# Classification of galaxies using machine learning methods

Guillem Ramírez and Ramón García

## 1 Introduction

Machine Learning (ML) involves statistical methods that mimic learning processes. In recent years, it has been proven to be an interesting approach to prediction problems that involve vast amounts of data. The aim of this project is to classify large sets of pictures of galaxies into three different categories using ML algorithms.

## 2 Machine Learning methods

A large number of problems involve huge amounts of data, for which ML techniques can be used. These techniques use automatic methods for making predictions based on past experience or knowledge (*data*) of a system's behaviour. Two main categories are distinguished. If the data used for the training has the output that needs to be predicted (labels), the problem belongs to *supervised learning*. *Unsupervised learning* focuses on clustering -discovering coherent groups in data *clusters*-, finding lower-dimensional data representations, estimating the probabilistic mechanism that generates data and finding anomalous data.

The labels used in supervised learning may be either categories such as colours (classification problems), or being continuous variables such as the height of a population (regression problems). Although classification and regression problems hold obvious differences, they can be solved similarly. Both types of supervised learning have been used for this project.

### 2.1 Estimators and data

#### 2.1.1 Definition

Let $X$ be the set of data being analysed. In supervised learning problems a set $Y$ is also given, which may have some -unknown- dependence with $X$. The learning problem consists in building an *estimator* $\hat{Y} = L(X)$, which predicts up to some error the output vector $Y$. Estimators can range from very simple and intuitive models to very complex ones.

#### 2.1.2 Train-test sample. Bias/variance

It is important to have in mind that the objective is to build a model through an algorithm. This is the reason why in ML problems the sample is divided into two main parts: train and test sets. Basically, the model is built using both the input $X_{train}$ and output $Y_{train}$ from the training sample. After this, the algorithm makes a prediction for the output of the $X_{test}$ samples. Comparing this prediction with the actual values of $Y_{test}$ gives an estimation for how accurate the model is, which could be used from that moment on in order to make other predictions.

The concept of bias-variance tradeoff is also important. Bias can be defined as the difference between the predicted and the real output for the training sample. The variance indicates the variability of an estimator when trained with different samples.

### 2.1.3 Cross-validation

Since data are often scarce, this may lead to bad estimations of error. To solve the problem, K-fold crossvalidation uses part of the available data to fit the model, and a different part to test it. Namely, the data is split into $k$ roughly equal-sized parts. In each of the $k$ iterations, the $k$-th part is used as the test sample in which the error is evaluated. The other parts are used as the training samples. The prediction error is calculated for that model, and all the different errors are averaged.

### 2.1.4 Nearest-Neighbor Methods

Given a point, nearest-Neighbor Methods take into account the closests training points. Namely, the $k$-nearest neighbor fit for $\hat{Y}$ is defined as follows:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{2.1}$$

where $N_k(x)$ is the neighborhood of $x$, defined by the $k$ closest points $x_i$ in the training sample. The parameter $k$ can be changed up to the value desired, taking into account the following considerations. First, the higher the value, the more computational power that will be required in order to build the model. Secondly, $k$ has to be optimised in order to achieve an acceptable error for both the training and test samples. Hence, in the case $k$=1, since only one neighbor is considered (the point itself) the training error is zero, but leading to a worse classification (in terms of the test sample) in comparison to higher values such as $k$=15. This is can be more easily seen in Figure 1). Although the classification with $k$=1 has no training error, it leads to a more complex and counterintuitive model.

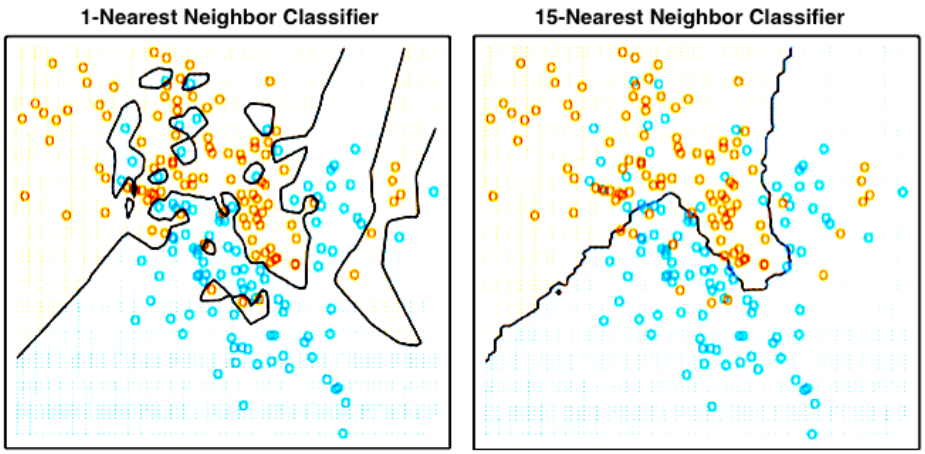



Figure 1: K-Neighbors method applied to a classification problem. On the left, it was used k=1, while k=15 was used in the right. Source: [1]

