Database Design and Photometric Redshift Estimation

Databases and Data Mining Final Report

Jorge Andrés Villa Vélez

*Leiden Observatory, Leiden University, The Netherlands**

January 26, 2018

Abstract

Taking advantage of different useful libraries and packages provided by **scikit-learn**, a database was created to organize the outcome data from a given survey. The database has been schemed with two different tables, one containing the information of all objects in each field (Field-1, Field-2, and Field-3), and filters (**Z**, **Y**, **J**, **H**, and **Ks**), and another one containing the information of the images. From the colors, J-H and Y-J computed for 18,428 stars, a new sample of 100,000 was drawn implementing Kernel Density Estimation and Gaussian Mixture Models. The best results were achieved with a combination of 13 Gaussians, though the KDE model was good enough to reproduce some features. Additionally, following a linear model as described by (Connolly et al., 1995) the photometric redshift for 74,309 was estimated using Linear Regressors (LR, Ridge, and LASSO) and Non-Linear Regressors (Random Forest, Bagging, and Gaussian Processes). The training and generalization errors were lower down to an average value of 0.005 and 0.0127 respectively, from an initial value of 0.0145 obtained with a simple Linear Regressor model. Different methods are tested and pros, and cons are pointed out as a final discussion.

1 Introduction

In the past years, an increase in new astronomical data has been perceived. New facilities such as LSST, ZTF or MASCARA are basic examples of the importance of time-domain in Astronomy. The optimum design of a database to hold data from those facilities is crucial in order to cleverly store and manipulate the valuable outcome of different projects. Here, a database will be designed based on The Vista Variables in the Vía Láctea Survey. This survey takes images of patches in the sky at almost regular intervals using five different filters (**Z**, **Y**, **J**, **H**, and **Ks**). Observations are repeated different times in band **Ks** to keep track of possible stellar variability. The number of observations of a particular sub-area varies between 50 and 300 (Final Project Guide, 2017). Moreover, taking into account that the Euclid mission will share three of this filters namely **Y**, **J**, and **H** a density estimation methods were used to draw new samples of stars following the same distribution as the one exhibited in the J-H, Y-J plane.

Besides this, another common task in Astronomy is to determine the distance of extra-galactic objects. This is performed via the redshift estimation of the source. Nevertheless, computing spectroscopic redshifts is highly time-consuming because one need the spectrum for every source and later characterize it. A different

*The author can be contacted via jvilla@strw.leidenuniv.nl

technique to estimate the redshift comes from a relation between the distance and the colors or magnitudes of the sources. This measurement is known as the photometric redshift. To address this, different regression methods will be applied to a given dataset in order to estimate in a good fashion the most reliable values for the photometric redshifts to construct a model and then use it to predict over a different bunch of data following the same structure. The dataset is conformed by M_r, $u-g$, $g-r$, $r-i$, $i-z$, and z_{spec} for $74,309$ objects.

In this report, I present a design of a suitable database and an estimation of photometric redshifts of galaxies. In Section 2 a brief introduction to the two different problems is presented. A solution for the design of the database and the simulation of new stellar objects following a certain distribution in a color-color plane is addressed in Section 3. A photometric redshift estimation using regular linear regression and more elaborated methods was performed in Section 4. A discussion and conclusions are provided in 5 and 6 respectively. A useful appendix is listed at the end of the report in 9.

2 Data

The project was divided into two main sections as said before. The first one is mainly related to the importance of time-domain surveys in Astronomy and how to deal with massive data resulting from different surveys. Moreover, creating suitable databases to address this helps us in a good fashion to properly organize and store our data. The first dataset corresponds to some test-data from a survey which takes images of a patch of the sky at fairly regular intervals. Every patch is observed in five different filters corresponding to bands: **Z**, **Y**, **J**, **H**, and **Ks**. Light curves of stars will be build up using band **Ks**, so multiple observations are performed. Different areas or patches in the sky are scanned following a given scanning-law for the survey yielding to a different amount of observations per patch. In our case, the dataset corresponds to three different *Fields* with $10,000$ objects each. In the band **Ks**, *Field-1* was observed three times whilst *Field-2* and *Field-3* were scanned one and two times respectively. When a given field is observed, the flux (**Flux1**, **Flux2**, **Flux3**) and calibrated magnitudes (**Mag1**, **Mag2**, **Mag3**) are obtained for all objects in three different apertures with its respective uncertainties in the measurements. A flag, corresponding to the kind of object is added where -1 corresponds to a Star, 0 to noise, $+1$ if non-stellar, and -2 if borderline stellar. A **RunningID** is provided but only to count-up objects in a given field and not properly to cross-match with different catalogs. However, each object is assigned a **StarID** to properly cross-match the object in any of the five different bands. Additionally, the position in pixels **X** and **Y**, and the Right Ascension **RA** and Declination in decimal degrees for each object is provided.

An additional file, corresponding to the observations details is provided. For each field and filter, a unique **ID** and **Filename** is assigned to keep track where the reduced images are stored in. The given **FieldID**, **Filter**, **MJD**, **Airmass**, and **Exptime** is also provided. We have to take into account that our database will be designed in a given way to meet the requirements of a "survey manager", thus, this original files will be modified and mixed to optimally address different queries. In Section 9, Table 9.1, this information is shown.

The second part of this project was aimed at predicting photometric redshifts of galaxies with different linear and non-linear methods. As it is known, the distance of a galaxy is one of the most challenging measurements in Astronomy. For most extra-galactic sources the true distance cannot be known, thus we rely on measuring the redshift. Using spectroscopy and measuring the amount of shift for a particular line one can predict the redshift (which will be very close to the real distance if the object is quite distant). However, this

is time consuming and lots of observations are needed in order to sample a given bunch of galaxies. Because of this, the photometric redshift technique allows us to compute the "distance" to an object based on the colors or magnitudes we measure. As part of the project, the main goal is to properly use known spectroscopic redshifts and colors to build up models which can help us predict in a good way photometric redshifts for a given amount of sources. This process consists of learning a function $f(\theta)$ that approximately predicts the redshift of a source when given the parameter θ which in our case corresponds to colors or magnitudes. The dataset is composed of files **File-A** and **File-B**, each one with different values of magnitude in the band r, colors and spectroscopic redshift (M_r, $u-g$, $g-r$, $r-i$, $i-z$, and z_{spec}) for $74,309$ objects.

In Section 3 and Section 4, I address the design of a suitable database for our survey and an estimation of the photometric redshifts of galaxies respectively. Results and procedures are thoroughly explained in each subsection.

3 A Database for a Time-Domain Survey

As has been stated above, I want to create a database suitable for a given survey. The main requirements for this database in order to meet the scientific goals of the team will be centered in keeping track of the location of the reduced images, the catalogs of detections in each image, the colors of stars, and its variability in the **Ks**-band. First, a schema for the database is created, taking into account some queries I want to perform. The layout will be entirely designed in order to provide an answer to a few specific questions. *SQL*-queries were used to tackle this down.

Moreover, using information from the color distribution in the **J-H** and **Y-J** color-plane, a mock sample of new stars were created using Kernel Density Estimators and Gaussian Mixture Models in order to have a larger dataset which can be used to model possible stars observed with the Euclid mission. These colors were chosen because Euclid will have three photometric filter corresponding to **J**, **H**, and **Y**. All the details and techniques used are explained in next sub-sections.

3.1 Database Schema

The creation of the database schema was designed in order to address five different queries in the most suitable manner. These queries are listed below:

- **R1**: Images observed between $56800 < \text{MJD} < 57300$ and stars with $\text{signal} - \text{to} - \text{noise} > 5$ in each image.
- **R2**: Objects with $J - H > 1.5$.
- **R3**: Objects whose **Ks** difference larger than 20 times the flux uncertainty from the mean flux.
- **R4**: Number of catalogs for a given field.
- **R5**: Magnitudes **Z**, **Y**, **J**, **H**, and **Ks** for stars with $\text{signal} - \text{to} - \text{noise} > 30$ in **Z**, **Y**, **J**, **H**, and **Ks** bands.

I want to generate our database in an automatic fashion which can be used with a different set of files or survey which meets the requirements of the schema. There are a few ambiguities which one needs to take

into account to properly tackle the queries and build up the schema for the database. As can be seen from the five queries to be performed, one needs to know the color **J-H** for all objects, so the most suitable way to do it is just simply computing first the colors and adding them up to the different files. In two queries, the signal-to-noise ratio is required. However, it really depends on how one defines it. I opted for a signal-to-noise defined as the inverse of the inverse of the relative error. So for example, if one has a relative error of 0.1 then the signal-to-noise has a value of $10:1$. In my case, the relative error will be chosen as the error in the **Flux1** which is the smallest aperture and will be less affected by background noise. In query **R1**, a comparison between the MJD is required and we have multiple files in **Ks**-band, so one needs to remind that files are ordered such as E001 is observed in an epoch before E003 for example. Also, the information inside the Observations file does not correspond to the same number as the one listed in column **MJD**, then, I will choose to work with the column value instead of the one in the filename. All the queries involving flux or magnitude and colors will be addressed using the first aperture values.

There are seven files for **Field-1**, five for **Field-2** and six for **Field-3** in FITS format. First of all, all files will be transformed into a more easily manipulable format as is the CSV in order to customize some new columns automatically from the name of the files or columns inside of them. A copy of the FITS files is created and read including the Observations information file to later be transformed and stored in new directories named *CSV* and *FITS* respectively. A directory called *DB* is also created and the database will be stored there. This is performed in order to work with a copy instead of the original data if something happens during the process and also to keep all different files ordered. Next, all the files corresponding to the **J**, **H**, and **Y** bands are read and colors **J-H** and **Y-J** are computed per each field and written-out in new files outside the main directory. This is due to the fact that some queries and Euclid mock sample can be created easily from this. Later, the color files are read simultaneously with all the provided files and some additional information is extracted from the file-names as the **FieldID** and the **ID** number from the Observations file for each particular image. The subtracted information will be written in new columns in the *CSV* files. The updated files contain now the same information as explained in Section 2 plus the two colors and the corresponding **FieldID** and **ID** columns.

After all the files and new information is in place, I proceed to create the database schema which can be checked in section *Creation of a schema for the database* in the notebook and Figure 1. Basically, all the information contained in each of the files are stored and concatenated using prof. Jarle Brinchmann's function explained in class. The final result from this is a database with two tables, a big one called Stars which will contain all the information of the 18 files (or images) and the corresponding information provided for the observations in a smaller table called Observations. The database file is stored in directory *DB* and can be modified multiple times using the function in the notebook if one wants to change the layout to properly meet some other different requirements of a survey. Once this is done, I proceed to perform the queries using SQL.

3.2 SQL Queries

The database is already created and can be finally used to perform the queries listed in Sub-Section 3.1. All the queries were addressed using SQL, however, for query **R2** it was easier to just use python because the way I created the database layout allows me to access directly to the colors and fields for each object.

