Databases and Data Mining in Astronomy

Esmee Stoop (1355880)

Introduction

In the past years, data is intensively used in Astronomy. The amounts and demands of data will most certainly increase in the upcoming years. In order to use big databases, such as the Gaia Catalogue (Gaia Collaboration et al., 2016a,b), a smart approach needs to be taken in structuring such a database. A database should be able to be user-friendly, easy to retrieve data, and tables should be correctly connected. And, of course, there should always be the possibility of easily extending the database with more data.

With the ability to use these big amounts of data, comes the possibility of more research. In order to handle the data, especially when using big datasets, smart and fast algorithms need to be utilised. New algorithms used in machine learning as well as classic methods, such as Linear Regression, provide the tools necessary for handling extensive data usage.

This report will combine both the process of structuring a database, as well as the data mining using a big database. First, a database is created to store data obtained by new facilities such as The Vista Variables in the Vá Láctea Survey. Our goal is to design a database able of containing such data as well as retrieving the requested data easily. Secondly, using a different dataset, we will explore the different methods utilized in Data Mining. The dataset is made in use of the following. Observing redshifts of galaxies using spectroscopy is a time-consuming process and thus very costly. To reduce the costs a different method is used, namely, the relation between the colours and redshift of a galaxy. However, this relation is still to be explored. Previous research was able to estimate redshifts accurately up to a redshift of 0.25. To be able to use this relationship in future surveys, the discrepancy is required to be below 0.01. We aim to reduce the error margins on the redshift of galaxies, which we calculate with photometric data and compare with the true redshift provided by spectroscopy.

The report is divided into two sections. Part I, where we will discuss the data, the structure of the database as well as the performed queries and its results. Secondly, in Part II the different approaches and methods used in order to define a good relationship between photometric and spectroscopic redshift are presented. An explanation of all methods is provided as well as the results obtained. The appendix presents the attempted SQL commands, necessary in Part I.

Part I: Creating a database for a time-domain survey

The data

The survey providing the data for the formation of the database was carried out as follows. Three different sections of the sky were observed between the Modified Julian Dates $57\,257.05$ and $57\,268.16$ The exposure times range between $16.0\,\mathrm{s}$ and $48.0\,\mathrm{s}$ depending on the filter used. Every field was imaged in the filters Z, Y, J, H and Ks, with the possibility of imaging a field in Ks multiple times at different epochs. The effective wavelength midpoint of the filters is found to be $900\,\mathrm{nm}$, $1020\,\mathrm{nm}$, $1220\,\mathrm{nm}$, $1630\,\mathrm{nm}$ and $2190\,\mathrm{nm}$, respectively. Per image, the number of observations varies between 50 to 300 objects. For every object, the survey provides the StarID, location, type of object, the flux and the magnitudes with uncertainties. The StarIDs vary between 0 - 9999, 170000 - 179999, and 300000 - 309999. These values are usable in between different images. The flux and magnitudes are given for different aperture diameters of $1\,\mathrm{arcsec}$ to $3\,\mathrm{arcsec}$. The type of objects the survey distinguishes are stars, noise, non-stellar and borderline stellar assigned as -1, 0, +1 and -2, respectively. The total number of objects observed is 30000 of which 26823 are stars.

The database layout

The database must be designed in such a way that it is easy to handle the data and should not take up unnecessary memory. As assigned by the client the database should be able to fulfil the queries found in Table 1. In order to do so the set up of the database as follows.

The database contains 3 separate tables, namely *Field Table, Star Table* and *Intensity Table* as shown in Figure 1. The tables and database were created in Python. The Field Table contains all important and general information about the field and the image such as the filename, the filter and the time of observation. The Star Table contains the general information about the objects observed in all fields. Such as the location of the object and the field in which it was observed. The so-called *FieldID* helps to link the Star Table

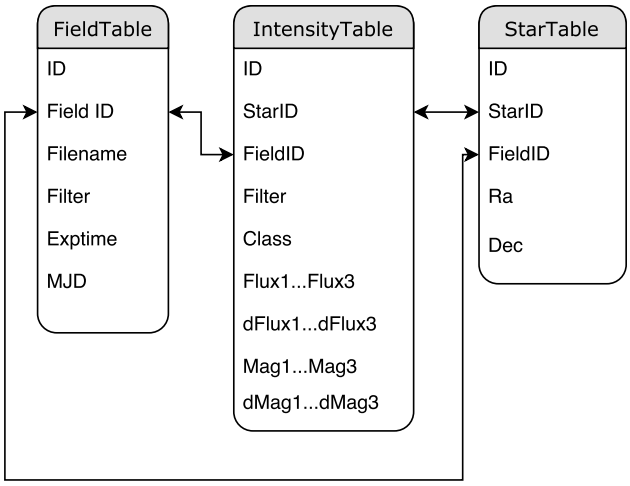


Figure 1: A schematic overview of the designed database. All tables are connect with one another via FieldID or StarID.

to the Field Table and visa versa. The Intensity Table consists of all fluxes and magnitudes, including the uncertainties, taken in all three apertures. To link this table to the Star Table and Field Table, the *StarID* and *FieldID*, respectively, are included. Furthermore, this table includes the filter used to observe these fluxes and magnitudes as well as the object type named *Class*. It was found more useful to include the Class and Filter in the Intensity Table when performing the queries. The colours are not included in this tables since these can be extracted from the included magnitudes and would result in unnecessary memory usage. All tables are linked to one another by the *FieldID* and *StarID*.

