessed: 19-01-2018.

[5] Photometric redshifts. http://www.ifa.hawaii.edu/reu/2006research/adams.pdf. Accessed: 19-01-2018.

[6] Random forests explained intuitively. https://www.datasciencecentral.com/profiles/blogs/random-forests-explained-intuitively. Accessed: 17-01-2018.

[7] seaborn: statistical data visualization. https://seaborn.pydata.org. Accessed: 19-01-2018.

[8] Time domain astronomy: a new frontier of astronomy. http://dame.dsf.unina.it/documents/DSF_2012.pdf. Accessed: 19-01-2018.

[9] A. Achúcarro, V. Atal, P. Ortiz, and J. Torrado. Localized correlated features in the CMB power spectrum and primordial bispectrum from a transient reduction in the speed of sound. *Physical Review D.*, 89(10):103006, May 2014.

[10] D. G. Bonfield, Y. Sun, N. Davey, M. J. Jarvis, F. B. Abdalla, M. Banerji, and R. G. Adams. Photometric redshift estimation using Gaussian processes. *Montly Notices in the Royal Astronomical Society*, 405:987–994, June 2010.

[11] A. J. Connolly, I. Csabai, A. S. Szalay, D. C. Koo, R. G. Kron, and J. A. Munn. Slicing Through Multicolor Space: Galaxy Redshifts from Broadband Photometry. *Astrophysical Journal*, 110:2655, December 1995.

[12] J. R. Taylor. *Introduction To Error Analysis: The Study of Uncertainties in Physical Measurements*. 2001.