SDSS Object Classification with Random Forest

Paloma Jol, Catherine Slaughter Group 23

Leiden Institute of Advanced Computer Science, The Netherlands

Abstract. Object classification is an ongoing, and often troublesome, task for astronomers. We apply machine learning techniques to this problem, using SDSS data to classify a number of galaxies, quasars and stars. We begin by testing a number of dimensionality reduction techniques, and attempt to apply unsupervised clustering algorithms to help classify our data. We then move to supervised techniques, assessing the impact of maximum depth on a single decision tree classifier in terms of over- and under-fitting models. We finally combine our reduced dimensionality data and the understanding gained from our decision tree test to create a random forest classifier. We perform metaparameter tuning for this classifier via a grid search CV method. We ultimately create a classifier which achieves a standard accuracy of 0.87153 ± 0.00118 when tasked with classifying astronomical objects.

1 Introduction

In the night sky, a multitude of astronomical objects can be observed. While these objects all seem very similar to the naked eye, the opposite is true. The most commonly known causes of these lights are the stars, a spherical structure composed of plasma held together by its own gravity, the light emitted is powered by nucleus fusion that occurs in its interior [15]. However, there are also larger structures that we can observe, like quasars or galaxies. Galaxies are the gravitationally bound large-scale structure that host stars, gas clouds, and dark matter. They can be thousands of light-years across and most of their emitted light comes from their stars [17]. Finally, a quasar is a quasi-stellar radio source, a very luminous active galactic nucleus. Meaning that while it looks small and unresolved like a star but it emits at a much broad range of frequencies [15]. This is only a small part of the objects that we observe, but the classification of these objects is one of the most fundamental in astronomy [11]. Providing automatic classification for these sources based on their observed properties is crucial for the increasingly larger data surveys that have been produced in astronomy in the latter years, such as the Sloan Digital Sky Survey (SDSS) [10]. Observations of stars, galaxies, and quasars were included in the Sloan Digital Sky Survey (SDSS), which released its first data release in 2003 and is currently planning on its fifth phase of SDSS-V. In this report, we will be using a modified data set of the SDSS to differentiate between stars, galaxies, and quasars using a Random Forrest classifier. In order to do that we will first do some pre-processing of the data in section 2. We will take out any outliers and look at possible missing values in the provided data set. In section 3 we will introduce and implement our methods, for many of the implementations in Python, we used the scikit-learn Machine Learning in package [14]. As part of this report, we do three implement three tasks a dimensionality reduction, a single classifier, and an ensemble classifier. Section 4 will report on the results of these experiments. First, we look at different dimensionality reduction algorithms and visualize the results of clustering on these reduced data sets. Secondly, we will implement a single Decision Tree and look at its accuracy and main features. Finally, we look at the Random Forrest classifier and compare this ensemble learning with the single Decision Tree to see how the accuracy changes. A summary and discussion of our findings can be found in section 5.

2 Data and Problem Formulation

The Sloan Digital Sky Survey (SDSS) released its first data release in 2003. The objects recorded in these data sets consist of objects that are labeled stars, quasars, and galaxies. Each object

has an associated five-band (u, g, r, i, z) imaging data with measurements of the flux for the Ultra Violet (u-band), Green filter (g-band), Red filter (r-band), Near-infrared filter (i-band), and Infrared filter (z-band). Additionally, information is gathered on the position of the object in the night sky, given by the Right Ascension angle (α), the Declination angle (δ) , and the field ID that identifies each field observed by the SDSS. The final physical property of the objects in the catalogue is the redshift. The rest of the features are identification variables, the date of observation (MJD) for the object, and the plate ID which are used to make the spectra of each object [10].

During the original data exploration, we made histograms for each of the features to see their spread. We found one data point in the u, g, and z flux bands that had a value of -9999.0. A flux is only defined for positive values, small negative values are often associated with noise from the background but large negative values are often used to indicate non-detections due to low flux measurements or errors in the instrument. In the case of SDSS, they have taken this -9999 as an indication for non-detection [12]. As part of our pre-processing, we removed this data point from our data set which leaves us with 99.999 samples. For the features, we made the correlation of Figure 1 to visualize the dependencies of the features. We see that each of the bands is correlated, this is not unexpected since most astronomical objects emit light over a spectrum, thus their flux will be spread over various bands. Having said that it often does not cover the whole range of the spectrum but is limited to neighbouring bands, which again is something we see in the correlation plot. The other correlation that we see between the three location parameters since each field is based on a part of the sky it will have specific alpha and delta values. The correlation between the alpha and delta in turn comes from the limitation of only observing the Northern Hemisphere [10].

