Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate (DiSTA)
Artifical Intelligence for Astrophysics Problems

Galaxy Classification

Final Project
Lucas Barbier-Goy

Academic Year 2022-2023

“Uno dei miei giorni più produttivi è stato quando ho buttato via 1000 righe di codice.”
- Ken Thompson (Scienziato informatico, sviluppatore del sistema operativo UNIX)

Contents

List of Figures

List of Tables

1 Introduction

The massive amount of data coming from telescopes or other types of sensors in physics makes manual classification and processing of these data impossible. To solve this problem, many techniques have been developed, including machine learning and deep learning, which allow, among other things, mass classification of these data or the creation of predictive models. Depending on the type of data being processed, some methods will be more appropriate than others.

In this project, we focus on the "Galaxy Zoo" dataset, a participatory scientific project in which images of galaxies were analyzed and classified by citizen volunteers. The dataset consists of two parts: a "processed" part where we find certain characteristics of galaxies as classified by the participants, and a raw part that presents the images as observed by telescopes. The objective is to predict whether a galaxy is more likely to be of spiral or elliptical type.

In the first part of this project (3.1), we compare the performance of different classical estimators, while in the second part (3.3), we use a convolutional neural network to make predictions on the images. We also explore the dataset using unsupervised learning to determine which attributes of the dataset have the greatest impact on the estimators' decision. This part (3.2), called Principal Component Analysis (PCA), will allow us to better understand the dataset studied.

Finally, the objective is to determine if the accuracy achieved by these methods is comparable to that of humans. In other words, would it be possible to use these machine learning methods to automatically sort the next SDSS data[1].

[1]It turns out that these methods are already being used in many cases for the processing of telescope images.

2 Dataset

In this study, we focus on the classification of spiral and elliptical galaxies using data provided by the citizen science project ”Galaxy Zoo”. For this purpose, we used two datasets.

The first dataset is a CSV table containing the classifications of galaxies made by Galaxy Zoo participants [1]. This table contains information on 667,944 galaxies and has sixteen columns. The columns include the unique identifier of the galaxy in the SDSS catalog, its position in the sky (right ascension and declination), the number of votes received for this galaxy, as well as the estimated probabilities by participants for six different categories: elliptical, clockwise rotating spiral, counterclockwise rotating spiral, galaxy with a blurry edge, galaxy with a dominant nucleus, and galaxy with multiple nuclei. Additionally, the table contains columns indicating the estimated type of the galaxy, as well as columns for probabilities corrected for classification biases (see Table 2.1).

The decision tree used by Galaxy Zoo (see Fig. 2.1) is based on a series of binary questions that distinguish between different shapes of galaxies. This process begins by distinguishing between elliptical and spiral galaxies based on the presence or absence of spiral arms. Next, spiral galaxies are classified based on the direction of rotation of their arms, while elliptical galaxies are divided into two subcategories based on their symmetry.

The second dataset is a set of image files of galaxies corresponding to the classifications provided in the first table. The goal is to demonstrate how machine learning

Table 2.1: Description of Galaxy Zoo 1 catalog data

Column	**Description**
OBJID	Unique identifier of the galaxy in the SDSS catalog
RA	Right Ascension (in degrees) of the galaxy
DEC	Declination (in degrees) of the galaxy
NVOTE	Number of votes obtained for this galaxy
P_{EL}	Probability that the galaxy is elliptical
P_{CW}	Probability that the galaxy is a clockwise spiraled galaxy
P_{ACW}	Probability that the galaxy is an anticlockwise spiraled galaxy
P_{EDGE}	Probability that the galaxy is a galaxy with a blurred edge
P_{DK}	Probability that the galaxy is a galaxy with a dominant nucleus
P_{MG}	Probability that the galaxy is a galaxy with multiple nuclei
P_{CS}	Probability that the galaxy is a galaxy with strange features
$P_{EL_{DEBIASED}}$	Probability of the elliptical classification, corrected for bias effect
$P_{CS_{DEBIASED}}$	Probability of the classification with strange features, corrected for bias effect
SPIRAL	Number of votes for the ”spiral” classification
ELLIPTICAL	Number of votes for the ”elliptical” classification
UNCERTAIN	Number of votes for the ”uncertain” classification

methods can handle and make high-precision predictions on raw (or processed) telescope images.

2.1 Data processing

Before starting the model training with the data presented above, it is necessary to process the dataset. Indeed, some features present in the original dataset have no influence on the class of the galaxy. Specifically, the first column is a unique identifier that cannot be a feature for our model. The second and third columns represent the positions of galaxies in the sky. Cosmological models assume that the universe is homogeneous and isotropic, so we assume that the position of the galaxy in the sky is not correlated with our classes/targets.

Once the unnecessary features for classification are discarded, it is important to check the regularity of our data. For example, the ”NVOTE” feature in our dataset has a very different scale and contains outlier values. These two characteristics make it difficult to visualize the data and, more importantly, they can alter the predictive performance of many machine learning algorithms. Unregularized data can also slow down or even prevent the convergence of many gradient descent-based estimators.[1] It is concluded in

[1]A discussion of the different scalers can be found here: https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/00-Data-exploration.ipynb.