Label	Query
R1	Find all images observed between MJD = 56800 and MJD = 57300 and give 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 retrieve the Y, Z, J, H and Ks magnitudes for all stars with S/N $>$ 30 in Y, Z, J, H and Ks.

Table 1: The queries the database should be able to execute.

Filename	Number of stars
H-ADP.2017-01-18T11:58:35.780.fits	7982
H-ADP.2017-01-18T11:58:35.780b.fits	7725
H-ADP.2017-01-18T11:58:35.780c.fits	8022
J-ADP.2017-01-18T11:58:35.781.fits	7022
J-ADP.2017-01-18T11:58:35.781b.fits	7354
J-ADP.2017-01-18T11:58:35.781c.fits	7248
Ks-ADP.2016-05-25T15:33:43.377.fits	7888
Y-ADP.2017-01-18T11:58:36.901.fits	6806
Y-ADP.2017-01-18T11:58:36.901b.fits	7215
Y-ADP.2017-01-18T11:58:36.901c.fits	7186
Z-ADP.2017-01-18T11:58:36.905.fits	6477
Z-ADP.2017-01-18T11:58:36.905b.fits	6929
Z-ADP.2017-01-18T11:58:36.905c.fits	6741

Table 2: The result of performing query R1

Performing the queries

Here the different queries are discussed. All commands to perform the queries in SQL can be found in the appendix. Here we will discuss the approximations and interpretations made. To finish, a sample of the magnitudes of 100.000 stars in the filters Y, J and H are given. This can be used for planning purposes for the Euclid Mission.

R1

The table containing the information regarding Modified Julian Date (MJD) is found in the Field Table while the Signal-to-Noise (S/N) can be calculated by dividing Flux1 by the uncertainty on the flux, dFlux1. Furthermore, we need to distinguish between stars and other objects, so only objects with class -1 are allowed. The results of the query are found in Table 2. It is important to mention that the dates in the fits files are not comparable with the dates and time in the file names. Only the dates in the fits files were used in this query.

R2

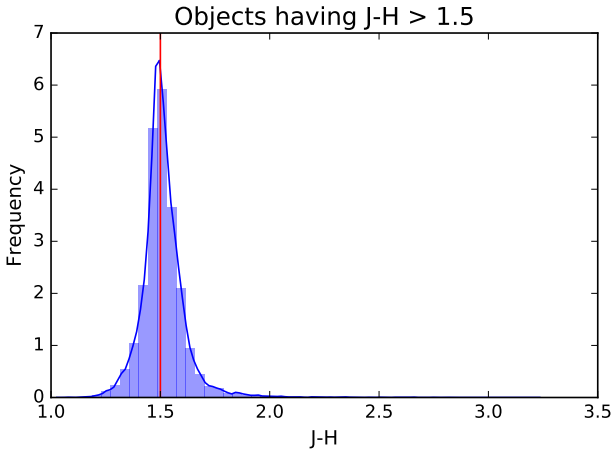
From the Intensity Table, all Mag1 and dMag1 were utilized which were observed in the J and H filter. The uncertainties were taken into account according to Equation 1. In doing so, 13980 objects were found meeting the query.

$$(J - H) \pm (dJ - dH) > 1.5 \quad (1)$$

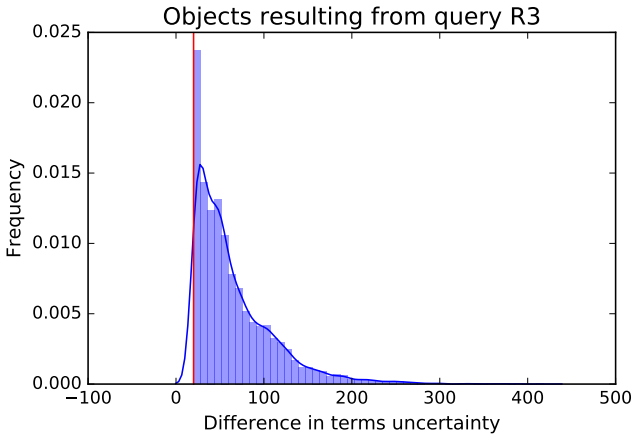
With dJ and dH the uncertainties on the J and H magnitudes. The magnitudes and uncertainties were calculated using only aperture 1.

R3

This query is assumed to be used to give insight on variable stars. If so, then what is mend with the mean flux is the mean of the flux for all epochs for 1 star. Furthermore, the uncertainty is taken to be the uncertainty on the flux for that star at a given epoch. In order to do so, we use Flux1 and dFlux1 as the flux in aperture 1. As a result, no stars from Field 2, with only one epoch, will be included in these result. Furthermore, in order for the output to be significant, we only



(a) Results from query R2. The uncertainties are included in the calculations allowing objects with lower J-H values to show.



(b) The objects obeying query R3. The distribution of the difference between the mean flux and the flux of the variable objects is given normalised to the uncertainty belonging to that flux of a specific object and epoch. The plot includes all the epochs of one object. The red line shows the benchmark of 20 times the uncertainty.