For query **R1** all the files with $56800 < \text{MJD} < 57300$ were selected and conditions on the class-flag to only select stars ($\text{Class} = -1$) and signal-to-noise above 5 were set. The query is shown in Section 9 and also

```
command = """CREATE TABLE IF NOT EXISTS Stars (RunningID DOUBLE,
                                    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,
                                    FieldID INT,
                                    ID INT,
                                    YJ_Color DOUBLE,
                                    JH_Color DOUBLE)""".format(table)
```

```
table = 'Observations'
command = """CREATE TABLE IF NOT EXISTS Observations (ID INT,
                                                      FieldID INT,
                                                      Filename varchar(40),
                                                      Filter varchar(2),
                                                      MJD DOUBLE,
                                                      Airmass DOUBLE,
                                                      Exptime DOUBLE,
                                                      UNIQUE(ID))""".format(table)
```

Figure 1: *left*: Schema for stars table. All the file information corresponding to each field and each band are stores in this table. *Right*: Schema for table Observations. The relevant information provided with the survey is stored here to keep track on relevant information.

Table 1: Query **R1** output. The relevant column for the query corresponds to the Number of Stars in file. Other information was added to keep track of the output meaning.

ID	Field	Filter	Image Name	Number of Stars in file
1	1	Z	Z-ADP.2017-01-18T11:58:36.905.fits	6447
2	1	J	J-ADP.2017-01-18T11:58:35.781.fits	7021
3	1	H	H-ADP.2017-01-18T11:58:35.780.fits	7982
6	1	Ks	Ks-ADP.2016-05-25T15:33:43.377.fits	7888
7	1	Y	Y-ADP.2017-01-18T11:58:36.901.fits	6790
8	2	Z	Z-ADP.2017-01-18T11:58:36.905b.fits	6917
9	2	J	J-ADP.2017-01-18T11:58:35.781b.fits	7354
10	2	H	H-ADP.2017-01-18T11:58:35.780b.fits	7725
12	2	Y	Y-ADP.2017-01-18T11:58:36.901b.fits	7212
13	3	Z	Z-ADP.2017-01-18T11:58:36.905c.fits	6717
14	3	J	J-ADP.2017-01-18T11:58:35.781c.fits	7245
15	3	H	H-ADP.2017-01-18T11:58:35.780c.fits	8022
18	3	Y	Y-ADP.2017-01-18T11:58:36.901c.fits	7182

the notebook. Table 1 summarizes the output results. Though the main question was related to the number of stars in each file and the output is not larger than 20, I decided to make a pie plot in order to show the percentage of objects found per field in every filter. The results are summed up in Figure 2. Most of the stars were found to be in band **H** and less in band **Z** for each field. In **Field-2** and **Field-3** no files or stars were found to reach our criteria.

Now, as said before, query **R2** the condition was evaluated using python instead of SQL because of the intrinsic design of the database. Part of the code can be checked in Section 9 and also the notebook for more details. All objects with color $J - H > 1.5$ were found and grouped by field. The final number of objects satisfying this condition was found to be 7154. In Figure 3, the distributions of colors per field is shown in the left panel while the position in the color-color diagram is shown in the right panel. From Figure 3 (a) one can observe how for **Field-3** and **Field-2** the color J-H distribution is highly concentrated to values closer

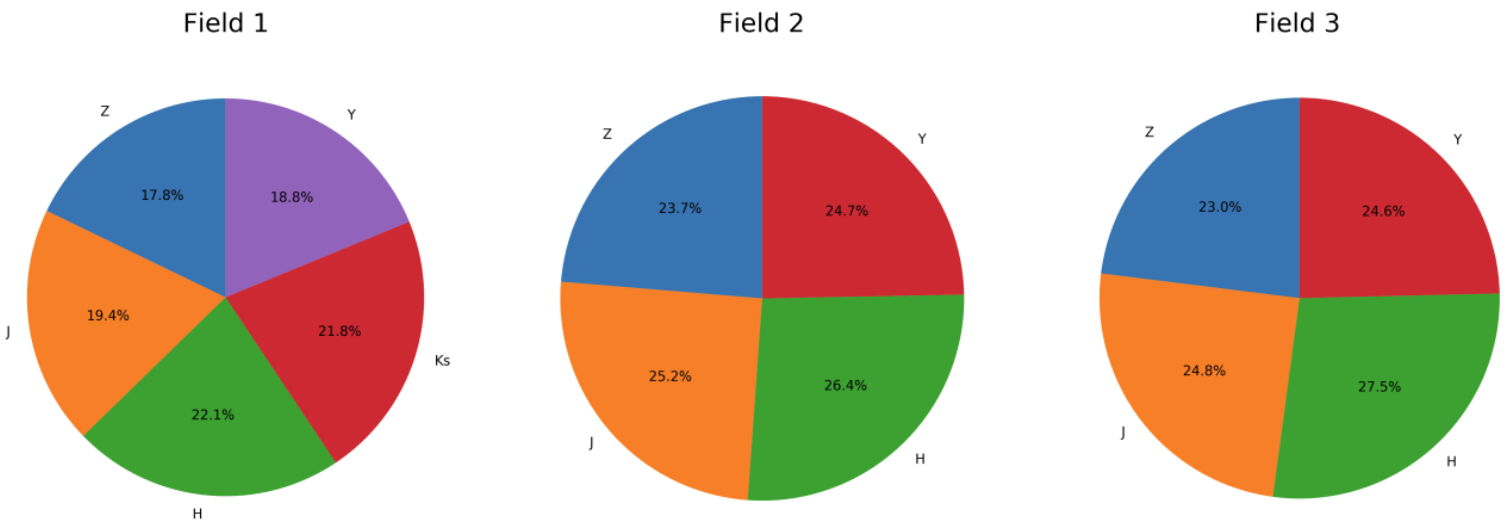

Figure 2: Pie diagrams corresponding to output of query **R1**. Each pie diagram is ordered by field and each color represents one of the five bands **Z**, **Y**, **J**, **H**, or **Ks**. The relative percentage respect the whole set of stars is also shown. In the case of **Field-1** the majority of stars are found to be in band **H** with 22.1%, while for **Field-2** and **Field-3** corresponds to 26.4% and 27.5% in the same band respectively. No stars were found to satisfy the conditions in the band **Ks** for fields 2 or 3

to 1.5 whilst for **Field-1** the distribution is more spread out to slightly higher values. In Figure 3 (b) the location of selected objects show no special location in color Y-J for these objects.

In query **R3** objects where the **Ks**-flux differs by a factor larger than 20 times, the flux uncertainty from the mean flux is retrieved. As shown in Section 9 or the notebook, the query was addressed in three steps, each one corresponding to each field. Pandas was also used to pass the query in order to save the data to perform the plot in a more friendly fashion. This query was a little bit confusing, however, I interpreted it as the whole bunch of objects where the difference between the flux in each **Ks** file for a given field and the average flux depending on whether the field has multiple **Ks** observations or not, was larger than 20 times the flux uncertainty in each **Ks** band file. I am not completely sure about the output for this query and one must be cautious in its interpretation. As I grouped by field StarID and performed three different queries, I was able to put all the data in a pie diagram which is shown in Figure 4. The majority of "variable" objects if I can call them like that, tend to lie in **Field-1** which is also the field with more catalogs in band **Ks**. However, as I am not completely sure about this query, I must refrain to make severe conclusion from the output data.

Finding the catalogs for a given field was addressed in query **R4**. The query is shown in Section 9 and the notebook. The output of this query was really easy because it is just the amount of files per field. So, for **Field-1** there are 7 catalogs, in case of **Field-2** there are 5 and for **Field-3** there are 6.

Retrieving the magnitudes for all bands (**Z**, **Y**, **J**, **H**, and **Ks**) where the signal-to-noise was larger than 30 in each band was addressed in the last query (**R5**). Once again, this query can be checked in Section 9 and the notebook. The summarized output is shown in Table 2. As a lot of stars are retrieved for each file, I decided to plot the histogram distribution of the objects for each field and filter. In first place, the histograms

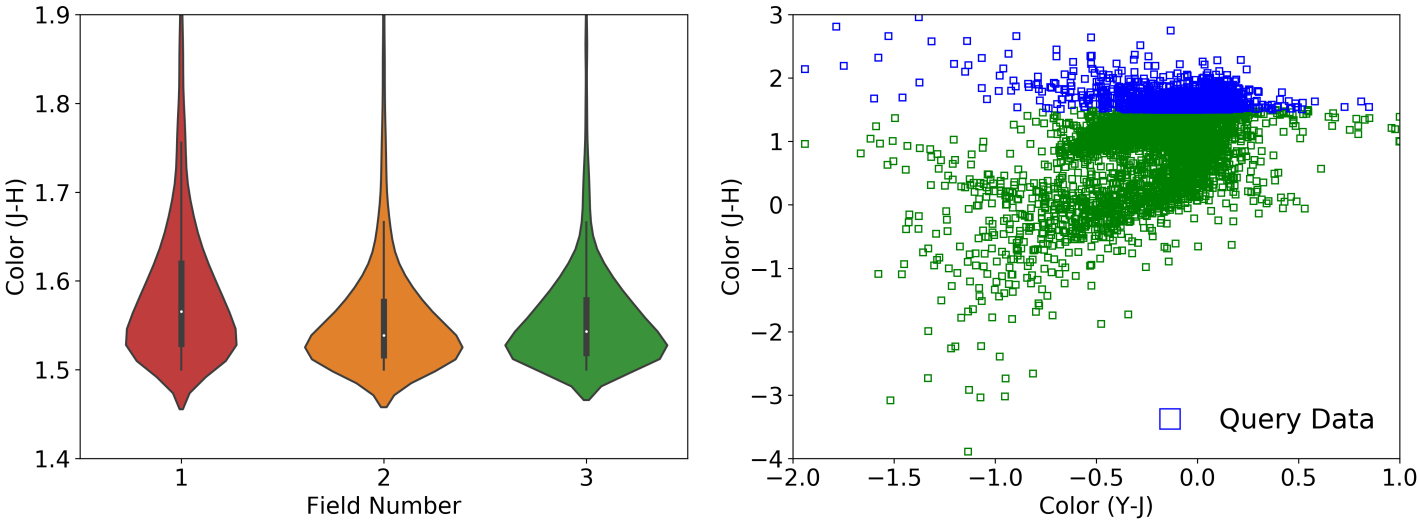

Figure 3: *left*: Violin diagram for query **R2** output. Each violin represents the individual distribution in color J-H for objects satisfying the condition. *Right*: Color-Color diagram. Additionally, the color Y-J was computed to show in which part of the color-color diagram this selected objects reside on. Blue squares represent the query output while green the whole set of original data.

were binned following the Scott's rule because the bin-width is proportional to the standard deviation of the data and inversely proportional to the cube root of the data size. It is not so robust to outliers but good enough to reproduce large datasets as the one we have. However, the figures looked overbinned so I decided to set the number of bins to 50 for better visualization. This is shown in figure 5. Each row represents **Field-1**, **Field-2**, and **Field-3** and each column corresponds to the different bands (**Z**, **Y**, **J**, **H**, and **Ks**). In the case of band **Ks** all the histograms are shown together if the field has more than one observation. All the fields show approximately the same magnitudes distribution in each band. However, one must note that for example in the case of bands **J**, **H**, and **Ks** a secondary peak pops up at lower magnitudes which may indicate a second different population of stars.