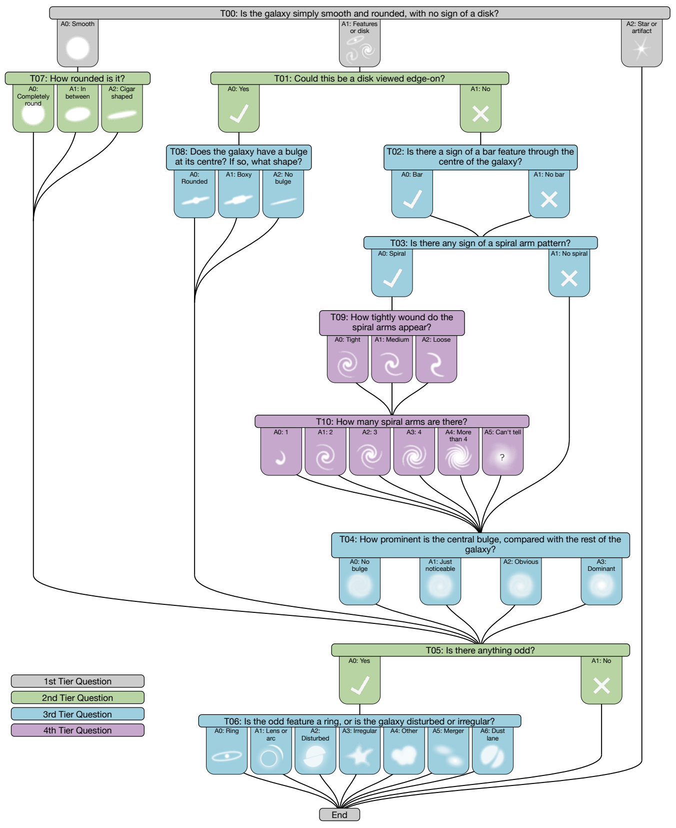
T00: Is the galaxy simply smooth and rounded, with no sign of a disk?
A0: Smooth
A1: Features or disk
A2: Star or artifact
T07: How rounded is it?
A0: Completely round
A1: In between
A2: Cigar shaped
T01: Could this be a disk viewed edge-on?
A0: Yes
A1: No
T08: Does the galaxy have a bulge at its centre? If so, what shape?
A0: Rounded
A1: Boxy
A2: No bulge
T02: Is there a sign of a bar feature through the centre of the galaxy?
A0: Bar
A1: No bar
T03: Is there any sign of a spiral arm pattern?
A0: Spiral
A1: No spiral
T09: How tightly wound do the spiral arms appear?
A0: Tight
A1: Medium
A2: Loose
T10: How many spiral arms are there?
A0: 1
A1: 2
A2: 3
A3: 4
A4: More than 4
A5: Can't tell
?
T04: How prominent is the central bulge, compared with the rest of the galaxy?
A0: No bulge
A1: Just noticeable
A2: Obvious
A3: Dominant
T05: Is there anything odd?
A0: Yes
A1: No
1st Tier Question
2nd Tier Question
3rd Tier Question
4th Tier Question
T06: Is the odd feature a ring, or is the galaxy disturbed or irregular?
A0: Ring
A1: Lens or arc
A2: Disturbed
A3: Irregular
A4: Other
A5: Merger
A6: Dust lane
End

Figure 2.1: Decision tree used by participants of Galaxy Zoo 2.

this analysis that the Scikit-learn's "Normalizer" scaler is the best choice, but in practice, it turns out that the "min-max" scaling gives the best results.

Finally, it is important to verify that the dataset is well balanced to avoid biases when training models. In our case, the dataset is highly imbalanced (see Fig. 2.2). To solve this problem, we randomly sample n galaxies from each class, where n is the number of points for the least represented class. As n is usually large, we select a much smaller value for the following examples[2]. A sample of the dataset actually used (before scaling) for training is shown in Table 2.2.

For practicality with the Scikit-learn API, we merged the classes into a single column and assigned a number to each of them:

- Spiral: 0
- Elliptical: 1
- Uncertain: 2

[2]Most of the models shown as examples are trained with $n = 150$.

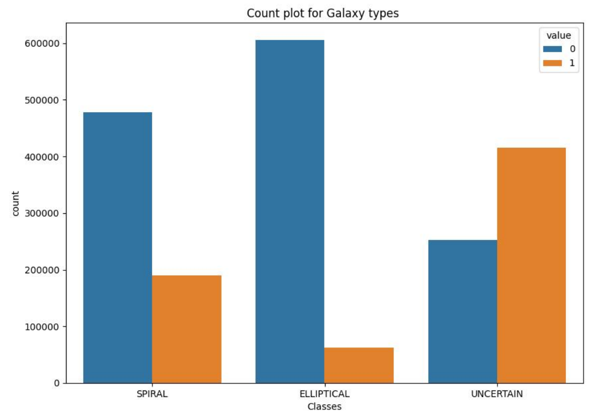

Figure 2.2: The total number of each galaxy is indicated by the orange column. The dataset is clearly imbalanced in favor of the "uncertain" class.

Table 2.2: Reduced Galaxy Zoo1 Catalog Data used for the training.

NVOTE	**P_EL**	**P_CW**	**P_ACW**	**P_EDGE**	**P_DK**	**P_MG**	**P_CS**	**P_EL_DBS**[3]	**P_CS_DBS**	**CLASS**
59	0.6100	0.0340	0.0000	0.1530	0.1530	0.0510	0.1860	0.6100	0.1860	2
18	0.6110	0.0000	0.1670	0.2220	0.0000	0.0000	0.3890	0.2030	0.7970	0
68	0.7350	0.0290	0.0000	0.1470	0.0740	0.0150	0.1760	0.4320	0.4280	2
52	0.8850	0.0190	0.0000	0.0580	0.0190	0.0190	0.0770	0.8850	0.0770	1
59	0.7120	0.0000	0.0000	0.2200	0.0680	0.0000	0.2200	0.64	0.29	2

3 Machine Learning

Machine learning is a branch of artificial intelligence that aims to provide computers with the ability to learn from data. This discipline encompasses several subparts, including supervised learning, unsupervised learning, and reinforcement learning.

In supervised learning, algorithms are trained on labeled training data, which is data for which the correct response is known. The algorithms learn from this data by adjusting their parameters to minimize the error between the predicted response and the actual response. Once learning is complete, the algorithm can be used to predict the response for new data.

In this analysis, we will focus on supervised learning techniques for galaxy classification. We will use classical supervised learning models to process tabular data, but also a deep learning model to process galaxy images. Deep learning is an advanced machine learning technique that uses neural networks to learn from data. These neural networks are capable of learning increasingly abstract representations of the data as they progress through layers of neurons. We will briefly introduce an unsupervised learning technique in section 3.2.

Thus, our analysis will combine supervised learning and deep learning techniques to classify galaxies using both image and tabular data.

3.1 Supervised Learning

[1]

Supervised learning is a sub-component of machine learning that deals with learning from labeled data. This field can be divided into two subcategories: classification and regression. Classification is used when the target variable is discrete, such as the classification of an iris species into three categories. In contrast, regression methods are used when the target variable is continuous, such as the price of a house.