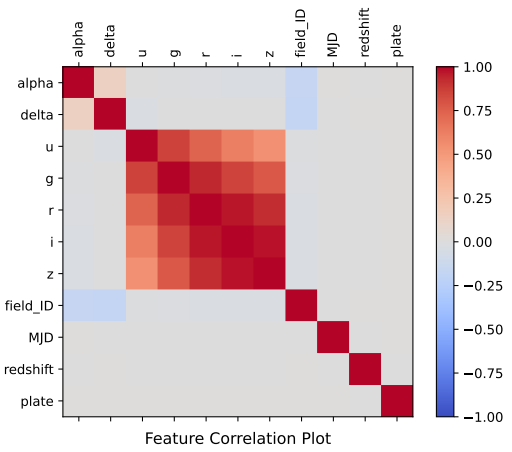


Fig. 1: Correlation plot for the 11 non-class features given in the data

Our features also contain some ID variables that we remove as part of the pre-processing. These are the field and plate IDs, for the simple reason they only indicate specifics on the SDSS survey. We also remove the date of observation, since the objects are not going to change based on when in the survey they are observed. Since both authors of this report are of an astronomy background we found it natural to also name the alpha and delta locations as ID parameters, this is based on our prior knowledge of the data set. Since the location is not going to be relevant to the label since by the uniformity of the universe [15] we expect there not to be a global preferred direction for e.g. stars, this is especially true in cases where the survey is not

yet finished, since then not everywhere has been observed yet [10], or technical limitations, such as that we cannot observe the southern hemisphere with SDSS and low delta sources are harder to observe in general. If our aim is to make a general model it should also work if new data, from new locations, are added [11]. Our pre-processed data thus consists of 99.999 data points each with six features; the five-band (u, g, r, i, z) imaging data and the redshift.

In this report, we try to solve the problem of using the SDSS data set provided to us to predict whether each object is a 'star', 'galaxy', or 'quasar' based on its spectral properties and redshift. We split this data into three parts a training set, a validation set, and a test set. The training set will be used for training the model iteratively, the validation set will be used for the cross-validation and the parameter tuning between the models and the test set will be used to measure the final accuracy of the model on unseen data. In order to build a good model we will try various experiments and see how it affects our predictive power on the set. We have set up this report with three main subjects; the dimensionality reduction, the single classifier, and the random forest classifier.

3 Methodology & Experiments

3.1 Dimensionality Reduction

Our training set has a total of size features to start with, so in this section, we are going to look at various dimensionality reduction methods. We tried a few different options three of which are presented below. Each method was implemented using the scikit-learn package [14] using a list of parameters for the different settings. For the dimensionality reduction methods, the training data is used, which is first scaled using the StandardScaler function in scikit-learn.

Principal Component Analysis (PCA) involves mapping the high dimensional data into a lower number (n) dimension primary components. In order to do this mapping we need the eigenvectors of the data, of which we take the n eigenvectors with the highest eigenvalues since these capture the most important information. The algorithm can find it challenging to capture non-linear relations since the mapping used is a linear transformation [16]. For the PCA we tried two different settings for the primary components, two components, and five components. We also varied the solver during some runs in 'auto', 'full', and 'arpack' [5]. Uniform Manifold Approximation and Projection (UMAP) provides a low dimensional representation of the data with retaining the topological structure as well as possible. The mathematical foundation is hard to interpret, but the mapping helps to interpret the model and understand the structure and allows non-linear relations to be fitted. This clustering method showed some promise so we did most of our experiments on this method. First, we varied the number of neighbors between 2 and 200. Secondly, we varied the minimal distribution between 0 and 0.5. Finally, we changed the used metric between the 'correlation', 'Euclidean', 'Minkowski', 'Mahalanobis', 'Jaccard', and 'Chebyshev' [9]. T-Distributed Stochastic Neighbor Embedding (T-SNE) is a method to reduce high-dimensional data to 2 or 3 dimensions. The core idea is that similar points in high dimensions should also be similar in lower dimensions. The model is not always interpretable since the dimensions have no clear meaning, the mapping cannot be reproduced for new points, and the moment of optimization reached is not always clear. This method has tunable parameters such as perplexity, exaggeration factor, and distance measures but we only ran some experiments changing the perplexity between 5 and 50, but the computation time was so long that we restrained from doing further experiments [8].