Figure 2

output the stars for which all epochs differ 20 times the uncertainty. As a result, 2854 objects were found. A distribution of the difference in terms of their uncertainty is given in Figure 2b. As can be seen in the figure, most objects are centred at a difference of 20 times the uncertainty. A maximum difference is found at 418.36 times the uncertainty.

R4

Only the Field Table is necessary to perform this query. The filename can be selected with the criteria that the FieldID belonging to that filename should meet the desired field number. The results are summarized in Table 3.

R5

This query specifically asks for stars and thus the class of the objects must be -1. Furthermore, all stars should fit the requirement of having a S/N higher than 30 in all filters. Here, we use the mean of the fluxes and uncertainties in the Ks epochs and require these values to give a S/N above 30. For the Ks band we will again take the average of the flux and magnitude over all epochs. The images with the Ks filter are taken with about 100 days apart from the other images. As a result, it is hard to tell when the variable stars are in the same phase as they were in the other images. Furthermore, we want to compare the Ks bands in different fields as well as stars within a field. So, taking the maximum would not be sufficient enough because some variables are likely to be in the maximum luminous state while others are never observed in that state. Taking the mean would then be a better option. However, this does impact the results on the second field which has only one epoch in the Ks-band. However, examining the results in Figure 3 this does not seem to be as significant.

Field	Filename
1	Z-ADP.2017-01-18T11:58:36.905.fits J-ADP.2017-01-18T11:58:35.781.fits H-ADP.2017-01-18T11:58:35.780.fits Ks-ADP.2016-05-25T15:33:39.546.fits Ks-ADP.2017-01-18T11:58:39.907.fits Ks-ADP.2016-05-25T15:33:43.377.fits Y-ADP.2017-01-18T11:58:36.901.fits
2	Z-ADP.2017-01-18T11:58:36.905b.fits J-ADP.2017-01-18T11:58:35.781b.fits H-ADP.2017-01-18T11:58:35.780b.fits Ks-ADP.2016-05-25T15:33:39.546b.fits Y-ADP.2017-01-18T11:58:36.901b.fits
3	Z-ADP.2017-01-18T11:58:36.905c.fits J-ADP.2017-01-18T11:58:35.781c.fits H-ADP.2017-01-18T11:58:35.780c.fits Ks-ADP.2016-05-25T15:33:39.546c.fits Ks-ADP.2017-01-18T11:58:39.907c.fits Y-ADP.2017-01-18T11:58:36.901c.fits

Table 3: The catalogues found for every field as a result from query R4.

Sampling 100.000 stars Using SQL the Y-J and J-H colours of the stars from the database were obtained to getter with the uncertainty. To be able to only have the data of stars in the output, the class was set to -1. In the resulting data, there are some colours not containing a value. These were removed from the table. Extreme deconvolution from the astroML (Vanderplas et al., 2012) package was chosen to estimate the true distribution of the dataset. This method fits a mixture of multiple Gaussian distributions to the true distribution. The algorithm is defined in Eq. 2, given by Bovy et al. (2011).The main advantage of this method is the utilization of the uncertainty in the estimation. However, the algorithm is time-consuming. The estimator was fitted with the colours and uncertainties and 100.000 samples were pro-

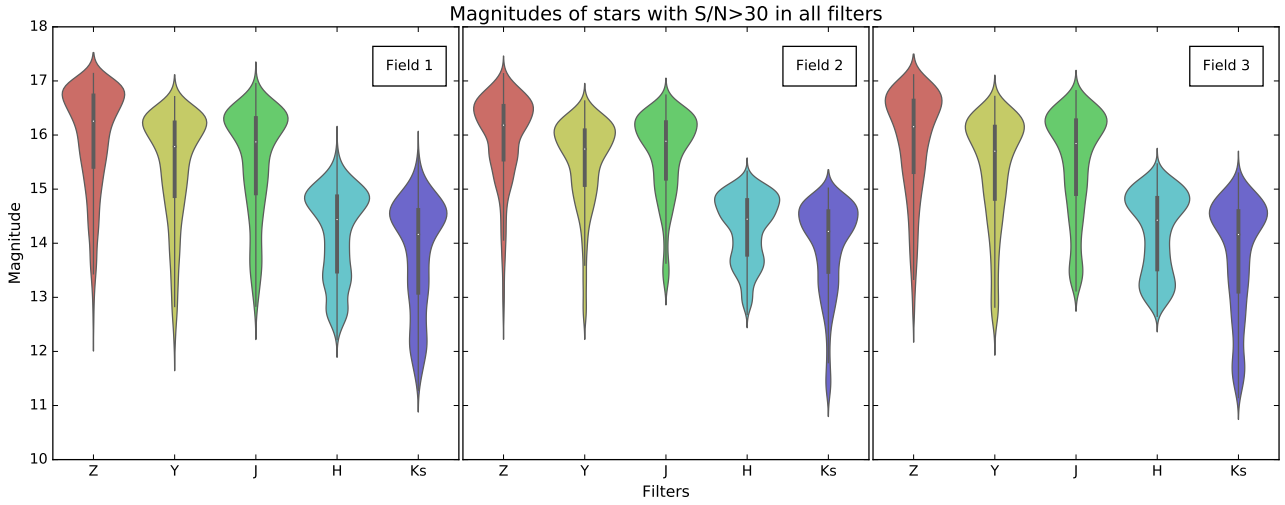


Figure 3: A violin plot of the results from query R5. All fields are shown with the magnitudes for which a star has a S/N > 30 in all filters.