In our case, we want to determine the type of galaxy, whether it is spiral or elliptical. This task is of a discrete nature and falls under classification. Therefore, we will focus on classification methods. Among the possible estimators, the following will be studied:

- Naive Bayes classifiers
- k-Nearest Neighbors
- Decision Trees
- Random Forests
- AdaBoost
- Support Vector Machines
- Perceptron
- Multi-layer Perceptron

These methods will be evaluated and compared based on their performance in terms of precision and recall. Note that recall measures the number of true positives detected by the classifier, compared to the total number of positives in the dataset. It is often used in conjunction with precision to evaluate the overall performance of a classifier.

3.1.1 Naive Bayes Classification

Naive Bayes methods are a set of supervised learning algorithms that are based on the application of Bayes' theorem. This theorem allows us to calculate the conditional probability of an event A given an event B, from the conditional probability of B given A and the marginal probabilities of A and B. Bayes' theorem is written as follows:

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \tag{3.1}$$

[1]The code for this section is available at: https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/01-Classifiers.ipynb.

Naive Bayes methods make the assumption of conditional independence between each pair of features, given the value of the variable belonging to the class. Mathematically, this is expressed by the following modification of Bayes' theorem:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)} \tag{3.2}$$

This formula is the basis for all naive Bayes classifiers. The only difference between these classifiers lies in the distribution $P(x_i|y)$ of (3.2), that is, the probability that x_i corresponds to class y. This type of classifier works well for document classification or spam filtering, for example. Naive Bayes classifiers have many advantages, such as requiring little training data, speed, or insensitivity to the curse of dimensionality. However, while they are good classifiers, they are poor estimators [2].

Among the probability distributions for $P(x_i|y)$, we find in particular the normal distribution, the Bernoulli distribution, and the multinomial distribution. Among the naive Bayes classifiers, we have:

- Gaussian Naive Bayes Classifier
- Bernoulli Naive Bayes Classifier
- Multinomial Naive Bayes Classifier
- Complement Naive Bayes Classifier

Using naive Bayes methods, we can therefore construct efficient classification models for problems with discrete or continuous features. The choice of the appropriate distribution will depend on the characteristics of the data and the assumptions we make about their distribution. To determine which classifier best corresponds to our problem, we will briefly introduce them in the following sections.

Gaussian Naive Bayes Classifier

In this version of the algorithm, it is assumed that the conditional probability distribution of each feature for each class follows a Gaussian distribution:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp -\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \tag{3.3}$$

where μ_y and σ_y^2 are respectively the mean and variance of feature x_i for class y. These parameters are estimated using maximum likelihood from the training data.

The Gaussian Naive Bayes classifier works well when the training data follows a normal distribution. However, if this is not the case, the classifier may be inaccurate. Additionally, this version of the Naive Bayes classifier is sensitive to outliers, which can also affect the accuracy of predictions. Nevertheless, despite its limitations, the Gaussian Naive Bayes classifier remains a popular choice for data classification due to its simplicity and speed of execution.

Multinomial Naive Bayes Classifier

This classifier is often used for text classification. The distribution is parameterized by vectors $\boldsymbol{\theta}_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features and θ_{y_i} is the probability $P(x_i|y)$ that feature i appears in a sample of class y. The parameters $\boldsymbol{\theta}_y$ are estimated using a regularized version of maximum likelihood. In particular, the relative frequency of occurrence $\hat{\theta} y_i$ is calculated for each feature i in each class y:

$$\hat{\theta} y_i = \frac{N y_i + \alpha}{N_y + \alpha_n}$$

where N_{yi} is the number of occurrences of feature i in samples of class y, N_y is the total number of samples in class y, α is a regularization hyperparameter, and $\alpha_n = n\alpha$ is the total sum of the regularization hyperparameter for all features. The value of α controls the degree of regularization: the larger α, the closer the $\boldsymbol{\theta}_y$ are to uniform values, while for $\alpha = 0$, the unregularized maximum likelihood estimation is recovered.

The multinomial distribution is often used to model data that count the number of occurrences of each feature in a sample, such as word frequencies in a text. In this case, features can be considered as word counts, and θ_{y_i} is the probability that word i appears in a document of class y.

Complement Naive Bayes Classifier

The Complement Naive Bayes classifier is a variant of the Multinomial Naive Bayes classifier. It was designed to better handle imbalanced datasets. Indeed, when certain classes are under-represented, the Multinomial Naive Bayes classifier may be biased in favor of the most represented classes.

The principle of the Complement Naive Bayes classifier is to consider the complements of each class, i.e., the features that do not belong to the class. Thus, for each class y, the complement probabilities $P(\neg x_i|y)$ are calculated for each feature i. The model parameters are then determined using these complement probabilities to compute the model weights. More specifically, the probability $P(x_i|y)$ is calculated using complementarity: $P(x_i|y) = 1 - P(\neg x_i|y)$. The model parameters are then estimated using the regularized maximum likelihood formula, similar to that of the naive multinomial Bayes classifier.

In summary, the complementary naive Bayes classifier uses complementary probabilities to better handle imbalanced datasets. The model parameters are estimated using a regularized maximum likelihood formula, similar to that of the naive multinomial Bayes classifier.

Model Selection and Verification

The features in our dataset are continuous variables taking values in the interval $[0, 1]$. Bernoulli or categorical classifiers, which rely on Bernoulli or categorical distribution laws, respectively, use discrete probability distributions and do not seem suitable for our problem. The multinomial naive Bayes classifier, as well as its complementary version, are suitable for discrete features such as counts and assume a multinomial distribution of features.

Therefore, the Gaussian naive Bayes classifier appears to be the optimal choice for our classification task given the dataset used. Additionally, it is reasonable to assume that our variables are distributed according to a normal distribution, since the variables are vote outcomes that should follow a normal distribution. All of these classifiers were tested using the Python scikit-learn library [3]. We sampled 150 elements from each class and split the dataset into a training subset and a validation subset, using a 70/30 ratio. For training, we used a ”min-max” scaler, particularly to scale the ”NVOTE” column. As expected, the Gaussian classifier performed the best, with an accuracy of 90%.