The definition provided works only in regression problems. However, the classification approach is essentially the same. It takes the most popular class among the $k$-closest neighbors instead of doing the average.

### 2.1.5 Decision trees and random forests

Decision trees can be understood as a set of consecutive decisions (nodes) which are split into two branches. Terminal nodes are called *leaves*. The simplicity of this method makes it very attractive, since the decision for a classification would be a cascade of binary conditions.

Decision trees used in this project are generated by following the CART algorithm without prunning [2]. At each node, the cut variable and the split-point are chosen amongst all the possibilities by minimising the Gini index. This is repeated until a maximum number of nodes or a maximum number of events/features on a leaf is reached.

In classification problems a class is assigned to every leaf, whereas in regression problems the assignment corresponds to a constant. The model function would be as follows:

$$\hat{Y}(x) = \sum_{m=1} c_m I(X_1, X_2) \in R_m \tag{2.2}$$

where $R_m$ is one of the regions the feature space has been divided into, and $c_m$ its constant associated.

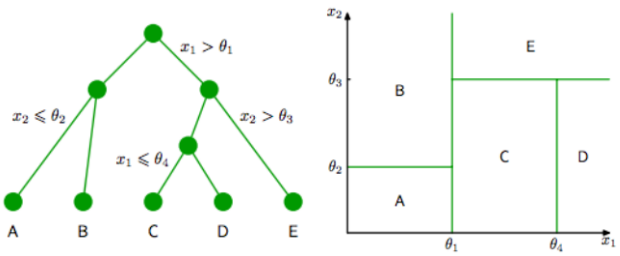



Figure 2: In the left, a tree of decisions and, in the right, its corresponding division on the feature space. Source: [2]

Random forests are essentially a set of $N$ decision trees that use bagging and bootstrapping. The idea in bagging is to average many different models, and hence reducing the variance. The bias of the combined set of trees increases with respect to the individual trees, and variance gets reduced. For each of the $N$ trees, the algorithm selects a bootstrapped (random) subsample from the training sample. It uses it to grow a decision tree in which only $m$ of the total available variables are used. These $m$ variables are selected randomly. Finally, all the generated trees are taken into account for predicting either the class or the value of the regression problem.

# 3 Data sample description

As stated before, the main goal of this project is to classify images of galaxies according to their type by using ML methods. A dataset of 70000 images of galaxies with their associated

types (training subset) as well as a dataset of 70000 images with no associated information (test subset) were provided by the Galaxy Zoo Project back in 2013 as part of a challenge in Kaggle [3] which consisted on developing an algorithm for galaxy classification.

The Galaxy Zoo Project was a citizen science project in which images of galaxies, taken by the SDSS telescope, were classified by people following the decision tree shown on Figure 3. The training subset mentioned before cointained a CSV file compiling people's answers to the classification tree for each galaxy. The goal of the challenge was to find the algorithm that could mimic the best the way people had classified galaxies.

This project builds on the mentioned challenge. While the classification tree for the Galaxy Zoo Project challenge was quite complete, our goal is only to classify a galaxy according to its shape into three categories: (i) Smooth (rounded) galaxy (ii) Galaxy with features or disk (iii) Star or artifact (no galaxy).