duced resulting in Figure. 4. The number of Gaussian distributions used was fixed at 10. This number is based on the log likelihood, which was the only possibility to evaluate on. Taking a range of numbers of components shows a steep increase. Using more than 10 components shows a decrease in steepness. In combination with the increase in runtime for every added component, 10 components seem sufficient.

$$p(x) = \sum_j \alpha_j N(x|\mu_j, \Sigma_j) + x_i = R_i x_i + \epsilon \qquad (2)$$

With the true values x_i, μ_j and Σ_j the jth mean and standard deviation respectively. The projected matrix is represented as R_i and the noise is given by ϵ.

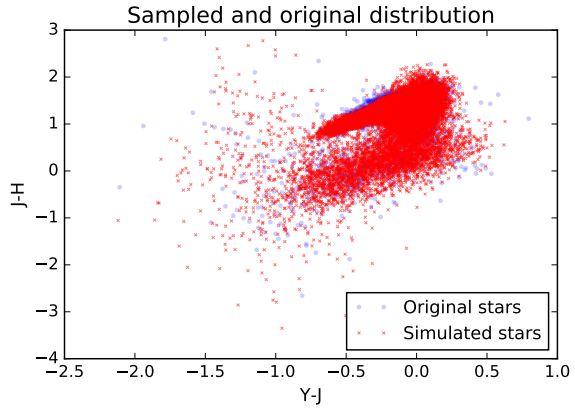


Figure 4: The obtained density distributions. In blue the true density is given. Red represents the sample containing 100.000 stars.

Discussion

The structure of the database was found useful. All queries were able to be done in SQL and without the use of other techniques. However, the SQL commands are very long in some cases. This could be experienced as not user-friendly by the client. This is most certainly the case for queries R3 and R5. A step by step use of the SQL codes and possibilities is thus included in the appendix. Despite this, the SQL codes are robust and will work for newly appended datasets to the database, as was our main focus.

Some of the queries can be interpreted differently, thus the SQL codes may need to be changed accordingly. This was especially the case for R3 for, which the main goal was thought to obtain the candidate variable stars. Furthermore, whenever the colours of the stars are required, the assumption is made that there is also a requirement for the uncertainty. Whether the uncertainty should be taken used in the criterion is open for discussion. Furthermore, one could consider changing the SQL command used for query R5 to get all epochs of Ks separately instead of only using the mean value. This will result in different results.

The samples obtained from extreme deconvolution align with the observational data nicely. The out layers nicely overlap and are well defined. However, a small part at the top bulk does not align well. A solution would be adding another Gaussian component to the data.

Part II: Data mining photometric redshifts of galaxies

The Data

The data consists of photometric content of the observed galaxies. It includes the magnitude R and the colours U-G, G-R, R-I, and I-Z. Finally, the spectroscopic redshift of the galaxy is included, as it will serve as the true redshift, z_{spec}.

The data was provided in two sets. Table A will be used for training and table B is utilised for testing the results. This is necessary as to be certain the model does not over-fit the train data. In that case, only the training data would be predicted correctly while we want to be able to use the model on other data with unknown redshifts. Using the test data we will know how well the model will behave on data without known redshifts.

In total there are 74309 and 74557 galaxies to train and test on. Our sample has a mean redshift of 0.32, just above the limit up to which previous research was able to verify the redshift with linear models. With z_{spec} varying between 0.01 and 0.94 in the training data and 0.00 and 0.89 in the test data.

We reduce the data as to get the separate magnitudes: R, U, G, I, and Z from the colours. This will prevent any extra dependencies in the model. An attempt on Principal Component Analysis (PCA) is made as to see the influence of different numbers of components on the error.

Models

A linear relation between the R, U, G, I and Z magnitudes and redshift is researched. We base our models on Connolly et al. (1995), whom used linear and quadratic fits to a sample of galaxies. The following models will be attempted:

$$f(\theta) = \theta_0 + \sum_{i=1,..,5} \theta_i M_i \tag{3}$$

$$f(\theta) = \theta_0 + \sum_{i=1,..,5} \theta_i M_i + \sum_{\substack{i=1,..,5 \\ j=1,..,5}} \theta_{i,j} M_i M_j \tag{4}$$

$$f(\theta) = \theta_0 + \sum_{i=1,..,5} \theta_i M_i + \sum_{i=1,..,5} \theta_i M_i^2 \tag{5}$$

With θ_0 the so-called intercept and θ_i and M_i the ith independent parameter and magnitude respectively. We will refer to Model 1, Model 2 and Model 3 as the equations found in Eq. 3, Eq. 4 and Eq. 5 respectively.

Algorithms

Here we will explain the algorithms used. First we will fit the parameter θ with a linear regression. Then an attempt on reducing the error with non-linear regressors is made. All algorithms are evaluated based on the error explained in Eq. 6.

$$E(\theta) = \text{median}\left(\left| \frac{z_{\text{spec}} - f(\theta)}{(1 + z_{\text{spec}})} \right| \right) \tag{6}$$

Large surveys require $\mathrm{E}(\theta)$ to be below 0.01 as will be the aim of this report.