Table 3.1: Confusion Matrix for Gaussian Classifier

	Predicted ”Spiral”	Predicted ”Elliptical”	Predicted ”Uncertain”
Actual ”Spiral”	27	0	1
Actual ”Elliptical”	0	32	1
Actual ”Uncertain”	1	6	22

Confusion Matrix

A confusion matrix is a table used to evaluate the quality of a classification system. It shows the number of correct and incorrect predictions made by the model compared to a set of test data, by comparing predictions to the true class labels.

It is also worth noting that the multinomial classifier still achieves an accuracy of 87%. The precision results for each classifier are summarized in the Tab. 3.2.

3.1.2 The K-Nearest Neighbors

The k-nearest neighbors (K-NN) method is a supervised algorithm that belongs to the category of neighborhood classifiers. It consists of assigning a class label to an unlabeled example by searching for the k most similar examples in the training set, called the

Classifier	**Accuracy (%)**
Gaussian	90
Multinomial	87
Bernoulli	57
Complement	69

Table 3.2: Accuracy Table of Naive Bayes Classifiers.

k-nearest neighbors, and assigning the most frequent class among these k neighbors to the unlabeled example. In other words, if $k = 5$, and for a given point, 4 of the 5 nearest neighbors to this point represent spiral galaxies, then the algorithm will deduce that the given point represents a spiral galaxy. Note that this algorithm can also be used for regression.

To determine the similarity between two examples, the algorithm uses a metric such as Euclidean distance or Manhattan distance. The choice of distance measure may depend on the type of data. Euclidean distance remains the most commonly used metric.

Due to its non-parametric nature, this algorithm is often very effective when it comes to classifying situations where the decision boundary is not clear. The only parameters that this algorithm takes are the number of neighbors k and the metric. These parameters naturally depend on the dataset. We often try to choose an odd number for k to avoid vote ties in binary classification. A value of k that is too small can lead to overfitting, while a value of k that is too large can lead to underfitting. During training, we will try several values of k in order to find the one that gives the best accuracy on the validation dataset.

Algorithms

Although the K-NN method remains the same, there are different algorithms that can be used to implement it. Each algorithm depends on the dataset, particularly the number N of samples and the dimension D of the features. For a small dataset, the brute force method can be used, which consists of calculating the distances between each sample in D dimensions. Although faster for small N, this algorithm has a complexity that grows as $\mathcal{O}(D.N)$ and becomes unusable for large datasets.

A solution to this problem is to build a K-dimensional tree, a generalization in K dimensions of the quadtree and octree data structures in 2 and 3 dimensions. This data structure avoids calculating the distances between all points. If two points are not on the same node or branch, their distance is not calculated. Once the tree is constructed, the complexity of the algorithm scales as $\mathcal{O}(\log N)$. The K-dimensional tree is very efficient for $D < 20$, but then suffers from the curse of dimensionality. A solution to this problem is the ball tree [4].

Given that our dataset contains only 10 features, either the brute force method or the K-dimensional tree will be used, depending on the number of samples selected when loading the dataset.

Training and results

During the training step, we used the same configuration as in the previous example, which we kept for the rest of the analysis. Although scaling the data had no effect on the naive Bayes classifiers, it had a significant impact on the performance of the K-nearest neighbors algorithm. Without scaling, the obtained accuracy is only 64%, while it reaches 84% with min-max normalization. Contrary to our expectations, scaling the data to have unit vectors results in slightly lower accuracy (83%).

Grid search indicates that the number of neighbors giving the best result is 9, using a metric called ”Manhattan”. It is important to note that these results vary depending on the number of points used for training.

Finally, we evaluated the performance of the K-NN algorithm on a separate test dataset. We obtained a confusion matrix showing the classification results for each class. The confusion matrix is presented in Fig. 3.1.

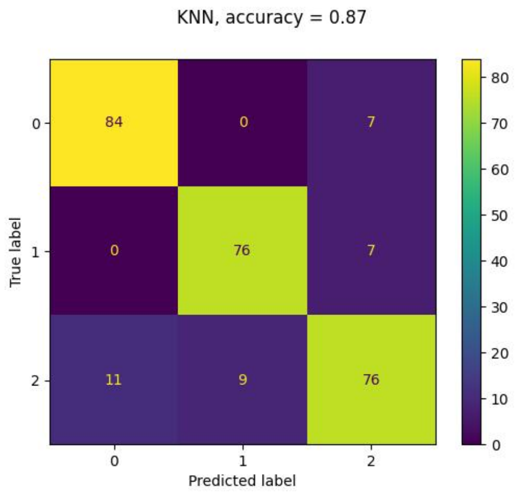

Figure 3.1: Confusion matrix for the K-NN algorithm. We can see, for example, that the algorithm confused 11 ”uncertain” galaxies with spiral galaxies. Note that the score appears higher because more points were used. For a very large number of points, the precision seems to converge towards 87%.

3.1.3 Support Vector Machine

Support Vector Machines (SVM) are a family of supervised learning algorithms used for classification and regression. The goal of SVM is to find an optimal decision boundary, called hyperplane, which separates different classes of data samples by maximizing the margin, i.e., the distance between the closest samples of each class (see Fig. 3.2). This approach is particularly useful for datasets with complex and nonlinear features, where a simple linear decision boundary is not sufficient to effectively separate the classes. SVMs can also be used for small to medium-sized datasets, but are not recommended for very large datasets as they can be quite resource-intensive.

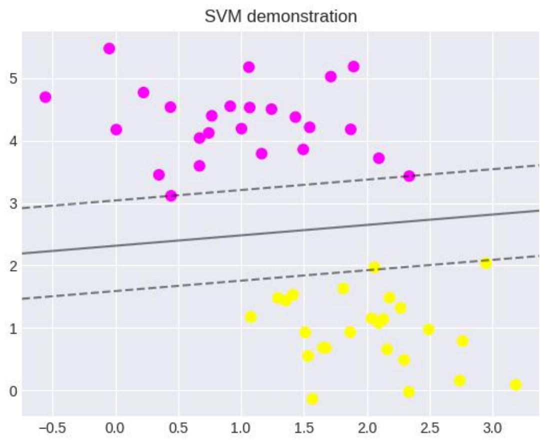

Figure 3.2: Illustration of decision boundaries created by the algorithm on a toy dataset in two dimensions.