In the next Sub-section, the creation of a mock sample of stars for the Euclid mission from the actual color-color distribution of stars is addressed.

3.3 Euclid: Y-J and J-H Color Estimation

As an additional topic, it is well-known that the Euclid mission will have three filters corresponding to the bands **J**, **H**, and **Y**. Above, I created two different files with colors corresponding to J-H and Y-J taking into account the different fields. However, these tables were created without any assumption on the object class. For the simulation, only those objects classified as stars are needed. Thus, the first step consisted in guaranteeing each object to be classified as a star in the three filters and with values different from a null-number. From this, I ended up with a sample of $18,428$ stars which can be used to model the color-color distribution and lately create a new larger sample of stars. In figure 6 the color-color diagram and histograms for each color are shown. The histograms were created with a 50 and 65 binning, and also normalized in

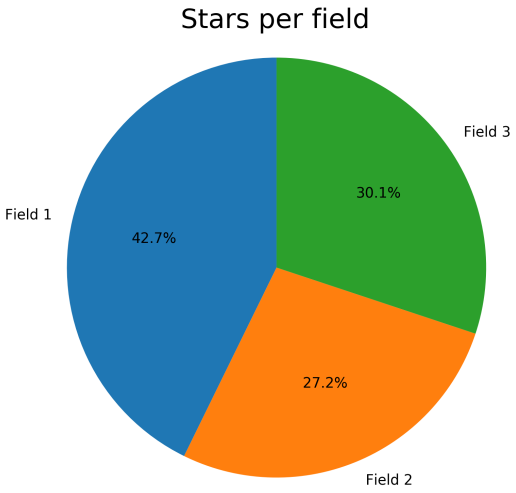

Figure 4: Pie diagram corresponding to output of query **R2**. Each color represents the different fields. The relative percentage of stars per field is shown. All the results corresponds to objects in the band **Ks**. The majority of objects which satisfy the condition and indirectly can be associated with variable objects is found to be in **Field-1**.

Table 2: Query **R5** output. The relevant column for the query corresponds to the Number of Stars with S/N $>$ 30 in each file. Other information was added to keep track of the output meaning.

Field	**Filter**	**Number of stars with S/N $>$ 30**
1	Z	3501
1	J	4636
1	H	6185
1	Ks	3948
1	Ks	4421
1	Ks	6421
1	Y	3986
2	Z	4432
2	J	5150
2	H	6048
2	Ks	3959
2	Y	4781
3	Z	4071
3	J	4918
3	H	6148
3	Ks	3899
3	Ks	4702
3	Y	4612

order to properly compare if the modeled distribution reproduces well the original dataset.

From Figure 6, one can see that most of the stars lie in a particular place of the Color-Color diagram. This is quite narrow for the J-H color with some spreading to the outskirts. However, for Y-J though the majority

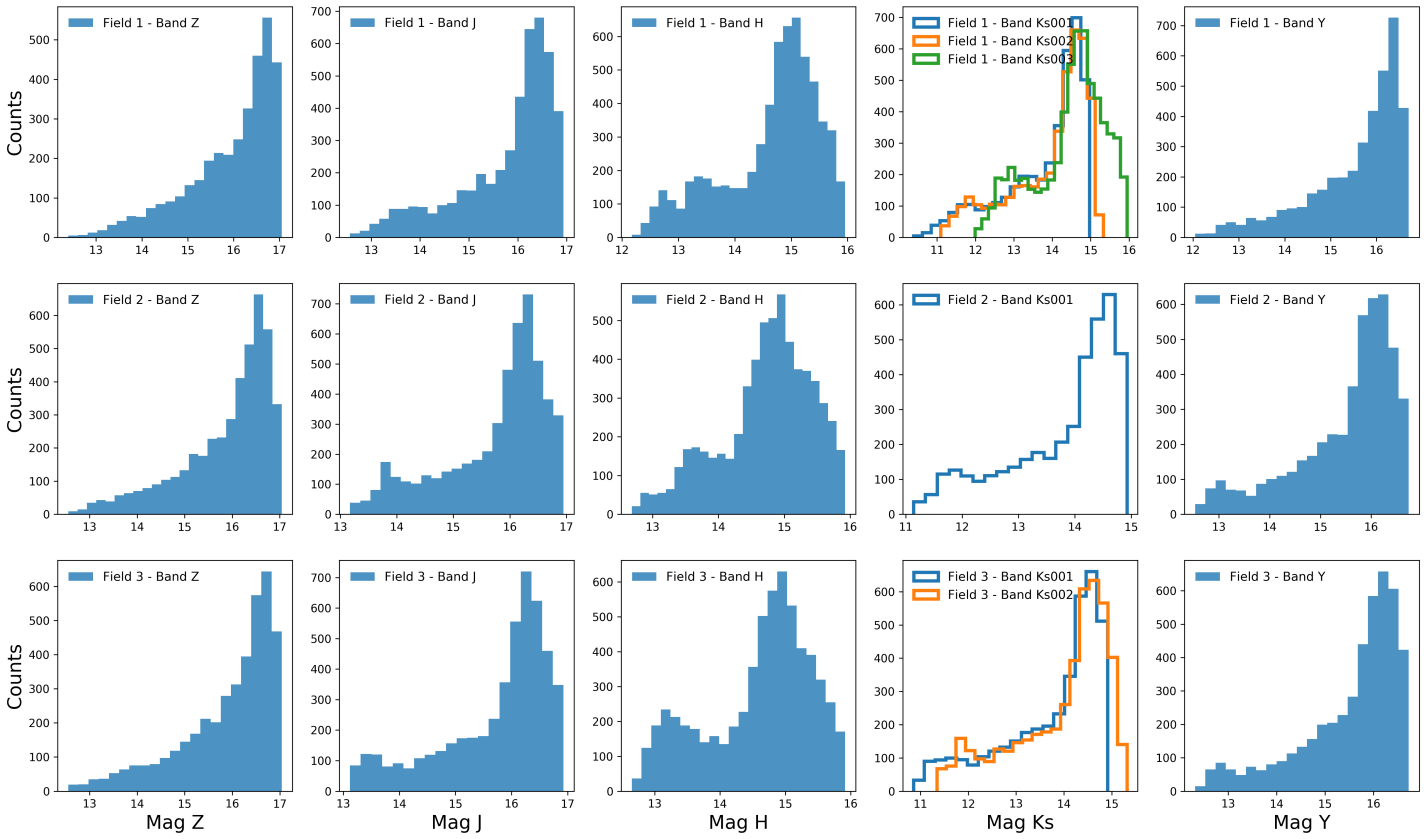

Figure 5: Histograms corresponding to output of query **R2**. Each row represents **Field-1**, **Field-2**, and **Field-3**. Each column corresponds to the different bands (**Z**, **Y**, **J**, **H**, and **Ks**). In the case of band **Ks** all the histograms are shown together if the field has more than one observation.

is still confined to a certain place, there is more spread, then I expect this color to be more complicated to reproduce by the density estimation models. As the main goal is to reproduce the density estimation to draw a new larger sample of stars following the same distribution in the J-H and Y-J plane, I opted to use two different approaches.

The first one is known as Kernel Density Estimation (KDE) and the second one called Gaussian Mixture Models (GMM). Both methods can be found as part of the **scikit-learn** Python library. First of all, the KDE is a neighbor-based approach which is used to learn a non-parametric generative model of a dataset to later generate efficiently new samples drawn from this model. Different metrics can be used with this model depending on the physical problem. In this case, I chose to use a regular flat-euclidean metric because of the Color-Color plane geometry. One important parameter to be known is the bandwidth which acts as a smoothing parameter, controlling the trade-off between bias and variance which is really important to faithfully reproduce the original dataset distribution. Large bandwidth means high-bias or very smooth density

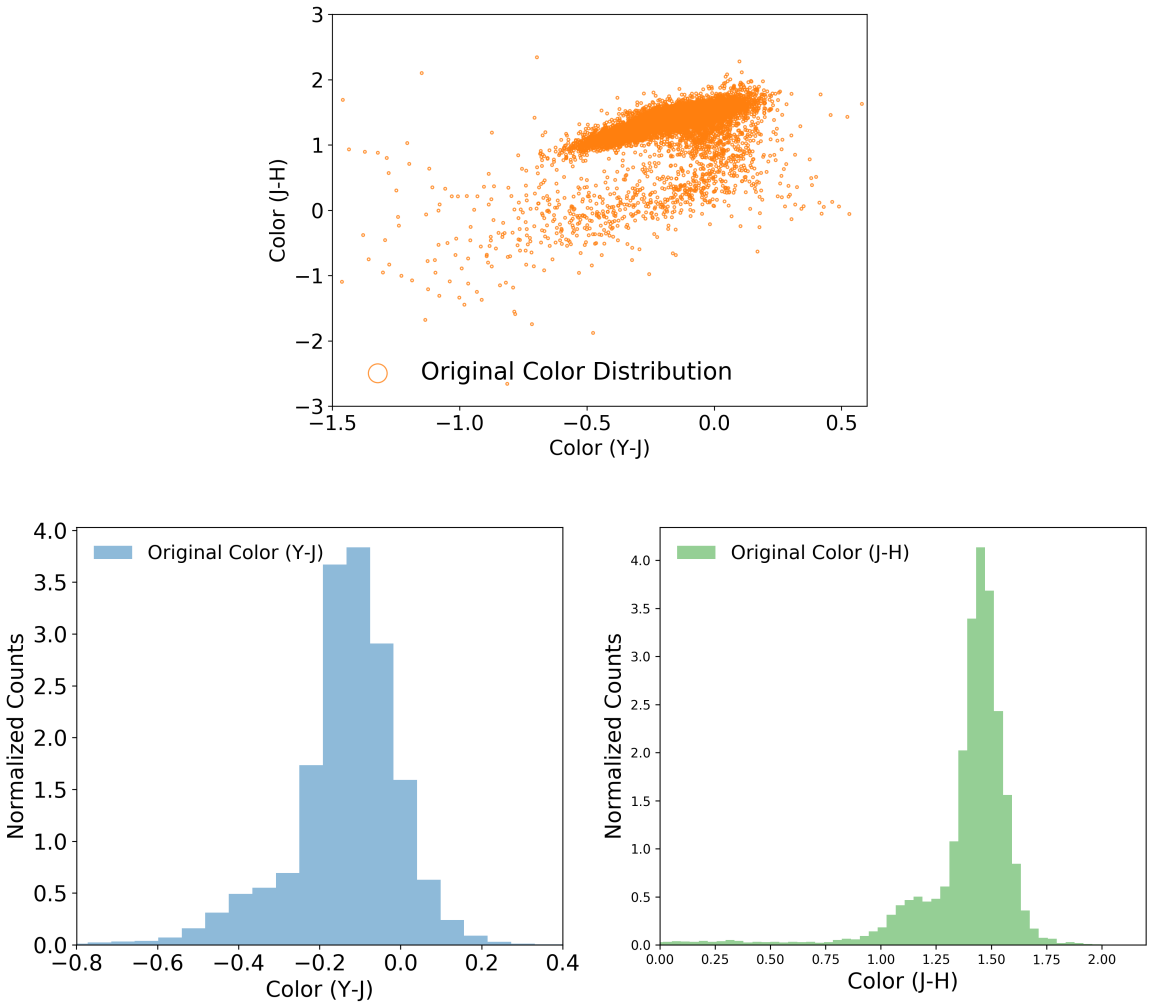

Figure 6: *top*: Color-Color diagram for J-H and Y-j. A sample of 18,428 stars were found to meet our requirements. *bottom*: Normalized histograms for Y-J and J-H. The distribution in J-H looks quite tight, while for Y-J is more spread out in a large range.