Linear regression The base of linear regression is fitting a straight line function to the data using independent variables. In this case, the independent variables are the magnitudes in the different filters. Linear regression is widely used in science due to its simpleness and its low memory costs.

The parameters of the model can be optimised using different kinds of linear regression. In general, the least squared error is used to train the optimal combination of parameters. The parameters for which Eq 7 minimizes will result in the best model.

$$\min || \hat{y}(\theta) - y ||^2 \tag{7}$$

With $\hat{y}(\theta)$ the model output and y the real values.

As an extension to the general linear regression, two other methods are discussed. One of these methods used to optimise the parameters is Ridge Regression. Ridge Regression reduces the possibility of overfitting by introducing penalties relative to the value of the parameters. Bigger values will get a bigger penalty. As a result the parameters will stay small in the fitting procedure. The algorithm is trained as follows from Eq. 8.

$$\min \left(|| \hat{y}(\theta) - y ||^2 + \alpha || \theta ||^2 \right) \tag{8}$$

With α the penalty term which controls the influence of the penalty on the model. A bigger value for α will give a bigger penalty on high values of θ

We expect Ridge Regression to perform the best as it will prevent overfitting of the data. The results will be compared to the ones obtained from normal linear regression.

The final linear regression algorithm to discuss is the Lasso regressor (Least Absolute Shrinkage and Selection Operator). The algorithm is very useful to reduce the number of variables as it will tend to a simpler model with fewer parameters. The model regulates them comparable to the Ridge Regression. In contrast to Ridge, Lasso is able to select parameters by setting them to zero. The algorithm is trained according to Eq. 9.

$$\min \left(\frac{1}{2n_{\text{samples}}} || \hat{y}(\theta) - y ||^2 + \alpha || \theta || \right) \tag{9}$$

Notice that the penalty is not quadratic in contrast to Ridge. It is expected that all parameters are necessary to obtain the optimal parameters. To do so, Lasso Regression is not included in the calculations.

In summary, Linear Regression is very useful to make simple and quick models. However, it is likely to oversimplify the real world. Therefore, other non-linear methods will be attempted in order to get a lower error margin.

Neural Network A neural network is originally based on how the human brain works. One could imagine that seeing a picture of a chair for the first time, will not allow you to recognize all types of chairs. Getting more experience with chairs and seeing more of them helps you in the end to recognize all types of chairs. With every chair you see, the brain will make more links in between nerves. This is exactly what a neural network does. The more information is given, the more links will be made and changed in order to recognize or estimate the desired output. The different nodes in a network can be seen as the different nerve cells. The connections between the nodes are responsible for giving a weight to the output of every node and to make some input of the nodes more or less important. A simple non-linear neural network of one layer is defined in Eq. 10

$$z_j = h\left(\sum_i w_{i,j} x_i\right) \tag{10}$$

Here z_j is the output of the jth node given input x_i. The connections and thus the weights between these nodes are described by $w_{i,j}$ the connection between node x_i and z_j. The h describes the activation function and will verify the output given by the nodes and decides whether the node should be activated by applying a threshold. Choosing a linear or non-linear h determines, whether a neural network is linear or non-linear. Adding more layers will result in a deeper and a more complex network.

A neural network is trained by finding the optimal weights which will give the most accurate output. The training of the weights is in general done using gradient descent. The weights are updated according to Eq. 11

$$w_i = w_i - \eta \frac{\partial E(w)}{\partial w_j} \tag{11}$$

$$E(w) = \sum_i (w1 + .. + w_n x_i - y_i)^2 \tag{12}$$

Here w_i is the ith weight changed according to the gradient of E(w). η is the learning rate which can be changed according to the preferred number to get lower weights faster. However, when this value is too great it will be harder to optimise the weights as the updated value will easily exceed required value.

The advantage of a neural network is its flexibility and it can, with the right number of layers and data, define any type of function. However, be able to get an accurate network, a large amount of data is necessary to train on, which not always available. Furthermore, the iterative training can be very time and memory expensive. Thus, a neural network is not always ideal. The algorithm is chosen based on its flexibility and the big amount of data available in this dataset. We are not interested in going into depth into the physical relations of the magnitudes with the redshift, but more in estimating these values. To do so a network containing 2 and 3 hidden layers will be tested. The number of hidden layers was chosen based on research done by Firth et al. (2003), who did similar research on photometric redshift using a Neural Network. The best fit was found using a network containing 3 hidden layers. The hyperparameters such as the number of nodes inside the layers will be found using brute force.

Random Forest Next, we will turn to a Random Forest (RF). This is a estimator that uses Decision Tree Regression on various samples within the dataset. In general, decision trees have the habit of over-fitting their training data when they become very deep. By averaging the found trees it will be able to improve the estimations while controlling the over-fitting of the data. A three will select at every split the best subset of features for that specific split. If a certain feature is strongly correlated to the desired output than this feature will be selected in more trees. And will be present in the resulting Random Forest. As a result of the Random Forest, the variance will be reduced with a slight increase in the bias.

Other research was done using RF. Carliles et al. (2010) used data provided by the Sloan Digital Sky Survey concluding that the use of RF is comparable to that of other machine learning techniques. However, the performance of RF offers robust results and does not require any parametric model as goes for others.