SVMs have several important parameters that need to be tuned to achieve the best performance for a particular dataset. Among these parameters, we can mention the choice of kernel function (e.g., linear, polynomial, or radial basis function (RBF)), regularization C, and gamma. The kernel function is used to transform the input data into a higher dimensional space where the classes are more easily separable. The most commonly used kernels are linear, polynomial, and radial kernels [5].

The regularization parameter C controls the trade-off between maximizing the margin and minimizing the error. Higher values of C result in narrower margins and more accurate classification of training data points, but may also lead to overfitting. Lower values of C allow for wider margins and less accurate classification of training data points, but may also reduce overfitting.

The gamma parameter is the coefficient of the RBF kernel. This parameter affects the shape of decision boundaries. Higher gamma values result in more complex decision boundaries, which can lead to overfitting. Lower gamma values result in smoother decision boundaries, which can lead to underfitting.

Training and Results

After exploring the data using PCA (see section 3.2), it appears that the samples are linearly separable. This suggests that the SVM method with a linear kernel would be well-suited to solving our classification problem.

Grid search indeed indicates that the parameters producing the best accuracy are a linear kernel with a C value of 1000. This configuration achieves an accuracy of 92%, surpassing naive Bayes classifiers and k-nearest neighbors.

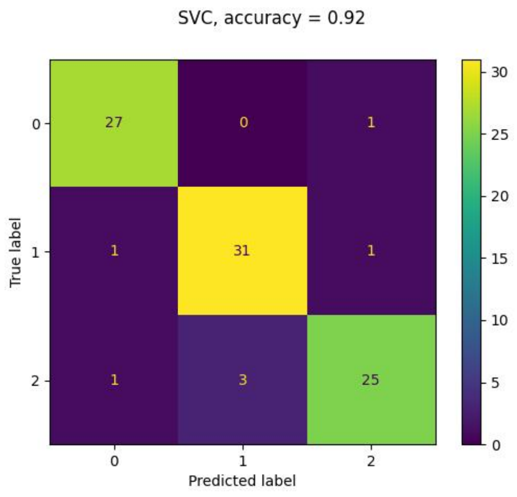

Figure 3.3: Confusion matrix for the SVM algorithm. The algorithm makes few confusions between the different classes.

3.1.4 Perceptron

The perceptron is a linear binary classification algorithm that creates a separating hyperplane to divide data into two classes. It is based on a step activation function that takes the weighted sum of inputs and returns a binary output. The model is trained by adjusting the input weights to minimize the classification error. The perceptron is most effective when data is linearly separable, meaning the two classes can be separated by a hyperplane. It was one of the first machine learning algorithms to be developed [6] and is often used as a basis for more complex models such as neural networks. It is also used for real-time classification and online learning. The perceptron is fundamentally a binary algorithm, so it is naturally used for binary classification problems. However, it is possible to extend it to multiple classes using ”one-vs-all” or ”one-vs-one” techniques.

In the former case, K binary perceptrons are trained, each distinguishing one class from the others, and in the latter case, $K(K-1)/2$ binary perceptrons are trained, each distinguishing two classes at a time.

Training and Results

The parameters that can be passed to the perceptron mainly concern the penalty methods. The most commonly used penalties for the perceptron are L1 and L2 penalties. The L1 penalty adds a constraint to the cost function that forces coefficients to be close to zero. This allows for automatic feature selection, as coefficients corresponding to less important features are reduced to zero. The L2 penalty adds a constraint that forces coefficients to be small and close to zero, but not exactly reduced to zero. This reduces overfitting by reducing the coefficients of features that have a low contribution to prediction. There are also other penalty methods for the perceptron, such as the ”ElasticNet” penalty that combines L1 and L2 penalties.

Grid search indicates that the L1 penalty gives better results with a learning rate of 10^{-4}. The accuracy achieved by the perceptron is only 74%, confirming that this estimator was not the most suitable for our problem.

3.1.5 Decision tree

As a non-parametric classification and regression method, the decision tree [7] is an essential tool in supervised learning. Its goal is to create a model capable of predicting the value of a target variable by learning simple decision rules from the data’s features. It should be noted that a tree can be seen as a piecewise constant approximation. The deeper the tree, the more complex the decision rules, and the better the model fits the data.

The advantages of this method are numerous: its simplicity of understanding and interpretation, easy visualization, low data preparation requirements, logarithmic cost $\mathcal{O}(\log N)$ for data prediction, ability to handle both numerical and categorical data, as well as multiple output problems. However, decision trees may present some disadvantages, such as creating overly complex trees that do not generalize well, instability, or creating biased trees if one class dominates the data.

The decision tree is a popular choice when the data is complex and heterogeneous, making it a suitable choice for astronomical datasets. However, it is important to note that the decision tree is better suited for smaller datasets. Given that the dataset we are studying comes from a decision tree (see Fig. 3.4), it is natural to consider the use of a decision tree as a good option for classification.

Training and results

Regarding the parameters, they include the tree's depth, the minimum number of examples required to split a node, the minimum number of examples required for a leaf node, the splitting quality measure criterion (gini, entropy, or classification error), and the quality measure threshold.

Grid search on the decision tree parameters for galaxy classification identified the optimal values of the splitting criterion and maximum tree depth, equal to entropy and 18, respectively. The results obtained from these parameters are satisfactory, with a maximum accuracy of 86%. The result is satisfactory although below what we had hoped for.

The fact that nodes are created randomly makes finding optimal parameters somewhat complicated, and those found here may not be the ones that give the best possible result. The tree that gives the best results is presented in Fig. 3.4. It is interesting to note that the decision is first made on the "P_CS_DEBIASED" column, and that the tree separates spiral and elliptical galaxies quite distinctly from the first node.