distribution, whilst small values lead to high-variance or unsmoothed density distribution (Pedregosa et al., 2011). In this case, I opted to find the optimal bandwidth using a function from the **scikit-learn** library in the Clustering routine where one needs to pass the data to be drawn and a quantile number which regulates the distance and amount of clusters for the estimation. In this case, was set to 0.01. The optimal bandwidth for this color dataset was found to be 0.046. Once this is fixed, I use the KDE passing the optimal bandwidth, and a Gaussian kernel because from the histograms drawn before we can see that our distributions follow a "Gaussian" profile. From this, I created a model and generated a new sample of 100,000 stars. In Figure 7, the color-color diagram and histograms for the original and new data sample are shown. As was mentioned before, the wide range in Y-J makes the estimation model spread out more for this color than in the J-H axis. This can be seen in the normalized histograms where the new distribution for J-H is reproduced quite good, except for some features around the peak, whilst for the Y-J the peak and the tails are not so well matched.

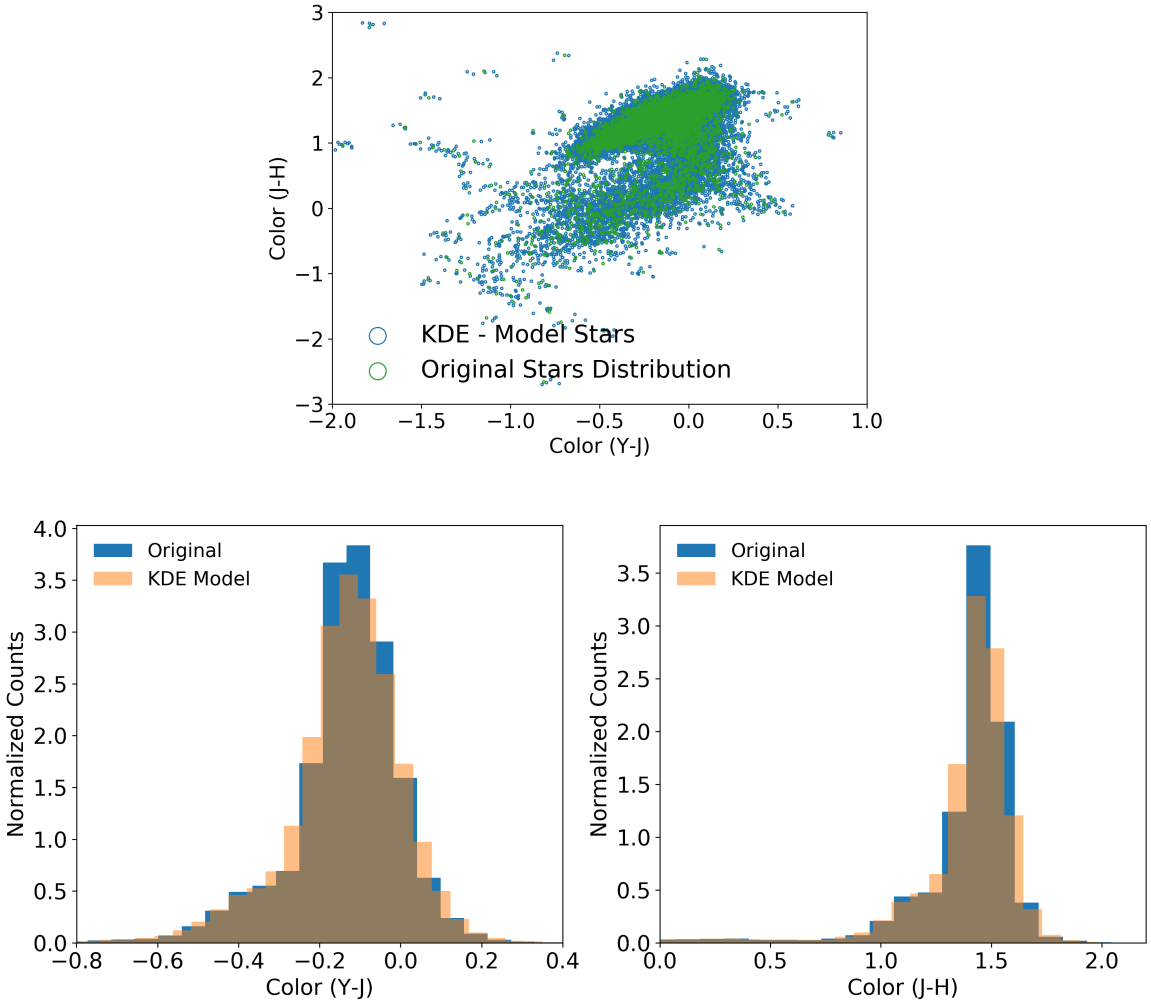

Figure 7: *top*: Color-Color diagram for J-H and Y-J. In green the original sample composed of 18,428 stars and in green a new sample of 100,000 stars drawn. *bottom*: Normalized histograms for Y-J and J-H, and both the new and original samples are shown.

The last model was quite good, but I was aiming to obtain a more accurate distribution for the new sample. I then decided to try Gaussian Mixture Models. This technique is also a density estimator quite useful as an unsupervised clustering scheme. This probabilistic model assumes that the sample data can be generated from a finite number of Gaussian distributions with unknown parameters. It is quite useful in our case because our distributions are Gaussian-like. Gaussian Mixture is the fastest algorithm for learning mixture models, maximizing the likelihood and not biasing the means towards zero (Pedregosa et al., 2011). Here, the main issue is to know which is the right amount of Gaussians which can reproduce faithfully our data sample. To address this, I decided to compute for a combination of 1 to 30 Gaussian models, the BIC and AIC retrieved from the internal model. AIC represents the relative distance to the unknown true likelihood function of the data and the fitted likelihood function of the model, so the lower the value the

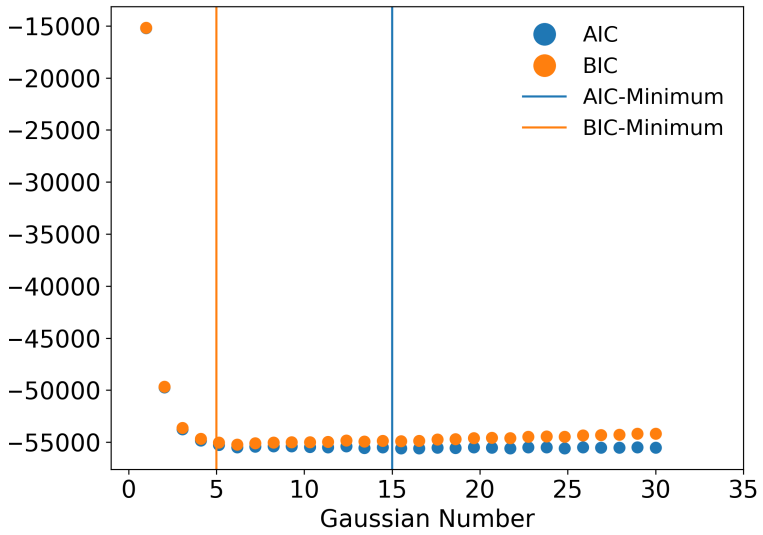

Figure 8: AIC and BIC criteria. Each point corresponds to different Gaussian models from 1 to 30 Gaussians combination. The vertical lines correspond to the minimum value achieved with each estimator setting the initial range of amount of Gaussians to be used in the model.

closer to the truth. BIC is an estimate of a function of the posterior probability of a model being the truth, under a certain Bayesian setup, so that a lower BIC, so the lower the BIC the closer to the true model. Both criteria are based on various assumptions and asymptotic approximations (The Methodology Center). It is useful to have in mind that AIC and BIC represent estimations on different statistics, thus, I ran the 30 models, computed the AIC and BIC, found the minimum value and decided to try all different combination of Gaussians number between them. I chose a model with 13 Gaussian because the optimal values according to this method which is pictured in Figure 8 tells me that the best model must be a combination of 5 to 15 Gaussians. Once this is tackled down, I proceed to apply the Gaussian Mixture Models with 13 components to my dataset and draw a new sample of $100,000$ stars from the estimated density distribution in the color-color diagram.

In Figure 9 the color-color diagram and the respective histograms are shown for comparison as was done before with the KDE-model. Here, one can see that the distribution of the new data (blue-dots) is highly concentrated around the original distribution (green-dots). In the case of J-H the distribution drawn from the new model reproduces very well the original sample as can be seen from the histogram and though for Y-J the distribution is still not as desired is way better than the one retrieved from the KDE-model. Then, I decided to keep the results from a Gaussian Mixture Models with 13 component as the model distribution of the density in the color-color diagram to be the one used to simulate what the Euclid mission would expect in the **J**, **H**, and **Y** bands.