Based on Carliles et al. (2010), this method is used in estimating the redshifts with photometric data. The depth of the trees included in the forest will be decided by brute forcing different depths. Furthermore, the random state is set to a specific value as to get the same random values at every training epoch. Different numbers of trees in the forest will be tested for a specific depth as to give insight into their influence. The final number will be chosen according to these results. As a follow up, boosting will be included in this method which will be explained here after.

Boosting The main goal of boosting is combining weak algorithms into a strong one. A weak learner is somewhat related to the true result while strong ones are strongly correlated. This is done by iteratively give bigger weights to the examples which are misclassified while others get a lower weight. As a result, in the next iteration, weak algorithms are forced to focus on correctly classifying the examples with a high weight. Finally, the results are averaged based on the weights assigned to ever algorithm. Boosting will reduce both the bias and variance and can be used in combination with different algorithms.

In general the boosting algorithm AdaBoost, Adaptive Boosting (Freud & Schapire, 1997), is used due to its success in different fields of science and its practical use. AdaBoost will thus also be used in this report. One should take the sensitivity on outliers and noisy data of AdaBoost into account. The boosting will be included using a constant value of estimators. First, this value is chosen as to see its influence on the results. A low value is preferred as the algorithm is time and memory expensive.

All above methods are implemented using the sklearn module in python (Pedregosa et al., 2011).

Results

Linear Regression First, the results from Linear and Ridge Regression are presented. As shown in Table 4 Linear Regression using Model 2 provides the best results. Next, is Model 3 with Linear Regression. Both models include a quadratic term, giving the appearance of a non-linear relationship between the magnitudes and the redshift. The obtained best model is as follows:

$$\begin{aligned} z_{\text{phot}} = {} & 0.04 - 1.97R + 1.27G - 0.56I - 0.09U \\ & + 1.29Z + 1.93RR - 0.01GG + 1.55II \\ & + 0.01UU - 0.10ZZ - 0.57RG - 3.67RI \\ & - 0.05RU + 0.51RZ + 0.78GI + 0.05GU \\ & - 0.31GZ - 0.06IU - 0.09IZ + 0.03UZ \end{aligned} \tag{13}$$

It is clear that the paired filters are not as big of an influence on the results but nevertheless still needed. Further-

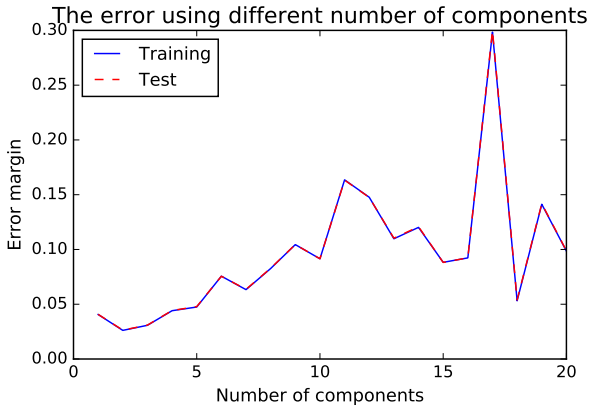


Figure 5: The effect of PCA on the training and test error using linear regression.

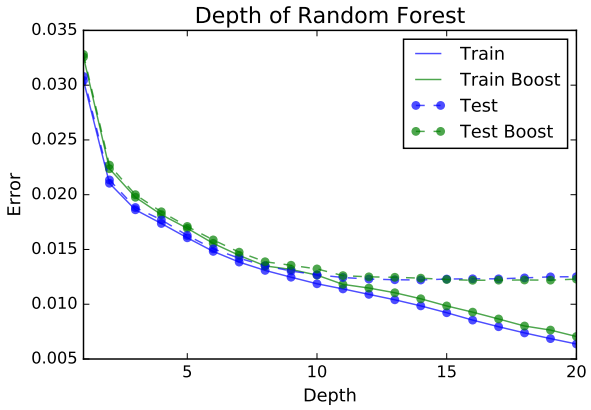


Figure 6: The results of attempting different depths in the forest and their effect on the error.

more, the magnitude of the filter R has the biggest influence on the output. When applying Principal Component Analysis (PCA) to this dataset, the importance of R is displayed. Using different numbers of dimensions with and without standardizing the data reduces the error to approximately 0.026 for both training and test data. The effect of PCA for the different number of components is shown in Figure 5.

Model	Regressor	Training Error	Test Error
1	Linear	0.0146	0.0146
	Ridge	0.0196	0.0198
2	Linear	0.0128	0.0127
	Ridge	0.0179	0.0180
3	Linear	0.0132	0.0133
	Ridge	0.0185	0.0186

Table 4: The results followed from applying Linear and Ridge Regression to the three different models explained above. Linear Regression applied on model 2 gives the error margins closest to the approved 0.01.

Neural Network The data used for the neural network only consists of the separate magnitudes R, G, I, U and Z. For the neural network, a ReLu activation and a constant learning rate of 1×10^{-3} was used. The ReLu activation ensures a non-linear network. To start, the input was distributed over 5 different nodes. The output is provided by 1 node as to get one value out of the network. Applying brute force led to the best combination of nodes per hidden layer. This resulted in 2 hidden layers in an error margin of 0.0149 for both the training and test data. This was established using a layer combination of 5:16:5:1. The same was done for a neural network containing 3 hidden layers which resulted in a training error of 0.0133 and a test error of 0.0134. These were provided by a neural network containing the layers 5:5:3:13:1.