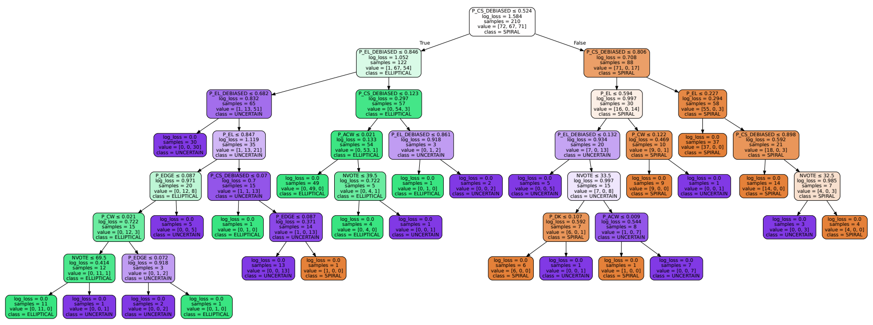

Figure 3.4: Internal structure of the decision tree created by the algorithm. For better visibility see https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/plots/arbre_decision.pdf.

3.1.6 Ensemble Methods

Although decision trees are a powerful tool, they can sometimes have limitations. In particular, if a single decision tree is used to classify data, it may be biased or not generalize well to the data, leading to overfitting. That is why it can be beneficial to use ensemble methods that combine several decision tree models to obtain a more robust and accurate classification.

The most common ensemble methods are random forests [8] and AdaBoost [9]. In both cases, multiple decision tree models (or perceptrons) are combined to achieve a more accurate classification.

Random forests are made up of several decision trees that are constructed randomly. Important parameters include the number of trees and the depth of each tree. Generally, the more trees in the forest, the more accurate the classification. However, this also increases the computation time. In our case, random forests can be particularly effective because they can handle complex data with multiple features and classes.

AdaBoost, on the other hand, trains decision tree models iteratively. At each iteration, the model focuses on the misclassified samples from the previous iteration. Important parameters include the number of iterations and the depth of each tree. AdaBoost can be very effective for difficult datasets, particularly when classes are imbalanced. However, it can also be sensitive to noise in the data.

In summary, ensemble methods are an effective approach for classifying complex data. In our case, random forests can be particularly useful for classifying galaxy data with many features, and AdaBoost can also be effective if the classes are imbalanced.

Training and Results

The grid search results show that for random forests and AdaBoost, the optimal number of estimators is 20 and 250, respectively. For random forests, the quality criterion used to split tree nodes is entropy, and the maximum depth is 34. For AdaBoost, the optimal estimator is a decision tree with an entropy quality criterion, a maximum depth of 3, and a learning rate of 0.1.

The accuracy achieved for the random forest is 93%, and for AdaBoost it is 81%. These results demonstrate that ensemble methods can be very effective for galaxy classification. Notably, this confirms the formidable efficiency of random forests.

3.1.7 Multilayer Perceptron

The Multilayer Perceptron (MLP)[10] is a type of artificial neural network that can be used for classification and regression. It is made up of several layers of neurons, where each layer is connected to the previous and next layer. The intermediate layers are called hidden layers because their outputs are not directly observed.

The MLP is trained by minimizing a cost function, which measures the difference between the model's predictions and the actual values. This cost function is minimized using an optimization technique such as gradient descent.

To implement the MLP, we need to specify several hyperparameters, such as the number of hidden layers, the number of neurons in each layer, the activation function used to compute the neuron outputs, the cost function used to train the model, as well as optimization-specific hyperparameters such as the learning rate.

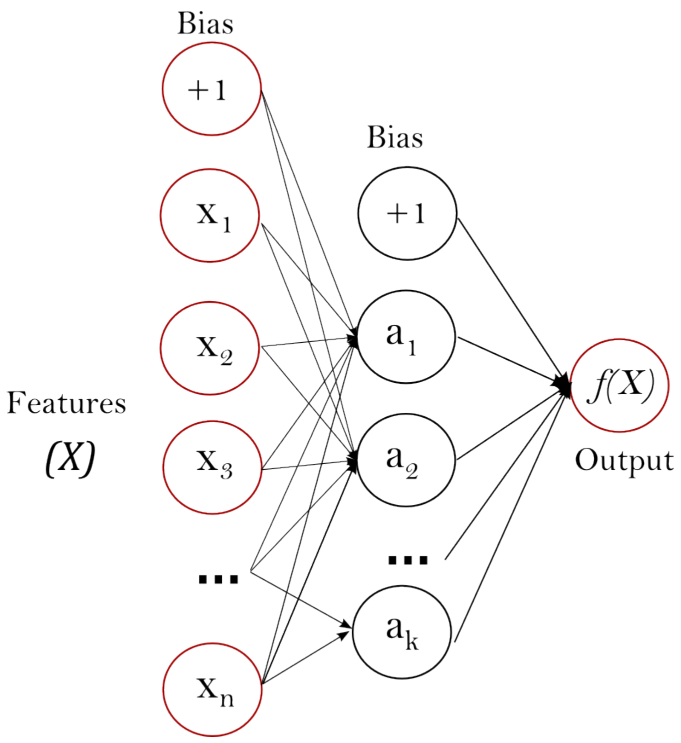

Figure 3.5: Structure of a perceptron with a hidden layer. Reference: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

Mathematical Details

A neural network is a mathematical function that can be represented as a graph of interconnected neurons (see Fig. 3.5). Mathematically, this function can be described as a series of nested non-linear functions. The output of each function feeds into the input of the next function, and so on, until the final output of the network is produced:

$$\hat{y}(\mathbf{x}) = w_L^T \phi_L \left(w_{L-1}^T \phi_{L-1} \left(\cdots \phi_2 \left(w_2^T \phi_1 \left(w_1^T \mathbf{x} + b_1 \right) + b_2 \right) \cdots \right) + b_{L-1} \right) + b_L \tag{3.4}$$

where $\mathbf{x}$ is the input, $\hat{y}(\mathbf{x})$ is the predicted output, $\mathbf{w}_l$ and $\mathbf{b}_l$ are the weights and biases of layer l, respectively, and $\phi_l(\cdot)$ is the activation function of layer l. In other words, each hidden layer is calculated by applying a linear transformation $\mathbf{w}_l^T \cdot$ to the output of the previous layer, followed by a non-linear activation function $\phi_l(\cdot)$. The final output is calculated by applying a final linear transformation $\mathbf{w}_L^T \cdot$ to the output of the last hidden layer, followed by a bias $\mathbf{b}_L$.