4 Photometric Redshifts of Galaxies

Estimating distance to extra-galactic sources is one of the most important measurements in Astronomy. This can be estimated from their spectrum features as the lines will be shifted by relative movement of the

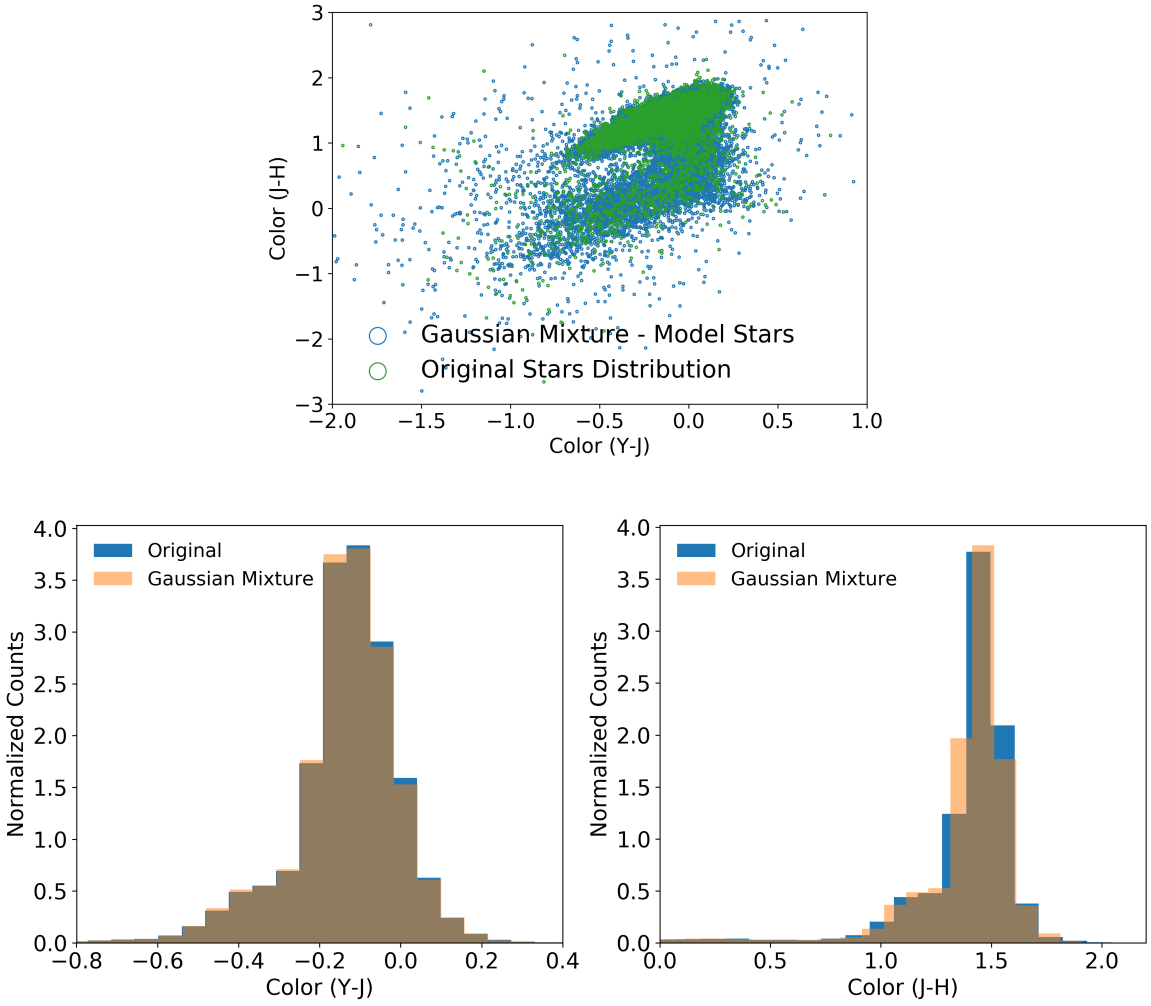

Figure 9: *top*: Color-Color diagram for J-H and Y-J. In green the original sample composed of $18,428$ stars and in green a new sample of $100,000$ stars drawn. *bottom*: Normalized histograms for Y-J and J-H, and both the new and original samples are shown.

source respect to us and one can predict its spectroscopic redshift based on that. Nevertheless, it is quite observational-time consuming and one needs to do this for several objects. Due to this, the photometric redshift technique is one of the most widely used which consists in predicting the distance using a combination of magnitudes or colors.

In the next sections, I will tackle this problem using different regressors on well-known colors and spectroscopic redshifts (see Section 2) of $74,309$ sources. First, a linear regression test was performed on dataset A to estimate the training error in the prediction of new redshifts from the original dataset. Later on, a second-file (dataset B) was used to estimate a generalization error which will reveal how accurate is our model to predict new redshifts on a totally different dataset. Finally, different non-linear regressors were used in order to reduce the training and generalization errors found with the linear regressors and improve

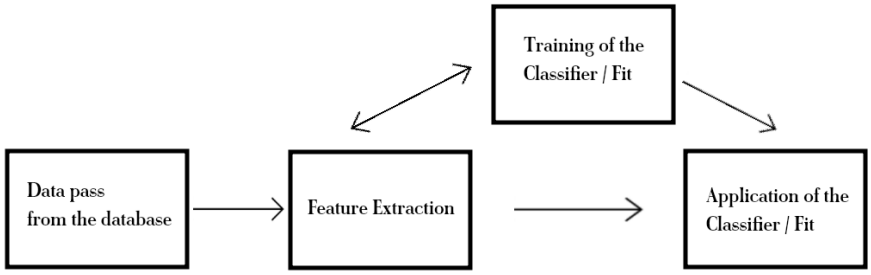

Figure 10: Flow scheme for feature extraction, training and application process.

our predictions on new photometric redshifts.

4.1 Feature Extraction

Feature extraction it refers basically to how one chooses the input data which will be passed to a training algorithm for a classification or fitting task, depending on what the user wants. This is one of the crucial parts of the Classifier/Fitting in data classification and data mining. It is strictly related to the scientific question and the main goal. For example, if one would like to classify/fit spectra, one would desire to use an input data emission and absorption lines to facilitate the training algorithm and reproduce faithfully the real spectral features. In the case of an image, one would prefer instead to use properties as magnitudes, colors, light concentration, light profiles, etc. The features are intended to be informative and non-redundant in order to facilitate the learning and generalization steps. This is strictly related to dimensionality reduction. When a model requires a large number of variables, then one needs to use a large amount of computer memory and training samples may be affected by overfitting resulting in a poor classification/fitting method. As a first step one can think about reducing the dimensionality to obtain the most important features in a given sample as part of the process of feature extraction. This features will be pass to the training algorithm which will classify/fit in a better fashion the new samples. Principal component analysis can be a good starting point to guide this choice.

In Figure 10 a flow diagram is sketched in order to broadly picture the feature extraction process. Initially one has a data sample which comes from a database and will be pass to a training algorithm. Not all the database is needed to be passed but maybe it is a wise choice to pass the most reliable features as stated before. Once this is done, one has two choices, one is to go directly to the application of this model to your dataset or send your features to the training algorithm and split it into sub-samples which can be compared back and forth with the features to lately apply them on the classifier/fit and obtain a more robust prediction. All this depends on the technique to be used and the physical problem one wants to predict. In Figure 11 there is more general flow diagram showing how the dimensionality reduction comes to play an important role in feature extraction and posterior classification/fitting.

In the next section, the linear regression method is applied and tested for different methods.

4.2 Linear Regressors: Linear Regression, Ridge and LASSO

For this specific problem, I used three different linear regression methods in the **scikit-learn** Python library which is: Linear Regression, Ridge and LASSO. As was said before, the dataset A has the magnitude in the

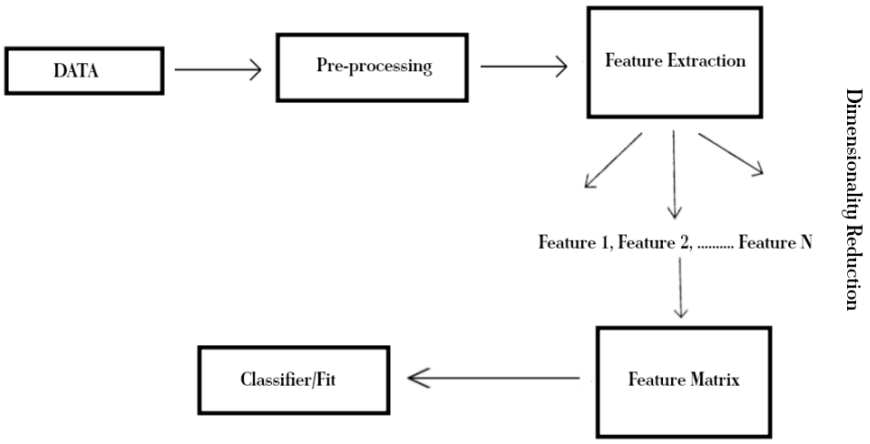

Figure 11: Flow scheme for feature extraction taking into account the dimensionality reduction.

r-band, the colors, and the spectroscopic redshift. First, I computed from the magnitude and color relations the rest of magnitudes because the model I plan to implement is based on the magnitudes. However, it is said that the discrepancy between using colors and magnitudes is not large so one can choose between these two models. The photometric redshift was then estimated using linear models following two different linear relations as presented in (Connolly et al., 1995) and (Gwyn, 2006). Equation 1 shows the linear combination of magnitudes (in our case u,g,r,i,z) where a_0 is the intercept and the sum is performed over the linear combination of magnitudes. In Equation 2 the quadratic combination in magnitudes is shown where a_0 is the intercept and the sum is performed over the linear combination of magnitudes and the second one over the possible combination of magnitudes.

$$Z_{phot} = a_0 + \sum_{i=0}^{N} a_i M_i \tag{1}$$

$$Z_{phot} = a_0 + \sum_{i=0}^{N} a_i M_i + \sum_{i=1}^{N} \sum_{j=i}^{N} a_{ij} M_i M_j \tag{2}$$

Differences between both models were not too large, so I decided to implement the linear model using the five magnitudes (see Equation 1). In the three approaches, the whole file was used to model and predict. However, there is an important parameter in the case of Ridge and LASSO regression which is known as the regularization strength. This value may allow to have different results and particularly in the case of LASSO a value of alpha = 0 leads to an ordinary least square and it is not advised for numerical reasons. The accuracy of the methods were evaluated according to the requirements of large cosmological surveys where the error in the predictions is expected to be less than 0.01 ($\langle|(Z_{phot} - Z_{spec})/(1 + Z_{spec})|\rangle < 0.01$). The training error which will then characterize and guarantee that a linear regressor is the better than other is given in Equation 3. I aimed to obtain a value close to 0.01 in the training error or lower. In the case of LASSO the parameters were poorly fitted and the training error resulted in $E(\theta) = 0.01944$ even using small values of the alpha parameter. For Ridge, the case was slightly better where the training error can be reduced to $E(\theta) = 0.01481$. However, the Linear Regressor gave me the best starting result with a training error of $E(\theta) = 0.01456$. All these values and procedure can be checked in the notebook. then, taking this