Additionally, we used two clustering mechanisms to evaluate how well the lower dimensionality data was clustered per label. We used K means, a clustering mechanism that uses distance to cluster similar points together [7]. The algorithm starts with the inputted k centroids and clusters each point in the data set to the closest centroid. The centroid is then updated by taking the mean of all of the data points that belong to that cluster. These steps are repeated until the data point clusters no longer change [13]. Additionally, we used DBSCAN, a Density-based

spatial clustering of applications with noise (DBSCAN) [2]. DBSCAN has a natural way to detect outliers in the data and to cluster the data points based on a continuous region of high point density increased density in the regions. For the DBSCAN we tried various parameters of epsilon between 0.001 and 0.05, and a minimum number of samples needed to be called dens of 50 to 10.000 which were chosen based on the requirement that we preferred to have lower number of clusters matching our labels and a data set of 69.999.

3.2 Decision Tree Classifier & Cross-Validation

We then moved on to creating a classifier with a single decision tree. Decision trees are a versatile learner that can be applied to a number of different tasks with both classification and regression applications. One must be cautious when using decision trees, as they have high variance when learning and are very prone to over-fitting if not pruned or otherwise limited. One way to analyze the over fitting of these classifiers is with a k-fold cross-validation method. K-fold cross-validation works by splitting the set of m training data points into k subsets, each of size m/k. The learner is then trained k times, where each time, only four of these subsets is used. The remaining set, called the "validation set" in the convention we chose, is then used to validate the resulting learner. In an overfit model, the accuracy of the training set will be significantly higher than that of the validation set. In an underfit model, both will be about the same, but rather low. Cross-validation can therefore be used as a tool to test the impact of various metaparameters on a model's output [16].

We implemented our decision tree with the Scikit Learn DecisionTreeClassifier class [3]. The sklearn class makes use of an ID3 algorithm for its classification, and provides users with a number of built-in metaparameters to define. In our case, we decided to test the maximum depth of the decision tree. This simple parameter can have a huge impact on the over-/under-fitting for such a learner [16]. Using our 6-parameter astronomical data, we created a number of decision trees with maximum depths varying from 1 to 20, leaving all other metaparameters in their default state. For each tree, we performed a standard k=5-fold cross-validation using the sklearn cross_validate method [1], making sure that the method returned the training data score in order to assess the over- or under-fitting. For ease of visualization, we then created a line plot, showing the average training and validation scores over the 5 iterations for each tree.

3.3 Random Forests & Metaparameter tuning

After doing our initial testing with a single decision tree, we decided to create a random forest for our final classifier. Our specific implementation uses sklearn's RandomForestClassifier [6]. This class makes use of bootstrapping to help reduce the correlation between individual ensemble members and help avoid over-fitting [16]. In order to further obtain the most well-fit, accurate classifier, we made use of grid-search cross-validation (GSCV) in order to simultaneously tune several metaparameters (see Table 1). Besides the parameters tuned, almost all other parameters were left in their default state [6], except the number of estimators, which is increased to 150. To perform our GSCV, we made use of sklearn's GridSearchCV class [4], which creates a "grid" of parameter sets, one for each possible combination of parameter inputs given. In our case, we analyzed 120 different possible parameter inputs. We also used a 5-feature PCA reduction as our input, in order to help remove some noise from our data. We also used the results of our single decision tree test subsection 3.2 to help choose the possible values for the maximum depth parameter.

From our GSCV output, we create a set of plots showing the effects of each parameter on the training and validation scores, averaging over the other three parameters.

It is important to note that sklearn's built-in GSCV functionality does not automatically check for over-fitting in selecting the "best" parameters. Because of this, we limit the possible hypothesis space by comparing the training and validation scores of each of the 120 parameter sets using a

Parameter	Description	Values Tested
criterion	The criteria used to calculate gain at each node	'gini', 'entropy'
max_depth	Maximum number of nodes any tree path may have	6, 7, 8, 9, 10
max_features	The number of data features considered at each node	1, 3, 5
max_samples	The fraction of samples to be selected to train each base estimator	.25, .5, .75, 1.