This allows the MLP to learn hierarchical representations of the input data, where each hidden layer extracts increasingly abstract features from the input data.

Typically, the activation functions are non-linear functions such as the sigmoid function or the rectified linear unit (ReLU) function. These functions are important because they allow the network to model non-linear relationships between the inputs and outputs.

Finally, to train a neural network, a gradient descent algorithm [11] is used to adjust the weights and biases of the network in order to minimize a cost function. This cost function measures the difference between the network's outputs and the actual values.

Training and results

Using grid search, we identified the optimal parameters for our model, namely: a hyperbolic tangent activation function and an architecture comprising of two hidden layers, each with 50 neurons. For the cost function, we used the cross-entropy function, which is commonly used for classification tasks. Our model achieves an accuracy score of 91%.

3.2 Unsupervised Learning

Unsupervised learning is an approach to machine learning that aims to discover hidden structures in data without using pre-existing labels or responses. This approach is particularly useful when we do not know exactly which features of the data are relevant or interesting.

One of the most commonly used methods for unsupervised learning is dimensionality reduction, which allows high-dimensional data to be projected onto a lower-dimensional space while preserving the most important characteristics. The objective of this method is to find linear combinations of variables that reduce the dimensionality of the data while preserving their variability. A popular algorithm in unsupervised learning is principal component analysis (PCA), which involves finding the axes of maximum variability in the data and projecting the data onto these axes to reduce dimensionality. Finally, clustering algorithms are also widely used in unsupervised learning. Clustering algorithms aim to group similar data points into distinct clusters. The most common clustering methods are hierarchical clustering and K-means. Hierarchical clustering groups data using either a top-down or bottom-up approach, while K-means partitions data into K clusters by minimizing the distance between each point and its corresponding cluster centroid. In summary, unsupervised learning is an approach to machine learning that enables the discovery of hidden structures in data without using pre-existing labels or responses.

3.2.1 Principal Component Analysis

We focus here only on the Principal Component Analysis (PCA) method, as it allows us to better understand our dataset and analyze the internal relationships between different variables. As described, this technique allows us to reduce the complexity of our data by projecting it onto a new lower-dimensional space, while preserving as much information as possible from the initial data. The principal components are determined by diagonalizing the covariance matrix of the data. This way, we can identify the most important variables and understand how they contribute to the different principal dimensions.

Results

We performed two principal component analyses (PCA) on our galaxy dataset, including or excluding the labeled "uncertain" data. Using PCA, we reduced our dataset from 10 dimensions to 2 dimensions, which allowed us to visualize the data on a two-dimensional plot based on their principal components.

The results of PCA showed that our galaxy dataset is linearly separable in two dimensions, especially if we exclude the uncertain classified data (see Fig. 3.6). Specifically, we observe that the first two principal components explain a large part of the variance of the data (see Fig. 3.7).

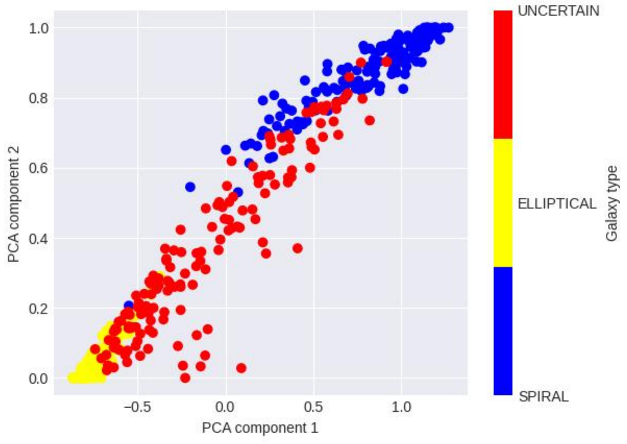

Figure 3.6: The points are distributed in the plane formed by the first and second principal components. These are given by the following linear combinations: CP1 = (-0.028 x NVOTE - 0.440 x P_EL + 0.145 x P_CW + 0.096 x P_ACW + 0.215 x P_EDGE - 0.016 x P_DK - 0.002 x P_MG + 0.457 x P_CS - 0.503 x P_EL_DEBIASED + 0.517 x P_CS_DEBIASED) and CP2 = (0.396 x NVOTE - 0.094 x P_EL + 0.675 x P_CW + 0.013 x P_ACW - 0.605 x P_EDGE - 0.018 x P_DK + 0.044 x P_MG + 0.083 x P_CS + 0.036 x P_EL_DEBIASED - 0.037 x P_CS_DEBIASED)

By using PCA, we were able to better understand the internal structure of our dataset and identify the most important features for variable separation. We could have started our analysis using PCA to observe this linear separation (in a different space) and choose

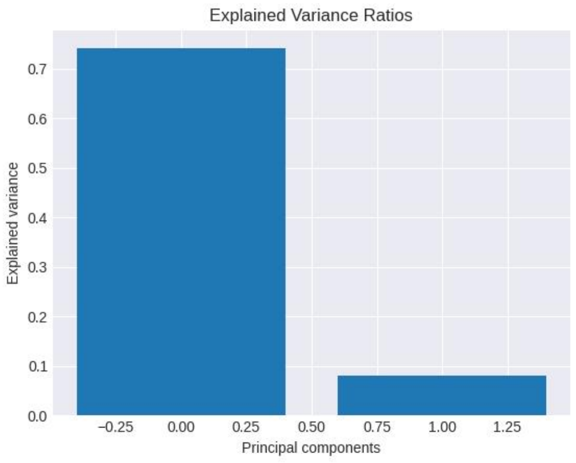

Figure 3.7: Explained variance ratios. On the left is the first component, and on the right is the second. The first component accounts for more than 74% of the variance, compared to only 8% for the second.

an appropriate classifier accordingly.