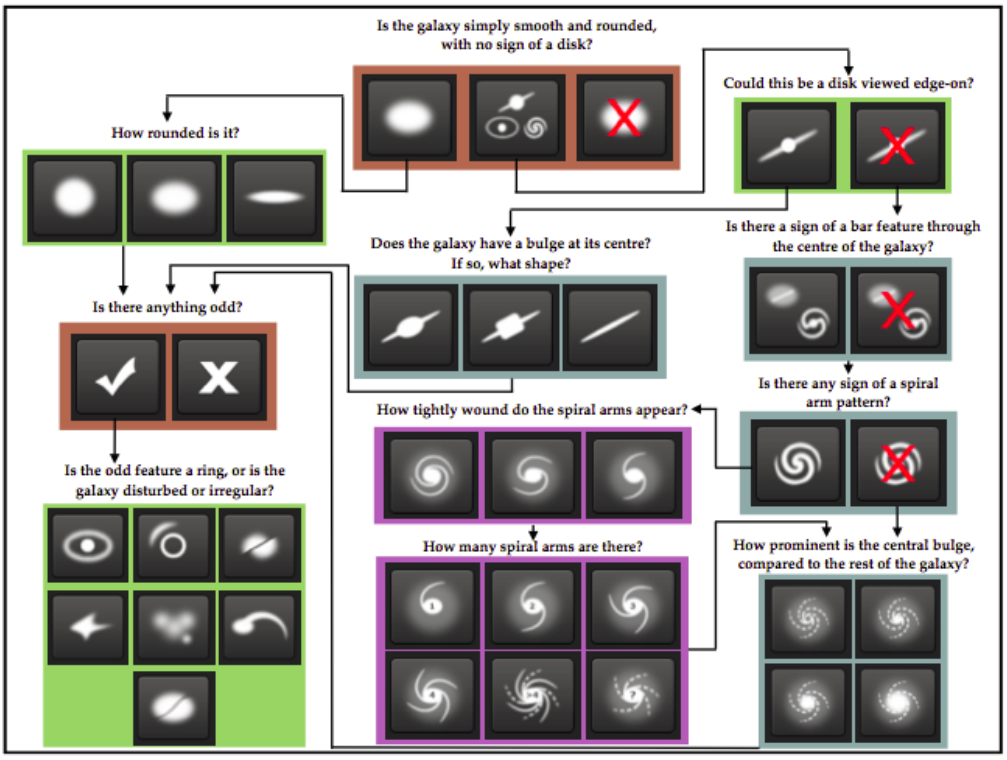



Figure 3: Decision tree of classifications for the Galaxy Zoo Project challenge (Willett et al. 2013) [5]. The goal of the project is to answer just the first question.

During the development of this project, only the training dataset is used. From this dataset, only the first 10000 images are taken. Although a ML model generally behaves better when it has been trained with a big dataset, it is also true that the time a model takes to be trained significantly increases with the number of training data it is fed with. Since computation time is a limited resource, a tradeoff has to be found. From these 10000 images, a random split is made and 7500 are used for the training subset, while the other 2500 constitute the test subset.

# 4 Galaxy classification results

In this section, results for the trained galaxy classification models are presented. Both regression and classification algorithms are used. A discussion on how to tune the best algorithm in order to improve the accuracy of its predictions is also included.

During the development of the project, Python is used along with the Jupyter Notebook IDE. Scikit-learn, an open-source library for ML in Python [4] is extensively used throughout the project, relying on its implementation of algorithms and methods. In order to use this library, NumPy, SciPy, and matplotlib libraries are also required.

## 4.1 Prediction by using regression algorithms

A first attempt by using regression algorithms is performed. In this case, each galaxy (object) is associated with a probability of being smooth, having features, or being point-like (continuous-valued attributes). These values range from 0 to 1.

From the three different ML algorithms described in Section 2 (K-Neighbors, Decision Trees, and Random Forests), the last two are used. Scikit-learn implements them as regression algorithms with *DecisionTreeRegressor*, and *RandomForestRegressor* classes.

The results obtained with the different regression algorithms when predicting the continuous values for the first question of the Galaxy Zoo Project decision tree are shown in Table 1. The mean errors in the table are the mean of training 10 different models with the same algorithm but with different randomly taken data:

| **Algorithm** | **Trees** | **Mean Error** |
|---|---|---|
| Decision Tree | - | $0.178 \pm 0.003$ |
| Random Forest | 10 | $0.138 \pm 0.002$ |

Table 1: Results for the different regression algorithms.

In order to compute the mean error, the difference between the predicted and the real number for each of the three categories is calculated -as an absolute value-. These three numbers are summed, so that we get the mean error for a galaxy. The mean errors of each galaxy are summed and divided by the number of galaxies in order to get the global mean error.

It can be seen that the Random Forest algorithm provides the best prediction.

## 4.2 Prediction by using classification algorithms

A second attempt is now made by using classification algorithms instead of regression. For the case of this project, objects are photos of galaxies, and the labels that have been picked are *smooth*, *features*, and *point*, as these are keywords which represent the three different categories in which galaxies can fit into. Galaxies of the training subset are assigned the label corresponding to the category which they have the maximum probability of belonging to.