Table 1: Metaparameters tuned with GSCV and the values tested.

percent difference calcuation: $\frac{Training - Validation}{Training}$. We set a threshold of 0.05, so any parameter set where the validation score was less than 95% of the training score would not be considered. From this subset of possible learners, we select the one with the highest GSCV validation score.

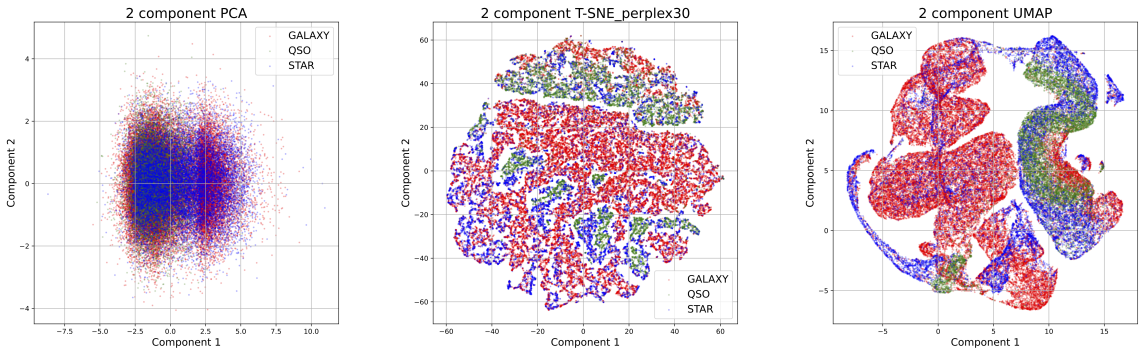
From here, we create and train a new random forest classifier with our selected parameters. We fit and test the model 5 times with a newly selected training set to calculate the average and standard deviation of its test accuracy. We do the same for a single tree classifier, with the same relevant selected parameters, for comparison purposes.

4 Results

In this section, we will discuss the results of the experiments we have done on the labeled SDSS data set. We start in section 4.1 with the results of the dimensionality reduction and the clustering. In section 4.2 we discuss the Decision Tree classifier, and the implemented cross validation. Finally, we look at the meta parameter tuning and feature importance in section 4.3.

4.1 Dimensionality Reduction

We implemented the dimensionality reduction methods PCA, UMAP and T-SNE explained in section 3.1 and ran the experiments on the pre-processed data. The best results for these methods can be found in Figure 2. The result for the PCA was produced with the 'auto' solver and the first two components are plotted. For the T-SNE a perplexity of 30 was used with early exaggeration of 12.0 and an euclidean metric. Finally, for the UMAP we used a two-components UMAP, based on ten neighbors, a minimal distribution of 0.3, and a correlation metric.

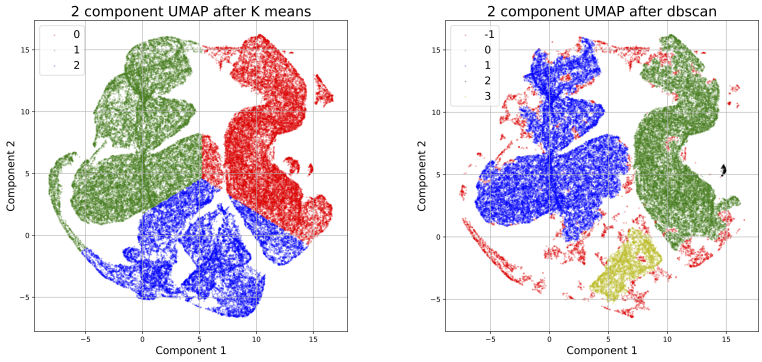


(a) PCA with its first two components (b) T-SNE with perplexity 30 (c) UMAP with two components

Fig. 2: Results of the dimensional reduction algorithm, the colors correspond to the labels: red for galaxy, green for quasar and blue for star.

We see that for the PCA the labels are not clustered separately. For the five components, we looked at the different pairs through the plotting but the overlap in the labels was in each case large. The components themselves are uncorrelated, as we would expect from the different eigenvectors but they do not separate well the galaxies, quasars, or stars. The T-SNE does show some clustering of the labels. The green seems to be clustered as multiple smaller islands embedded in the multitude of red. The blue labels which correspond to the stars are not clustered but seem to be spread across the figure, with only some clustering in the lower left bottom.