3.3 Deep Learning

The Galaxy Zoo data used so far are based on classification probabilities assigned by a large number of participants, rather than direct measurements of the physical characteristics of the galaxies. We have created classifiers based on numerical data that could eventually determine the classes of galaxies still classified as ”uncertain”. Although these probabilities are useful for predicting the type of a galaxy, they require significant human processing beforehand. We could speed up the classification process by using a model that would classify galaxies directly using telescope images.

Here, we will implement[2] a convolutional neural network (CNN) to classify galaxy images. The CNN is a popular choice for image classification because of its ability to automatically extract important features from images using convolution filters [12].

3.3.1 Convolutional Neural Network

The operation of a CNN is based on multiple layers of information processing. The first layer processes the raw image and applies convolution filters to extract visual features such as edges, corners, textures, etc. The second layer applies new filters to extract

[2]The code can be found at the following link: https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/03-Deep-learning.ipynb.

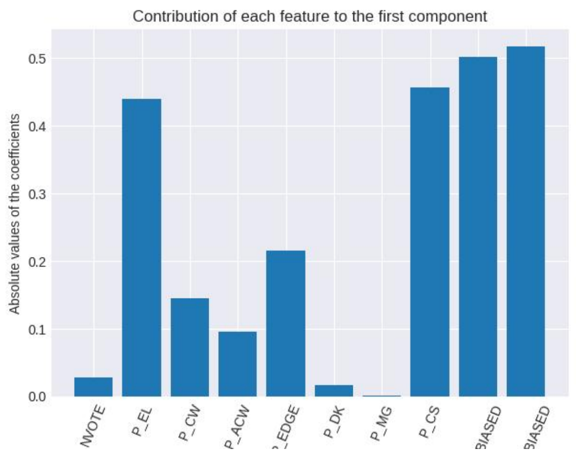

Figure 3.8: Absolute values of coefficients of the first principal component of the PCA. We observe that similarly to the decision tree, the feature ”P_CS_DEBIASED” is predominant.

more complex features from the results of the first layer. Subsequent layers apply even more complex filters to extract increasingly abstract and hierarchical features. After the features have been extracted, the results are flattened into a single dimension and sent through one or more dense neuron layers for classification.

3.3.2 Training of the network

The model consists of two convolutional layers, each followed by a pooling layer to reduce the size of the image representation. The first convolutional layer has 32 filters of size 3x3, while the second layer has 64 filters of the same size. The pooling layers use a window of 2x2. After the convolutional and pooling layers, the image is flattened into a one-dimensional vector, which is then connected to an output dense layer with a ”softmax” activation function to classify the image as either a spiral or elliptical galaxy. The model was trained on a dataset containing 10,000 images of spiral galaxies and 10,000 images of elliptical galaxies. The dataset was split into training and validation sets, with an 80/20 ratio. The model was trained using stochastic gradient descent (SGD) with a learning rate of 0.01 and a batch size of 32 images. The accuracy of the model was then evaluated on a test set of 5000 images of each type of galaxy. The achieved accuracy is 97.6%.

4 Conclusion

In this study, we compared several classifiers (see Table 4.1) to predict the nature of galaxies based on data about their shape or, in the case of the CNN, processed telescope images.

Classifier	Accuracy
Perceptron	74%
K-nearest neighbors	84%
Naive Bayes	90%
SVM	92%
Decision tree	86%
Random forest	93%
AdaBoost	81%
Multi-layer perceptron	91%
Convolutional neural network	98%

Table 4.1: Summary of results from different classifiers.

The results show that accuracy varies significantly among classifiers. The highest accuracy was achieved with the convolutional neural network, trained on images of spiral and elliptical galaxies, with an accuracy rate of 98%. Among other classifiers, the SVM and random forest also showed remarkable performances, with accuracy rates of 92% and 93%, respectively.

However, it is important to note that some estimators require more or less data to achieve their optimal performance. For example, the MLP was trained with more data than other estimators. We also noticed that classifiers have different computation times, which can be important in some practical applications. Additionally, the CNN was trained using only images of spiral and elliptical galaxies. PCA or k-means clustering allowed us to see that the boundary between elliptical and spiral galaxies is very clear, but that points corresponding to ”uncertain” galaxies are scattered between the two. Thus, by excluding uncertain galaxies during the CNN test, we greatly facilitated its task and hence the final accuracy.

It would be interesting to reproduce this analysis by excluding uncertain data from the training of all classifiers. We could then use our best classifier to determine the true nature of galaxies classified as ”uncertain”. Alternatively, we could use our current best classifier and use the classification probabilities given by the estimator.

Finally, for a more in-depth tabular analysis, the dataset of Hart et al. [13] is more precise and detailed. For such a dataset, including more features (around 230), the PCA techniques discussed in section 3.2 may prove indispensable.

Bibliography

[1] K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. Casteels, E. M. Edmondson, L. F. Fortson, S. Kaviraj, W. C. Keel, *et al.*, "Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 435, no. 4, pp. 2835–2860, 2013.

[2] H. Zhang, "Exploring conditions for the optimality of naive bayes," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 02, pp. 183–198, 2005.

[3] 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*, vol. 12, pp. 2825–2830, 2011.

[4] S. M. Omohundro, *Five balltree construction algorithms.* International Computer Science Institute Berkeley, 1989.

[5] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, pp. 199–222, 2004.

[6] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[7] F. Breiman, "Olshen, and stone," *Classification and Regression trees*, 1984.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[9] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[10] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[11] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural networks: Tricks of the trade*, pp. 9–50, Springer, 2002.

[12] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] R. E. Hart, S. P. Bamford, K. W. Willett, K. L. Masters, C. Cardamone, C. J. Lintott, R. J. Mackay, R. C. Nichol, C. K. Rosslowe, B. D. Simmons, *et al.*, “Galaxy zoo: comparing the demographics of spiral arm number and a new method for correcting redshift bias,” *Monthly Notices of the Royal Astronomical Society*, vol. 461, no. 4, pp. 3663–3682, 2016.