Method	Training Error	Test Error
Linear Regression	0.0128	0.0127
Neural Network	0.0133	0.0134
Random Forest	0.0099	0.0122
Random Forest Boosting	0.0093	0.0122

Table 5: A small summary of the results obtained throughout part II. The error margins for on both the train and test set are presented.

Random Forest and Boosting First, the results using RF will be discussed then the results including boosting to the method are presented. The number of trees in the random forest is set to 10. Applying different values resulted in very small changes in the error. Computationally, a small number of trees is preferred. Including this value in the calculations, the depth is determined. A range of 1 to 20 node layers is used in order to ascertain the optimal maximum depth of the trees. The results from these depths are found in Figure 6. Here, the separation between train and test data shows the point of over-fitting. The best results were found using a RF with a maximum depth of 14 with the error margins 0.0099 and 0.0122 for train and test data respectively.

The same procedure is used when boosting the algorithm with AdaBoost. The obtained result gives the errors 0.0093 and 0.0122 for the train and test data for a depth of 16 layers. There is only a small difference found between the two methods regarding the training error. However, the error on the test set is equal.

A small summary of the obtained results of all attempted methods is found in Table 5.

Discussion

Different models were used to find the relation between the redshift and the magnitudes. The best result was expected using the Ridge Regression as it prevents overfitting of the data. However, the best results were found using Linear Regression trained as a function of the Least Squared Error. An

explanation would be that some magnitudes are stronger related to the redshift and thus should have greater parameter values, which is not supported by Ridge Regression. Furthermore, the use of Model 2 suggests that the relationship between redshift and magnitudes is a combination of linear and quadratic functions. As an initial simple fit to the data, the results seem very promising. Nevertheless, they do not fit the required error margin below 0.01. (Connolly et al., 1995) also shows, though with other filters, that a model like Model 2 gives a better fit to the data. However, their difference in χ^2 between Model 1 and Model 2 is only 1.01.

The best results using a Neural Network were found with a 5:5:3:13:1 network. Comparing the results with the other methods, the neural network performs worse. Moreover, it is very time-consuming in comparison. We can compare our results to Firth et al. (2003) by calculating the Root Mean Square (RMSE), which is found to be 0.04. This is twice the value obtained by Firth et al. (2003). This difference is mainly caused by the use of different filters and initial weights. The network can be improved by training the network on a bigger set for which fine-tuning of the hyperparameters is again required.

Finally, the Random Forest gives the best results, with a train and test error of 0.009 and 0.012 respectively. The Random Forest without boosting seems sufficient enough, taking the calculation time and depth of the trees into account. The method on itself comes with some advantages over other methods as it does not use a underlying statistical model on the data distribution.

We conclude that the RF is found to produce a robust model and meets the $E(\theta) < 0.01$ term on the training data. It is a great improvement on the Linear Regression method.

References

S. Carliles, T. Budavri, S. Heinis, et al., 2010, *Astron. J.*, 712, 1

A. J. Connolly, I. Csabai, A.S. Sxalay et al., 1995, *Astron. J.*, 110, 2655

F. Pedregosa, G. Varoquaux, A. Gramfort, et al., 2011, *Journal of Machine Learning Research*, 12, 2825–2830

Y. Freud, R.E. Schapire, 1997, *Journal of Computer and System Sciences*, 55, 119–139

J.T. Vanderplas, A. Gray, Ž. Ivezić et al., 2012, *Conference on Intelligent Data Understanding (CIDU)*, 47–54

J. Bovy, D.W. Hogg, S.T. Roweis, 2011, *The Annals of Applied Statistics*, 5, 1657–1677

A.E. Firth, O. Lahav, R.S. Somerville, 2003, *Monthly Notices of the Royal Astronomical Society*, 339, 4, 1195–1202

Gaia Collaboration et al. 2016a, *Summary description of Gaia DR1*

Gaia Collaboration et al. 2016b, *Description of the Gaia mission (spacecraft, instruments, survey and measurement principles, and operations)*

Appendix

SQL commands

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.

```
SELECT f.Filename, COUNT(i.StarID)
FROM FieldTable AS f
JOIN IntensityTable AS i
  ON (
      (f.FieldID = i.FieldID)
      AND (f.Filter = i.Filter)
      )
WHERE (i.Flux1/i.dFlux1 > 5)
  AND (
      f.MJD BETWEEN 56800
        AND 57300
      )
  AND (i.Class = -1)
GROUP BY f.Filename
```

R2 Find the objects that have J-H > 1.5.

```
SELECT ij.StarID, ij.Mag1 - ih.Mag1
  AS 'J-H', ij.dMag1 + ih.dMag1
  AS 'd(J-H)'
FROM IntensityTable AS ij,
  IntensityTable AS ih
WHERE (ij.StarID = ih.StarID)
  AND (ij.Filter = 'J')
  AND (ih.Filter = 'H')
  AND (
      (ij.Mag1 - ih.Mag1 +
        ij.dMag1 + ih.dMag1 > 1.5)
      OR (ij.Mag1 - ih.Mag1 -
        ij.dMag1 - ih.dMag1 > 1.5)
      )
```

R3 Find the objects where Ks differs by more than 20 times the flux uncertainty from the mean flux.