The UMAP shows a complex clustering of the labels with multiple regions which show a higher density of one specific color. The blue labels are still quite scattered but also have their own denser regions like the green and red labels do. Because it visually looks like it does better at separating the clusters we decided to run the clustering algorithms on the two-component UMAP. Before clustering, the labels look like shown in Figure 2c. The labels given by the clustering methods, K means with three clusters, and DBSCAN are given in Figure 3. The DBSCAN clusters were made using an epsilon of 0.05 and minimal samples of 50. For the K means, we could compare the clustering before and after dimensionality reduction because we can input the number of clusters. We found that using the processed data as input and the known labels we get a K means clustering accuracy of 0.71 and an accuracy of 0.81 when using the dimensionally reduced data. Thus we see a slight increase in the accuracy K means algorithm when using the reduced data. On the other hand, DBSCAN chooses how many clusters it makes which makes it harder to link each cluster wiht one given label of 'galaxy', 'quasar' or 'star'. We do see in Figure 3b that the clustering picks up on some of the denser regions that were visible in the UMAP of Figure 2c, however many parameters we tried gave either a very large amount of clusters or labeled all of the data as outliers.



(a) K means clustering (b) DBSCAN clustering

Fig. 3: Results of the clustering algorithms, the colors correspond to different cluster labels.

4.2 Decision Tree & Cross-Validation

The results for the single decision tree cross-validation test can be found in Figure 4. The training score, as well as the difference between the training and validation scores, increases with maximum depth. In comparison, the validation score increases then decreases. This plot informed our decision to use values of 5 through 9 for the maximum depth in our GSCV.

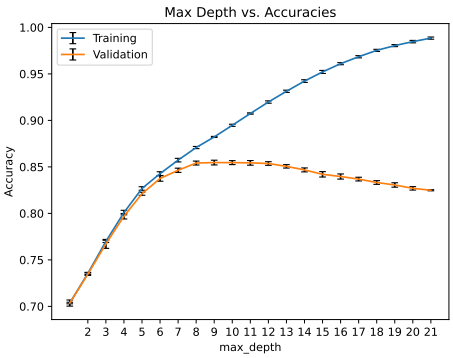


Fig. 4: The training and validation set scores at each cross-validation iteration for various inputs of max_depth, where 'None' allows the tree to be as deep as it needs to be

4.3 Random Forest & Metaparameter Tuning

The output of our GSCV can be seen in Figure 5. These plots show the average validation scores for each value of the 4 parameters we are tuning. This is done by selecting all the runs with a given value for a specific parameter, and averaging over them. Doing so should generally average out the impacts of the other parameters, so that we can still see the general trend. For the gain criterion, we see that the entropy criteria yields a slightly better validation score, on average, and the gini impurity. The trend for the tree depth generally mimics that seen in Figure 4, where the score generally increases with increased depth, flattens out. The number of features selected from at each node creation starts very low at 1, maximises at 3, and drops down slightly at 5. Finally, the score generally decreases as the maximum number of samples selected in bootstrapping increases.

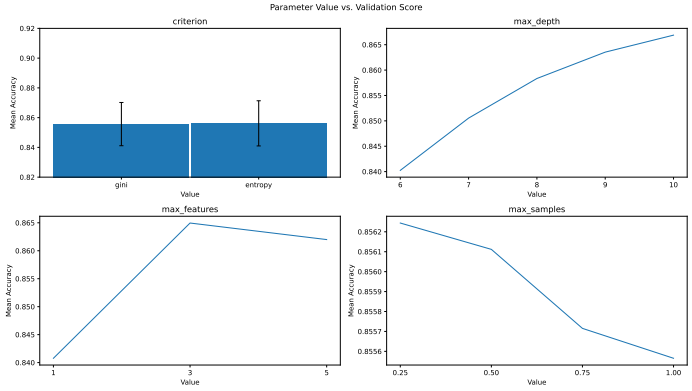


Fig. 5: The validation scores for the different values of each of our 4 tuned parameters, where the other parameters are averaged over.

The final selected best parameters can be found in Table 2. These are the parameters that maximise the validation score while ensuring the percent difference between the training and validation score remains less than 5%.

Parameter	Value
criterion	'entropy'
max_depth	10
max_features	3
max_samples	.75

Table 2: Final selected parameters.