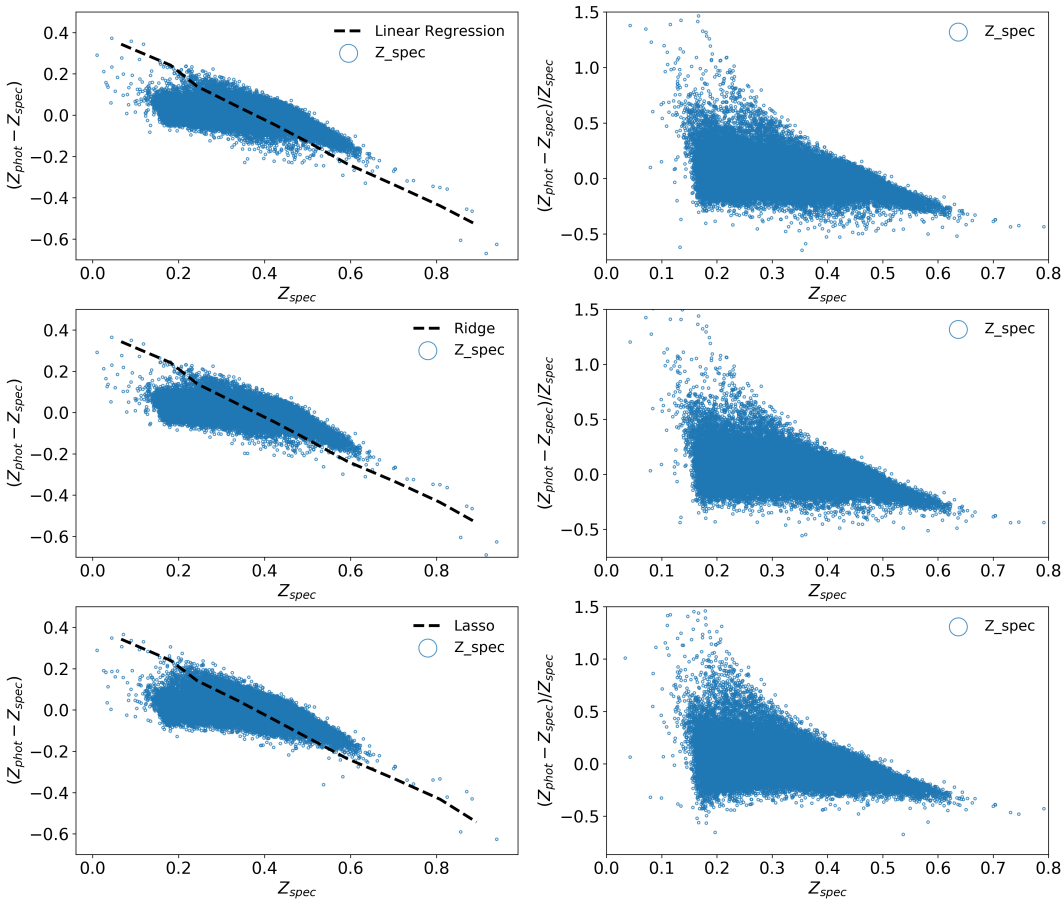

Figure 12: Linear regression methods from top to bottom: Linear Regression, Ridge, and LASSO. The first column corresponds to the difference between the photometric and spectroscopic redshift versus the spectroscopic redshift. The black line shows the binned mean value for both axis. The second column represents the residuals after performing any of the linear regression methods.

into account the model to use for later comparison will be the Linear Regressor. The fitted parameters are shown in Equation 4.

$$E(\theta) = \text{median}\left(\left|\frac{Z_{phot} - f(\theta)}{1 + Z_{spec}}\right|\right) \tag{3}$$

$$Z_{phot} = -0.783 + -0.013\text{M}_\text{u} + 0.096\text{M}_\text{g} + 0.372\text{M}_\text{r} + -0.395\text{M}_\text{r} + -0.021\text{M}_\text{z} \tag{4}$$

The same procedure was used to examine the model with quadratic terms and also combinations of magnitudes and colors but the differences were not large enough to take this into consideration. In the next section, a brief explanation of why the training error is not a reliable estimation of the general error of an estimator is explained and the error is also generalized using file B.

4.3 Generalization Error

In the last section, I computed the training error based on how well the Linear Regression method performed on a dataset A. However, it is necessary to generalize the error because one cannot use the same file to model and then predict over the same bunch of data. Here, the generalization error concept must be introduced. The generalization error is a measure of how accurately an algorithm used for classification or fitting is able to predict the outcome values. The learning algorithm is evaluated on finite samples, then it may be sensitive to sampling error. Thus, measurements of prediction error on current data as was performed in the last section with dataset A may not provide much information about the predictive ability on new dataset. It is important then to know how good is the model one has created to predict a different dataset. The error defined in Equation 3 weights the error equally. However, the generalized error must weight differently those pairs that correspond to higher joint probability and less those which rarely occur. Thus, the generalization error will be simply the same as in Equation 3 but the true value will be those values of a different dataset (dataset B) and the predicted ones will correspond to the ones obtained after applying the model computed with dataset A on B. In this way, the accuracy of our model is better estimated.

In order to do this the first approach consisted in applying the Linear Regression method on dataset A to create a model and predict over dataset A to obtain a training error (basically what was done in Sub-Section 4.2). After the model is created I use the model to predict on dataset B and compute the error. In this case, I decided to introduce a weight function in the Linear Regression method inverse to the redshift aiming to weight differently high redshift values which can be less accurate than the smaller one. However, the change in the training error is not so significant. The training error was found to be $E(\theta)_T = 0.01433$ and the generalization error $E(\theta)_G = 0.01441$. The model seems to predict very well the new redshifts on the new dataset because the generalization error is just slowly higher than the training as is expected. However, we were aiming to obtain a smaller value for the training error in order to predict better on different datasets. Then, the method used to achieve this was performing a Principal Component Analysis in order to transform the data files in a more suitable way which can allow us to predict in an easier and more accurate way the redshift. However, after different experiments, it was impossible to reduce the training error for dataset A which was the main goal of the exercise. The implementation of the PCA method is shown in the notebook. What I want to achieve with this as explain in Sub-Section 4.1 is to project data into the most dominant components of the data to somehow reduce dimensionality. Nevertheless, in this case, I did not reduce the components, because the main goal was just to transform data and not throw it away. Both files were passed to the PCA routine and then used in the regular Linear Regression model as explained above to compute new training and generalization errors. Results turn out to be the same with or without PCA due to corrupted files. In Figure 13 the predicted photometric redshift and actual spectroscopic redshift are shown after performing Linear Regression on the PCA new files A and B. The black line corresponds to a one-to-one relation just to compare how tight or spread is the distribution and this will be useful when comparing more elaborated methods in the upcoming Sub-Section.

The next section will be dedicated to exploring different methods besides from the linear regressors to achieve a smaller generalization error than the ones derived before.

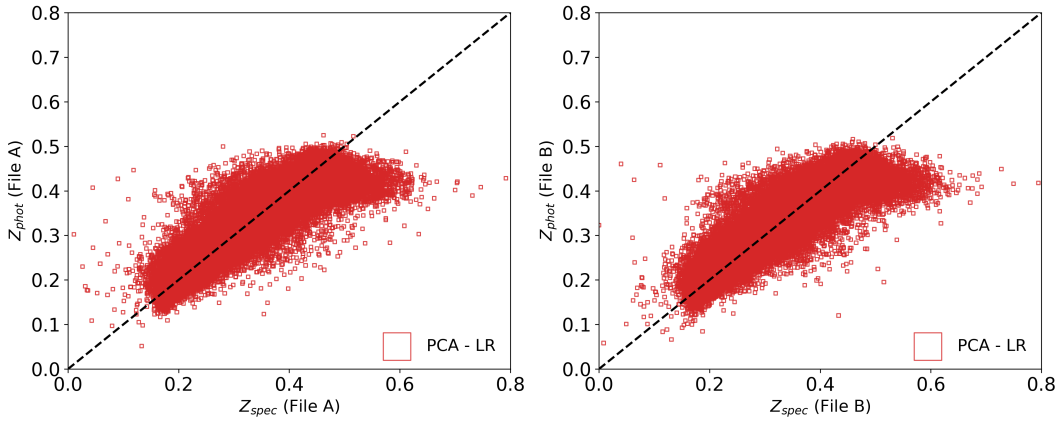

Figure 13

4.4 Different Regressors and Comparison

After implementing the Linear Regression to predict the photometric redshifts, the most common and natural question is: Can you obtain a lower generalization and training error for the same datasets using a different estimator?. This was addressed using three main estimators corresponding to Random Forest, Gaussian Processes, and Bagging Regressor. Nevertheless, K-Nearest Neighbors, ADABoosting, and Gradient Boosting were also explored, but will not be explored in detail.

In the case of **Gaussian Processes**, this was proposed mainly because, in (Bonfield et al., 2010), they suggest that *GP* regression has desirable properties for photometric redshift estimation, particularly for deep, high-redshift surveys where it is difficult to obtain a large, complete training set. The Gaussian Processes method consists in a generic supervised learning method., which predicts in a probabilistic fashion (Gaussian) allowing to compute empirical confidence intervals and decide based on those if one should refit the prediction in a given region of interest. Different kernels can also be specified, in this case, RBF(0.1). However, the whole samples/features information is used to predict which makes the implementation fairly slow. It is also useful to have in mind that the efficiency in high dimensional spaces, namely when the number of features exceeds a few dozens, this method loses its efficiency. The hyperparameters of the kernel are optimized during the fitting maximizing the log-marginal-likelihood (LML). This can be controlled with "n-restarts-optimizer" which was set to 3 in this particular case.
As was stated before, this method is computational and time expensive, thus two simple tests were performed taking into account the availability of computer power I have access to. The first one was addressed using only 3% of the dataset A to create the model and then apply it to predict on A and B. This is shown in Figure 14. The training error and generalization errors obtained were 0.01914 and 0.01994 respectively. A second trial was performed using another machine but only with 50% of the dataset A which led to the results shown in 15 and a training error of 0.00826 and generalization error of 0.01931. The improvement in the training error is notable which gives a tighter relation for the predicted redshifts compared to the spectroscopic ones, even so, the generalization error does not improve that much. It would be really nice to test this method with all the data sample instead of sub-samples to compare it with other estimators used here.

Although one would expect to obtain better results based on the Gaussian Processes Regressor, the time

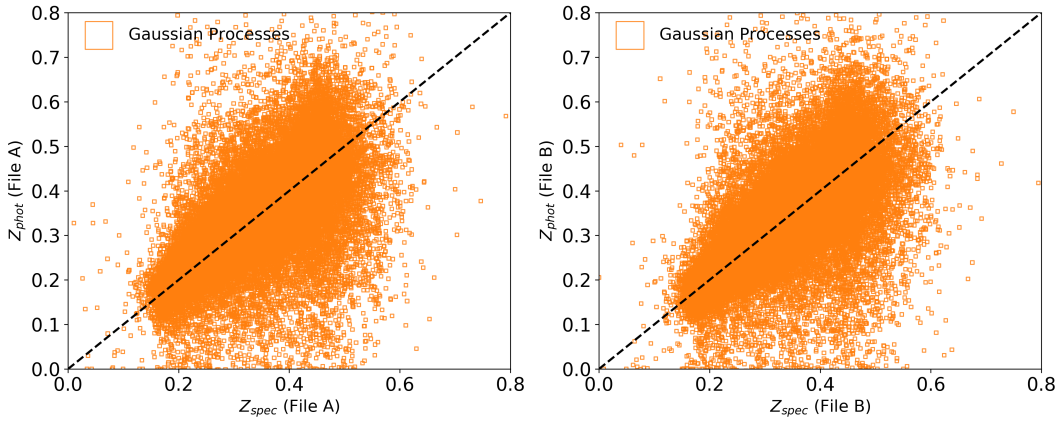

Figure 14: Gaussian Processes Regressor. The left panel show the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-A*. The right panel shows the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-B*. The black line is a one-to-one relation just for comparison of how tight or spread is the relation between the original data set and the model.