Now, all three algorithms described in Section 2 are used. Scikit-learn implements them as classification algorithms with *KNeighborsClassifier*, *DecisionTreeClassifier*, and *RandomForestClassifier* classes.

When predicting the label for each of the test subset images, Table 2 shows how how the different regression algorithms behave. The incorrect guesses are the mean of training 10 models with the same algorithm but with different randomly taken data:

| **Algorithm** | **Neighbors/Trees** | **Incorrect Guesses (%)** |
|---|---|---|
| K-Neighbors | 5 | $31.69 \pm 0.7$ |
| Decision Tree | - | $34.33 \pm 0.9$ |
| Random Forest | 10 | $28.28 \pm 0.8$ |

Table 2: Results for the different classification algorithms.

It can be seen that again -as like with the regressors-, the Random Forest algorithm provides the best prediction. However, classifiers produce worse predictions compared to regressors. This can be explained from the fact that when working with continuous variables, there is more freedom in order to adapt the model compared to just choosing from a discrete set of labels.

## 4.3 Final results with model selection

It has been shown how different models trained using algorithms with different parameters behave. Concluding which is the best algorithm for training a good ML model, and especially which are the best parameters to tune this algorithm is not a trivial task.

For this purpose, Scikit-learn provides the `model_selection` module. This module allows to compare, validate, and choose parameters and models. The `validation_curve` class of this module is used in this project in order to evaluate the number of trees to use with the *RandomForestClassifier* algorithm, which as stated before, is the algorithm which provides a model with the best predictions.

When plotting the results of this piece of code, we get the following validation curve:

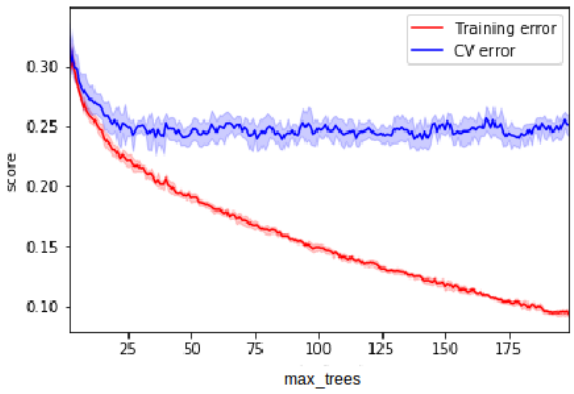



Figure 4: Validation curve for the RandomForestClassifier algorithm

It can be concluded, by looking at the CV (cross-validation) error, that the best performance of the algorithm is achieved when the number of trees is around 25. When recalculating the regression and classification results for the Ranfom Forest algorithm with these number of estimators, the mean error (for the case of the regression) drops from $0.138 \pm 0.002$ to $0.134 \pm 0.002$ and the incorrect guesses (for the case of the classification) drops from $28.28 \pm 0.8\%$ to $25.05 \pm 0.9\%$.

### 4.4 Results discussion

Models trained with regression algorithms seem to perform better than the ones trained with the classification algorithms, as they tend to produce lower prediction errors. However, experience gained during the development of the project shows that they also take quite more time to be trained and predict data.

Better results would have been obtained with classification algorithms if training data had been properly treated first. Tuning of the algorithms by optimizing their parameters would have had as well a positive impact. Had there been more time to devote to the project, more accurate models would have been obtained by implementing these suggestions, which are thus left for possible future reviews of this work.

## 5 Acknowledgements

Javi has been an awesome leader. We really appreciate his dedication, both in terms of time and effort. He built a fantastic local server to store all of the stuff required, he allowed us to use his computer in order to perform some calculations, the many references he has provided to us... He is one of those people who really enjoy what they do and he has shared his passion with us. Also, many thanks to all the *palomos* of PANDA for the crazyness and enjoyment provided.

## References

[1] T. Hastie, R. Tibshirani and J. Friedman: *The elements of Statistical Learning*, Springer, 2009.

[2] C. M. Bishop: *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] *Galaxy Zoo - The galaxy challenge*. First acces: June 2017. https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge .

[4] Pedregosa et al, *Machine Learning in Python*, JMLR 12, 2825-2830, 2011.

[5] Willett et al., *Galaxy Zoo 2: detailed morphological classifications for 204,122 galaxies from the Sloan Digital Sky Survey*, MNRAS, 2013.