We created, trained, and tested a forest with our selected parameters 5 times, and calculated the mean and standard deviation test accuracy. For comparison purposes, we did the same for a single tree classifier with our same parameters (except, of course, n_estimators). We found the forest had an average accuracy of 0.87153 ± 0.00118. In comparison, the single tree had an average accuracy of only 0.81504 ± 0.00325. We determined 5 runs was sufficient given how small the standard deviations were.

5 Conclusions

5.1 Dimensionality Reduction

We successfully performed dimensionality reduction on our large data set using three different methods: PCA, T-SNE, and UMAP. The effectiveness of each method is varied, with the clearest large-scale structure forming from the UMAP method. In general, it appears that this data may simply be too complicated to be sorted with unsupervised clustering algorithms on 2-dimensional reduced data. Future research to this end could seek out more advanced clustering methods, or perhaps attempt clustering with a 3-component reduced dataset. In any case, however, our PCA dimensionality reduction served useful as an noise-reduced input for our supervised methods.

5.2 Decision Tree & Cross-Validation

Our decision tree cross-validation experiment shows very clearly just how important tree length is when trying to avoid over- and under-fitting. For this task, we can clearly see from the plot that a single decision tree can become very overfit very quickly. For maximum depths between 1 and 5, the training and validation scores are basically the same. They start to diverge a little between 5 and 8, and very rapidly separate after. It is interesting to note, however, that the validation score does continue to increase briefly even after the training score begins to diverge. This indicates that there is some amount of difference between the two that is acceptable.

5.3 Random Forest & Metaparameter Tuning

We successfully built and trained a random forest that is able to correctly classify our celestial objects $\sim 87\%$ of the time. We achieve this in part by conducting a grid search cross validation on four metaparameters, namely the gain criterion, maximum tree depth, maximum number of features considered at each node, and maximum number of samples used to train each weak learner. In general, our selected best parameters (Table 2) are in line with the general trends seen from the GSCV (Figure 5), with the exception of the max_samples parameter. This most likely indicates that max_samples has a generally smaller or more variable effect on the ensemble as a whole, and would be an interesting topic of future study. It may also be interesting to either expand the metaparameter space input into the GSCV, or to study a similar space with higher resolution. Additionally, this project could be expanded by utilizing more features in the original data set, perhaps by making use of other astronomical catalogs, like the WISE catalog, as done in [11].

References

1. Cross-validate. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html, accessed: 2022-12-16
2. Dbscan - density-based spatial clustering of applications with noise. finds core samples of high density and expands clusters from them. good for data which contains clusters of similar density. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html, accessed: 2022-12-18
3. Decision tree classifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, accessed: 2022-12-08
4. Gridsearchcv. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, accessed: 2022-12-16
5. Principal component analysis (pca). https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, accessed: 2022-12-03
6. Random forest classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, accessed: 2022-12-08
7. sklearn k-means clustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, accessed: 2022-12-18
8. T-distributed stochastic neighbor embedding. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html, accessed: 2022-12-08
9. Umap. https://pypi.org/project/umap-learn/, accessed: 2022-12-05
10. Abazajian, K., Adelman-McCarthy, J.K., Agüeros, M.A., Allam, S.S., Anderson, S.F., Annis, J., Bahcall, N.A., Baldry, I.K., Bastian, S., Berlind, A., et al.: The first data release of the sloan digital sky survey. The Astronomical Journal **126**(4), 2081 (2003)
11. Clarke, A., Scaife, A., Greenhalgh, R., Griguta, V.: Identifying galaxies, quasars, and stars with machine learning: A new catalogue of classifications for 111 million sdss sources without spectra. Astronomy & Astrophysics **639**, A84 (2020)
12. Deokkeun, A., Johnson, J.A.: ugriz sdss/segue photometry of globular and open clusters (Sep 2014), https://classic.sdss.org/dr7/products/value_added/anjohnsono8_clusterphotometry.html
13. Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H.: The elements of statistical learning: data mining, inference, and prediction, vol. 2. Springer (2009)
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
15. Ryden, B., Peterson, B.M.: Foundations of astrophysics. Cambridge University Press (2020)
16. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: From theory to algorithms. Cambridge university press (2014)
17. Sparke, L.S., Gallagher III, J.S.: Galaxies in the universe: an introduction. Cambridge University Press (2007)