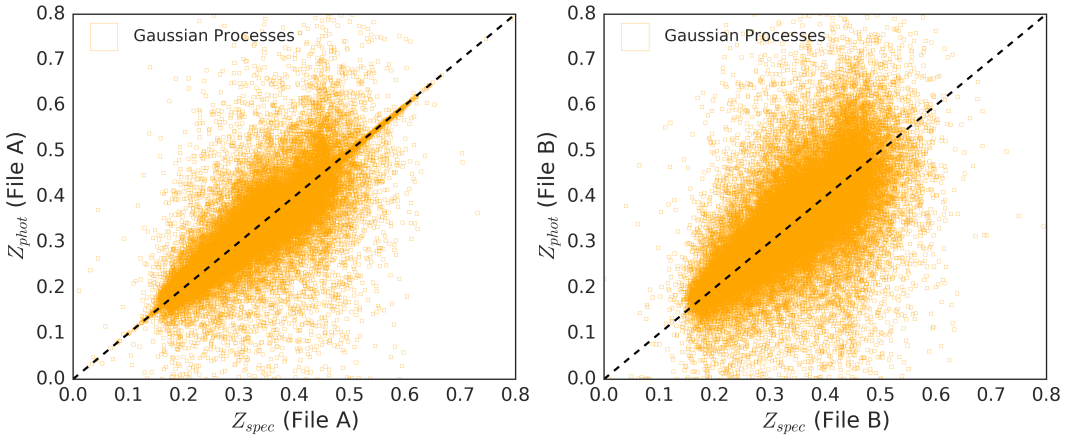

Figure 15: Gaussian Processes Regressor with a different input dataset. The left panel show the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-A*. The right panel shows the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-B*. The black line is a one-to-one relation just for comparison of how tight or spread is the relation between the original data set and the model.

it takes to execute does not allow me to make severe statements about the outcome. Thus, the next method to be used was **Random Forest Regressor**. *RF* fits a number of decision trees on various sub-samples of the dataset and uses the average of them to improve the predictive accuracy, controlling over-fitting and reducing the variance. The averaged prediction of the individual classifiers is the prediction of the ensemble. Each tree is built from a sample drawn with replacement from the training set when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features (Pedregosa et al., 2011). As the name suggests, this process is random, leading to an increase in the bias which is not too large and largely

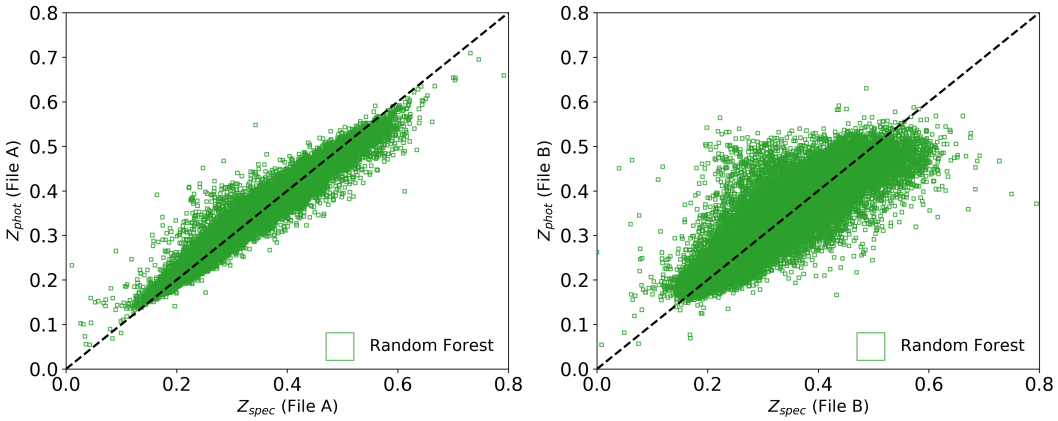

Figure 16: Random Forest Regressor. The left panel show the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-A*. The right panel shows the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-B*. The black line is a one-to-one relation just for comparison of how tight or spread is the relation between the original data set and the model.

compensated by the improvement in the variance, yielding to an overall better model estimation. The main parameters used by this routine are the "n-estimators" and "max-features" which will control the number of random walkers (the number of trees in the forest) and the sub-samples drawn from the original dataset. The number of estimators is suggested to be taken as large as possible because one the routine converges it will stop independently of the chosen number. In my case, I just set the number of trees in the forest to 10. The training error and generalization error obtain with this implementation are 0.00464 and 0.01285 respectively. These values are quite small compared to the ones obtained before, and show how good and fast this method can be to interpret, analyze and draw new samples and perform fitting to our original dataset. In figure 16, both the predicted and measured redshifts are shown for both files. One can compare this figure with the one obtained using Gaussian Processes and realize how accurate is the estimation of new redshifts just by looking at the spread along the black line which represents the one-to-one relation. This is expected as the training and generalization errors obtained were quite small which was the main goal of this test.

Even though this method was good enough to reduce the generalization error, and the main goal was achieved, I decided to test another method called **Bagging Regressor**. This method is an ensemble meta-estimator which also fits on random subsets of the original dataset and then aggregates individual predictions voting or averaging, to form a final prediction. It is quite similar to Random Forest because it also tries to reduce the variance of a black-box estimator as could be a decision tree by using randomization and then building up an ensemble out of it. However, the samples here are drawn with replacement, that is why this method is called bagging. In many cases, bagging methods constitute a very simple way to improve with respect to a single model, without making it necessary to adapt the underlying base algorithm (Pedregosa et al., 2011). The parameters used here were, the number of estimators (set to 20) once again which controls the number of trees in the forest, and "max-samples" and "max-features" which controls the number of sub-sets and features to be used in the prediction. In this case, the values obtained for the errors were really close to those in the Random Forest Regressor corresponding to a training error of 0.00554 and a generalization error of 0.01251. Which once again, satisfy the condition imposed above over reducing the generalization error. The photometric redshift (predicted) and spectroscopic redshift (real value) are shown in Figure 17

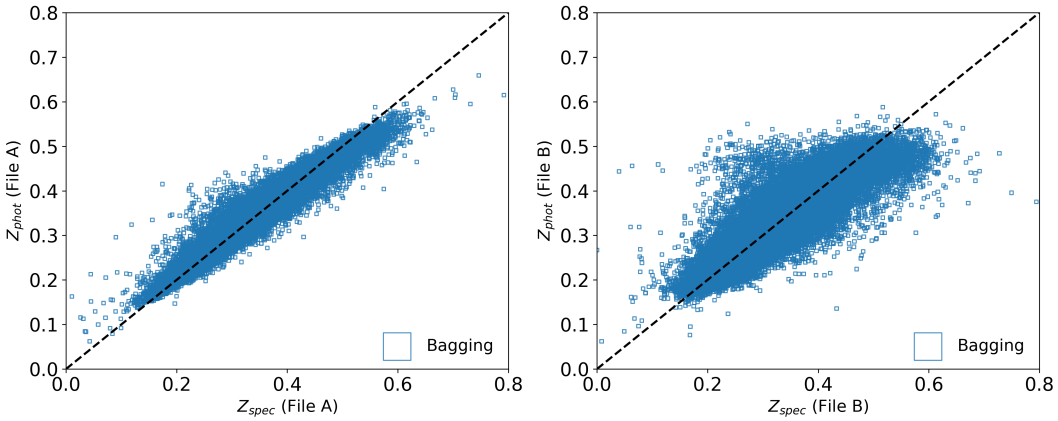

Figure 17: Bagging Regressor. The left panel show the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-A*. The right panel shows the photometric redshift computed using the GP method versus the spectroscopic redshift for *File-B*. The black line is a one-to-one relation just for comparison of how tight or spread is the relation between the original data set and the model

for both files. As expected from the small values of the errors, the prediction is way better than the one obtained only using a Linear Regressor and quite similar to the Random Forest estimation.

Finally, I tried other different methods as ADABoost, KNN and Gradient Boosting. Nevertheless, most of the results did not vary that much, so I just decided to stick with the last three methods explained above which I considered was the more important and reliable based on the information I found in different documentation and also the nature of the physical problem. In the last Sub-Section I will address the Pros and Cons of the different methods used before and a summary of the most important features/values found here.

4.5 Regressors: Pros and Cons

In last Sub-Sections, I discussed some useful methods tested in order to lower down the generalization error. Even so, all of them have their pros and cons. In Table 3 I summarize the main features which make a task suitable or not to be used and also the training and generalization error obtained when applied to our datasets.

This table basically summarizes what I have performed to achieve as the final results in the last Sub-Section. The different methods used show a decreasing training and generalization error when the random methods are used. In spite of this, the Linear Regressor is the simplest model and allows us to obtain a really good estimate of the errors and a good prediction of the redshifts, at least to start with, and later refine with more elaborated regressors. All different methods have pros and cons and one must be aware of them when applying them to a certain physical problem. Learning the theory behind, exploring the routines, comparing the outcomes is vital in order to reliably reproduce a particular feature, either using estimators or fitting.

Table 3: Different regression methods testes. The training and generalization error obtained is summarized. The pros and cons for each method is listed. ((Pros and Cons), (Regression Pros and Cons))

Regression Method	Training Error	Generalization Error	Pros	Cons
Linear Regressor	0.01456	—	Extremely simple method. Really good if data follows a linear trend. It is nice as a first approach when a data is unknown to learn how to play with the parameters.	It can impose rather strong constraints on the solution space adding more bias than reducing variance. Oversimplifies real life problems which are mainly non-linear.
LASSO	0.01944	—	Useful to fit linear models. Solves a lot of problems which may pop-up using regular linear regression. Better than usual automatic selection methods.	It can produce models that make no sense. It ignores non-significant variables which may be important.
Ridge	0.01481	—	Can reduce the variance, though increases the bias. Can improve predictive performance. Mathematically simple.	Not able to shrink coefficients to exactly zero. Cannot perform variable selection.
Linear + PCA	0.01433	0.01441	Projection method. Allows the extraction of information and remove noise. Reduces dimensionality / Compression	Throws away data which could be reliable. Depends on feature size and scale.
Gaussian Processes	0.01914 0.00826	0.01994 0.01931	Predicts in a probabilistic fashion. Very neat way to tune hyper-parameters by maximizing the marginal likelihood. Tends to consistently give very good fits without any need for cross-validation.	Most people use it without understanding the theory behind which is really import. It is not a descriptive tool. It takes longer to predict and fit.
Random Forest	0.00464	0.01285	Decorrelates trees (relative to bagged trees). Reduced variance. It runs efficiently on large databases.	If the trees get lost, it is complicated to interpolate and extrapolate. Not as easy to visually interpret. If the data contain groups of correlated features of similar relevance for the output, then smaller groups are favored over larger groups
Bagging	0.00554	0.01251	Reduces variance in comparison to regular decision trees. Can provide variable importance measures. Can easily handle qualitative (categorical) features	Not as easy to visually interpret. Does not reduce variance if the features are correlated.