To be able to make this readable the code is split into different steps.
Step 1: The command to get the average per over all Ks epochs per star from aperture 1 is given by:

```
SELECT StarID, AVG(Flux1) Average
FROM (
      SELECT StarID, Flux1
      FROM IntensityTable
      WHERE Filter = 'Ks_E001'
      UNION ALL
      SELECT StarID, Flux1
      FROM IntensityTable
      WHERE Filter = 'Ks_E002'
      UNION ALL
      SELECT StarID, Flux1
      FROM IntensityTable
      WHERE Filter = 'Ks_E003'
      )
GROUP BY StarID
```

Step 2: To get the StarIDs and fluxes of all stars and epochs we use

```
SELECT StarID, Flux1, dFlux1
FROM (
      SELECT StarID, Flux1, dFlux1
      FROM IntensityTable
      WHERE Filter = 'Ks_E001'
      UNION ALL
      SELECT StarID, Flux1, dFlux1
      FROM IntensityTable
      WHERE Filter = 'Ks_E002'
      UNION ALL
      SELECT StarID, Flux1, dFlux1
      FROM IntensityTable
      WHERE Filter = 'Ks_E003'
      )
ORDER BY StarID
```

Step 3: We only want the objects for which all epochs have a difference of 20 times the uncertainty. In order to do so, we will need the number of epochs per star. This command returns the StarIDs and the number of epochs used to calculate the average.

```
SELECT StarID, COUNT(*) AS idAmount
FROM (step1)
GROUP BY StarID
```

Step 4: Now search for the stars for which the epoch has a difference between the flux at a certain epoch and the mean over all epochs of more than 20 times the uncertainty on a certain epoch. Here, *step2* is the command to be able to get all StarIDs and fluxes from step 2 and *step1* is the command given in step 1.

```
SELECT i.StarID AS StarID
FROM (step2) AS i
JOIN (step1) AS i0
   ON i0.StarID = i.StarID
WHERE (
      ABS(i.Flux1 - i0.Average)
      > i.dFlux1 * 20
      )
```

Step 5: From step 4 we have a number of outputs for stars which meet the query. However, a distinction between stars which obey this query in all epochs and which do not has not yet been made. The following command looks for the number of epochs per star which do obey the query. *step4* involves the command given at step 4.

```
SELECT StarID, COUNT(*) AS idAmount
FROM (step4)
GROUP BY StarID
```

Step 6: This command checks whether the number of epochs a star is observed in total is equal to the number of epochs which obey the query. Only the stars which do will be given in the output. Here

```
SELECT i.StarID
FROM (step5) i
```

```
JOIN (step3) AS nr
   ON nr.StarID = i.StarID
WHERE (nr.idAmount = i.idAmount)
```

R4 Find all catalogues that exist for a given field.
Here the input for the desired field can be given by the user in *fID*.

```
SELECT f.Filename FROM FieldTable AS f
WHERE f.FieldID = fID
```

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 desired field can be given by the user in *fID*.

```
SELECT iy.StarID, iy.Mag1 AS Y, iz.Mag1 AS Z,
    ij.Mag1 AS J, ih.Mag1 AS H, ik.Mag1 AS Ks,
    iy.FieldID AS Field
FROM IntensityTable AS iy, IntensityTable AS iz,
    IntensityTable AS ij, IntensityTable AS ih
JOIN (
    SELECT StarID, AVG(Flux1) AS Flux1,
       AVG(dFlux1) AS dFlux1, AVG(Mag1) AS Mag1,
       FieldID, 'Ks' AS Filter
    FROM (
       SELECT StarID, Flux1, dFlux1, Mag1,
          FieldID
       FROM IntensityTable
       WHERE Filter = 'Ks_E001'
          AND FieldID = fID AND Class = -1
       UNION ALL
       SELECT StarID, Flux1, dFlux1, Mag1,
          FieldID
       FROM IntensityTable
       WHERE Filter = 'Ks_E002'
          AND FieldID = fID AND Class = -1
       UNION ALL
       SELECT StarID, Flux1, dFlux1, Mag1,
          FieldID
       FROM IntensityTable
       WHERE Filter = 'Ks_E003'
          AND FieldID = fID AND Class = -1
          )
    GROUP BY StarID
    ) AS ik
ON iy.StarID = ik.StarID
WHERE (iy.StarID = iz.StarID)
    AND (iz.StarID = ij.StarID)
    AND (ij.StarID = ih.StarID)
    AND (iy.Filter = 'Y')
    AND (iz.Filter = 'Z')
    AND (ij.Filter = 'J')
    AND (ih.Filter = 'H')
    AND (iy.FieldID = fID)
    AND (iz.FieldID = fID)
    AND (ij.FieldID = fID)
    AND (ih.FieldID = fID)
    AND (iy.Class = -1) AND (iz.Class = -1)
    AND (ij.Class = -1) AND (ih.Class = -1)
    AND (iy.Flux1/iy.dFlux1 > 30)
    AND (iz.Flux1/iz.dFlux1 > 30)
    AND (ij.Flux1/ij.dFlux1 > 30)
    AND (ih.Flux1/ih.dFlux1 > 30)
    AND (ik.Flux1/ik.dFlux1 > 30)
```