5 Discussion

As was discussed in all the sections above, the main goal of this project was to tackle down the creation of a database to follow up ma particular survey specifications and the accurate estimation of photometric redshifts for extra-galactic sources using linear and non-linear regression. Both topics are part of forefront problems of current Astronomy. Designing suitable databases to deal in a good manner with all the new data from different surveys is quite a challenge that we as Astronomer need to be aware of, and also need to imagine new ways to cleverly store, analyze and manipulate this information. An Astronomer is not just a vessel which analyses the data but also a main character in charge of dealing with the way to properly store it and distribute it to their peers.

On the other hand, estimating the redshifts to extra-galactic sources as explained above is one of the main tasks which constitutes an important of the building block of the modern Cosmology. A lot of Cosmology parameters and calculations reside on how well distances are measured. Then, having in mind and knowing

the difficulties of computing this may allow Astronomers in someway to wonder about a better idea to predict the real distributions of distances in our universe using the powerful tools of computational sciences along with the Astronomy and Physics knowledge.

This project reflects how different, and easy tools are useful to tackle down modern questions in Astronomy if are well used. Designing a database, performing queries, checking the outputs, having in mind that different people will make use of this makes you think in clever ways to make research life easier for your peers. Also, acknowledging that statistical and computational tools can help you out to better analyze regular tasks which may lead to useful techniques, in some way allows the Astronomy community to grow researching speaking, in a broad way to a better future.

6 Conclusions

Data corresponding to a survey in five different bands (**Z**, **Y**, **J**, **H**, and **Ks**) was used to create a suitable database and solve different queries using mainly SQL, which were aimed to facilitate the track of different parameters and image information for a mock survey. The final database was decided to contain two tables, one with all the information from each field and band, and another one with the observation information. The database schema can be modified automatically if a new user wants to mold it to their preferences. Colors for the different objects are computed separately allowing the use of this data in an easy fashion to compute a mock sample of new stars (100,000) to be used in a simulation for the Euclid mission.
Gaussian Mixture and Kernel Density estimators were used to reproduce the density distribution in the color-color plane (J-H, Y-J). The best model was achieved using 13 Gaussian combination based on different tests performed. However, the KDE is not that bad and can also be kept as a good starting point to explore the data.

For the second part, Linear and Non-Linear regression methods were applied to two different datasets A and B in order to predict the photometric redshift for $74,309$ objects. the linear regression method, the simplest one was really useful to explore data and realized that there was a strong linear correlation between the redshift and the magnitudes which were obtained from the colors in the original dataset following common models found in the literature. This allowed me to compute a training error. The model created with dataset A was applied on dataset B to generalize the error and see if the created model was good enough to predict the redshift in a totally different dataset. The generalization error and also the training error was reduced significantly while using different methods as Random Forest and Bagging, which was the main goal of the project. However, as was mentioned, in literature the Gaussian Processes method is widely used and seems to lead to smaller errors. In this case, due to computational limitations, the whole power of the model was not tested so one cannot fairly compare this with the latest models.

The Linear Regression method was combined with PCA in order to lower down the error before applying more sophisticated methods. Nevertheless, the PCA using all the component, not dropping out any information, but just transforming the data was not successful due to corrupted files. Still, this approach could be really useful before tackling down a problem known to be linear and to have a prior estimation of the behavior of the dataset instead of using other methods which cannot fairly describe the physical problem.

In general the creation of databases, the manipulation of data information, the creation of new models from known density distributions and the prediction of new samples from known trends as the case of photometric

redshifts are really useful in Astronomy and every Astronomer should be able to use this techniques and tools to properly answer current questions in Astrophysics. Understanding the theory and mathematics behind each process and not its application it is vital in order to properly apply a method to solve a given problem. The interpretation of different results under the light of mathematical and physical theories are really important to properly connect the computational science techniques and the physical approaches.

7 Acknowledgments

I would like to thank my friends Guadalupe Cañas, Esmee Stoop, Louis Martin, Felipe Ramos and Charlotte Brand for the useful discussions and comments, and acknowledge support along the development of this project. All the extensive conversations which helped somehow to shed a light on different parts of this work and aimed to improve it.

8 References

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. *Monthly Notices of the Royal Astronomical Society*, 405(2):987–994, 2010. doi: 10.1111/j.1365-2966.2010.16544.x. URL +http://dx.doi.org/10.1111/j.1365-2966.2010.16544.x.

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. URL +http://adsabs.harvard.edu/abs/1995AJ....110.2655C.

Final Project Guide. A database for a time-domain survey and photometric redshifts of galaxies. 2017.

Stephen D.J. Gwyn. Photometric Redshifts. *NED*, 2006. URL +https://ned.ipac.caltech.edu/level5/Glossary/Essay_photredshifts.html.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pros and Cons. Classification model pros and cons. URL +https://github.com/ctufts/Cheat_Sheets/wiki/Classification-Model-Pros-and-Cons.

Regression Pros and Cons. Regressions (pros and cons). URL +https://www.quora.com/.

The Methodology Center. AIC vs. BIC. *PennState*. URL +https://methodology.psu.edu/AIC-vs-BIC.

9 Appendix

9.1 Observations table information

ID	FieldID	Filename	Filter	MJD	Airmass	Exptime
1	1	Z-ADP.2017-01-18T11:58:36.905.fits	Z	57267.1671072	1.6405	40.0
2	1	J-ADP.2017-01-18T11:58:35.781.fits	J	57257.0504323	1.0105	48.0
3	1	H-ADP.2017-01-18T11:58:35.780.fits	H	57257.044108	1.006	16.0
4	1	Ks-ADP.2016-05-25T15:33:39.546.fits	Ks	56788.346937	1.023	16.0
5	1	Ks-ADP.2017-01-18T11:58:39.907.fits	Ks	56561.0020158	1.079	16.0
6	1	Ks-ADP.2016-05-25T15:33:43.377.fits	Ks	56829.0390512	1.6085	16.0
7	1	Y-ADP.2017-01-18T11:58:36.901.fits	Y	57267.1596647	1.5605	40.0
8	2	Z-ADP.2017-01-18T11:58:36.905b.fits	Z	57268.1671072	1.6405	40.0
9	2	J-ADP.2017-01-18T11:58:35.781b.fits	J	57258.0504323	1.0105	48.0
10	2	H-ADP.2017-01-18T11:58:35.780b.fits	H	57258.044108	1.006	16.0
11	2	Ks-ADP.2016-05-25T15:33:39.546b.fits	Ks	56789.346937	1.023	16.0
12	2	Y-ADP.2017-01-18T11:58:36.901b.fits	Y	57268.1596647	1.5605	40.0
13	3	Z-ADP.2017-01-18T11:58:36.905c.fits	Z	57268.1671072	1.6405	40.0
14	3	J-ADP.2017-01-18T11:58:35.781c.fits	J	57258.0504323	1.0105	48.0
15	3	H-ADP.2017-01-18T11:58:35.780c.fits	H	57258.044108	1.006	16.0
16	3	Ks-ADP.2016-05-25T15:33:39.546c.fits	Ks	56789.346937	1.023	16.0
17	3	Ks-ADP.2017-01-18T11:58:39.907c.fits	Ks	56562.0020158	1.079	16.0
18	3	Y-ADP.2017-01-18T11:58:36.901c.fits	Y	57268.1596647	1.5605	40.0

9.2 Queries

```
"""Query R1"""

Rows_1 = con.execute("""SELECT o.ID, o.FieldID, o.MJD, o.Filter, o.Filename, COUNT(s.StarID)
    \
FROM Stars as s LEFT JOIN Observations as o ON (o.ID = s.ID AND 1.0/s.dMag1 > 5 AND s.Class
    = -1) \
WHERE (o.MJD > 56800 AND o.MJD < 57300) \
GROUP BY o.filename ORDER BY o.ID""")
```

```
"""Query R2"""

Color_JH = pd.read_csv('Q1/Tables/Color_JH.csv')
Color_YJ = pd.read_csv('Q1/Tables/Color_YJ.csv')

ColorJH = []
ColorYJ = []
Field   = []

for i in range(len(Color_JH)):
    if Color_JH['Color_JH'][i] > 1.5:
        ColorJH.append(Color_JH['Color_JH'][i])
        ColorYJ.append(Color_YJ['Color_YJ'][i])
        Field.append(Color_JH['ID'][i])

print("The number of objects with color J-H > 1.5 is: %g"%len(ColorJH))
```

```
"""Query R3"""

ROW_3_1 = """SELECT s.StarID FROM Stars as s LEFT JOIN Observations as o ON (o.ID == s.ID) \
WHERE o.Filter == 'Ks' AND o.FieldID == 1 AND \
ABS(s.Flux1 - (SELECT AVG(ss.Flux1) FROM Stars as ss LEFT JOIN Observations as oo \
WHERE (oo.Filter = 'Ks' AND ss.FieldID == oo.FieldID) GROUP BY ss.StarID) ) > ABS(20*s.
    dFlux1) GROUP BY s.StarID
"""

ROW_3_2 = """SELECT s.StarID FROM Stars as s LEFT JOIN Observations as o ON (o.ID == s.ID) \
WHERE o.Filter == 'Ks' AND o.FieldID == 2 AND \
ABS(s.Flux1 - (SELECT AVG(ss.Flux1) FROM Stars as ss LEFT JOIN Observations as oo \
WHERE (oo.Filter = 'Ks' AND ss.FieldID == oo.FieldID) GROUP BY ss.StarID) ) > ABS(20*s.
    dFlux1) GROUP BY s.StarID
"""

ROW_3_3 = """SELECT s.StarID FROM Stars as s LEFT JOIN Observations as o ON (o.ID == s.ID) \
WHERE o.Filter == 'Ks' AND o.FieldID == 3 AND \
ABS(s.Flux1 - (SELECT AVG(ss.Flux1) FROM Stars as ss LEFT JOIN Observations as oo \
WHERE (oo.Filter = 'Ks' AND ss.FieldID == oo.FieldID) GROUP BY ss.StarID) ) > ABS(20*s.
    dFlux1) GROUP BY s.StarID
"""

Field_1 = pd.read_sql_query(ROW_3_1, con)
Field_2 = pd.read_sql_query(ROW_3_2, con)
Field_3 = pd.read_sql_query(ROW_3_3, con)
```

```
"""Query R4"""

Rows_4 = con.execute("""SELECT NCat, Field FROM (SELECT o.FieldID as Field, \
o.ID, o.FieldID, COUNT(o.ID) as NCat \
FROM Observations as o GROUP BY o.FieldID)""")
```

```
"""Query R5"""

Rows_5 = con.execute("""SELECT o.FieldID, o.Filter, COUNT(s.StarID) \
FROM Stars as s LEFT JOIN Observations as o ON (o.ID = s.ID AND 1.0/s.dMag1 > 30 AND s.Class
    = -1) \
WHERE (o.ID > 0) \
GROUP BY o.filename ORDER BY o.ID""")
```